An Autonomous Explore/Exploit Strategy

Alex McMahon
University of Reading
Berkshire
RG6 6AY, UK
+ 44 (0)118 378-5123
siu01ajm@rdg.ac.uk

Dan Scott
University of Reading
Berkshire
RG6 6AY, UK
+ 44 (0)118 966 6893
siu01ds@rdg.ac.uk

Dr Will Browne
University of Reading
Berkshire
RG6 6AY, UK
+ 44 (0)118 378 6705
w.n.browne@rdg.ac.uk

ABSTRACT

In reinforcement learning problems it has been considered that neither exploitation nor exploration can be pursued exclusively without failing at the task. The optimal balance between exploring and exploiting changes as the training progresses due to the increasing amount of learnt knowledge. This shift in balance is not known a priori so an autonomous online adjustment is sought. Human beings manage this balance through logic and explorations based on feedback from the environment. The XCS learning classifier system uses a fixed explore/exploit balance, but does keep multiple statistics about its performance and interaction in an environment. Utilising these statistics in a non-linear manner, autonomous adjustment of the explore/exploit balance was achieved. This resulted in reduced exploration in simple environments, which could increase with the complexity of the problem domain. It also prevented unsuccessful 'loop' exploit trials and suggests a method of dynamic choice in goal setting.

Categories and Subject Descriptors

F.2.2 Nonnumerical Algorithms and Problems

General Terms

Algorithms, Design.

Keywords

Learning Classifier Systems, Genetics-Based Machine Learning, Explore/Exploit Strategy.

1. INTRODUCTION

"The dilemma is that neither exploration nor exploitation can be pursued exclusively without failing at the task." Sutton and Barto [1]. The optimal balance between exploring and exploiting changes as the training progresses due to the increasing amount of learnt knowledge and is not known a priori. Many reinforcement learning algorithms, such as XCS [2], use a fixed explore/exploit ratio that may lead to unnecessary exploration steps, repetitive loops in exploit trials and an increased time for training. Additionally, Whitley [3] identifies that repetition (resampling) is important to avoid in learning algorithms, which is to be considered in this work.

Gecco'05, June 25-29, 2005, Washington, DC, USA.

1.1. Aims and objectives

The aim of this project is to investigate whether the statistics held by an LCS can be used to create an autonomous explore/exploit strategy. Both Markov decision processes (MDP) and partially observable Markov decision processes (POMDP) will be tested to observe the effect of the strategy [4].

2. TYPES OF ENVIRONMENT

2.1. Single/multi step

In a single step environment the choice of action from any given state has no bearing on any future state of the environment, with the result of the action being an instantaneous reward level. If the choice of action affects the future states of the environment, with reward being (potentially) received after multiple steps then the environment is considered multi-step. The predominant multi-step environment used in Learning Classifier System (LCS) research is the 'Woods' environment (e.g. Figure 1).

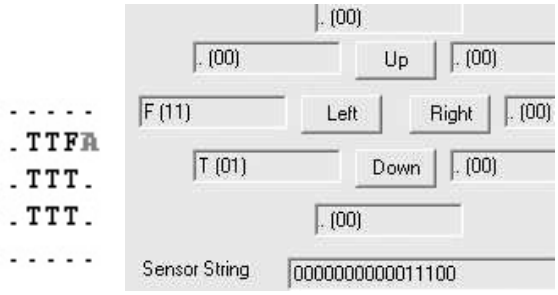

Figure 1. Woods 1.

The Woods environments are cell based environments where each cell contains one of several objects which may have a reward associated with it; the AI agent (or

animat) must navigate this environment to find the reward. In the most common woods environment there are 2 types of objects: T (tree) is an impassable object with zero reward, F (food) bestows a positive reward. Empty cells are indicated by a full stop and have zero associated reward. The animat has the ability to sense the objects in the surrounding 8 cells.

2.2. Markovian/non-Markovian

Within multi-step environments there is a distinct division between two types of environment, "If an environment has the Markov property, then its one-step dynamics enable us to predict the next state and expected next reward given the current state and action." [1].

