KNOWLEDGE GROWTH IN AN ARTIFICIAL ANIMAL

by

Stewart W. Wilson

Rowland Institute for Science, Cambridge MA 02142

ABSTRACT

Results are presented of experiments with a simple artificial animal model acting in a simulated environment containing food and other objects. Procedures within the model that lead to improved performance and perceptual generalization are discussed. The model is designed in the light of an explicit definition of intelligence which appears to apply to all animal life. It is suggested that study of artificial animal models of increasing complexity would contribute to understanding of natural and artificial intelligence.

INTRODUCTION

The science of understanding and realizing intelligence in artificial systems needs a definition of intelligence. Every science needs good definitions of the problems it addresses. But in the artificial intelligence field there has been a hesitancy about defining intelligence. For example, on the first page of a recent, widely used AI textbook we find: "A definition in the usual sense seems impossible because intelligence appears to be an amalgam of so many information-representation and information-processing talents."[1] For many AI goals, this omission is not important. But the lack of a good working definition can lead to uncertainty in evaluating progress toward understanding intelligence *per se*, even though results are in other respects substantial.

This paper reports work using an artificial, behaving, animal model to study intelligence at a primitive level. An explicit definition of intelligence is adopted, and guides construction of the model. The definition has intuitive appeal and apparent applicability to the range of life from human beings to very primitive animals. Because of this range, some results with the primitive animal model should provide insight into intelligence in general.

A DEFINITION OF INTELLIGENCE

A good definition should be relatively simple and yet cover most of the things we regard as belonging to the concept and few we regard as not belonging. The psychological literature offers a number of useful similar efforts but the best definition of intelligence we have found is the following, from the physicist van Heerden:

> Intelligent behavior is to be repeatedly successful in satisfying one's psychological needs in diverse, observably different, situations on the basis of past experience.[2]

This definition (vH) is suitable for the computer study of intelligence because it is comprehensive and its terms are not difficult to define computationally for experimental purposes. A high rate of receipt of certain reward quantities can correspond to "repeatedly successful in satisfying one's psychological needs" (on the simplest level, somatic needs). To "diverse, observably different, situations" can correspond sets of distinct sensory input "vectors" with each set having a particular implication for optimal action. To "past experience" can correspond a suitable internal record of earlier interactions with the environment, and their results.

THE ANIMAT MODEL

Computer modeling of human levels of intelligence is complex. VH's apparent applicability to both simple animals and human beings (assuming appropriate translations of its terms) suggests the usefulness of the easier course of considering basic problems that simple animals must solve, and constructing behaving models aimed at solving them. Observation of the models should aid understanding of all intelligence, and the construction of more complex models.

To define our model, we abstract four basic characteristics of simple animals:

1) The animal exists in a sea of sensory signals. At any moment only some signals are significant; the rest are irrelevant.

2) The animal is capable of actions (*e.g.* movement) which tend to change these signals.

3) Certain signals (*e.g.* those attendant on consumption of food), or certain signals' absence (*e.g.* absence of pain) have special status for him.

4) He acts, both externally and through internal operations, so as approximately to optimize the rate of occurrence of the special signals.

An animal's sensory-motor situation is described in very general terms by (1) and (2). Characteristics (3) and (4) are assumptions which provide a way of making definite the notion of "needs" and their satisfaction. Together, the four characteristics form the basis of our artificial animal model. For brevity, we call such a model an "animat".

We take as the animat's basic problem the generation of rules which associate sensory signals with appropriate actions so as to achieve the optimization of (4), above. For this, the major questions are adaptive, namely:

1) How to discover and emphasize rules that work.

2) Get rid of those that don't (since memory space is limited and noise is undesirable), and

3) Optimally generalize the rules that are kept (since space is limited).

There is some previous work along these lines. Notable were Grey Walter's: *machina speculatrix*, which was a sort of sub-animat which chose actions based on needs and the sensory situation, but did not adapt its rules; and *m. docilis*, which could be taught a conditioned response[3]. More recently, Holland and Reitman[4] exhibited successful performance by a rule-adaptive animat-like system which optimized its rate of satisfaction of two distinct needs. Booker[5] experimented with an animat-like "hypothetical organism" which adapted its rules in a simple environment that contained both attractive and aversive stimuli; he also provides a review of earlier systems. The present investigation is indebted to the last two works.

