Revisiting Genetic Selection in the XCS Learning Classifier System

Faten Kharbat, Larry Bull & Mohammed Odeh
School of Computer Science
University of the West of England
Bristol BS16 1QY, U.K.
{faten2.kharbat, Larry.bull, Mohammed.odeh}@uwe.ac.uk

Abstract- The XCS Learning Classifier System has traditionally used roulette wheel selection within its genetic algorithm component. Recently, tournament selection has been suggested as providing a number of benefits over the original scheme, particularly a robustness to parameter settings and problem noise. This paper revisits the comparisons made between the behavior of tournament and roulette wheel selection within XCS in a number of different situations. Results indicate that roulette wheel selection is competitive in terms of performance, stability and generated solution size if the appropriate parameters are used.

1 Introduction

Learning Classifier Systems (LCS)[Holland, 1976] have traditionally used roulette wheel selection within their Genetic Algorithm (GA)[Holland, 1975] component. Holland's developed system [e.g., Holland, 1986] relied upon fitness sharing to maintain suitable rules within the evolving rule-base population and therefore its use of roulette wheel selection was crucial (see [Bull & Hurst, 2002] for discussions). The vast majority of LCS research has made a shift away from Holland's original formalism after Wilson introduced XCS [Wilson, 1995]. XCS uses the accuracy of rules' predictions of expected payoff as their fitness (Holland's original implementation also considered accuracy [Holland & Reitman, 1978]). As a result of some simple Markov-chain analysis of the selection pressure in accuracy-based LCS, it was suggested that other forms of selection schemes should be explored within such systems [Bull, 2001; 2002]. Recently, the use of tournament selection within XCS has been investigated [Butz et al., 2003]. Results indicate that tournament selection may be more robust in noisy environments and less dependent upon the value of the learning rate parameter (β). Also, it was shown that using roulette wheel selection caused XCS to fail if a low β value was used, i.e., a high dependency between the β parameter and the roulette wheel selection was shown to exist. Moreover, XCS using roulette wheel selection was tested with Gaussian noise added to the payoff and shown to fail to find the optimal solution even with a small amount of noise. XCS with tournament selection was shown able to cope with significantly greater levels of noise.

Using simple accuracy-based LCS, it has been shown that increasing the amount of separation between fitnesses can aid performance when using traditional roulette wheel selection in harder problems [Bull, 2004; 2005]. The same effect can be achieved within XCS. Therefore in this paper we revisit the tasks used to compare the two selection schemes. This comparison reveals the equality between the two alternatives and that the correct parameters' values are essential for both systems. It is shown that tournament selection does not outperform roulette wheel selection in all the problems previously used. The investigations reveal that roulette wheel selection is robust and independent of the β parameter if the appropriate fitness separation is used. On the other hand, tournament selection needs to have the correct tournament size to be able to generate an appropriate selection pressure. Most significantly, rule compaction is found to be more efficient under roulette wheel selection due to its inherent bias towards the most fit individuals within a population (e.g., see [Baker, 1987] for early discussions).

2 XCS

On each discrete time step, a set [M] is created consisting of those rules which match the current binary input vector, where rules use a trinary encoding with a '#' allowing generalization. A system prediction is then formed for each action in [M] according to a fitness-weighted average of the predictions of rules in each action set [A]. The system action is then selected either deterministically or randomly (usually 0.5 probability per trial). If [M] is empty, covering is used to make a rule whose condition matches the current input. Fitness reinforcement in XCS consists of updating three parameters via the Widrow-Hoff delta rule: predicted payoff (p), the error in that prediction (ε), and fitness (F) for each appropriate rule. The fitness is updated according to the error relative to those of the other rules within the set. In delayed reward tasks, the maximum value of the system's prediction array is discounted by a factor γ and used to update rules from the previous time step. Thus XCS exploits a form of Q-learning [Watkins, 1989] in its reinforcement procedure.

The GA acts in action sets [A], i.e., niches. Two rules are selected based on fitness from within the chosen [A]. Rule replacement is global and based on the estimated size

0-7803-9363-5/05/$20.00 ©2005 IEEE.

of each action set a rule participates in with the aim of balancing resources across niches. The GA is triggered within a given action set based on the average time since the members of the niche last participated in a GA. The intention is to form a complete and accurate mapping of the problem space through efficient generalizations. In this way, XCS represents a way of using reinforcement learning on complex problems where the number of possible state-action combinations is very large. The reader is referred to [Butz & Wilson, 2001] for a full description of XCS.