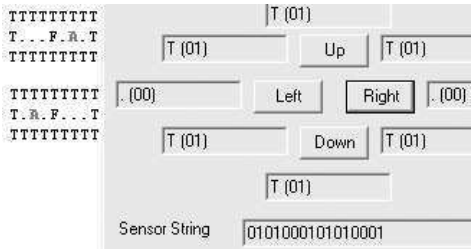

Figure 2. Woods 100, 2 positions with identical sensor strings

This is perhaps best illustrated with an example; in Woods1 (see Figure 1) each possible position of the animat results in a unique sensor string (note that the environment wraps around on each side) and so after learning the animat can tell instantly which cell it is in. However in Woods 100 (see Figure 2), there are two positions that give identical sensor readings, but with different subsequent states (and eventual rewards), this is therefore a POMDP environment.

2.3. Static/dynamic

In the standard woods environment nothing changes over time, to test more advanced aspects of AI the environment can be made to be dynamic, for example the objects could move over time, or certain properties of objects could change. There is also the possibility of having a Multi-Agent System (MAS) which would introduce the possibilities of competition and collaboration. Initially the system will be tested in Static environments only; although it is hypothesised that dynamic environments would require the simulation of more complex exploration.

3. AI ARCHITECTURES

3.1. Learning Classifier Systems

Learning Classifier Systems are rule-based evolutionary learning systems. The fact that they are rule-based gives them an advantage over some learning systems in that they have a high level of knowledge transparency, as the rules governing the LCS can be interpreted relatively easily. The original LCS [5] has been adapted and improved by various researchers, of key importance is XCS [2], which (amongst other things) modifies Holland's LCS to use an accuracy-based fitness approach. There has also been relevant further work extending LCS to anticipate the effect of its actions, in the shape of Anticipatory Classifier Systems (ACS) [6]. Wilson outlined both global and local explore/exploit strategies and concluded much further work is needed as they are crucial for true autonomy [7].

3.2. Hybrid Explorer Classifier System

The proposed system is a combination of a standard implementation of XCS [8] and the novel 'explorer' explore/exploit strategy. It supports the principle behind reinforcement learning in that an intelligent agent should learn from the interaction with its environment, and not from an explicit teacher. This leads to a Hybrid Explorer Classifier Systems (HECS).

4. EXPLORER ARCHITECTURE

The proposed explorer architecture for HECS bases the exploration level of the animat on three factors, whilst storing the explorer level as two separate real variables ranging from -1 to 1 (with 1 being fully satisfied). The equations proposed introduce several nonlinearities to the system, see figure 3, the precedence for the use of non-linear equations in LCS can be seen in their use for several other calculations, such as accuracy calculation, particularly where a sharp discrimination is required between similar inputs.

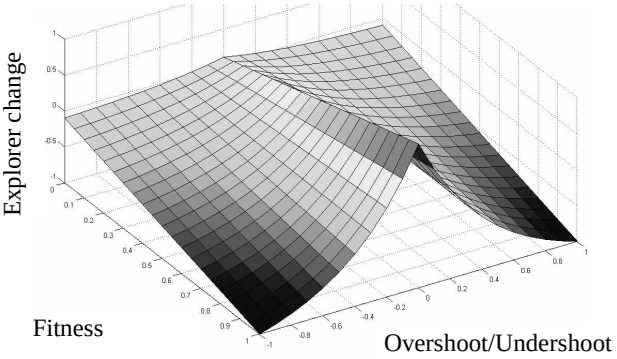

Figure 3. Graph of explorer change based on prediction fitness and overshoot/undershoot

4.1. Accuracy induced exploration

The proposed explorer framework takes into account the accuracy of the predictions made by the animat, this calculation is based on the fitness of the prediction (how confident the animat is in the accuracy of its prediction) and the amount of overshoot/undershoot of the prediction (relative to maximum reward/punishment):

$$\Delta E_A = f(F) \times \left[g(M_{PA}) \times \left(e^{-|O_s| \times M_{PA}} - 1 \right) + 1 \right] \quad (1)$$

The function $f(F)$ is dependent on the fitness of the prediction (F) and a parameter that limits the maximum change in exploration (0-1).

