# Classifier systems, endogenous fitness, and delayed rewards: A preliminary investigation

**Lashon B. Booker**
The MITRE Corporation
1820 Dolley Madison Blvd
McLean, VA 22102-3481
*booker@mitre.org*

## Abstract

Previous work has shown the potential advantages of using endogenous fitness schemes in classifier systems. The basic idea behind endogenous fitness is to reinforce successful system performance with "resources" that rules need in order to reproduce. Instead of storing explicit quantitative estimates of performance, each rule has one or more reservoirs that are used to store resources. When enough resources have been accumulated, a rule utilizes some of its resources to reproduce and the reservoir level is reduced accordingly. This paper extends this concept to accommodate environments having delayed rewards. Reinforcement learning techniques for solving average-reward Markovian decision processes are combined with a simple endogenous fitness scheme in a classifier system. We describe initial tests of this approach on state-space search problems used in previous classifier system studies.

## 1 Introduction

Classifier system implementations have traditionally used explicit measures of utility — such as predicted payoff, accuracy, payoff-derived strength, etc. — to quantify the utility and fitness of classifier rules. This research is investigating classifier systems that determine the utility and fitness of rules endogenously, without computing explicit estimates.

The basic idea behind endogenous fitness schemes is straightforward. The rules advocating an action on a given time step (i.e., the action set) take all of the credit for whatever reinforcement is received. Each reinforcement event leads to the distribution of some nominal resources among those rules, and the acquired resources accumulate over time in internal reservoirs. Following the way resources are used in Echo [6], the endogenous fitness scheme allows rules to reproduce only when they have accumulated resources in excess of some threshold amount. When enough resources have been accumulated, a rule utilizes some of its resources to reproduce and the reservoir level is reduced accordingly. A distinction is made between reinforcement events that are better than "average" and events that are worse than average. The cumulative difference in resources a rule has received for these two outcomes is used to determine eligibility for reproduction. The relative amount of resources received for these two outcomes is the basis for a prediction about the outcome (i.e., reinforcement better or worse than "average") expected whenever a rule is active.

Previous work [4] described one implementation of this idea. That classifier system has a fairly conventional design, borrowing elements from previous work on GOFER [2, 3] and XCS [11]. The population of classifiers has a fixed size $\mathcal{N}$, each classifier having a single condition on the left side and a single action on the right side. Each classifier $\xi$ has two associated reservoirs: the $\Delta_+(\xi)$ reservoir that stores resources obtained from "better than average" reinforcement events, and the $\Delta_-(\xi)$ reservoir that stores resources obtained from "worse than average" reinforcement events. The only other parameters stored with each classifier are: age $\alpha(\xi)$, which is used in the procedure for deleting classifiers; an estimate $\pi(\xi)$ of the average reward available when $\xi$ is included in the match set; and, a counter $\nu(\xi)$ that records the number of times $\xi$ has been included in the match set on an "explore" trial. Classifiers are eligible to reproduce when the difference $|\Delta_+(\xi) - \Delta_-(\xi)|$ is larger than some threshold. Empirical performance of the endogenous fitness scheme implemented in this way were encouraging. The system performs as well as utility-

based classifier systems such as XCS [11] on the multiplexor problem.

One of the important research issues not addressed by this previous work on endogenous fitness in classifier systems is how to solve multi-step reinforcement learning problems involving sequences of actions and delayed rewards. In this paper we describe work in progress that is extending the endogenous fitness scheme to handle such problems.

We begin with a brief discussion of our latest approach to implementing endogenous fitness in classifier systems. That discussion is followed by a description of work in progress that integrates average-reward reinforcement learning techniques into the endogenous fitness paradigm.