3 Roulette wheel vs. Tournament selection

Tournament selection [Goldberg & Deb, 1991] is among the most widely used selection schemes in genetic algorithms. In a single iteration a tournament is formed by selecting some number of individuals (usually called tournament size τ) from the population. The most fit individual wins the tournament. The size of the tournament controls the selection pressure; a bigger tournament size generates higher selection pressure. Tournament selection depends on the individuals' rank rather than on their relative fitness. Therefore, it is not affected by the fitness distribution through the population and doesn't require any global knowledge [e.g. Eiben & Smith, 2003], which can be considered a big advantage.

Since its beginning, XCS used the roulette wheel selection within its genetic algorithm component (so-called XCSRW). Recently, tournament selection was suggested to be used within reproduction to "enable XCS to solve a much broader class of problems in general" [Butz et al., 2003]. Butz et al. argued that using fitness proportionate selection causes XCS to have a strong dependency on β and an inability to solve noisy problems because of its dependency on the fitness scale rather than fitness rank. That is, fitness proportionate selection cannot differentiate between similar accuracies in an action set, thereby reducing the fitness pressure which "hinders XCS from evolving an accurate payoff map of the problem" [Butz et al., 2002].

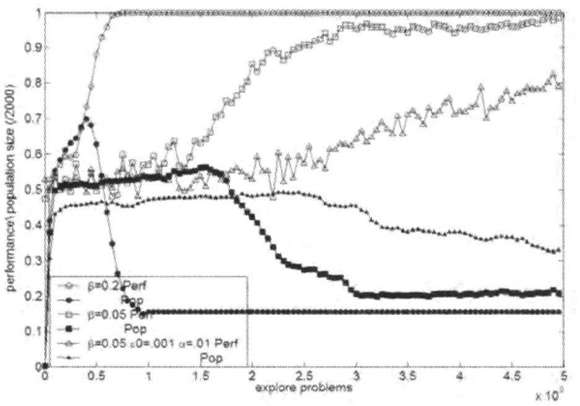
To reveal this XCS was applied to versions of the well-known multiplexer task. These Boolean functions are defined for binary strings of length $l = k + 2^k$ under which the first k bits index into the remaining 2^k bits, returning the value of the indexed bit. A correct classification results in a payoff of 1000, otherwise 0.

Performance (fraction of correct responses) and population (number of unique macroclassifiers in the rulebase) from exploit trials only are recorded, using a 50-point running average, averaged over ten runs (after [Wilson, 1995]).

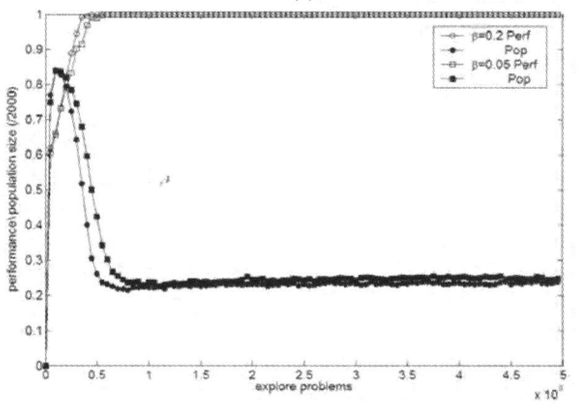
Tournament selection is implemented as suggested in [Butz et al., 2002], where the tournament size is a fraction $\tau = (0,1]$ of the current action set size (τ=0.4 is the suggested value). In order to select two parents, two independent tournaments are held by choosing the classifiers in each tournament randomly from the action set and then select the best one. If the size of the tournament will be less than one (i.e., the action set size is very small), a random classifier is chosen. Within this work, XCS using tournament selection is called XCSTS.

3.1 Parameter β Dependance in Boolean Problems

Figure 1 shows the behaviour of the roulette wheel selection and tournament selection within XCS on the 20-bit multiplexer problem, where the parameters are as in [Butz et al., 2003]: N=2000, $p_{\#}$=1.0, θ_{GA}=25, uniform crossover χ=0.8, free mutation μ=0.04, α=1, δ=0.1, ε_0=1.0, θ_{sub}=20, ν =5 and GA subsumption is used with different values of β[1] . XCSRW (Figure 1(a)) solves the problem with β=0.2 but fails with β =0.05; while XCSTS (Figure 1(b)) has the ability to solve both cases.