$$f(F) = \Delta E_{Max} \times F \qquad (2)$$

M_{PA} is a parameter which defines how close to 'perfect accuracy' a prediction has to be for a positive explorer increase (affects the gradient of the curve).

$$g(M_{PA}) = \frac{2}{1 - e^{M_{PA}}} \qquad (3)$$

O_S is the scaled overshoot

$$O_s = \frac{O}{R_{range}} \qquad (4)$$

Where:

$\Delta E_{\max}$ is the parameter which limits the maximum change in exploration (0-1)

F is the Fitness of prediction

O is the Reward Overshoot, (positive value indicates reward was greater than prediction)

R_{range} is the animat's calculation of the range of rewards (max reward-max punishment)

M_{PA} is a parameter that defines the tolerance to perfect accuracy for a prediction to the reward obtained.

4.2. Reward induced exploration

When the animat receives immediate positive reward an explorer level is increased by an amount related to the amount of reward received relative to the maximum reward available, if the animat receives negative reward (or punishment) then the explorer level is decreased by an amount relative to the maximum punishment. The equation relating immediate reward to explorer change is, see figure 4:

$$\Delta E_R = h(M_{RS}) \times sign(R_S) \times \left(e^{|R_S| \times M_{RS}} - 1 \right) \qquad (5)$$

$$h(M_{RS}) = \Delta E_{\max} \times \frac{1}{e^{M_{Rs}} - 1} \qquad (6)$$

Where:

R_S is the scaled reward/punishment

M_{RS} is a scaling factor which affects the slope of the curve

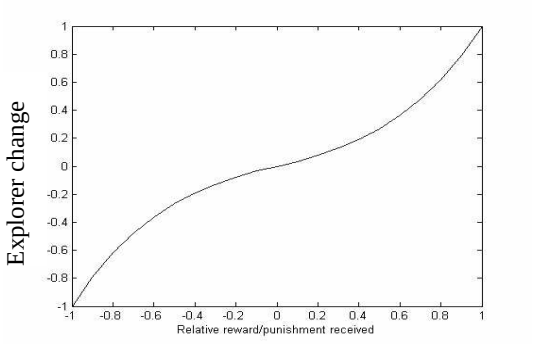

Figure 4. Graph of explorer change against immediate reward

4.3. Effect of time on explorations

The effect of reward on the animat's explorer needs is made to decay over time so that as time proceeds the animat becomes less satisfied with the reward it received in the past, the opposite occurs with negative reward so that the animat begins to forget the negative reward it received in the past. For positive rewards this decay is delayed until the animat has made more steps than it has estimated as being the maximum amount required to reach a reward (n_{max}) based on the minimum and maximum predictions in the population, and the discount factor (γ) (see figure 5):

$$n_{\max} = \log_{\gamma}\left(\frac{P_{\min}}{P_{\max}}\right) \qquad (7)$$

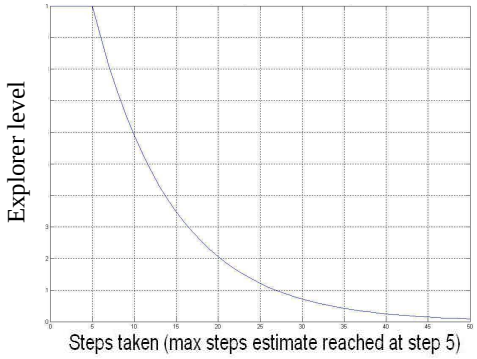

Figure 5. Explorer level decay as steps taken passes max steps estimate

4.4. Effect of explorations

The two explorer levels are used for two purposes: Firstly at the start of every trial the animat chooses whether to explore or exploit during this trial. This choice is based on the accuracy explorer level; if the animat is not satisfied with its accuracy it should explore more so as to improve its knowledge (and therefore accuracy), if however the animat has been largely accurate then it can assume that the environment has been learnt to a sufficient level and therefore the best choice of action is to maximise the immediate reward by exploiting this knowledge.

$$P_{\text{explr}} = \left(0.5 - \frac{0.9 \times E_A}{2} \right) \quad (8)$$

Note in eqn. (8) the factor of 0.9 is an arbitrary value between 0 and 1 which means there is always a chance of both exploration and exploitation (see figure 6).

