# Greedy Utile Suffix Memory for Reinforcement Learning with Perceptually-Aliased States*

**Leonard A. Breslow**
Navy Center for Applied Research in AI (Code 5510)
Naval Research Laboratory
Washington, DC 20375, USA
*breslow@aic.nrl.navy.mil*

### Abstract

Reinforcement learning agents are faced with the problem of perceptual aliasing when two or more states are perceptually identical but require different actions. To address this problem, several researchers have incorporated memory of preceding events into the definition of states to distinguish perceptually-aliased states. Recently, McCallum (1995b) has offered Utile Suffix Memory (USM), an instance-based algorithm using a tree to store instances and to represent states for reinforcement learning. USM's use of online instance-based state learning permits state definitions to be updated quickly based on the latest results of reinforcement learning. The use of a *fringe* (an extension of the tree to a prespecified depth below the 'real' tree) provides the algorithm a limited degree of lookahead capability. However, the use of a fringe incurs a large cost in terms of tree size and provides only limited lookahead capability. We introduce a modification of USM, Greedy Utile Suffix Memory (GUSM) to address these concerns. GUSM uses a *positive* criterion for splitting a leaf node, similar to that used in USM, based on the immediate *advantage* of node splitting and resultant tree expansion. In addition, GUSM includes a *negative* split criterion, based on the *inadequacy* of a leaf node and the state it represents for determining the next correct action to take. GUSM is able to solve problems on which USM will sometimes fail (i.e., depending on the setting of the fringe depth parameter). In addition, GUSM usually produces trees that contain fewer nodes overall (i.e., real + fringe nodes) and solves problems as rapidly as USM.

**Keywords**: reinforcement learning, perceptual aliasing, instance-based learning

*NCARAI Technical Report No. AIC-96-004.

# 1 Introduction

Reinforcement learning agents are reactive systems that learn to map situations, or states, into actions so as to maximize future discounted reward. Such agents are faced with the problem of perceptual aliasing when two or more states are perceptually identical but require different actions (Whitehead and Ballard, 1991). Purely reactive policies do not produce optimal performance in such situations. For instance, a robot attempting to locate the mail room in an office building may reach an intersection that is perceptually identical to other intersections in the building. To address this problem, some researchers have incorporated memory of preceding events into the definition of a state. For instance, the robot might learn that if it enters the intersection *after* first passing the elevators on its left and then passing the stairwell on its right, it should turn left at the intersection. Thus, perceptually-aliased states may be differentiated by including information on events (i.e., observations and actions) that precede the current observation. The goal in methods that incorporate memory into state definitions is to incorporate memory *selectively*, i.e., when and only when memory helps the agent disambiguate perceptually-aliased states and thereby learn the next correct action to take in such states.

Accordingly, methods for differentiating aliased states engage in two simultaneous learning processes: learning the correct *state representation* and reinforcement learning of the correct *policy* of actions to take in each state. Both learning processes feed into and influence one another. Some methods conduct state learning offline from reinforcement learning (Chrisman, 1992; McCallum, 1993). They use statistical tests to determine when a state is perceptually aliased and needs to be split into two or more states based on their histories. These algorithms learn slowly because offline state learning lags behind reinforcement learning and thus reinforcement learning often proceeds on the basis of a state representation that is not as up-to-date as it could be given the agent's experience to date.

Instance-based learning (IBL)(Aha et al., 1991) offers an approach to state learning that appears more consistent with the dynamic, interactive relationship between state learning and reinforcement learning. The entire sequence of past events is stored and continuously reinterpreted by the agent in the light of subsequent experience. In one IBL approach, McCallum (1994a;1994b) introduced a $k$-nearest neighbor algorithm, Nearest Sequence Neighbor (NSM), that constructs state definitions dynamically for each new observation the agent encounters. NSM has complementary strengths and weaknesses to the offline learning approaches. It learns much more rapidly, probably due to its dynamic incorporation of history information by online state learning. However, NSM provides less optimal solutions, since it does not assess the relevance of the history information that it incorporates into states.