## 2 Implementing Endogenous Fitness In Classifier Systems

In the current implementation there are also no explicit, individual performance estimates associated with classifiers. Each classifier $\xi$ has the two associated reservoirs $\Delta_+(\xi)$ and $\Delta_-(\xi)$. The reservoirs are initialized to be empty and the initial classifiers are generated at random. The following additional parameters are stored with each classifier: an estimate $\pi(\xi)$ of the average reward available when $\xi$ is included in the match set; estimates $\delta_+(\xi)$ and $\delta_-(\xi)$ of the average reward available when the best and worst actions in $\mathbf{M}$ are selected (as determined by the performance system); a counter $\alpha(\xi)$ that records the number of times $\xi$ has been included in the action set; and, an estimate $\omega(\xi)$ of the proportion of times the reward available exceeds $\pi(\xi)$ when $\xi$ is in the action set. These parameters are used to help characterize the flow of resources in a match set. Here we provide a brief summary of the key details, focusing primarily on those that differ significantly from the description given in [4].

### 2.1 Performance System

The system performance cycle is fairly routine. For each input message $i$, the system first determines the set of classifiers $\mathbf{M}$ eligible to classify the message. Matching classifiers are always included in $\mathbf{M}$. Following the procedures in GOFER, if there are fewer than $\mathcal{N}_m$ matching classifiers available, classifiers with the highest partial match scores are deterministically selected to fill out $\mathbf{M}$. We use the simple partial match score

$$\mu(\xi, i) = \begin{cases} s + l & \text{if } \xi \text{ matches the message } i \\ l - n & \text{otherwise} \end{cases}$$

where $l$ is the length of the input condition in $\xi$, $s$ is the specificity, and $n$ is the number of positions where the condition doesn't match the message.

For each action $a$ represented in the match set $\mathbf{M}$, the system computes an *action mandate* that captures the system's knowledge about the likelihood of a "better than average" outcome if action $a$ is chosen. Each classifier $\xi$ in $\mathbf{M}$ computes the value

$$\lambda(\xi) = 2 \mid \omega(\xi) - 0.5 \mid$$

as the mandate for its action. Note that $\lambda(\xi)$ is 1 whenever the outcome associated with $\xi$ is consistently better or worse than average ($\omega(\xi) = 0$ or 1) and 0 when the outcome is random ($\omega(\xi) = 0.5$). When $\omega(\xi) \geq 0.5$, this contribution from each rule is added to an *action selection array*. When $\omega(\xi) < 0.5$, this contribution from each rule is subtracted. The rationale for this approach is to give a higher net weight to those actions that, based on previous experience, have the highest likelihood of being followed by a "better than average" outcome.

As in XCS, the information in the action selection array is used to determine which action is selected. The members of $\mathbf{M}$ that agree with the selected action constitute the *action set* $\mathbf{A}$. The system then sends that action to the effectors, and the environment may respond with reinforcement.

### 2.2 Reinforcement

On every time step, parameters of the classifiers in $\mathbf{M}$ are adjusted and some amount of resource $\mathbf{R} > 0$ is made available to the classifiers in $\mathbf{A}$. Competition for this resource is the primary mode of interaction among the rules in the population. The following sequence of steps is used to determine how the resource is distributed:

- The $\pi(\xi)$ parameter is revised for all classifiers in $\mathbf{M}$ using the simple update rule

  $$\pi_t(\xi) = \begin{cases} \frac{(\pi_{t-1}(\xi)\nu_{t-1}(\xi)) + \mathcal{R}}{\nu_t(\xi)} & \text{if } \nu_t(\xi) \neq \nu_{t-1}(\xi) \\ \pi_{t-1}(\xi) & \text{otherwise} \end{cases}$$

  where $\mathcal{R}$ is the reward received, $\nu(\xi)$ is a counter that records the number of times $\xi$ has been included in the match set, $\pi_0(\xi) = 0$, and $\nu_0(\xi) = 0$. If the action $a$ is the best (or worst) option available, then a similar update is made to $\delta_+(\xi)$ (or $\delta_-(\xi)$).