IMPLEMENTATION

Within the above framework we make the model definite by defining the animat's: environment, sensory channels, repertoire of actions, its association rules, and then its performance and adaptation algorithms.

Environment:

A rectangle on the computer terminal screen 18 rows by 58 columns and continued toroidally at its edges defines the environmental space. Alphanumeric characters at various positions represent objects; the animat itself is denoted by *. Some, possibly many, positions are just blank.

Sensory Channels:

In studies so far, * has been given the ability to pick up sensory signals from objects which happen to be one step (row and/or column) away, in any of the eight (including diagonal) directions; nothing is detected from more distant objects. Thus the "sense vector" has eight positions. With * located, for example, as shown below left, the sense vector would be as shown at the right:

```
T T
* F      T T F b b b b b ,
```

where b stands for blank. To form the sense vector, the circle of positions surrounding * is mapped, clockwise starting at 12 o'clock, into a left-to-right string.

But this vector is not the final sensory input. We imagine that an object is ultimately sensed as the outcome of measurements upon it by one or more feature or attribute detectors. Without loss of generality we assume each detector produces either a 0 or 1 output. If there are d detector types, an object translates into a binary string d bits in length. The sense vector as a whole thus translates into a "detector vector" of $8d$ bits. Detector translations or encodings of objects are fixed in *'s "low-level" sensory hardware. They are assigned at the beginning of an experiment. For example, in experiments discussed here, "F" (food) is encoded as "11"; "T" (tree or obstacle) as "01"; and "b" (open space) as "00". [The first bit might be thought of as the output of a "food smell?" detector; the second, of an "opacity" detector.] Thus the above sense vector translates into the detector vector:

01 01 11 00 00 00 00 00 .

The associative apparatus takes the detector vector as input.

Repertoire of Actions:

*'s actions are restricted to single-step moves in each of the eight directions. The directions are numbered 0-7 starting at 12 o'clock and proceeding clockwise; for example, a move in direction 3 would be south-easterly.

The animat may move, or attempt to move, to a position occupied by an object. The environment's response for each kind of object is predefined. In present experiments, if the move is into a position whose encoding is 00 (the blank object), there is no response (though the new sense vector will in general be different). If * steps into a space occupied by an object whose encoding has the first bit equal to 1, * is regarded as having eaten the object and receives a reward signal. If * tries to step toward an adjacent object whose encoding is 01, the step is not permitted to occur (a collision-like banging may be displayed).

The foregoing establish a semi-realistic situation in which sensory signals carry partial, but uncertain, information about the location of food, and avail-

able actions permit exploration and approach. Environmental predictability can be varied through the choice and arrangement of the objects. The number of object types which may be experimented with is limited only by the number of bits in the detector encoding scheme.

Association Rules:

For its association rules, the animat uses a rudimentary form of Holland's[6] "classifier" rule. The animat's rules each consist of a "taxon" and an "action". The taxon is a sort of template capable of matching a certain set of detector vectors. The action is some one of the available actions. The animat's classifier says, in effect, "if my taxon matches the current detector vector, then consider taking this action." It is a kind of hypothesis about what to do given a certain sensory situation (class of detector vectors). An example of a classifier would be:

0# 01 1#. 0# 00 00 0# 0# / 2 .

The matching rule requires that for any taxon position having a 0 or 1, the same value must occur in the detector vector; taxon positions with # (don't care) match unconditionally. Because of the #'s, which confer a kind of generality on the classifier, the above taxon, for example, will match 32 possible detector vectors, including the one discussed earlier.

It is worth making a few further observations about this classifier. First, it is a pretty good one because if food is present in direction 2 and the classifier matches the detector vector, the action recommended is to move in direction 2 and not some other direction! Second, in directions 0, 3, 6, and 7, the taxon only requires that the object be, in effect, non-food, it being irrelevant whether these directions have obstacles or are blank. Directions 1, 4, and 5 have not been so generalized. Broadly speaking, a classifier is more useful to the animat to the extent it is general (matches many detector vectors) without being so general that it makes too many errors (*i.e.*, that in certain matching situations its recommended action is inappropriate).