As a compromise between these two approaches, McCallum (1995b) introduced Utile Suffix Memory (USM). Like NSM, USM is an instance-based algorithm. Like the offline learning approaches, USM uses statistical tests to determine the relevance of history information. USM uses a *prediction suffix tree* (PST) (Ron et al., 1994) to organize event instances

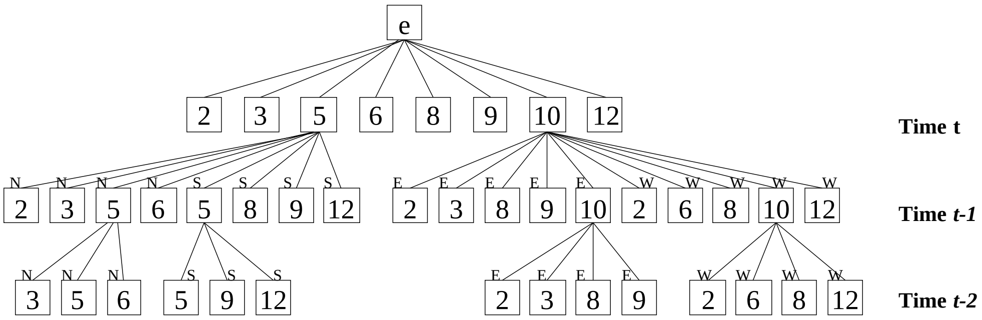


Figure 1: Suffix Tree for maze task (fringe nodes not shown).

into states based on their past histories. The agent constructs the tree progressively from the top down during the course of its problem solving and learning. Each current observation is represented by a node at level one of the tree and a tree path from level one to a leaf represents a sequence of actions and observations in reverse chronological order. The tree construction process performs a limited degree of lookahead search by the maintenance of a fixed-depth *fringe* below the tree. The fringe consists of virtual tree nodes of prespecified depth beneath the true leaf nodes of the tree. There are two limitations to the use of a fringe for lookahead. First, the fringe is very expensive in terms of memory and storage, since a tree grows exponentially relative to its depth. Second, the fixed range of lookahead provided by the fringe is not sufficient to solve problems requiring greater lookahead.

We introduce an improved version of USM, Greedy Utile Suffix Memory (GUSM), to address these problems. Whereas USM uses a *positive* criterion for state splitting (i.e., the immediate improvement in predictions of the next action to take), GUSM uses a *negative* criterion as well; states are split if they *fail* to provide clear recommendations for correct actions even if the splitting provides no immediate advantages. In this way, a minimal fringe (one node deep) is sufficient to solve problems requiring indefinite amounts of lookahead without excessive tree expansion. With greedy state splitting, the tree is expanded only in places where it is most in need of improvement. GUSM is shown to learn action policies as quickly as USM, to succeed on problems on which USM fails, and to generate trees of smaller overall size (ie., real + fringe nodes) than USM.

In Section 2, we describe the USM algorithm in more detail. In Section 3, we describe the GUSM algorithm. In Section 4, we report experimental comparisons between the algorithms. In Section 5, we describe directions for future research. Finally, in Section 6, we conclude our discussion.

## 2 Utile Suffix Memory

The prediction suffix tree in Figure 1 is the most parsimonious state space represenation adequate to learn the maze in Figure 2, from McCallum (1995b). The root node is empty. The

| | | | | | | |
|---|---|---|---|---|---|---|
| 9 | 10 | 10 | 8 | 10 | 10 | 12 |
| 5 | | | 5 | | | 5 |
| 5 | | | G | | | 5 |
| 5 | | | 5 | | | 5 |
| 3 | 10 | 10 | 2 | 10 | 10 | 6 |

Figure 2: McCallum's maze navigation task.

level-1 nodes each represent a different *current* observation, at time $t$, denoted by numbers. Nodes at lower levels represent a previous observation and subsequent action (denoted by the small letters above each pictured node).[1] The path from a leaf node to a time-$t$ node represents the sequence of events preceding the observation at time $t$ that are relevant to determining the next correct action to take at $t+1$. Each such path represents a state in the state space; the leaf node of the path contains the set of event instances whose histories match the event sequence of the path. Nonaliased states are represented simply by a single time-$t$ leaf node (e.g., observations 2,3,6, etc.), since no history information is needed to determine the next action to take at $t+1$. The tree is expanded only below time-$t$ nodes that represent aliased states and only to the extent such expansion improves the selection of the correct next action to take at $t+1$.

To determine the next action to take, the agent first finds the tree path that matches the event history of the current observation. The event instances stored in the leaf node of each path are subdivided by the possible actions at time $t+1$. Each event contains a Q-value estimate of the future discounted reward, which is updated periodically by reinforcement learning. The action with the highest average Q-value in the leaf is selected.