- The members of $\mathbf{M}$ collectively estimate the average reward $\Pi$ for the current state as the central

tendency of the values $\pi(\xi)$ in $\mathbf{M}$. Since $\mathbf{M}$ will often include overly general rules with inaccurate values, it is helpful to take some steps to avoid having this estimate contaminated. Order statistics can provide a robust estimate of the central tendency. We use a conservative boxplot criterion [8] to identify outlying values and exclude them from the computation. The boxplot criterion computes the median $\tilde{x}$ of the data values, the lower quartile $q_1$, and the upper quartile $q_3$. Any value that lies $3(q_3 - q_1)$ above the upper quartile or below the lower quartile is labeled as an outlier. The trimean estimator [1], given by

$$\hat{x} = \frac{q_1 + 2\tilde{x} + q_3}{4}$$

is used to obtain a simple and reasonably robust estimate of the central tendency $\Pi$. While there are many other ways to compute the central tendency that give adequate results, the methods using order statistics have given the best results so far.

In an analogous manner, there is a collective determination of the central tendencies $\hat{\delta}_+$ and $\hat{\delta}_-$ of the parameters $\delta_+(\xi)$ and $\delta_-(\xi)$ respectively.

- The resource $\mathbf{R}$ available on each time step is scaled to reflect the size of the reward $\mathcal{R}$ relative to what is expected in $\mathbf{M}$. It is sufficient to use a simple linear scaling given by

$$\mathbf{R} = \bar{\mathbf{R}} \left( 1.0 + \frac{\mathcal{R} - \hat{\delta}_-}{\hat{\delta}_+ - \hat{\delta}_-} \right)$$

where $\bar{\mathbf{R}}$ is a system parameter indicating the minimum amount of resource made available on each time step. Given two classifiers that are consistently associated with above average rewards, this procedure gives a modest selective advantage to the classifier that is best from a payoff standpoint.

- Each classifier in $\mathbf{A}$ receives a share of the resource given by

$$\rho(\xi) = \left( \frac{\lambda(\xi)\mathbf{H}(\xi)}{\sum_{\xi \in \mathbf{A}} \lambda(\xi)\mathbf{H}(\xi)} \right) \mathbf{R}$$

where $\mathbf{H}(\xi)$ is a hypergeometric probability that helps bias the distribution of resources to favor sets of rules that efficiently cover all input messages. This computation is strongly related to the familiar fitness-sharing schemes used in GA implementations to solve multimodal optimization problems. See [4] for more details. When $\mathcal{R} \geq \Pi$, $\rho(\xi)$ is added to $\Delta_+(\xi)$; otherwise, it is added to $\Delta_-(\xi)$.

Under this regime, rules that are consistently associated with only one type of outcome will quickly achieve a large net accumulation of resources in one of the reservoirs since all of their resources are stored in one place. Conversely, rules associated with both outcomes will distribute their resources over both reservoirs, taking longer to attain any large net accumulation. This is significant because the frequency of reproduction is tied to the net accumulation of resources in the one of the reservoirs.

### 2.3 Rule Discovery

After the rule reservoirs have been updated, any classifier in $\mathbf{M}$ having a sufficient net excess of resources in its reservoirs becomes eligible to reproduce. An excess of resources is indicated by

$$|\Delta_+(\xi) - \Delta_-(\xi)| > \tau$$

for some threshold $\tau$.

If there is more than one classifier in $\mathbf{M}$ eligible to reproduce on a given cycle, all eligible classifiers are designated as parents and allowed to produce one copy of themselves. Parents then have the reservoir containing the excess decremented by $\tau$, which can be viewed as the cost of generating an offspring. The reproduced copies are modified by mutation and crossover, and the resulting offspring are inserted into the population. Classifiers are stochastically selected for deletion based on $\alpha(\xi)$, so that general classifiers are more likely to be chosen. This is the simplest kind of deletion technique used in Echo-like systems. Future research will investigate the potential advantages of charging each rule a "maintenance cost" every time it is active, changing the resource flow to allow parents to share resources with their offspring, and deleting rules with empty (or nearly empty) reservoirs.

Note that rules consistently associated with above average (or below average) outcomes will consistently enjoy a reproductive advantage over their competitors. In combination with a deletion technique biased against general classifiers, this exerts considerable selective pressure against overly general rules.

### 2.4 Initial Tests

Figure 1 and Figure 2 show the performance of this revised classifier system on the 11-bit and 20-bit multiplexor problems. Results are averaged over 10 runs.