Besides taxon and action, each classifier possesses a "strength", a quantity serving as the principal measure of a classifier's value to the animat. There may be other associated quantities, as well.

The animat keeps a classifier population [P] of fixed size. Usually, [P] is initialized by filling all the taxa with 0, 1, and # according to some random rule; actions are similarly filled in. As the animat's CRT "life" evolves, the classifier population changes, as will be described.

PERFORMANCE ALGORITHM

*'s basic cycle is one "step", within which events having purely to do with immediate behavior are very simple. First, the current detector vector is calculated. Second, [P] is searched for classifiers which match it; these form the "match set" [M]. Third, a classifier is selected from [M] using a probability distribution over the strengths of [M]'s classifier's; that is, the probability of selection of a particular classifier is equal to its strength divided by the sum of strengths of classifiers in [M]. Fourth, * moves according to the action of the selected classifier, or tries to. The environment's response to the move will be as described earlier.

It can be seen that *'s move choice tends to be the one having the greatest total strength among the [M] classifiers advocating it. Thus, overall, * first asks which classifiers of [P] "recognize" the current sensory situation, then from these tends to pick the move with the greatest associated strength. The subset of [M] consisting of classifiers whose action is the same as the chosen action is called the "action set" [A].

ADAPTATION ALGORITHM

The adaptation algorithm has three distinct aspects: 1) reinforcement of classifier strengths; 2) "genetic" operations on classifiers yielding new classifiers; and 3) direct creation of classifiers.

Reinforcement:

As discussed in the last section, a classifier's strength is a major determinant of its ability to influence *'s action and therefore performance. We consequently want strength to reflect the performance which tends to result when this classifier is in [A]. That would be straightforward if every step were rewarded: we could, for example, adjust the classifier's strength by an amount proportional to the reward. Classifiers which got bigger rewards would be stronger, thus more likely to be an [A], etc.

Realistically, however, it is usually the case that only some of an organism's actions receive a definite reward from the environment. Actions leading up to, or setting the stage for, a rewarded action are themselves not directly rewarded, but they must somehow be encouraged or the final payoff will not occur. Holland[7] addressed this problem in proposing a "bucket-brigade" algorithm in which, very briefly, 1) classifiers make payments out of their strengths to classifiers which were active on the preceding cycle, and 2) the same classifiers later correspondingly receive payments from the strengths of the next set of active classifiers. External reward goes to the final active set in the chain. In effect, a given amount of external reward will eventually flow all the way back through a reliable chain, reinforcing every precursor classifier.

Our basic implementation of this idea is as follows. On each step:

1) all classifiers in [A] have a fraction e of their strengths removed;

2) the total strength thus removed from [A] is distributed to the strengths of any classifiers in [A-1], defined as the action set in the previous step;

3) * then moves and if external reward is received it is distributed to the strengths of [A]; if external reward is not received, the classifiers of [A] replace those of [A-1].

Thus every [A] participates in general in two transactions, one paying out, the other receiving. We can write

$$S'_A = S_A - eS_A + p \quad ,$$

where S_A is [A]'s total strength on one step, S'_A its total on the next, and p is the total payoff received (either external reward or from the next [A]). If p is the same over time, S_A approaches a constant value given by p/e, so that under reasonably steady payoff conditions, S_A is an estimator of typical payoff. Similarly, the strength of any individual classifier is an estimator of its typical payoff.

The total payoffs to [A] and [A-1] are in the simplest case shared equally by the recipient classifiers. This has the consequence that the more classifiers are in, say, [A], the less payoff each gets.

Genetic Operations:

Consider two classifiers which match similar situations:

0# 01 1# 0# 00 00 0# 0# / 2

and

0# 0# 11 01 00 0# 0# 0# / 2

Each is good, but each still lacks something in generality since, for example, the matching requirements for 01 in bits 2-3 and 6-7, respectively, of each are perhaps unnecessarily restrictive. Suppose we make a new classifier by combining bits 5-9 of the first with bits 0-4 and 10-15 of the second. The result would be the slightly more general classifier:

0# 0# 1# 0# 00 0# 0# 0# / 2 .