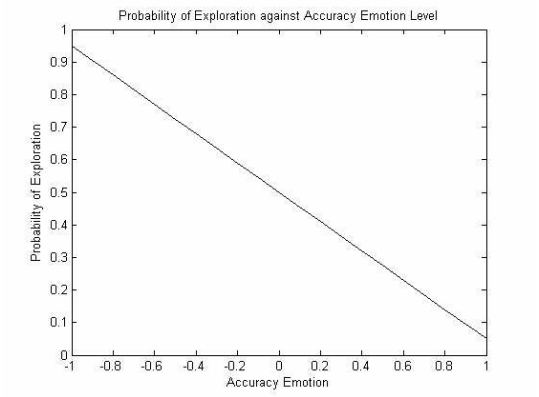

Figure 6. Graph showing the probability of the animat exploring at any given trial

Secondly the animat uses the reward induced explorer level to 'escape' from unsuccessful exploit trials; at the start of every exploit step the animat has the possibility of switching to explore mode if its explorer level is too close to zero; as the explorer level does not begin to decay until the estimated maximum number of steps have been taken there is little chance of switching until several exploit steps have failed to find a reward.

$$P_{\text{SwitchToExplr}} = 1 - \left| E_R \right| \quad (9)$$

5. RESULTS

5.1. Woods1 – XCS

The implementation of XCS in HECS was verified as it correctly reproduced known results in the Woods1 environment [8]. HECS successfully learnt the best actions for each of the available positions (see figure 7)

```
46321
4TTF4
4TTT7    789
7TTT8    4 6
13331    123
```

Figure 7. Woods1 - Best Actions (and key)

Once the soundness of the basic XCS was established, the explorer side of the architecture was enabled and the results compared. Each experiment was run 10 times with the average values shown in table 1. The deviation between each run was not significant.

With the explorer enabled the animat remained largely in explore mode for a period at the start, until it was satisfied with its accuracy, at which point it switched mainly to the exploit mode with occasional exploration trials. This suggests that the explore/exploit strategy enabled the animat to decide when to exploit effectively.

Table 1. Woods1, effect of the explorer strategy on performance

	XCS	HECS
Average Steps from start of training	2.792	2.727
Average Steps last 100 trials (optimum 1.6875)	1.708	1.725
Ave. No. Of Explores	2000	138
Actual Time taken (s)	402	65

Over the complete training cycle, 2000 exploit trials, HECS took fewer steps, which may be important during online learning where resources need to be conserved. The average steps taken to reward at the end of training was close to the optimum value, which was confirmed as the optimal policy was present in both systems.

The number of explore trials was significantly reduced by the novel explore/exploit strategy. This resulted in a significant timesaving for the overall algorithm.

The graph for macro population size, see Figure 8, shows that with explorations on HECS is still able to generalize, albeit less effectively than XCS. The graph is somewhat misleading in terms of the x axis, as HECS uses much less exploration trials and achieves a level of generalisation very close to that achieved with XCS after a greater number of exploration trials.

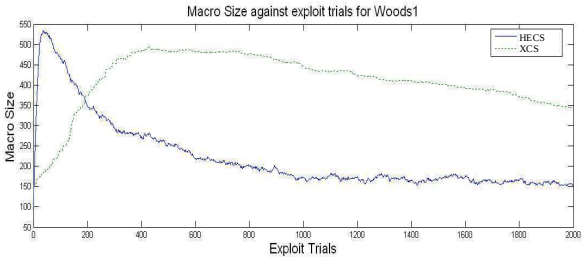

Figure 8. Woods1 - graph showing effect of explorations

5.2. Maze 4

```
TTTTTTTT
T..T..FT
TT..T..T
TT.T..TT
T......T
TT.T.A.T
T....T.T
TTTTTTTT
```

Figure 9. Maze4

'Maze4' see figure 9, is also a static Markovian environment, but is a significantly larger environment than Woods1. This domain was used to test whether the

explore/exploit benefits of HECS scaled to larger environments.