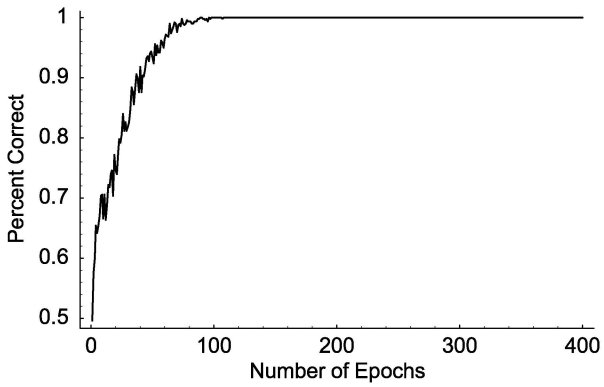


Figure 1: Performance on 11-bit multiplexor

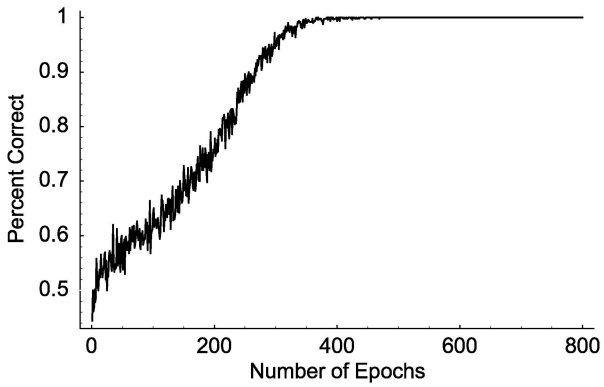


Figure 2: Performance on 20-bit multiplexor

Performance was measured by the proportion of correct decisions over a learning epoch consisting of 50 randomly generated input strings. Each 11-bit experiment was run for 400 epochs (20,000 input strings) and each 20-bit experiment was run for 800 epochs (40,000 input strings). The reward scheme pays $+1$ for correct responses and $-1$ for incorrect responses. The action-selection regime was based on the one used by Wilson [11]. This regime makes a random (probability 0.5) choice between "exploit" mode — where the system selects the best action as indicated by the action selection array — and "explore" mode where the system selects an action at random. During explore trials, a correct decision is recorded whenever the system would have made the correct "exploit" decision. The system parameters[1] used were: $\mathcal{N}_m = 16$, $\mathbf{R} = 2000$, $\tau = 500$, initial reservoir levels of 0 for new offspring, a mutation rate of $1/(3\ell)$, and a crossover rate of 1.0. The 11-bit experiments used $\mathcal{N} = 400$ while the 20-bit experiments used$\mathcal{N} = 800$.

Note that the 11-bit problem was solved after about 100 epochs (5,000 inputs) and the 20-bit problem was solved after about 500 epochs (25,000 inputs). This is roughly half the time reported previously for the XCS system on these problems [12]. Though it is difficult to draw any definitive conclusions on the basis of these results, it is clear that in these problems the endogenous fitness scheme does an effective job of discovering accurate rules. Work in progress is studying how this approach scales to larger multiplexor problems.

## 3 Accommodating Delayed Rewards

One of the important research issues not addressed by previous work on endogenous fitness in classifier systems is how to solve multi-step reinforcement learning problems involving delayed rewards. This section briefly describes an initial approach to extending the endogenous fitness scheme to handle such problems.

Classifier systems traditionally solve problems involving delayed rewards by using the bucket brigade algorithm [7] or some other algorithm from the reinforcement learning literature [9]. These algorithms all compute and manipulate explicit estimates of the reward expected when a specific action is taken in a given state. Since the endogenous fitness scheme only computes explicit estimates of the average reward expected in a state, the most natural starting point for our investigation is to consider average-payoff reinforcement learning algorithms [10].

[1] Classifier input conditions were initialized so that each possible symbol in $\{1, 0, \#\}$ was equally likely to occur.