The above operation on two classifiers resembles a kind of crossing-over or recombination of chromosome parts in genetics. It is an operation in which two "parent" classifiers produce an offspring that is possibly an improvement over both of them. Another "genetic" operation, this time using just one parent, would first clone the parent, then mutate one or more of the clone's taxon positions. Other types of operations on classifier structure can be imagined (one will be discussed later). In each case the attempt is to use existing classifiers as the starting points for improved classifiers.

But the crossover points above were chosen quite carefully; otherwise the offspring might have been no improvement, or even a retrogression (to a classifier more specific than either parent). We do not expect the animat to know where best to cut and mutate. How can we expect genetic operations to be of any use?

Holland[8] presents a mathematical theory showing that a population of individual symbol strings, in which each string can be assigned a numerical worth, will progressively increase in average worth as its members undergo reproduction, genetic operations on or among the offspring, and deletion of individuals to maintain constant population size. The key requirement is that an individual's probability of reproduction be proportional to its worth. Holland extended the theory to include classifier systems. In employing genetic operations, our animat constitutes an exploration and test of the theory.

The specific algorithm employed is as follows:

1) A first classifier $c1$ of [P] is selected with probability proportional to its strength;

2) If $c1$ is merely to be reproduced, a copy of it is made and added to [P]. To make room, some classifier is deleted;

3) If $c1$ is to be crossed with another classifier, a second, $c2$, is selected, also with probability proportional to strength, but from the subset of [P] of classifiers having the same action as $c1$. Two cut points are chosen as above, but at random, and an offspring $c3$ constructed out of the parts. $c3$ is added to [P] and some classifier is deleted.

Note that the parents are kept (unless one happens to suffer the deletion, but this is unlikely). The offspring, in effect, go into competition for payoff with the parents. Better (higher strength) offspring should proliferate more rapidly than their parents, driving them out; for worse offspring, the reverse should be the case.

"Create" Operations:

Occasionally, as * executes the performance algorigthm, a detector vector may occur that no classifier of [P] matches, *i.e.*, the situation is unrecognized. The animat's response is to create a new, matching, classifier. A taxon is made by adding some #'s at random to the detector vector; an action is chosen randomly. The created classifier is added to [P] and one is deleted. The new classifier immediately matches the previously unrecognized situation and action occurs by the normal mechanism.

EXPERIMENTAL PROCEDURE

The animat model was designed with the vH-intelligence definition as a guide. In experiments with the model we are interested in finding procedures and parameter values that seem to give *

greater rather than less vH-intelligence. For this two measures have been adopted. One is a performance measure: given an environment, how many steps does * take, on average, to find food objects? The other is a generality measure: does * evolve classifiers each tending to be useful in a number of distinct situations? Generality is important because it suggests that a high level of performance developed in one environment will carry over to a somewhat different environment.

The experimental procedure is to fix *'s methods and parameters, then have him do a large number of "problems" in a particular environment E. The measures of performance and generality are tracked. A "problem" always consists of starting * at a randomly selected blank position in E; then * moves until he eats some food, at which point the problem ends. The number of steps between start and food is recorded; a moving average of this quantity over the previous 50 problems is the performance measure, STPSAV.

To track generality, we calculate a histogram over the "periods" of all classifiers in [P]. The period of a classifier is a moving average of the number of steps by * between occurrences in [A] of this classifier. Thus a frequently used classifier will have a low period. [P] will then be general to the extent the histogram of periods is largest at low period. As [P] evolves we expect the histogram peak to move toward lower period, if [P]'s generality is increasing.

