HIERARCHICAL CREDIT ALLOCATION IN A CLASSIFIER SYSTEM

Stewart W. Wilson
The Rowland Institute for Science
100 Cambridge Parkway
Cambridge MA 02142 USA

ABSTRACT

Learning systems which engage in sequential activity face the problem of properly allocating credit to steps or actions which make possible later steps that result in environmental payoff. In the classifier systems studied by Holland and others, credit is allocated by means of a "bucket-brigade" algorithm through which, over time, environmental payoff in effect flows back to classifiers which take early, stage-setting actions. The algorithm has advantages of simplicity and locality, but may not adequately reinforce long action sequences. We suggest an alternative form for the algorithm and the system's operating principles designed to induce behavioral hierarchies in which modularity of the hierarchy would keep all bucket-brigade chains short, thus more reinforceable and more rapidly learned, but overall action sequences could be long.

I INTRODUCTION

Many learning systems face the problem of temporal credit allocation: the proper reinforcement of activities which do not directly result in need satisfaction or external reward but are nevertheless essential precursors to such outcomes. Animals learn extensive hunting, stalking, or foraging behaviors aimed at the ultimate payoff of something to eat. A person who values others' cooperation must discover and reinforce effective precursor strategies. In the message-passing, rule-based *classifier systems* (Holland, 1986), credit is allocated by means of a "bucket-brigade" algorithm to earlier-acting rules which "set the stage" for later actions that bring external payoff. The essential idea is that classifiers which match messages and become active on a given time step "pay" a fraction e of their "strengths" to the strengths of the classifiers which posted the messages and were active on the previous time step. When finally external payoff enters the system, it is added to the strengths of the then currently active classifiers. If over time a given payoff-achieving sequence gets repeated, strength increments will in effect flow back to reinforce its early-acting classifiers. Consequently, early-acting classifiers that indeed participate in sequences that make possible later payoff will, by the algorithm, receive due credit.

In certain other AI systems which learn to perform multiple-step tasks [*e.g.*, Mitchell's LEX system for symbolic integration (Mitchell, Utgoff, and Banerji, 1983), and the ACT* cognitive architecture of Anderson (1983)], credit is assigned to early steps by keeping and analysing a record of all pre-payoff actions, both considered and taken, and the associated reasoning. In contrast, Holland's bucket brigade technique does not depend on retrospective analysis but operates locally, during performance, in the strength transaction between steps, with the better classifiers at each step being selected statistically over time. The bucket-brigade principle would consequently appear appropriate for systems—such as animals and autonomous robots in on-going interaction with uncertain environments—where storage and analysis of raw experience is expensive or impractical.

In this paper, however, we suggest that the bucket-brigade may lose effectiveness as action sequences grow long. As a remedy, we propose a modification of the algorithm that makes it more directly reflect the hierarchical nature of behavior.

II LONG CHAINS

The mechanics of the bucket brigade suggest that a classifier whose early action indeed contributes to later payoff may still have difficulty getting reinforced if the number of message-posting cycles from its activation to payoff is large. As a simple example, suppose that classifier C posts a message which, by triggering an effector, causes an action (*e.g.*, application of hand pressure to a restaurant door) that leads eventually, n time-steps (message-postings) later to payoff in the form of satisfaction at the taste of food. N will at least equal the number of intervening elementary actions, which may be very large. If over time C is to be properly reinforced as a member of the sequence, the sequence will have to be repeated as many times as it takes the strength increment due to the food payoff to "reach" C.

To estimate the number of repetitions required, we used a simple simulation in which a list of n strengths represented a bucket-brigade chain of n classifiers. Setting the strengths initially at zero, we provided external payoff R at one end and ran the chain repeatedly, according to the bucket-brigade rule of the previous section, until the nth strength reached 90% of R/e, where (as can be shown) R/e is the assymptotic value approached by all strengths in the chain. The number of repetitions required, $M_{90\%}$, was, to a close approximation,

$$M_{90\%} = (3 + 1.2n)/e,$$

for values of e in the range from 0.1 to 0.4. E should be kept small so that classifier strengths average over a

number of payoff events; typically, e is chosen to be no greater than 0.1. Given that value, our equation says that a "stage-setting" classifier just 10 steps from environmental payoff will require no fewer than 150 repetitions of the sequence to be properly reinforced.