(a)



(b)

Figure 1: **20-bit multiplexer with β=0.2 and β=0.05 on XCSRW (a) and XCSTS(b)**

Figure 2 shows the second case that was discussed in [Butz et al., 2003], where a Gaussian noise is added with the mean zero and standard deviation σ to the environment payoff. Figure 2(a) shows that XCSRW fails to solve the problem with σ greater than 200. In fact even with σ equals to 100, it finds the problem difficult to solve. On the other hand, XCSTS in Figure 2(b) shows more performance stability.

[1] If not stated differently these settings are to be used for all the 20-bit multiplexer problem within this work.

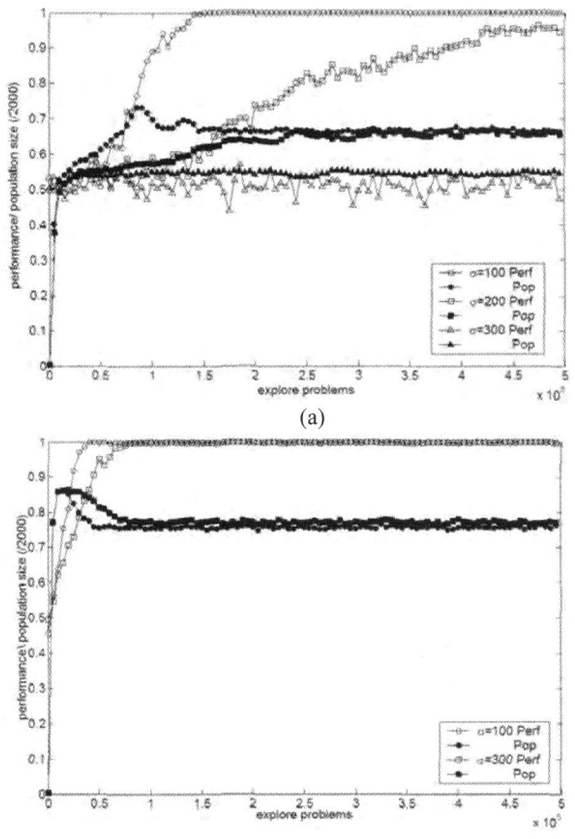


Figure 2: 20-bit multiplexer with different Gaussian noise values on XCSRW (a) and XCSTS (b)

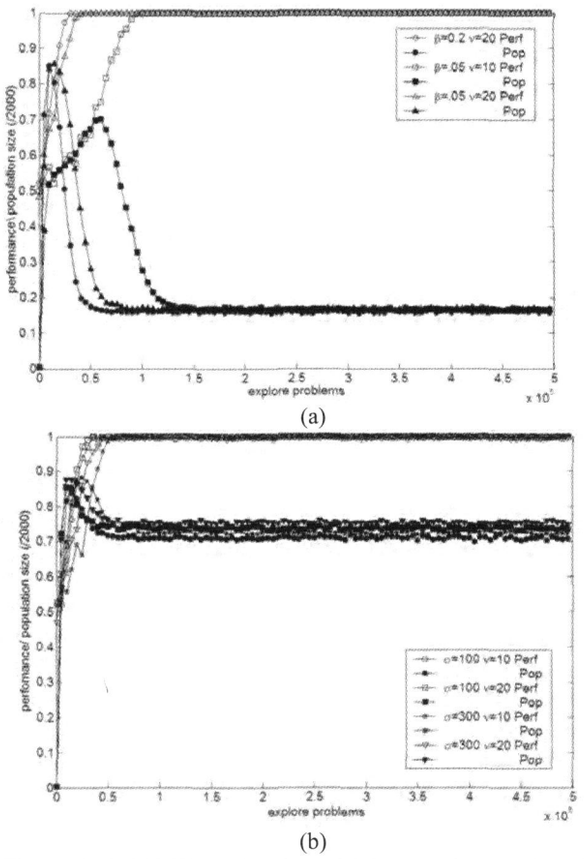


Figure 3: XCSRW on 20-bit multiplexer using ν=10 and ν=20 (a) and with σ=100 and σ =300 (b)

Accurate system parameters estimation is essential to ensure a proper fitness pressure [Butz et al., 2003] and therefore it is essentail to understand the nature of any selection scheme to determine the best parameter values that enable it to generate an appropriate selection pressure. Roulette wheel is a proportional selection which uses ν to separate between fitnesses. Increasing the separation between similar fitnesses can be obtained by a steeper function curve [Bull, 2004; 2005]; that is, a higher separation between fitnesses enables a higher ability in identifying the accurate rules. As a result, increasing ν (e.g. ν=20) shows that XCSRW is able to solve the 20-bit multiplexer problem with β=0.05 (Figure 3(a)).