As a concrete example, refer to the maze problem in Figure 2. On each trial, the agent is placed in an arbitrary starting cell in the maze and moves one cell at each time step until it reaches the goal cell, labeled $G$. The agent's observation in a cell is defined entirely by the (non)existence of a barrier at each of the four compass directions. Thus, cells labeled with the same number (e.g., 5) are all perceptually identical and the corresponding states are perceptually aliased if different actions are required in the cells. Aliased states are differentiated by history information using the state tree in Figure 1. For example, the rightmost leaf node, with observation 12 and action West, contains the set of past instances

[1] The trees used by McCallum (1995b) contain separate nodes for actions and observations. We consolidate corresponding action and observation nodes, since aliased states typically cannot be differentiated on the basis of previous actions alone in the class of problems of concern here where possible previous actions are deducible from current observations. This consolidation results in a great reduction in tree size. The conclusions of this paper should apply equally to trees in which observation and actions nodes are differentiated.

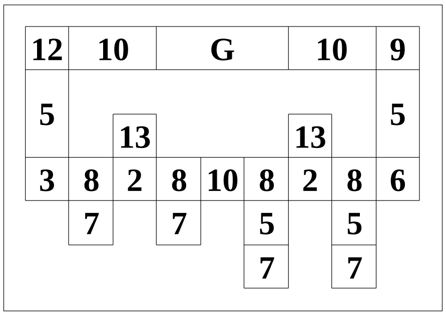


Figure 3: A noncontiguous maze navigation problem.

of observation 10 at time $t$ that were preceded by the event sequence: observation 12 and action West, following by observation 10 and action West. This sequence unambiguously identifies this particular instance of observation 10 with the maze cell directly East of the cell labeled 8 (See Figure 2). The agent should learn that the next correct action to take from this state is West. In contrast, the correct action to take from certain other cells with observation 10 (e.g., the one West of 8) is East.

The tree initially consists only of a root and level-one (nonaliased) nodes. The tree is expanded online during reinforcement learning based on the ongoing results of learning. The goal of tree expansion is to produce a state space using only history information that is *relevant* to the selection of correct actions and, thus, to reinforcement learning of action policies. Relevance is determined by a test of statistical significance[2] conducted following each action the agent makes. If the distribution of Q-values for possible actions in a leaf node differs significantly from the distribution in a potential descendant of that node, then the leaf node is split, becoming an internal node, as the descendants become leaf nodes.

Potential leaf nodes are represented in USM in a fixed-depth *fringe*, an extension of the actual tree below the "real" leaf nodes. The fringe provides a limited degree of lookahead; a leaf node is compared to each of its descendant nodes in the fringe. A statistically significant difference in Q-value distributions between a leaf node and one of its descendant nodes, $f$, results in the "splitting" of the leaf node, i.e., the extension of its subtree down to the depth of $f$. The former leaf node becomes an internal node, while node $f$ and its siblings become leaf nodes. The fringe is then extended to the prespecified depth below the new leafs.

USM has been shown to learn even in the presence of noise. The reader is referred to McCallum (1995b) for further details of the USM algorithm.

# 3 Greedy Utile Suffix Memory

[2] The statistical test is the Kolmogorov-Smirnov nonparametric two-sample test (Siegel, 1956).

Two disadvantages of the use of a fixed-depth fringe are (1) the increased size of the resulting tree and (2) the limitations of the lookahead offered by the fringe as a means to overcome horizon effects in tree expansion. With regard to the first, the addition of a fringe of fixed depth $d_f$ adds exponentially to the size of the tree that must be stored or retained in memory. The same depth of fringe is maintained uniformly below the leaf nodes of a tree, including those leafs that provide an adequate basis for prediction.