A typical updating scheme for average-payoff reinforcement learning is given by:

$$Q_{t+1}(x_t, a_t) = (1 - \beta(x_t, a_t))Q_t(x_t, a_t) + \beta(x_t, a_t)[\mathcal{R}(x_t, a_t) - r_t + \max_{a' \in \mathbf{A}_{t+1}} Q_t(x_t, a')]$$

where $x_t$ is the state, $a_t$ is the action taken, $Q_t(x_t, a_t)$ is the payoff expected when taking action $a_t$ in state $x_t$, and $r_t$ is the sample average of the payoffs received for greedy actions. Note that a discounting factor is not needed to assure that the updated values remain bounded, since anchoring the computation to $r_t$ accomplishes that.

In order to use this approach in a classifier system, we must identify something that plays the role of $Q_t(x_t, a_t)$. The endogenous fitness scheme used here only maintains explicit reward estimates associated with the match set $\mathbf{M}$, so there is no explicit information available about the payoff of an arbitrary state–action pair. However, since the resource flow experienced by a classifier is correlated with the size of the reward expected when that classifier belongs to $\mathbf{A}$, it is reasonable to consider modifying the resource flow as an alternative to updating an explicit parameter. Moreover, the parameter $\hat{\delta}_+$ provides explicit payoff information about one very important state–action pair: the one associated with the best action in $\mathbf{M}$. Consequently, the following heuristic counterpart to the average-reward reinforcement learning update is used in the endogenous fitness computation: at time $t$, the value $\hat{\delta}_+$ is passed back to the classifiers in $\mathbf{M}_{t-1}$ (i.e., it is added to whatever external reward was received by that match set). The computations in $\mathbf{M}_{t-1}$ then proceed as usual using the augmented reward in place of the external one.

Grefenstette's state space search problem [5] was used to test of how well this average-payoff version of the classifier system can discover action sequences leading to external reward. The state space contains 288 states arranged in a $9 \times 32$ rectangular grid. The first row contains the 32 initial states where all searches begin. Three transitions are possible from any one state to some neighboring state. If we identify each state using a row index $i$, $0 \leq i < 9$, and a column index $j$, $0 \leq j < 32$, then the states accessible from state $(i, j)$ are the states

$$(i+1, j-1 \bmod 32) \quad (i+1, j) \quad (i+1, j+1 \bmod 32)$$

The last row in the grid contains the 32 final states, each of which is associated with a fixed reward. Rewards range from 0 to 1000 and are distributed as shown in 1.

| Reward | Column Index of Final State |
|---|---|
| 0 | 0,1,14,15,16,17,30,31 |
| 50 | 2,13,18,29 |
| 75 | 3,12,19,28 |
| 125 | 4,11,20,27 |
| 250 | 5,10,21,26 |
| 500 | 6,9,22,25 |
| 1000 | 7,8,23,24 |

Table 1: Distribution of rewards in the state space problem.

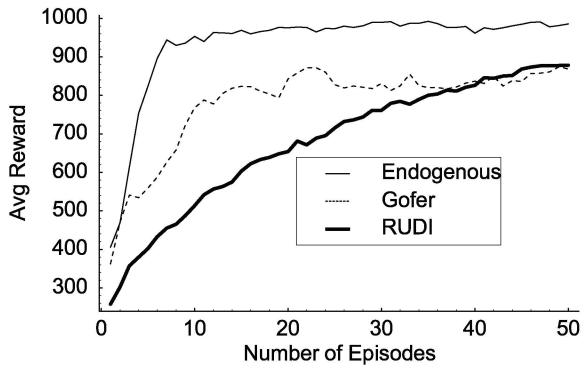


Figure 3: Performance on state space problem (compared to Gofer and RUDI)

The challenge in this problem is to learn a sequence of state transitions from each initial state that maximizes the reward obtained at the end of the sequence. It is a difficult learning problem because, from some initial states, the early moves determine whether or not it is even possible to achieve the maximum reward. Effective credit assignment is therefore a pivotal issue. Another difficulty is that there is a "hamming cliff" in the binary representation of the final states associated with the optimum reward (between columns 7 and 8, and columns 23 and 24). This complicates the categorization task faced by the genetic algorithm.