HECS managed to greatly improve its performance by using explorer based exploration as can be seen in the results, table 2, where both the number of explore trials and actual time taken were significantly reduced. The 'average test steps' is approximately the same. This environment is more complex than Woods1 due to the increase in average steps required to reward and less generalisation being possible. HECS was also able to generalise to a similar degree as XCS.

Table 2. Results for Maze4

	XCS	HECS
Average Steps from start of training	8.18	4.74
Ave. No. Of Explores to optimum. Max explorer change = 1.0	2000	208
Actual Time taken (s)	1765	225
Ave. No. Of Explores to optimum. Max explorer change = 0.5	2000	3852

Exploration could be increased in HECS by reducing the effect of the explorer, which shows that the explore/exploit strategy can be adapted to an environment. Despite this greater exploration, the time taken for the experiment was only slightly longer than XCS due to the explorer preventing the animat becoming stuck in long exploit trials, which occurred 147 times in XCS.

It is noted that similar levels of generalization were achieved partly due to the low levels of generalisation possible, see figure 10. This figure also shows with the explorer the system found a good strategy quickly, whereas without the explorer the system kept trying alternative policies.

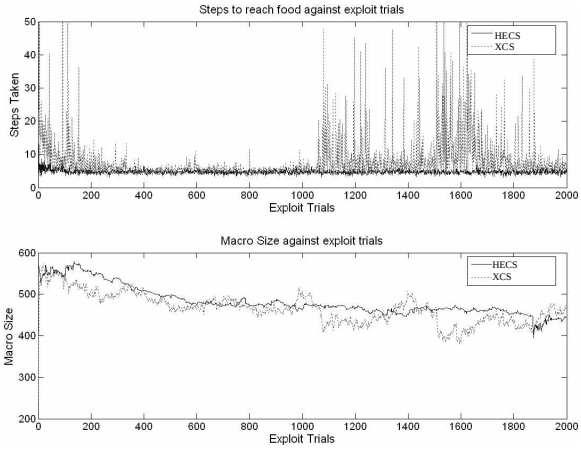

Figure 10. Graphs showing steps taken and macro population size for Maze4

5.3. Other environments

The program was tested in Woods100, as this environment is a POMDP. Standard XCS is unable to learn rules that give the best actions. Extensions to the standard XCS, such as memory, or a different LCS architecture, such as ACS, are required in such environments. Therefore, it is not surprising that HECS failed to evolve appropriate classifiers. The explorer could not make an improvement, so the animat was always below the required accuracy, and so very rarely exploited.

This does not indicate a problem with the implementation of the explorer; instead it suggests that it could be used to change algorithmic methods or algorithm goals. It may allow a system to recognise that its environment is not simple/Markovian and adjust behaviours accordingly.

5.4. Effect of the explorer parameters

The introduction of the explorer has increased the number of parameters required for HECS to run; it is worth noting which of these parameters HECS is most sensitive to, and which parameters are very robust. The 'Reward Slope Factor', M_{RS} in eqn. (5) seems to be fairly robust to changes in value. Increasing the 'Perfect Accuracy Weighting', M_{PA} in eqn. (1), improved the generalization ability in Woods1, however the same increase in Maze4 caused HECS to do many more exploration trials, never reaching a satisfactory accuracy level to begin exploitation. This suggests that this parameter is not very robust in the current implementation, although perhaps it could be adapted in relation to the estimate of the environment size.

An interesting parameter is the factor which multiplies the accuracy induced explorer level in eqn. (8), currently set at 0.9. If this factor is removed then the animat performs well in terms of requiring less explore trials (in Woods1) however it fails to generalise very much, as in effect it finds a set of classifiers which are accurate and lead to a reward from every location, and so it stops

exploring the environment and simply exploits its knowledge. Introducing this factor forces the animat to explore occasionally, and so causes the animat to continue to learn, whilst reducing the number of exploration trials required (compared to an unbiased explore/exploit choice).