Figure 3(b) also shows that XCSRW is able to solve the same problem in Figure 2(a) if an appropriate ν value is chosen. Variant values of ν were tested and ν=20 shows a better performance speed and stability in the non-noisy environment. This value may need to be bigger if a harder problem is to be solved as will be shown later, but the main point remains the same. i.e., to control the selection pressure by using a steeper function to separate similar errors.

Furthermore, Figure 4 and 5 are related to the 37-bit multiplexer problem using the same settings as in [Butz et al., 2001]: N=5000, $p_{\#}$=0.6, θ_{GA}=25, χ=0.8, μ=0.04, α=1, δ=0.1, ν =5, ε_0=10.0 and θ_{sub}=20, where learning (i.e. updating the action set) is taking place in both exploit and explore trails, It is clear that XCSRW fails to solve the problem with low β and low ν as shown in Figure 4(b), but improves in terms of speed and population size if the required selection pressure is used. i.e., an appropriate ν value is used, ν=20.

In fact, XCSTS in Figure 5(b) fails to reach the goal with low τ and low β, although τ was chosen from the accepted range [0.2-0.8] ([Butz et al., 2003]).Nevertheless, changing the value of τ can improve the result (as shown in Figure 5(a) & (b)). That is, high selection pressure is needed to solve this problem in the expected time. Having shown a competitive performance speed in all problems, XCSRW generated *more* compacted solution than XCSTS as will be discussed in section 3.3.

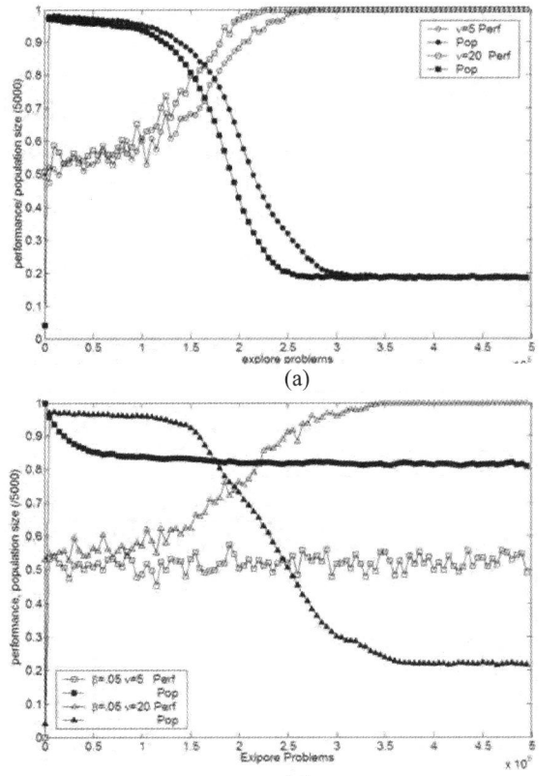


Figure 4: 37-bit multiplexer on XCSRW with β=0.2 (a) and β=0.05 (b) using different ν values