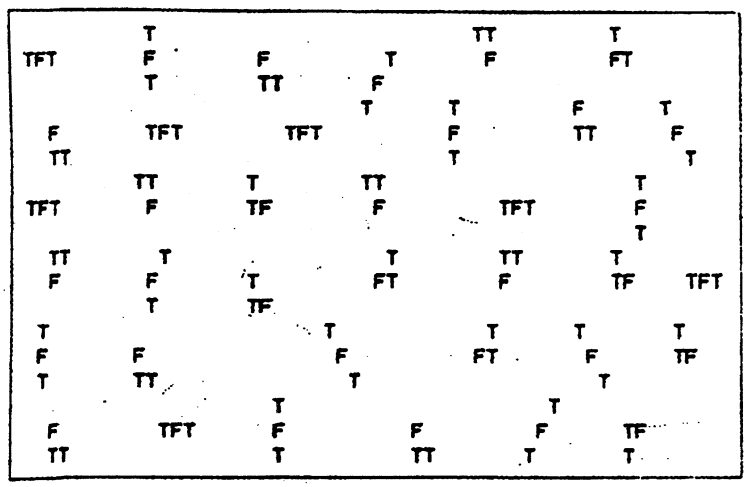

Figure 1. The Environment "WOODS7".

An environment used for many of the experiments is "WOODS7", shown in Fig. 1. Although WOODS7 may look easy, it actually contains a total of 92 distinct sense vectors, so *'s need to discover and generalize is substantial. To obtain performance baselines, we can start * randomly, then let him also move completely randomly until food (F) is bumped into. For WOODS7, the long-term average of the number of steps this takes is about 41 steps. We may also ask [9]: what is the best possible performance (if, say, the animat had human capabilities)? For every starting position, the number of steps to the nearest F can be found and averaged over all starting positions. The result for WOODS7 is 2.2 steps.

RESULTS AND DISCUSSION

Fig. 2 shows a performance curve for a combination of procedures and parameter settings that is among the best so far found. There is an initial rapid improvement within the first 1000 problems (untypically good during the first 100 problems, where STPSAV usually stays above 15), followed by very gradual improvement thereafter. The performance at 8000 problems, between 4 and 5 steps, is quite respectable compared with "perfect" (2.2 steps), especially since * has no information whatsoever until he is next to a nonblank object.

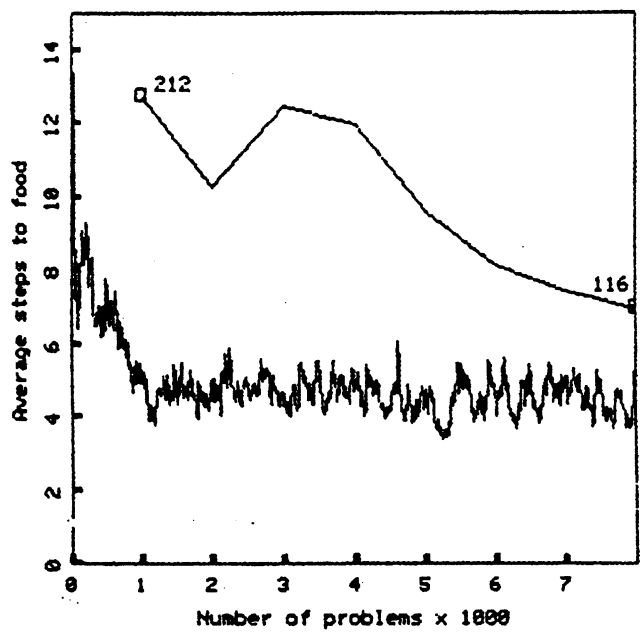

Figure 2. STPSAV (ragged line) and Period Average (broken line) for * to 8000 problems. Period values as marked.

For the same animat, Fig. 3 shows the histogram of periods of [P] at 8000 problems. There is a definite bulge for low periods; the average period is 116. For comparison, the broken line in Fig. 2 shows the trend of the period averages at earlier epochs, indicating gradual generalization in the sense we have defined.

Qualitatively, a * such as this one gives the impression of "knowing" the Woods quite well. When next to F, * nearly always takes it directly; occasionally he will move one step sideways and take it from that direction. When next to one or more T's,

but with no F immediately in sight, * quite reliably steps around the obstacle(s) and finds the F. When * is "out in the open", *i.e.*, the sense vector consists of blanks, he has no information about the best way to go, as in a thick fog. One might expect *'s behavior to resemble a random walk but this is not the case. Instead, the movements look more like a general "drift" in some direction, with some superimposed randomness. After several problems the drift may shift to another direction.

