Toward a Better Understanding of Rule Initialisation and Deletion

Tim Kovacs
Dept. of Computer Science
University of Bristol
Bristol, BS8 1UB, U.K.
kovacs@cs.bris.ac.uk

Larry Bull
School of Computer Science
U. of the West of England
Bristol, BS16 1QY, U.K.
larry.bull@uwe.ac.uk

ABSTRACT

A number of heuristics have been used in Learning Classifier Systems to initialise parameters of new rules, to adjust fitness of parent rules when they generate offspring, and to select rules for deletion. Some have not been studied in the literature before. We study the interaction of these heuristics in an attempt to improve performance and detect any unnecessary methods. We evaluate the two published methods for initialisation of new rules in XCS and find the one based on parental values results in better evolutionary search but larger population sizes than the one based on population means. In preliminary work we demonstrate that when the difficulty of the 6 multiplexer is increased by reducing the population size limit and turning off subsumption we can improve performance by discounting the fitness of both parents and children.

Categories and Subject Descriptors

I.2.6 [**Artificial Intelligence**]: Learning—*Concept learning*; I.2.8 [**Artificial Intelligence**]: Problem Solving, Control Methods, and Search—*Heuristic methods*

General Terms

Algorithms

Keywords

Learning Classifier Systems, XCS, steady state genetic algorithms, deletion, fitness initialisation

1. INTRODUCTION

A number of heuristics have been used in Learning Classifier Systems (LCS) to initialise parameters of new rules, to adjust fitness of parent rules when they generate offspring, and to select rules for deletion. We study the interaction of these heuristics in an attempt to improve performance and detect any unnecessary methods. We work with the widely-studied XCS system [13], with which, for lack of space, we must assume familiarity, but in section 4 import ideas from other LCS.

GECCO'07, July 7–11, 2007, London, England, United Kingdom.

1.1 Rule initialisation methods

The original XCS paper indicates rule parameters should be initialised to fixed default values ([13] p. 6). However, this method appears to have been used only with random initial populations, which were used in early studies [11]. Later, Wilson started experiments with empty initial populations and relied on covering to seed the population. With this approach a new method of rule initialisation was used. It is not mentioned in [13] but is in [6] (p. 18).

> "The genetic algorithm creates two classifiers whenever it acts. If no crossover occurs, the offspring simply inherit the predictions of the parents. If crossover does occur, each offspring's prediction is set to the mean of the parents' predictions. Again, this is a policy of "minimal disruption". The offspring errors are set to one-fourth of the population mean error. The fitnesses are set to one-tenth of the population mean fitnesses." [11] section 3.3.3

We refer to this as the *population initialisation method*. The XCS algorithmic description of 2001 defines initialisation differently:

> "If the offspring are crossed, their prediction, error, and fitness values are set to the average of the parents' values." [4] section 3.9, p. 13.

We refer to this as the *parental initialisation method*. We hypothesise it improves on the earlier method as the parental values are likely to be better estimates of the childrens' values than the population averages are.

The XCS Java version 1.0 by Martin Butz [5] implements parental initialisation with a discount of the offsprings' error by one-fourth and fitness by one-tenth (as in the population method). We refer to the undiscounted parental initialisation as p1 and the discounted version as p2.

1.2 Deletion schemes

We will compare various deletion methods. In all cases rules are selected for deletion using a roulette wheel and all rules are eligible for deletion, not just those in the action set.

The first two methods, which we label t1 and t2 for techniques 1 and 2, are from [13]. In t1 the size of the slot on the roulette wheel occupied by a rule is its estimate of the average size of the action sets it participates in. The intention is to balance allocation of rules between action sets so they tend to have approximately the same size. This technique is based on one introduced in [1] for the same purpose.

In t2 the slot size is as in t1, except if a rule's fitness is less than a small fraction δ of the population mean fitness, in which case the slot size is multiplied by the population mean fitness divided by the rule's fitness. This has the effect of increasing the deletion probability of low-fitness rules.

A third scheme t3, a hybrid of the first two, was introduced in [7], and has since been adopted as the standard approach in XCS [4]. In this scheme a rule's slot size depends on how much experience it has. Rules with less than an experience threshold θ_{del} are treated as in t1, while those exceeding the threshold are treated as in t2. This scheme protects newly evaluated rules from the low-fitness deletion penalty of t2 until they have been well-evaluated, at which point useful rules should have gained enough fitness to escape penalisation.[1] The final scheme is uniform random deletion.

Results in [7] indicate that t1 is very similar to random deletion in terms of both %[O] and population size (see the following section) and we therefore omit it from the present study. That work did not consider rule initialisation methods or the fitness discounting of section 4.