The second disadvantage, horizon effects, manifests itself when the algorithm is applied to a problem having states with *noncontiguous relevant events*. USM shares this disadvantage with the offline learning approaches discussed earlier. All of them assess history information for inclusion in a state definition one event at a time in reverse chronological order, the order in which the USM tree is expanded. As soon as one event fails the assessment, these algorithms do not explore events further back in time. This limitation could prevent the algorithm from differentiating perceptually-aliased states in certain circumstances. For instance, in the maze in Figure 3, the two cells labeled 2 cannot be differentiated on the basis of the immediately prior event, since in both cells the previous event is equally likely to be an East action from a cell with observation 8, a West action from a cell with observation 8, or a South action from a cell with observation 13. Expanding the tree below node 2 at level $t$ to level $t-1$ would, therefore, not be justified, since it offers no immediate advantage. However, branching further below to $t-2$ *does* provide a useful distinction, differentiating the two aliased observation-2 states. The child nodes of the observation-2 node in which the correct next action is East are characterized by previous observations 10-East-8-East, 5-North-8-East, 5-North-8-West, and 6-West-8-West. The children of the observation-2 node in which the correct action is West are 3-East-8-East, 7-North-8-East, 7-North-8-West, and 10-West-8-West. (Note that 8-East and 8-West appear at the ends of both sets of sequences.) For convenience, I refer to problems involving noncontiguous relevant events, such as the maze in Figure 3, as *noncontiguous problems*.

USM has the advantage over the offline learning methods of using a fixed-depth fringe of nodes to provide some lookahead capability. However, the amount of lookahead may not always be adquate to solve noncontiguous problems. In the problem just described, a fringe of depth $d_f = 1$ would be inadequate to solve the problem. A fringe of depth $d_f = 2$ is adequate for problems such as this having a "relevance gap" of one time step in the event sequence, but would be insufficient to handle a noncontiguous problem with larger relevance gaps.

To handle noncontiguous problems, we introduce the Greedy Utile Suffix Memory algorithm (GUSM). In GUSM, a leaf node is split if *either* a *positive* or *negative* criterion is satisfied. A positive criterion for node splitting, such as the criterion used in USM, is one that depends on the advantages offered by a potential new child of the leaf node being considered for splitting. A negative split criterion is based only on the *inadequacy* of the current leaf node, whether or not its child nodes offer any improvement. The negative criterion used in GUSM is satisfied if a leaf node fails to offer a clear action recommendation.

Specifically, it is satisfied if the recommended action in the node is *not* significantly superior to the action having the second-highest value (and if the node has a sufficient quantity of event instances to allow meaningful statistical comparisons.) On the basis of this negative criterion, the agent can expand a tree below a leaf node of inadequate predictivity until it reaches a leaf node that is predictive. Rather than requiring a large fixed-depth fringe to handle hard noncontiguous problems, GUSM can handle all such problems with a fringe of depth $d_f = 1$. The positive split criterion allows GUSM, like USM, to expand the tree in places where there is evidence of immediate positive benefits from expansion. Using the negative criterion, GUSM expands the tree in places where the tree provides an inadequate basis for prediction. One disadvantage of the negative criterion is that it may produce an indefinite amount of tree expansion in *indeterminate* states in which no one action is objectively superior to the others. In practice, we have found that the agent successfully learns the problem before tree expansion proceeds very far in indeterminate states.

We modified USM in one additional respect. USM performs many unnecessary statistical comparisons. Specifically, USM compares a leaf node to each of its descendant nodes in the fringe on each possible next action. If *any* of these comparisons reveals a statistical difference, the leaf node is split. This will occur even if the descendant nodes do not make a different *recommendation* from the leaf. The resulting tree expansion provides no benefit in such cases. In contrast, GUSM restricts statistical comparisons to those that are potentially useful. Specifically, a node is compared with one of its descendant nodes only if it makes a *different* recommendation from the descendant, and the two nodes are compared only on their respective recommended actions.

## 3.1 Details of the Algorithm

The *event history* is represented as a list of events records. Each event, at time step $i$, includes four pieces of information: an observation $o_i$, the action taken following the observation $a_i$, the immediate reward obtained from the action $r_i$, and a Q-value measure of future discounted reward $q_i$. The event history is represented as a doubly linked list:

$$\langle o_1, a_1, r_1, q_1 \rangle \Longleftrightarrow \langle o_2, a_2, r_2, q_2 \rangle \Longleftrightarrow \cdots \Longleftrightarrow \langle o_t, a_t, r_t, q_t \rangle$$

Each event record also contains a pointer to the leaf node in which it resides.

Each leaf and internal *node* in the PST tree contains:

- A relative time step label $t - n$ (i.e., relative to level 1 nodes representing the current observation at time $t$).
- Observation $o_{t-n}$,
- Action $a_{t-n}$,
- Average Q-values for each possible action following $t$: $Q_{a_i}$.
- The sets of events for each possible action following $t$: $\{E_{a_i}\}$.