III BEHAVIORAL MODULES

Clearly, something is wrong if the reinforcement algorithm must feed strength increments back through the enormous number of elementary steps between, say, the push on a restaurant door and the enjoyment of food. Intuitively, that sequence consists of just a few big steps: <enter restaurant>, <get a table>, <get food>, <eat>. If the algorithm treated *these* as the bucket-brigade units, reinforcement would be faster since the chain would be short. Somehow we must also reinforce the smaller steps which compose the big ones. But we note that <enter restaurant> can be broken into the sequence: <find door>, <open door>, <go through>, and that <open door> expands, in turn, into a short sequence one of whose components is <push>. Albus (1979), among others, shows how any complex activity can be decomposed into a hierarchy of behavioral modules each consisting of just a few "steps". If the bucket brigade could apply hierarchically to module steps, we might be able to reinforce quite extended activities without encountering the "long chain" problem.

Our approach to this objective is to modify the standard classifier system's performance and reinforcement algorithms so as explicitly to encourage behavioral modules and short bucket-brigade chains [see Holland (1985) for a different suggested approach]. The basic change is to use a hierarchical message list instead of the standard homogeneous one in which all messages have equal status, and, for the moment, to allow at most one message at a time on a given level. Our plan of exposition is first to take the reader through an example, then to present the new algorithm, and finally to discuss questions which the algorithm raises.

IV AN EXAMPLE

Figure 1 illustrates the operation of the hierarchical performance and reinforcement algorithm over a certain interval of 17 time-steps. The figure shows principally the contents of the system's message list, but also indicates environment changes, actions, and bucket-brigade flows.

At time t_0 we imagine that the message M_1 spontaneously appears on a previously empty message list. M_1 is special in that it represents an internal system need, *e.g.*, <get food>, in which case we could say that the system has just felt (renewed) hunger. We note that M_1 stays on the list until the very end of the epoch, when food (R) is received. M_1 is in effect the *name* of a behavioral module (intent, plan, subprogram) with the purpose <get food>.

At t_0 the message from the environment was E_1 (top of figure). Since the system took no external action on that time step, the same environment holds (we assume) at t_1. But the overall situation is different at t_1 since the message list contains M_1. The system now forms a *match set* consisting of all classifiers which match both E_1 and M_1. From the match set, a single classifier is picked (through a competition based on classifier strengths) and that classifier's message, M_2, is posted on the list on the next level down. The interpretation is that M_2 names a module of M_1 that applies when the environment is E_1.

Still the system has not made an external action. At t_2, a match set is again formed with the proviso that its members must match E_1 (still unchanged) and M_2, illustrating the matching rule: "if no external action occurred in the previous time-step, compare only against the *lowest* level message on the list in forming the match set." The rationale is that the lowest level message represents the system's most immediate intent, which should have priority. Again, the figure shows the posting of message M_3 and thus a deepening of the hierarchy.

At t_3 something new happens. Following the same rules as above, the system picks a winning classifier whose message specifies an external action A_1. In this case the action is taken and no new message is posted (action messages cause only actions). We have reached the level of a module (M_3) whose components are not intents or submodules but external activity.

At t_4, a new matching rule applies: "if an external action occurred during the previous time-step, compare against *all* messages on the list; if the (again) *single* winning classifier matched a message on level k of the list, erase all lower-level messages (if any) and post the winner's message one level below k." In the current case, we see from the figure that M_3 must have been the highest message matched (since no messages got erased) and that the winner's message was the action A_2. The interpretation is that the system simply executes another action belonging to M_3.

At t_5, bigger changes occur. The second matching rule (the "ascent" rule) again applies, and this time the winning classifier matched M_2 (and E_3), resulting in erasure of M_3 and the posting of M_4. Here the interpretation is that, given environment E_3, module M_2 moves on to its second submodule M_4; *i.e.*, its first submodule, M_3, has been successfully carried out.

We now have enough information to understand the rest of the figure. From t_6 through t_8, the system executes the actions of M_4, but this also completes M_2. At t_9 the system enters "descent" (the first matching rule applies) and begins execution of the module M_5, which lasts until t_{16}. Note that the first three steps of M_5 are actions but the fourth is a submodule. Finally, the fifth step, the action A_{10}, results in external reward entering the system, which causes erasure of the entire message list.

The "ascent" matching rule, which acts most dramatically at t_8 and t_{16}, is designed to identify the highest-level module to which the environment resulting from the current action is relevant, and to terminate all lower level modules. This corresponds to the observation that completion of a high-level subplan usually means completion of all subplans which immediately underlie it. For example, completion of the subplan <get a table> under the plan <get food> also completes <take a seat> and, under that, <pull the chair back up to the table>, etc.

The operation of the bucket brigade in figure 1 is illustrated by the small arrows, which indicate strength

flows. An arrow from one message to another, as between M_4 and M_3, means a payment from the classifier which posted M_4 to the classifier which posted M_3. As usual, the amount involved is a fraction of the strength of the source classifier, and it is added to the strength of the recipient. Similarly, an arrow from one action to another (or from a message to an action, or vice versa) means a payment between the two corresponding classifiers. The special case of an arrow leaving the first step of a module, as with M_2, means a fraction of the strength of the posting classifier is simply removed and "thrown away".