1.3 Experimental method

We used the venerable 6 multiplexer Boolean function and sampled inputs uniformly from the entire function, a method which has been widely used with LCS (see e.g. [13, 7]). We used XCS with the standard parameter settings for the 6 multiplexer given in figure 1. We used GA subsumption but not action set subsumption. We will measure performance in terms of %[O] which indicates the percentage of the optimal solution present in the rule population. The optimal solution is defined as the minimal irredundant solution in the ternary language often used with XCS. In the figures higher %[O] is better. We will also evaluate the population size. Lower population size is better. These measures are explained in more detail in [8].

2. COMPARING INITIALISATION METHODS

Figure 2 compares the three initialisation methods. Of the two parental methods (p1 and p2) p2 has lower population size. They have very similar %[O], although p1 appears to lead very slightly early on. A tentative conclusion is that p2 is the better method.

As hypothesised in section 1.1 the later parental initialisation methods are superior in terms of %[O] to population initialisation. However, population initialisation is better in terms of population size. Our explanation of the population size difference is as follows. From one trial to the next population initialisation produces rules with very similar initial prediction error and fitness (since they are based on the same population means). In contrast, parental initialisation will produce rules with very different prediction error and fitness

[1]An earlier study which protects rules until they are well-evaluated is [10].

Parameter		Value
Subsumption threshold	θ_{sub}	20
GA threshold	θ_{GA}	25
t3 deletion threshold	θ_{del}	25
Covering threshold	θ_{mna}	1
Low-fitness deletion threshold	δ	0.1
Population size limit	N	400
Learning rate	β	0.2
Accuracy falloff rate	α	0.1
Accuracy criterion	ε_o	0.01
Accuracy exponent	ν	5
Crossover rate	χ	0.8
Mutation rate	μ	0.04
Hash probability	$P_\#$	0.33

Figure 1: Standard XCS parameter settings for the 6 multiplexer.

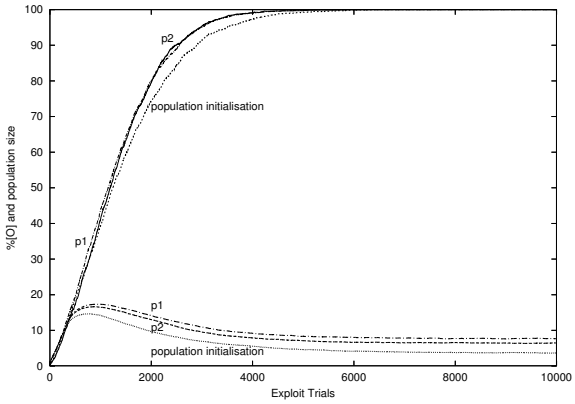

Figure 2: Two parental (p1 and p2) initialisation methods and population initialisation compared on the 6 multiplexer with 400 rules using t3 deletion. Curves averaged over 200 runs.

(since they are based on the parents, and parents have very different values). Parental initialisation will give some rules low initial error and high fitness. Among these, some will turn out to be low-quality rules. We hypothesise that the increased population size seen with parental initialisation is due to the persistence of low-quality rules which have inherited low error and high fitness from their parents. We hypothesise the increase in population size in p2 is less than in p1 because p2 discounts child error and fitness and hence its low-quality offspring are less disruptive than p1's.

3. COMPARING DELETION METHODS

In section 2 we we used t3 deletion because we earlier [7] found it superior to the the other deletion methods of section 1.2 in terms of %[O], and t3 was adopted as part of the XCS specification [4]. However, an initial experiment indicated the advantage of t3 over t2 had disappeared. We therefore compare a range of deletion methods in this section.

Figure 3 shows t2, t3 and random deletion with parental initialisation (p2). This not only fails to replicate t3's lead over t2 on %[O] but suggests t2 is slightly better, although its lead may not be statistically significant.

We therefore replicate experiments from [7] in which pop-

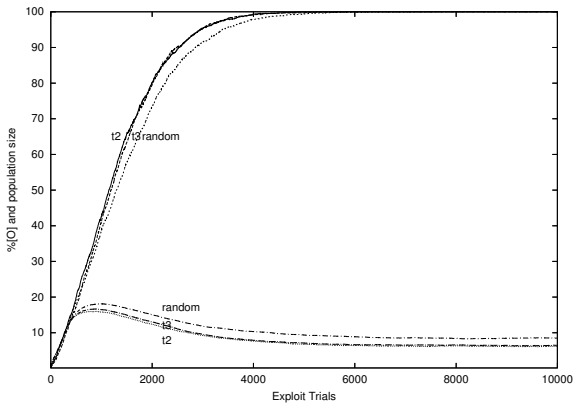

Figure 3: Deletion schemes compared on the 6 multiplexer with 400 rules and the p2 parental initialisation method. Curves averaged over 200 runs.