- A pointer to the parent node, $P$.
- Pointers to child nodes, $C_a^o$, corresponding to possible events (i.e., observation-action nodes) at time $t-n-1$. For leaf nodes, child nodes are implemented as virtual *fringe* nodes. When a fringe node becomes a leaf node, new fringe nodes are added beneath it. A fringe node is identical in structure to a leaf node, except that it has no children.

GUSM starts with a tree consisting only of the root node and with an empty event chain. The GUSM algorithm consists of the following steps.

1. **Select the next action** for the current observation $o_t$ in event $e_t$.
   (a) Find the leaf node in the tree whose history matches the history of observation $o_t$ (including $o_t$ itself). ( Matching is based on observation and actions only, not on rewards.) If no matching leaf node exists, create a new leaf node for the event.
   (b) Select the action in the leaf node having the highest mean Q-value. With some exploration probability $p$, select an action randomly instead.
2. **Perform the selected action.** Record the action and immediate reward within the event record of $e_t$. Create record $e_{t+1}$ and record within it the new observation $o_{t+1}$.
3. **Update the Q-values** in all events $i$ in the leaf node (and fringe node) that contain the chosen action, using the Q-learning rule
$$q_i \leftarrow (1-\beta)q_i + \beta(r_i + \gamma U_{t+1}) \tag{1}$$
where $\beta$ is the learning rate, $\gamma$ is the discount rate, and $U_{t+1}$ is the utility of the next event, defined as
$$U_{t+1} = \max_a Q_{t+1}(a) \tag{2}$$
The utility of a node $U$ is the Q-value of the action recommended by the node.
4. **Update node data.** For each node on the path from the leaf to the root, in order, update the mean Q-value of the action selected.
5. **Revise the tree if necessary.** At the leaf node:
   (a) Determine whether the criterion for branching is satisfied, according to both the positive and negative criteria discussed earlier.
   (b) If either branching criterion is satisfied, branch below the leaf, making it an internal node and making its children into leaf nodes. Create fringe node children below the new leaf nodes.
6. **Stop iterating if $o_{t+1}$ is the goal.** Otherwise, increment the time counter ($t \leftarrow t+1$) and return to step 1.

Table 1: Average Number of Tree Nodes on McCallum's Maze Problem.

| Algorithm | Real Tree | Fringe | Total |
|---|---|---|---|
| USM ($d_f = 1$) | 36.0 | 50.0 | 86.0 |
| USM ($d_f = 2$) | 34.3 | 96.9 | 131.2 |
| GUSM | 57.7 | 52.6 | 110.3 |

Table 2: Average Number of Tree Nodes on Noncontiguous Maze Problem.

| Algorithm | Real Tree | Fringe | Total |
|---|---|---|---|
| USM ($d_f = 2$) | 53.8 | 88.3 | 142.1 |
| GUSM | 80.0 | 48.4 | 128.4 |

# 4 Experimental Results

The GUSM and USM algorithms were compared on McCallum's maze problem (Figure 2) and on the noncontiguous maze problem (Figure 3). USM was tested with fringe depths $d_f$ set to 1 and to 2 (the USM-1 and USM-2 conditions, respectively). McCallum (1995b) reports using USM with several reinforcement learning methods. We have chosen one of these methods, Q-learning (Watkins and Dayan, 1992), with $\beta = .25$ and $\gamma = .33$. Results reported are averages over 20 runs of each algorithm. Each run consists of a series of trial blocks in each of which the agent is randomly assigned to a different starting position in the maze. All possible starting positions are used once in a trial block.

All three algorithms successfully learned McCallum's maze problem. No significant differences were found in learning speed. However, differences were found in the number of tree nodes generated (see Table 1). GUSM produced a larger "real" tree than USM with either fringe depth. However, the results of the USM $d_f = 2$ condition shows that an increase of the USM fringe depth to even one level more than that in GUSM results in more nodes being produced overall (i.e., "real" + fringe nodes).

On the noncontiguous maze problem, GUSM and USM with $d_f = 2$ were equally successful but USM with $d_f = 1$ was almost uniformly unsuccessful, failing on 19 of the 20 runs (see Figure 4). Results in Table 2 show that GUSM produced a larger real tree than the successful USM variant, but produced fewer nodes overall.