The revised classifier system was tested on this problem in an experiment involving 50 learning episodes, each consisting of 1000 traversals of the state space. The system parameters that differed from the multiplexor experiments were $\mathcal{N} = 2000$, $\mathcal{N}_m = 24$, $\mathbf{R} = 500$, and $\tau = 2500$. The action-selection regime differed slightly from the one used for the multiplexor problems. For each traversal of the state space, the system first makes a random choice between an "exploit" traversal in which the best action is taken on every step, or an "explore" traversal in which action selection is controlled by the mulitplexor action-selection

regime. The results are summarized in Figure 3, which compares the performance with previous results on this problem reported for Grefenstette's [5] system RUDI and Booker's [3] system GOFER. The revised classifier system quickly achieves good performance on this task and steadily improves toward optimum (1000 reward level) performance, clearly outperforming both RUDI and GOFER.

## 4 Conclusions

While these results are preliminary, they do show that an endogenous fitness scheme is compatible with reinforcement learning algorithms for problems involving delayed rewards. This makes endogenous fitness a more suitable alternative for implementing classifier systems to solve interesting problems. Moreover, the performance of the endogenous fitness approach is comparable to that obtained by systems like XCS and shows the potential to do even better. Current research efforts are conducting more experiments with this enhanced version of the endogenous fitness scheme in order to better assess its strengths and weaknesses. It is clear that this approach is promising enough to warrant further investigation.


### † Acknowledgments

This research was funded by the MITRE Sponsored Research (MSR) program. That support is gratefully acknowledged.


## References


[1] Vic Barnett and Toby Lewis. *Outliers in Statistical Data*, Third edition. John Wiley and Sons, Chichester UK, 1994.

[2] Lashon B. Booker. Classifier systems that learn internal world models. *Machine Learning*, 3:161–192, 1988.

[3] Lashon B. Booker. Triggered rule discovery in classifier systems. In J. David Schaffer, editor, *Proceedings of the Third International Conference on Genetic Algorithms (ICGA89)*, pages 265–274, Fairfax, VA, 1989. Morgan Kaufmann.

[4] Lashon B. Booker. Do We Really Need to Estimate Rule Utilities in Classifier Systems? In Pier Luca Lanzi, Wolfgang Stolzmann, and Stewart W. Wilson, editors, *Learning Classifier Systems: From Foundations to Applications*, volume 1813 of *LNAI*, pages 125–142, Berlin, 2000. Springer-Verlag.

[5] John J. Grefenstette. Credit assignment in rule discovery systems based on genetic algorithms. *Machine Learning*, 3:225 245, 1988.

[6] John H. Holland. Echoing emergence: Objectives, rough definitions, and speculations for Echo-class models. In G. Cowan, D. Pines, and D. Melzner, editors, *Complexity: Metaphors, Models, and Reality*, volume XIX of *Santa Fe Institute Studies in the Sciences of Complexity*, pages 309–342. Addison-Wesley, Reading, MA, 1994.

[7] John H. Holland, Keith J. Holyoak, Richard E. Nisbett, and P. R. Thagard. *Induction: Processes of Inference, Learning, and Discovery*. MIT Press, Cambridge, 1986.

[8] Boris Iglewicz and David. C. Hoaglin. *How to Detect and Handle Outliers*, volume 16 of *American Society for Quality Control Basic References in Quality Control: Statistical Techniques*. ASQC Quality Press, Milwaukee WI, 1993.

[9] Leslie Pack Kaelbling, Michael L. Littman, and Andrew W. Moore. Reinforcement learning: A survey. *Journal of Artificial Intelligence Research*, 4:237–285, 1996.

[10] Satinder P. Singh. Reinforcement learning algorithms for average-payoff Markovian decision processes. In *Proceedings of the Twelfth National Conference on Artificial Intelligence*, pages 700 706, Seattle, WA, 1994. The AAAI Press.

[11] Stewart W. Wilson. Classifier fitness based on accuracy. *Evolutionary Computation*, 3(2):149–175, 1995.

[12] Stewart W. Wilson. State of XCS Classifier System Research. In Pier Luca Lanzi, Wolfgang Stolzmann, and Stewart W. Wilson, editors, *Learning Classifier Systems: From Foundations to Applications*, volume 1813 of *LNAI*, pages 63–81, Berlin, 2000. Springer-Verlag.