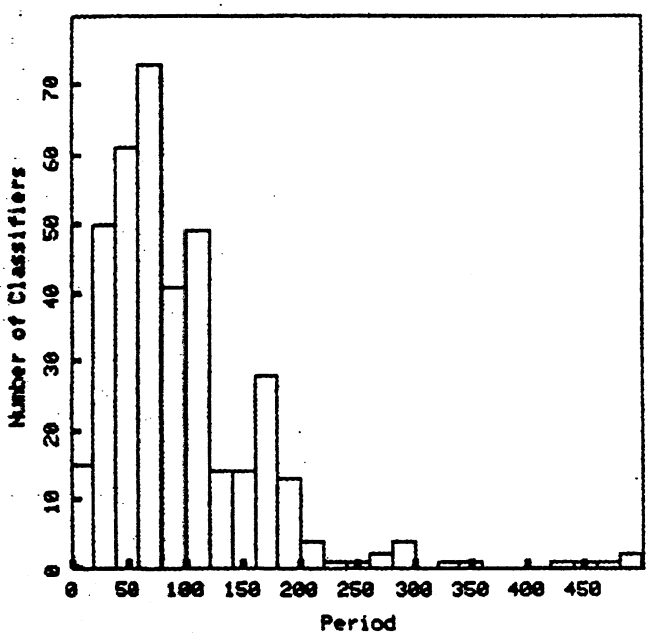

Figure 3. Histogram of classifier periods for the * of Figure 2 at 8000 problems.

Parameter Values:

Parameter values for the animat of Fig. 2 were arrived at by experiment. Three basic parameters are discussed in this section, with observations about setting them reasonably.

For Fig. 2, |P| contained 400 classifiers. A suitable value for this number appears related to the number of distinct sense vectors or "scenes" (here, 92) in the environment. Too small a ratio of classifiers to scenes results in "forgetful" behavior in which * keeps losing good moves that appeared well learned. A small ratio means that for some scenes deletion has a high probability of eliminating all matching classifiers. For ratios above about four, the forgetting is much less noticeable. To the extent * generalizes, more and more classifiers match each sense vector, further reducing the problem.

The "estimator fraction", e, was set at 0.2, *i.e.*, a classifier lost 20 percent of its strength each time it entered [A]. In general, smaller values of e mean that a classifier's strength reflects a weighted average of payoffs that reaches farther into the past. Conversely, a larger value makes the strength more sensitive to recent payoffs. It was found that $e = 0.4$ produced a noticeably more erratic STPSAV curve, whereas changing from $e = 0.2$ to 0.1 did not affect the curve significantly. Strength should accurately estimate a classifier's typical payoff. In this problem, payoff fluctuations are apparently large enough so that $e = 0.4$ results in too short an averaging interval for good estimation. If e is too small, though, newly formed classifiers may get evaluated too slowly; we therefore kept e at 0.2.

The rate at which genetic operations occurred was set proportional to the problem rate. Specifically, at the end of each problem, a single genetic event (as described earlier) took place with probability RGPROB. Given the event, crossover occurred with probability XPROB. Settings were typically 0.25 and 0.50, respectively. These seemed to ensure that, on average, classifiers would be fully evaluated by the reinforcement process by the time they were selected for a genetic operation (or deleted). Typically, a problem took five steps in which each set [A] had about 10 members, giving about 50 evaluations. The above value for RGPROB then implies 200 evaluations per genetic event. This seems excessive except that some classifiers are much more frequently used than others and we wanted to allow for the well-rewarded but infrequently called-upon classifier. It is possible our results would have been speeded up, without adverse side effects, by a higher genetic rate.

Distance Estimation:

Performance in the earliest animat experiments was far below the level of Fig. 2. One defect was a kind of "dithering" in which while * would tend toward F's, the path would have unnecessary sidesteps and wanderings. It was then realized that the basic reinforcement algorithm does not care whether a path from point A to food is long or short; there is nothing which preferentially reinforces the most expeditious classifiers. Any path, even a looping one, will come to equilibrium at a high strength level in its constituent classifiers.