The findings provide evidence that GUSM without parameterization can solve problems that the USM algorithm with a given fringe-depth parameter setting is unable to solve. The finding that GUSM produces larger real trees than USM is not surprising, since GUSM includes an additional basis for node splitting and tree expansion (i.e., the negative criterion). However, in practice, GUSM would be expected to produce trees with fewer nodes overall (i.e., including real and fringe nodes) than USM, since many problems require a fringe depth greater than 1 and it is often not known in advance what fringe depth is required for a given

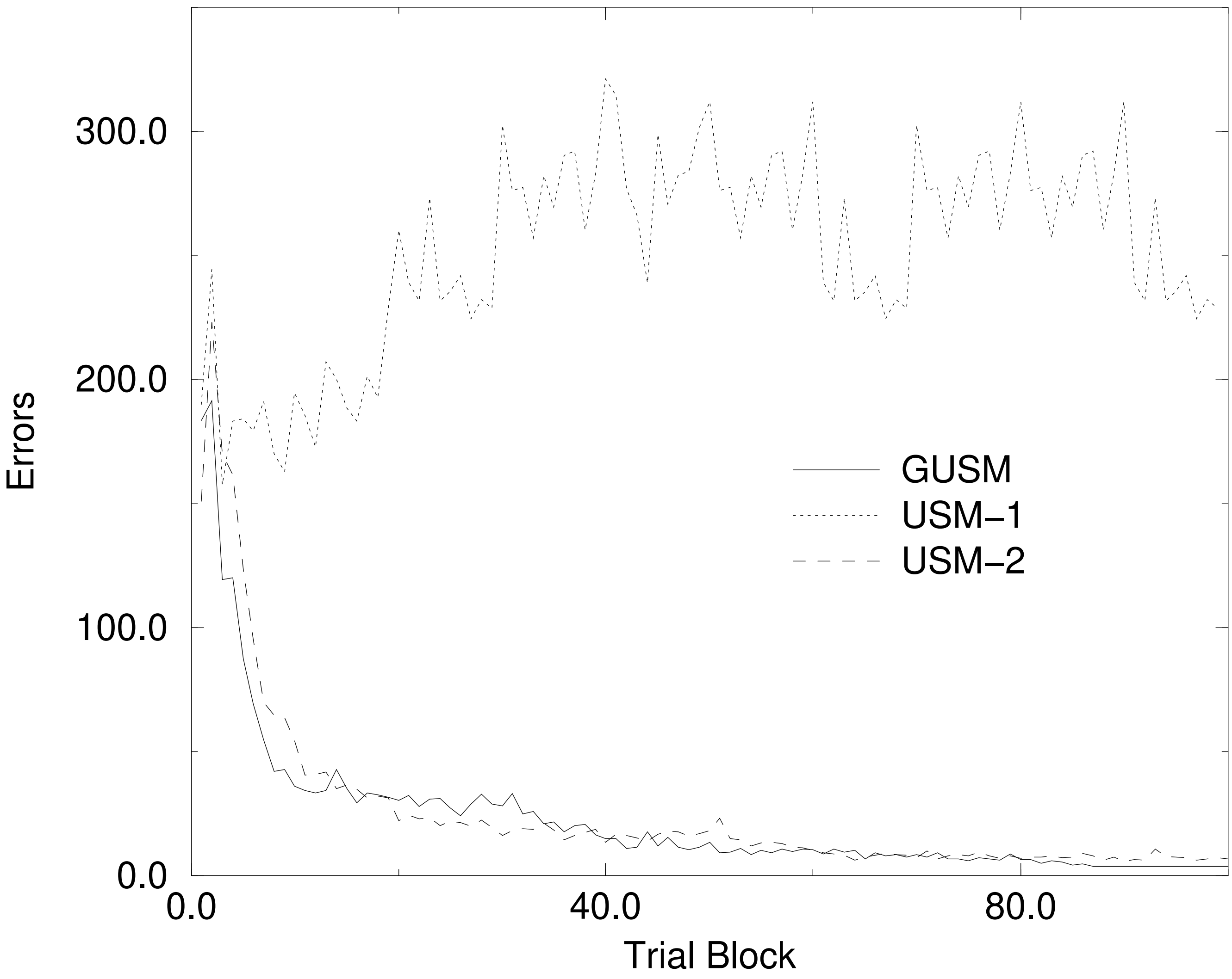


Figure 4: Comparison on noncontiguous maze problem.

problem. In problems with a higher degree of noncontiguity of relevant events in the state definitions, the required fringe depths would result in trees that are much larger than those produced by GUSM. The GUSM algorithm is not confronted with the problem of tradeoffs between lookahead and tree size faced by the USM algorithm.