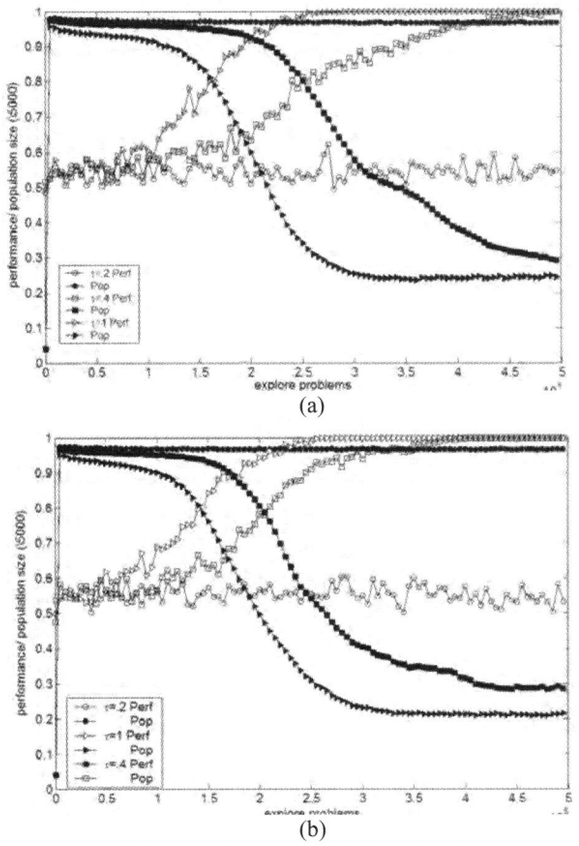


Figure 5: 37-bit multiplexer on XCSTS using β=0.2 (a) and β=0.05 (b) with different τ values

The count ones problem is another binary task that was used to compare between the two selection schemes in [Butz et al., 2003]. It uses the previous settings except N=3000 with a string length 20, k=7 relevant bits and ν=5. The problem can be described as follows: if the number of ones in the k relative bits is greater than $k/2$, then the classification should be 1 else it should be 0. The 1000/0 reward scheme is used. The results in Figure 6 are identical to the ones in [Butz et al., 2003] except for the population compaction level, and it is clear from the figures that the difference between the two systems is very marginal and is perhaps directed towards XCSRW. XCSRW in Figure 6(a) generates a more compacted solution than XCSTS in Figure 6(b) as in the previous cases. Moreover, it reaches the anticipated performance faster with a higher mutation rate.

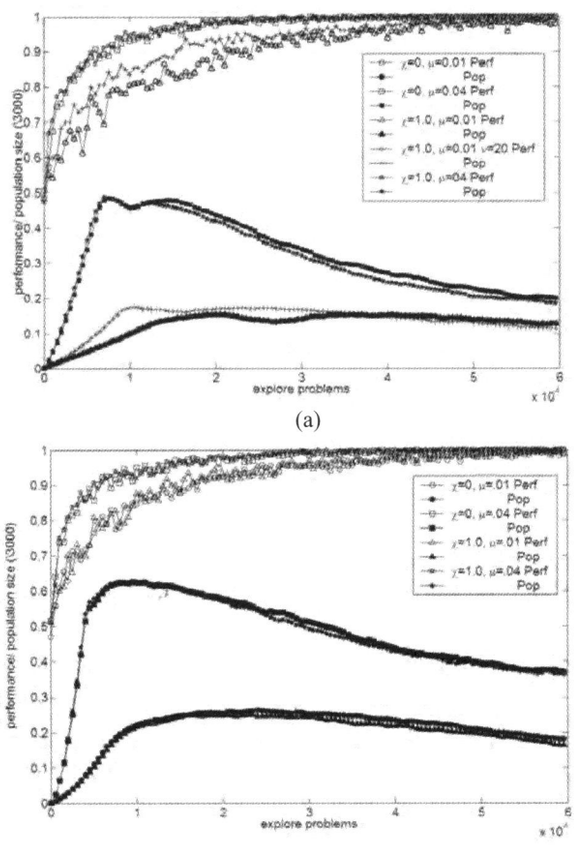
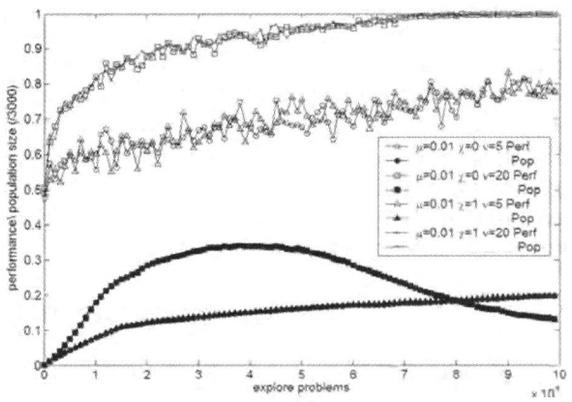