The solution had to be more subtle than simply penalizing long paths. What is required is a technique that, at every position, tends to prefer the most direct of several possible moves, but does not prevent the setting up of a long path if that is actually the shortest path available. Our solution was twofold. First, each classifier was made to keep an estimate of its distance (in steps) to food. This did not require elaborate look-ahead. Instead each classifier in [A-1] adjusted its distance estimate according to an average of the distance estimates of [A]; when reward was received, the members of [A] were similarly adjusted, using the quantity 1. This tech-

nique, with each estimate an average over the last few updates, is quite satisfactory.

The distances are employed as follows. In the performance cycle, selection from [M] is based on probability proportional to strength/distance instead of just strength. Consequently, a move tends to be selected that is not only strong, but also "short". Now comes the second part of the solution. At the same time as [A] is formed, the set NOT[A] of the remaining classifiers in [M] is taxed by a small amount (typically five percent): the "longer" classifiers thus tend to incur a loss by not being selected. This "lateral inhibition" induces a sort of catastrophe in which the shorter classifiers become even more likely to be picked and the longer become ever weaker, and can disappear entirely. Note that the competition is purely local and does not work against the setting up of minimal long paths.

This technique is very effective against "dithering"; the progressive takeover of a match set by a discovered shorter move has been repeatedly observed. Our solution is not perfect, however, because to suppress the special case of occasional looping situations we had to impose a small tax (five percent) on [A]. Since [A] is the set which receives payoff, the tax has little effect except if a loop is taking place, and then the tax is soon very effective. Still, in principal, even a small tax on [A] reduces the strength flow in very long chains, putting them at a reproductive disadvantage. This residual problem may be an indication that as paths grow, they should be "condensed" into units of behavior longer than one step.

Extensions to "Create"

A second area of changes which improved performance had to do with the "Create" operations. As discussed, Create at first only occured when [M] was empty. It was found that * sometimes also got stuck looping among situations with nonempty [M]'s. The tax on [A] enabled recognition of these loops because the total strengths in each [A] would tend to zero. We put in a threshold that triggered Create if the strength of any [M] got too low. This suppressed looping dramatically and improved performance.

It was also found important to trigger Create randomly, at a very low rate (typically, with probability 0.02 per step). * is engaged in path construction, using the best available current evidence. This can lead to good but nevertheless suboptimal paths which might be improved if * would only try something different. Random Creates are one way to introduce a new move direction. Usually the new classifier is no improvement. But when it is, and it gets tried (gets in [A]), it will be (often heavily) reinforced and therefore given a good chance at eventual reproductive success.

A different type of Create was also found useful. Instead of randomly picking the action in a Created classifier, * may make an educated guess, as follows. From its current position, * steps tentatively into a randomly selected adjacent position. There, [M] is determined and the strength-weighted average of the distances of its classifiers, MNDIST[M], is formed. The same is done for several adjacent positions. These values are then compared with MNDIST[M] for the starting position. Several decision schemes are possible, with the general idea of picking an action direction corresponding to the shortest apparent path. If, however, none of the adjacent MNDIST[M]'s is better by more than 1 than the current position's value, it is preferable not to create a new classifier. This technique is important early in *'s existence, when very little is yet known; but, interestingly, it appears that * should not rely entirely upon it. Some suboptimal paths get set up which tend not to be improved. The problem goes away if random Creates are also available.

Effect of Genetic Operations:

Finally, we shall discuss what the experiments suggest about the role of the genetic operations. To begin, it is helpful to define a "concept" as a set of classifiers from [P] having exactly the same taxon and action, and for which there is no other classifier in [P] with that taxon and action. The basic effect of *'s genetic operations then appears to be to exert a pressure tending to increase the generality of [P]'s concepts. That is, with time, the periods of the concepts in [P] tend to decrease. The pressure is restrained by the requirement that the concepts be more or less correct (* must get the food expeditiously). The precise point of balance appears to depend on the parameter regime.

An important experiment is to evolve an animat with reinforcement and Create going as usual, but with genetic operations turned off. The result is a performance almost as good as Fig. 2. But significant generalization does not occur; the curve of histogram averages remains essentially flat at a value of about 270. There thus appears to be a division of effort: Create introduces the raw material, the specific examples to be evaluated; and the genetic operations produce more general concepts from the examples.