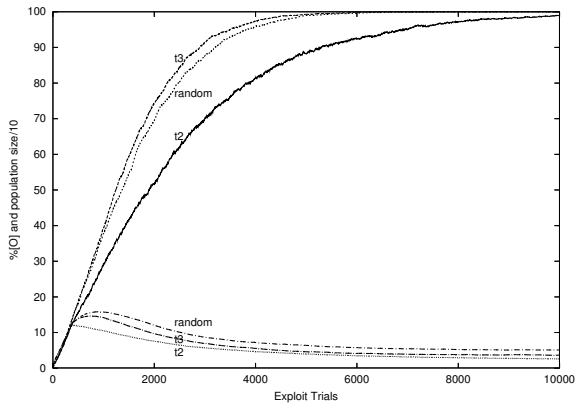

Figure 4: Deletion schemes compared on the 6 multiplexer with 400 rules and the population initialisation method. Curves averaged over 200 runs.

ulation initialisation is used. The results in figure 4 fit those in [7], indicating that the optimal deletion method depends on the rule initialisation method.

The best overall performance so far (on both %[O] and population size) is by t2 and parental initialisation (specifically p2). We might therefore conclude t3 and its protection of new rules is an unnecessary complication of the t2 scheme on which it is based. However, we hypothesise the protection principle is sound and that in other circumstances it might prove its worth. Specifically, we believe it will have an advantage in situations where good rules must wait longer for their fitness to rise and hence have more need for protection. This should occur when good rules are very specific, for examples in the parity problem. It should also occur when there is noise slowing the rise in fitness, or when there is a sever imbalance in the class distribution. Furthermore, recent work [9] has resulted in a more principled and less coarse method of protecting rules which should enhance t3's effectiveness.

4. FITNESS DISCOUNTING

We hypothesise that discounting the fitness of *both* the child and the parent will improve the convergence of %[O] by making it easier for children to compete with parents than when only the child is discounted. In fact, this approach introduces generalisation pressure because, of two equally accurate rules with the same discounted fitness, the more general will recover its true fitness more quickly (because it occurs in more action sets) and hence increase its reproductive probability more quickly. Parental fitness discounting is used to provide generalisation pressure in ZCS [12] where half the fitness of the parent is donated to the child. It has also been used in MCS [2] and YCS [3]. A more general discussion can be found in [3].

In XCS the main source of generalisation pressure is the niche GA and its effect is far stronger than that introduced by discounting parental fitness. However, we hypothesise that using both sources of generalisation pressure in the same system will improve performance over using only one. Our technique is to subtract 90% of one microclassifier's fitness from each parent.

Figure 5 compares discounting both parents and children

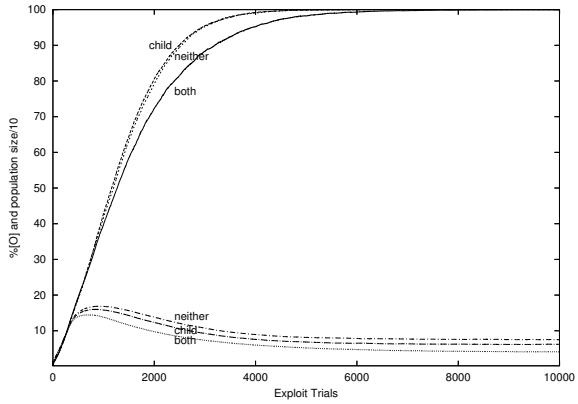

Figure 5: Fitness discounting methods compared on the 6 multiplexer with 400 rules, t2 deletion, p1 parental initialisation and fitness discounting as specified on each curve. Curves averaged over 1000 runs.

to discounting only children and discounting neither. Discounting both results in the smallest population size but much worse %[O]. Therefore we find no support for our hypothesis that %[O] will be improved by discounting both.

5. REDUCED POPULATION SIZE

We have failed to find support for the following hypotheses: i) that t3's protection of new rules will be beneficial when parental rule initialisation is used, and ii) that discounting both parents and children would improve %[O]. However, we strongly suspect that is because the 6 multiplexer is too simple a problem. In order to make it more difficult we now reduce the population size limit to 200 rules (from 400) and remove subsumption deletion. The results in figure 6 demonstrate that although discounting children leads on %[O] it does not converge to 100% [O]. Eventually discounting both overtakes it and does converge to 100% [O]. Discounting both also has the lowest population size. This supports the idea that more difficult problems will show the hypothesised advantages but more study is clearly needed.