At time-steps 5, 8, 15, and 16, a more complicated payment pattern occurs. For example, at t_8, the standard strength fraction is deducted from the classifier which sent M_5, but the resulting quantity is paid to *each* of the three recipients indicated by the arrows. That is, if an amount Q is deducted from the source classifier, each recipient has its strength incremented by Q. Similarly, at t_{16}, the payoff quantity R (and not one-third of R) is paid to each of the three recipients shown. The intent of this "non-splitting" of payoff is to encourage hierarchical deepening where appropriate; a different rule may of course turn out to be better.

We may note in figure 1 how the bucket-brigade pattern causes strength flows along the constituent steps of each module, thus reinforcing the steps in the spirit of the original bucket-brigade principle. But this "hierarchical bucket brigade" also achieves our objective of reducing the length of any individual chain. Note that the overall activity of figure 1 consists of ten action steps (and 17 time-steps) yet no classifier is more than five payment steps from the external reward. More generally, hierarchy means that the average payment sequence length will be of the order of log n, where n is the number of steps in the overall activity.

V HIERARCHICAL ALGORITHM

We now state the hierarchical performance and reinforcement algorithm.

1) Obtain the current message E from the environmental input interface.

2) If phase="descent", form the *match set* [M] of all classifiers which match both E and the lowest-level message on the message list, else

   If phase="ascent", form the match set [M] of all classifiers which match both E and any message on the message list.

3) Compute the *bid B* of each classifier C in [M] by taking the product of C's strength and a small constant (say 0.1).

4) Select a classifier C* from [M] using a procedure in which higher-bidding classifiers are more likely to be selected.

5) Reduce C*'s strength by the amount of its bid; then

   If phase="ascent", pay an amount equal to B to each of the classifiers (if any) which sent messages lower on the list than the message matched by C*, erase those lower messages, and pay an amount B to the classifier whose action was carried out on the previous time-step;

6) If C*'s message is an external action, set phase="ascent",

   Else post the message on the next lower empty level of the message list and set phase="descent".

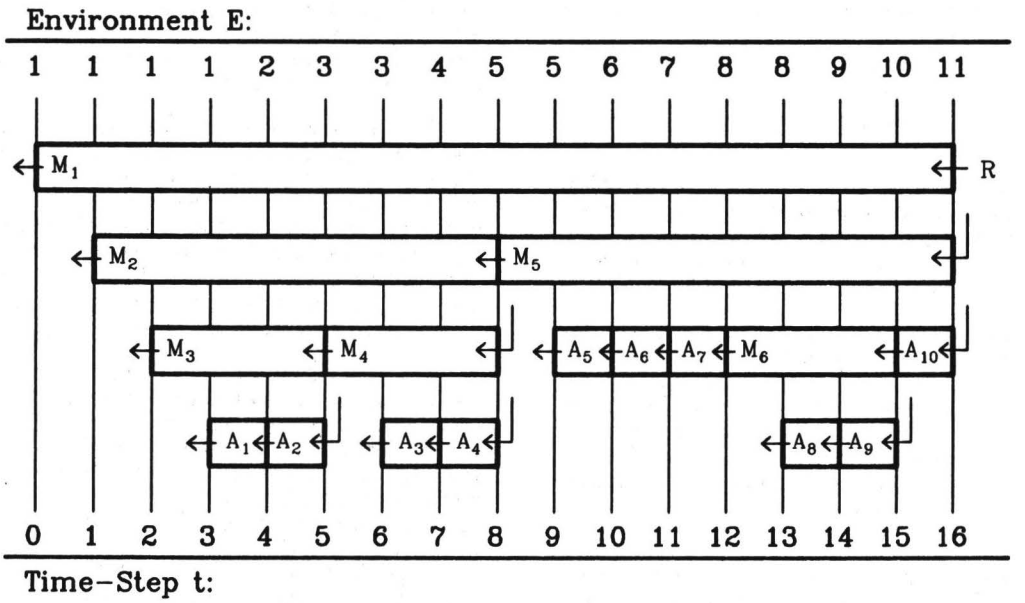

Figure 1 Example of Hierarchical Performance and Reinforcement Algorithm

7) If the message of step 6 was an action, take it.

8) If payoff R is received from the environment, pay amounts equal to R to each of the classifiers which sent messages now on the list, erase all messages, and pay an amount R to the classifier whose action was just taken.

   Set phase="descent".