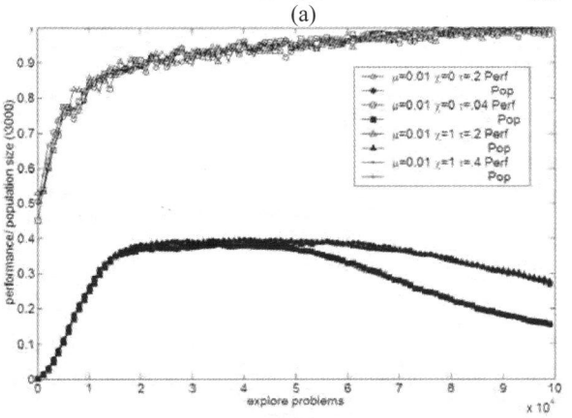
Figure 6: Count One l=20, k=7, N=3000 over XCSRW (a) and XCSTS (b)

Butz et al. [2002] have constructed the layered count ones problem which was also used in [Butz et al., 2003] to reveal the effectiveness of recombination in XCS and to expand the comparison between XCSTS and XCSRW. The layered count ones problem is similar to the count ones except for the returning reward which was defined to be $\sum_{i=0}^{k} value[i]$ in the correct classifications and 1000 - $\sum_{i=0}^{k} value[i]$ in the wrong ones. Figure 7 concurs with the results from Butz et al. [2002] about considering having

the correct selection pressure to be one of the "… important [criteria] for a reliable convergence." [ibid]. XCSRW fails to solve the problem with low selection pressure (i.e., ν=5). Both systems are identical in terms of performance speed when the XCSRW is using ν=20, despite the low mutation rate or the absence of crossover.



(a)



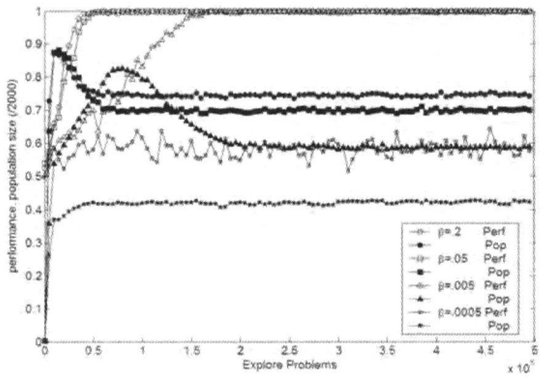
(b)

Figure 7: **Layered Count One *l*=20, *k*=7, *N*=3000 over XCSRW (a) and XCSTS (b)**

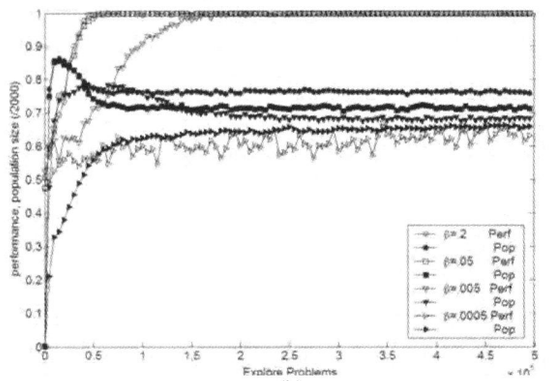
3.2 Noise Dependance in the multiplexer problem

Figure 8 shows another comparison between XCSRW and XCSTS to reveal the extent to which both systems depend on parameter β in the noisy environments. Figure 8(a) shows that XCSRW using ν=20 is robust in noisy problems up to σ= 250 and β=0.005 as well as XCSTS in Figure 8(b). The two systems show identical behavior. However, both systems XCSRW and XCSTS fail to solve the problem with σ= 250 and β=0.0005 or with σ= 500 as shown in Figure 9.

Beside the effect of ν and τ on the selection pressure, the mutation rate μ is also a sensitive parameter that affects the population specificity (the higher the mutation rate the higher the population specificity) and generates a pressure that - in addition to the fitness and set pressures - results XCS being able to generate a maximally accurate general solution [Butz & Pelikan, 2001] [Butz & Sastry, 2003]. Butz et al. [2003] also mentioned the effect of high mutation rate which could disturb the generated solution.