# 5 Directions for Future Research

One disadvantage of the GUSM algorithm is the relatively large size of the "real" trees produced. We have tested various methods for *postpruning* the trees and found the results to be variable in terms of the effects on reinforcement learning. However, we believe it worthwhile to explore further the question of pruning of PST trees to determine if there are pruning methods or problem domains in which pruning has beneficial effects. The vast literature on pruning of inductive decision trees (Breiman et al, 1984; Quinlan, 1987; 1993) may well shed light on the problems involved in "right-sizing" PST trees for reinforcement learning.

We have referred earlier to the problem of indeterminate states, in which no action is clearly superior. While such states have not posed any difficulties to GUSM on the problems to which it has been applied thus far, we anticipate that indeterminate states may result in overbranching on more complex problems. It would be worthwhile to find a nonparametric heuristic method to limit the tree expansion produced by GUSM's negative split criterion. For instance, one could prevent a path from being extended by the negative criterion to a lower level in the tree than has been reached thus far using the positive criterion.

# 6 Discussion

Greedy Utile Suffix Memory is introduced to build on the contribution of the McCallum's USM algorithm to the problem of reinforcement learning in the context of perceptually-aliased states. Evidence is presented that GUSM solves reinforcement problems with aliased perceptual states as quickly as USM without parameterization. In addition, it succeeds on problems which USM with a specific parameter setting (i.e., fringe depth) cannot solve. While GUSM produces larger "real" trees than USM, the overall size of the trees produced (i.e., including "real" and fringe nodes) is often smaller than that produced by USM.

## Acknowledgements

The author wishes to thank David Aha for the continual advice and encouragement he provided during all stages of this research. He also wishes to thank John Grefenstette, Behzad Kamgar-Parsi, and Ralph Hartley for their comments on earlier drafts of this paper.

### References

Aha, D. W., Kibler, D., & Albert, M. K. (1991). Instance-Based Learning Algorithms. *Machine Learning*, *6*, 37–66.

Breiman, L., Friedman, J. H., Olshen, R. A., & Stone, C. J. (1984). *Classification and regression trees.* Belmont, CA: Wadsworth International Group.

Chrisman, L. (1992). Reinforcement Learning with Perceptual Aliasing: The Perceptual Distinctions Approach. In *Proc. Tenth National Conference on Artificial Intelligence.*

McCallum, R. A. (1992a). First Results with Utile Distinction Memory for Reinforcement Learning. Technical Report 446, University of Rochester Computer Science Department.

McCallum, R. A. (1992b). Using Transitional Proximity for Faster Reinforcement Learning. *The Proceedings of the Ninth International Machine Learning Conference (ML-92)*, Aberdeen, Scotland.

McCallum, R. A. (1993). Overcoming Incomplete Perception with Utile Distinction Memory. *The Proceedings of the Tenth Internation Machine Learning Conference (ML-93)*, Amherst, MA.

McCallum, R. A. (1994a). First Results with Instance-Based State Identification for Reinforcement Learning. Technical Report 502, University of Rochester Computer Science Department.

McCallum, R. A. (1994b). Reduced Training Time for Reinforcement Learning with Hidden State. *The Proceedings of the Eleventh International Machine Learning Conference (ML-94)*, New Brunswick, NJ.

McCallum, R. A. (1995a). Instance-Based State Identification for Reinforcement Learning. *NIPS-95.*

McCallum, R. A. (1995b). Instance-Based Utile Distinctions for Reinforcement Learning. *Proc. of the Twelfth International Machine Learning Conference (ML-95).*

Quinlan, J. R. (1987). Simplifying decision trees. *International Journal of Man-Machine Studies, 27,* 221–234.

Quinlan, J. R. (1993). *C4.5: Programs for machine learning.* San Mateo, CA: Morgan Kaufmann.

Ron, R., Singer, Y., & Tishby, N. (1994). Learning Probablistic Automata with Variable Memory Length. In *Proc. of the Seventh Annual ACM Conference on COLT,* New Brunswick, NJ.

Siegel, S. (1956). *Nonparametric Statistics for the Behavioral Sciences.* McGraw-Hill, New York, NY.

Watkins, J. C. H., & Dayan, P. (1992). Q-learning. *Machine Learning. 8(3).* 279–292.

Whitehead, S., & Ballard, D. H. (1991). Learning to perceive and act by trial and error. *Machine Learning. 7(1)*, 45–83.