6. DISCUSSION

This explore/exploit strategy is novel as it uses nonlinear combinations of feedback from reward, prediction accuracy and temporal performance. Although there are similarities to a number of explore/exploit strategies suggested by Wilson [7] nearly 10 years ago, the strategies have not been widely adopted. This is partly due to XCS without advanced explore/exploit strategies performing very well in terms of final 'average steps to reward' and rule generalisation where appropriate. However, the novel autonomous explore/exploit strategy enabled HECS to significantly reduce the number of exploration steps (and hence time taken) in MDP remains, whereas XCS is stuck with its *a priori* explore/exploit balance. The algorithm also adjusted to the needs of the system as training progressed, with the balance weighted towards exploration at the start of training.

The use of a temporal measure based on the expectation of reward was successful in preventing loop 'exploits' trials, which were a problem for XCS. Setting a limit of 50 consecutive unsuccessful exploit trials in XCS would reduce the time taken, but requires domain knowledge.

When tested in a POMDP domain, both systems failed as they did not have been required structure. A different structure, such as ACS, could be triggered by the explore/exploit strategy, although this would be a sequential process. The use of memory has been proposed to improve performance in such problem domains. A modified strategy could allow HECS to consider its memory register only when considering a state where rules had low confidence of action.

Implementing a memory system could also help prevent resampling in two ways. When a message is presented to the system it could be compared with a list of known difficulties, so exploration can be triggered. Secondly, a set of past training examples may be kept in memory along with associated rewards. Memory could be used so that the system exploits information until an unknown area of search space is reached. The system would actively search for areas that need greater exploration. This is similar to work done by Butz on biased exploration in anticipatory learning classifiers systems [9].

Care must be taken not to autonomously tune one parameter by introducing two parameters that require tuning. In the case of this explore/exploit strategy, the parameters introduced are reasonably robust and can lead to significant time savings.

7. CONCLUSIONS

A novel explore/exploit strategy has been introduced that provides a dynamic choice for each trial and escapes from unsuccessful exploit trials. This was achieved through the utilisation of statistics from environmental feedback and results in behaviour that offers performance benefits where domain knowledge cannot be used to set up a standard XCS.

Although tested in a limited number of domains, HECS appears to scale well and have reasonably robust parameter settings. Further investigation is required, including how far statistics used to guide LCS performance can be taken.

Acknowledgements: The authors would also like to thank Jan Drugowitsch for his useful advice.

8. REFERENCES

[1] R. Sutton and A. Barto, 'Reinforcement Learning: An Introduction', Cambridge, MA: MIT Press, 1998.

[2] S. Wilson. 'Classifier fitness based on accuracy', Evolutionary Computation, vol. 3(2), pp. 149-175, 1995.

[3] L. D. Whitley, Plenary Lecture 'Representation, Search and Learning.', Abstract Proc 3rd Int. Conf. on Artificial Neural Networks and Genetic Algorithms (ICANNGA97), Eds. Smith G. D., Steele N. C. and Albrecht R. F., Springer-Verlag Wein, New York, pp 624, 1997.

[4] M. V. Butz, 'Rule-base evolutionary online learning systems: learning bounds, classification and prediction.' PhD thesis University of Illinois, Illinois, 2004.

[5] J. H. Holland, 'Adaptation', in Progress in theoretical biology, Vol. 4, R. Rosen & F. Snell (Eds.), New York: Academic Press, 1976, pp. 263-293.

[6] M. V. Butz, 'Anticipatory learning classifier systems', Boston, MA: Kluwer Academic Publishers, 2002.

[7] S. Wilson, 'Explore/Exploit Strategies in Autonomy', From Animals to Animats 4. Proceedings of the Fourth International Conference on Simulation of Adaptive Behaviour, eds. P. Maes, J. Pollack, J. A. Meyer and S. W. Wilson, The MIT Press/Bradford Books, Cambridge, 1996.

[8] TSI Artificial Intelligence. 'Animat performance of ZCS, XCS and ACS' http://www.ai.tsi.lv.

[9] M. V. Butz, 'Biasing exploration in an anticipatory learning classifier system', Advances in Learning Classifier Systems: 4th International Workshop, IWLCS 2001, eds. P-L Lanzi, W. Stolzmann, and S. W. Wilson, Berlin: Springer-Verlag, 2001, pp. 3-22.