It is clear that crossover is capable of making a more general classifier out of two less general parents; this was illustrated earlier. We are not sure, however, just why for * the more general concept has a selective advantage. Somehow, greater generality must lead to greater concept strength; there is no other way to win out. Yet being active more frequently does not in itself result in greater strength: strength is an estimator typical payoff, not payoff rate.

Our tentative hypothesis stems from noting that a more specific concept will always have to share payoff with any more general offspring that comes into existence. This initially weakens the specific concept so that the number of classifiers making it up tends to fall (at equilibrium, numbers are proportional to total strength). Consequently, the specific gets even less of the payoff, since payoff is shared. The result is a cascading situation in which the more general concept wins out. The odds favor the general because it has more than this one source of payoff.

While general classifiers appear to have a selective advantage, this is of no use unless such classifiers can be formed and introduced in the first place. Crossover is adequate for some types of generalization. But a natural operation for the purpose is obviously intersection. We have implemented this operation as follows. Two parents are chosen and a new taxon is formed by intersecting copies of the parents' taxa over a randomly selected interval. In that interval, if the parents differ at a position, the new taxon gets a #; if not, the new taxon gets the common value. Outside the interval, the new taxon is filled in from parent 1.

Intersection is a "hot" operation which should be used cautiously because it can introduce #'s at a high rate. Nevertheless, our results show increased generalization with little performance loss when crossover and intersection are both available to *.

Space remains only discuss the deletion technique. The simplest method, conceptually, is to delete at random. Then, to a first approximation, the equilibrium number of classifiers in a concept—or in any subset of [P] whatsoever—is proportional to its total strength. A drawback of random deletion is that a valuable concept that happens to consist of one classifier is at considerable risk until it reproduces. This is not a problem on average if [P] is large enough. Yet one wonders whether "deleting the weak" might not be better.

Several methods have been tried, all but one clearly worse than random deletion. The possibly better method is to delete with probability proportional to the reciprocal of strength. This has the obvious effect of tending to protect the precious classifier just mentioned. It can also be shown that the probability that a concept [C] will lose a member under this type of deletion is proportional to the square of its number, which places a strong restraint on over-expansion.

The * of Fig. 2 employed both intersection (along with crossover) and inverse-strength deletion.

CONCLUSION

In its simple way, * meets the definition of intelligence stated at the beginning. * becomes good at satisfying its need for food in a Woods of diverse object configurations on the basis of experience. Though not yet tested, *'s rule generalization over time suggests that performance would be maintained in a somewhat different Woods, or if the Woods slowly changed.

While the present animat has numerous limitations (sensory, motor, memory, etc.) there does not seem to be any essential barrier to removal of the limitations and to carryover of the present algorithms to a more sophisticated model in more complicated environments.

ACKNOWLEDGEMENT

The author wishes to acknowledge valuable conversations with C.G. Shaefer of the Rowland Institute.

REFERENCES

[1] Winston, P.H. *Artificial Intelligence*, 2nd ed. Reading, Massachusetts: Addison-Wesley, 1984.

[2] van Heerden, P.J. *The Foundation of Empirical Knowledge*. Wassenaar, The Netherlands: Wistik, 1968.

[3] Walter, W.G. *The Living Brain*. New York: Norton, 1953.

[4] Holland, J.H., & Reitman, J.S. Cognitive systems based on adaptive algorithms. In *Pattern-Directed Inference Systems*, Waterman, D.A., & Hayes-Roth, F., (eds.). New York: Academic Press, 1978.

[5] Booker, L. *Intelligent Behavior as an Adaptation to the Task Environment*, Ph.D. Dissertation (Computer and Communication Sciences), The University of Michigan, 1982.

[6] Holland, J.H. Adaptation. In *Progress in Theoretical Biology, 4*, Rosen, R., & Snell, F.M., (eds.). New York: Plenum, 1976.

[7] ____ . Genetic algorithms and adaptation. In *Adaptive Control of Ill-Defined Systems*, Selfridge, O.G., Rissland, E.L., & Arbib, M.A., (eds.). New York: Plenum, 1984.

[8] ____ . *Adaptation in Natural and Artificial Systems*. Ann Arbor: University of Michigan Press, 1975.

[9] Martha Gordon, personal communication.