9) Return to step 1.

The new algorithm leaves some operational questions unanswered. For instance, we are not told what to do in step 2 if the match set [M] is null (this is also not covered in the standard algorithm). In "descent", the sensible thing would seem to be to assume that the most recent posting (lowest message) was a "mistake", erase it, and retry the match against the next higher message. In "ascent" the situation is more complicated, but failure to match is less likely since the whole list is matched against. A possible response would be to "reverse" the last action (if possible) and retry the match. In both cases, the stochastic element in the selection of C* (step 4) would permit alternative outcomes. If the system became truly stuck in a certain state or loop, a "fatigue" process could come into play, causing messages gradually to drop from the list. All these questions are more properly addressed at the level of the system routine of which the hierarchical algorithm is a component.

VI DISCUSSION

An important difference between the hierarchical classifier system outlined here and the standard system is that parallelism appears to be greatly reduced. The standard classifier system permits numerous messages to be posted in each cycle, whereas the hierarchical system permits the addition, to those already on the list, of no more than one message per cycle. Parallelism in the standard system is intended to serve several functions (Holland, 1986). Having numerous classifiers active on each cycle should allow the system simultaneously to test numerous hypotheses about the best way to get to payoff. Over time, those that profit in the bucket brigade should win out (and, under the discovery algorithm, become progenitors of new, possibly even better, classifier hypotheses). Secondly, parallelism is intended to give the system gracefulness in the sense that when control is divided among a cluster of rules, the failure or absence of one rule can be expected to have only a marginal effect on performance. Finally, complex situations may be more flexibly represented internally by a set of numerous activated rules, each responding to an element of the situation, than by a few, or just one, rule which would have to encompass all relevant aspects in its condition. In short, multiple activation is intended to give the system a more powerful *mental model* of the world (Holland *et al.*, 1986).

This is clearly an important objective for any learning system. We suggest, however, that the hierarchical system is not so narrow as it may appear. At any moment, in general, the message list contains a number of messages, which could be taken to represent a mental model, but in this case an hierarchical one. The higher level messages represent broader, more general, aspects of the situation than the lower level messages. Selection of a message for posting on a given level is the result of a competition which on another occasion could well pick a different classifier's message for testing. Furthermore, the hierarchical system can be modified to permit more than one message on each level (essentially, one lets C* be a set instead of a single classifier, but there is not space here to go into detail). The single place where the hierarchical system is clearly "narrower" than the standard one is in "descent", where our rule is that only the lowest-level message gets matched against, corresponding to the principle that a plan cannot be achieved before its subplans. (To prevent insensitivity to environmental surprises, a multi-level interrupt can be provided by adding "If E differs from the previous E, set phase='ascent'" to step 1.)

Further research is needed to determine the hierarchical classifier system's merit. We intend to apply it in an extension of our previous "animat", or artificial animal, simulations (Wilson, 1985, in press). In concluding, we would stress the hierarchical system's two apparent plusses: short bucket-brigade chains and explicit modularity.

REFERENCES

[Note: *ICGATA* abbreviates J.J. Grefenstette (Ed.), *Proceedings of an International Conference on Genetic Algorithms and Their Applications.* Pittsburgh: Carnegie-Mellon University.]

Albus, J.S. (1979). "Mechanisms of planning and problem solving in the brain." *Mathematical Biosciences*, **45**, *247-293.*

Anderson, J. R. (1983). *The architecture of cognition.* Cambridge: Harvard University Press.

Holland, J.H. (1985). "Properties of the bucket brigade algorithm." *ICGATA*. pp. 1-7.

Holland, J.H. (1986). "Escaping brittleness: The possibilities of general-purpose learning algorithms applied to parallel rule-based systems." In R.S. Michalski, J.G. Carbonell & T.M. Mitchell (Eds.), *Machine learning; an artificial intelligence approach. Volume II.* Los Altos, California: Morgan Kaufmann Publishers, Inc.

Holland, J.H., Holyoak, K.J., Nisbett, R.E., and Thagard, P.R. (1986). *Induction: Processes of Inference, Learning, and Discovery.* Cambridge, MA: MIT Press.

Mitchell, T. M., Utgoff, P. E., & Banerji, R. (1983). "Learning by experimentation: acquiring and refining problem-solving heuristics." In R. S. Michalski, J. G. Carbonell & T. M. Mitchell (Eds.), *Machine learning, an artificial intelligence approach.* Palo Alto, California: Tioga.

Wilson, S.W. (1985). "Knowledge growth in an artificial animal." *ICGATA*, *16-23.*

Wilson, S.W. (in press). "Classifier systems and the animat problem." *Machine Learning.*