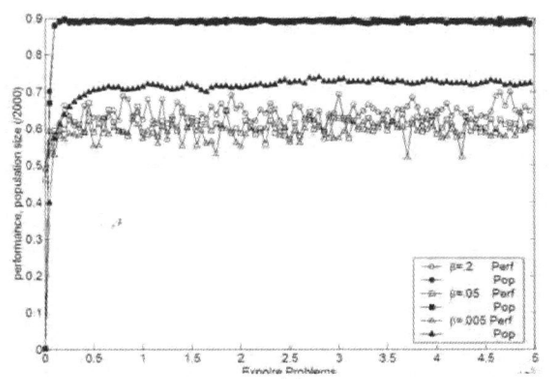


(a)

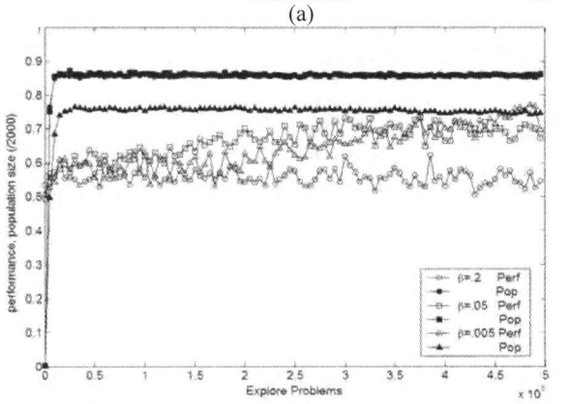


(b)

Figure 8: **20-bit multiplexer with σ =250 on XCSRW using ν=20 (a) and XCSTS (b) with different values for β**



(a)



(b)

Figure 9: 20-bit multiplexer with σ =500 on XCSRW using ν=20 (a) and XCSTS (b) with different values for β

In the noisy environments Butz and Pelikan [2001] suggested that the more specific the classifier is, the higher fitness it achieves. Therefore, their reproduction probability will increase and their deletion probability will decrease, which will cause the population specificity to become higher than expected. Consequently, in the high noise cases, either the mutation rate should be decreased to allow XCS to produce the maximally accurate general solution or the population size should be increased to ensure the occurrence of the reproductive opportunity [Butz et al., 2003]. The latter suggestion is not suitable since we could not determine the extent to which the population size should be increased. Thus, balancing the mutation pressure was explored.

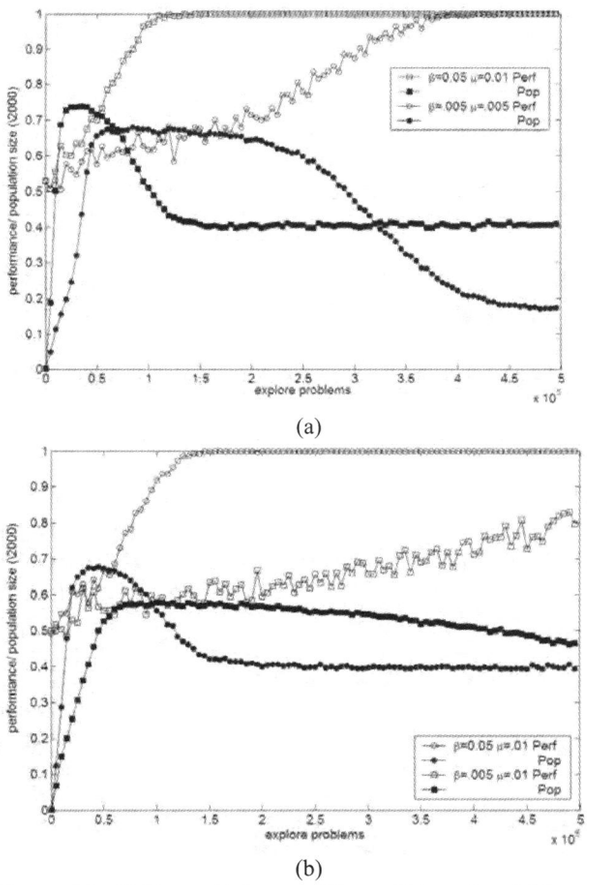


Figure 10: **20-bit multiplexer with σ =500 and μ=0.01 on XCSRW using ν=50 (a) and XCSTS with τ=1 (b) with different values for β**

Figure 10(a) shows that XCSRW with a steeper function (ν=50) is able to solve the noisy problem that is in Figure 9(a) if the correct value of μ is used. In fact, the combination between the two parameters is strong and a single modification does not work without the other (not shown). The same scenario applies for XCSTS in Figure 10(b), which also solves the problem using the correct value of μ. Tournament selection was able to solve the problem with β=0.05 and τ=1 using μ=0.04, but it needs longer time (not shown). It was also able to solve the problem with β=0.005 and τ=1 using μ=0.01 and p#=0.6 after 700000 trials (not shown). This confirms results in [Kharbat et al., 2004] where the mutation's affect on performance was highlighted. Moreover, XCSRW was able to solve the problem in Figure 8(a) with β=0.0005 and σ =250 using the same settings as shown in Figure 10, while XCSTS could not achieve the accurate solution (not shown).