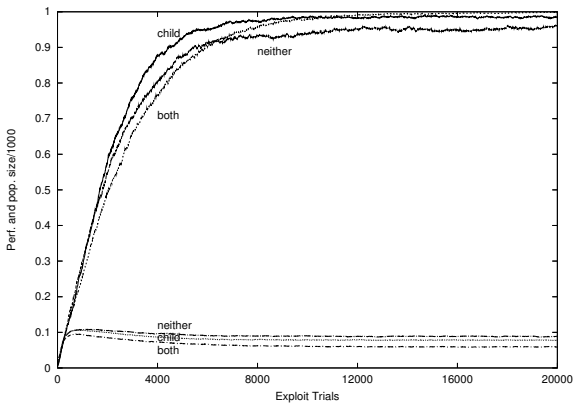

Figure 6: Fitness discounting methods compared on the 6 multiplexer with 200 rules and no subsumption deletion. Deletion and initialisation as in figure 5. Curves averaged over 200 runs.

6. CONCLUSION

We conclude that parental initialisation is superior to population initialisation, and that p2 may be superior to p1. We have found the optimal deletion method depends on the initialisation method used. We have failed to show t3 improving performance but strongly suspect it will do so on harder problems that the one used here. This will be the subject of further research.

We have shown that discounting the fitness of both parents and children can allow XCS to find 100% [O] when the multiplexer is made more difficult. Whether this holds on other more difficult problems will be the subject of further research.

The present study is entirely empirical and an avenue for future work would be to investigate principled ways of discounting fitness during reproduction.

7. REFERENCES

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

[2] L. Bull. A simple payoff-based learning classifier system. In X. Yao, E. Burke, J. A. Lozano, J. Smith, J. J. Merelo-Guervos, J. A. Bullinaria, J. Rowe, P. Tino, A. Kaban, and H.-P. Schwefel, editors, *Parallel Problem Solving from Nature - PPSN VIII*, pages 1032–1041. Springer Verlag, 2004.

[3] L. Bull. Two Simple Learning Classifier Systems. In L. Bull and T. Kovacs, editors, *Foundations of Learning Classifier Systems*, number 183 in Studies in Fuzziness and Soft Computing, pages 63–90. Springer-Verlag, 2005.

[4] M. V. Butz and S. W. Wilson. An Algorithmic Description of XCS. In P. L. Lanzi, W. Stolzmann, and S. W. Wilson, editors, *Advances in Learning Classifier Systems*, volume 1996 of *LNAI*, pages 253–272. Springer-Verlag, Berlin, 2001.

[5] M. V Butz. XCSJava1.0: An Implementation of the XCS classifier system in Java. Technical Report 2000027 at the Illinois Genetic Algorithms Laboratory, 2000. http://www.psychologie.uni-wuerzburg.de/i3pages/butz/

[6] T. Kovacs. Evolving Optimal Populations with XCS Classifier Systems. Master's thesis, School of Computer Science, University of Birmingham, Birmingham, U.K., 1996. Also technical report CSR-96-17 and CSRP-96-17 ftp://ftp.cs.bham.ac.uk/pub/tech-reports/1996/CSRP-96-17.ps.gz.

[7] T. Kovacs. Deletion schemes for classifier systems. In W. Banzhaf, J. Daida, A. E. Eiben, M. H. Garzon, V. Honavar, M. Jakiela, and R. E. Smith, editors, *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-99)*, pages 329–336. Morgan Kaufmann, 1999. Also technical report CSRP-99-08, School of Computer Science, University of Birmingham.

[8] T. Kovacs. *Strength or Accuracy: Credit Assignment in Learning Classifier Systems*. Springer, 2004.

[9] J. A. R. Marshall, G. Brown, and T. Kovacs. Bayesian Estimation of Rule Accuracy in UCS. *To appear in T. Yu, editor, the proceedings of the 2007 workshops on genetic and evolutionary computation.* ACM, 2007.

[10] C. Melhuish and T. C. Fogarty. Applying A Restricted Mating Policy To Determine State Space Niches Using Immediate and Delayed Reinforcement. In T. C. Fogarty, editor, *Evolutionary Computing, AISB Workshop Selected Papers*, number 865 in Lecture Notes in Computer Science, pages 224–237. Springer-Verlag, 1994.

[11] S. W. Wilson. NetQ question on "Classifier Fitness Based on Accuracy". http://www.eskimo.com/~wilson/netq. Accessed 22/3/2007.

[12] S. W. Wilson. ZCS: A zeroth level classifier system. *Evolutionary Computation*, 2(1):1–18, 1994.

[13] S. W. Wilson. Classifier Fitness Based on Accuracy. *Evolutionary Computation*, 3(2):149–175, 1995.