Changing the mutation rate to μ=0.01 results in improved performance in the case of the 37-bit multiplexer problem and allows XCSRW to solve the problem in less than 200,000 trials with both β=0.2 and ν=20 (not shown).

3.3 Final Population Size

XCSRW seems to generate more compacted populations than XCSTS. For example, in the case of the 20-bit multiplexer problem XCSRW generates 0.18 of the population size compared to XCSTS which generates 0.25 of the same population. Moreover, XCSRW generates 0.19 of the population size in the 37-bit multiplexer where it is more than 0.23 in XCSTS. In fact, even in the noisy environment XCSRW seems to generate more compacted populations than XCSTS.

Tournament is a rank based selection scheme where each classifier in the population has a selection probability based on its rank within its niche. Hence, tournament selection tends to explore all the classifiers within the population and this can have several consequences on the population distribution: (1) overgeneral classifiers with high experience and high prediction error are kept until picked for deletion for their low fitness, (2) over specific classifiers will be kept as well for the GA subsumption sooner or later, and (3) the maximally accurate general classifiers will have the chance to stay in the population.

Whereas in the roulette wheel selection the bias is towards the most fit individuals within a population (e.g., [Baker, 1987]). Therefore, reproduction is not relying on deletion as in tournament selection.

Furthermore, anaylsis shows another reason that prevents tournament from generating the same level of roulette wheel compaction which is the existance of overlapped classifiers. It is known that in some action sets, non-overlapped classifiers are present without the existence of any overlapped ones. The fitness for the latter classifiers will be shared relatively with the non-overlapped ones and therefore will approximate a lower fitness value [Butz et al., 2003]. In other words, the fitness of the overlapped classifiers will not reach the maximum value (which is one) whereas the non-overlapped classifiers will have the maximum value when they exist alone in some other action sets.

Because of the nature of having a bias toward the better individual, roulette wheel selection imposes the strength of the non-overlapped classifiers by selecting them consistently for reproduction due to their high fitness, and naturally the overlapped classifiers will die from not having the opportunity for reproduction.

On the other hand, the overlapped classifiers still have their chance for reproduction in the rank based system and

therefore they will not die but will have both lower numerosity and fitness than the non-overlapped ones.

Butz et al. [2003] suggested, however, that relative-fitness-based XCS tends to produce a non-overlapped accurate maximally general solution that can describe each environmental niche. That is correct if the roulette wheel selection is used as it is able to eliminate the overlapped classifiers keeping only the minimum number of general accurate classifiers.

4 Conclusions

This paper has revisted a comparison between roulette wheel and tournament selections within XCS. The behavior of roulette wheel was investigated as well as tournament selection by applying them to different types of problems. The value of ν guides the selection pressure within roulette wheel since it represents the steepness of the fitness curve. Therefore, choosing the correct value of ν gives the system the ability to create the required amount of pressure that can solve the problems used by [Butz et al., 2003] including the noisy ones. It also controls the roulette wheel's ability to cope with the low values for other parameters.

Also, it has been observed that tournament selection is sensitive to the tournament size τ as has been shown in the 37-bit multiplexer problem. Choosing the correct value for τ is essential to create the correct amount of pressure to solve problems. From this investigation the best range for ν seems to vary between 5 and 50, where τ can be chosen from the previous suggested range [0.2-0.8] in [Butz et al., 2003] taking into consideration that the value τ=1.0 is required in some cases.

As well as the learning rate (β) needing to be decreased in noisy environments to generate accurate payoff estimation, so does the mutation rate (μ). This is another parameter that should be optimized if a problem is to be solved. It is well known that the mutation rate (μ) controls the diversity within the population which guides the search for the accurate solution. The previous results show that the mutation rate (μ) not only affects the specificity [Butz et al., 2001] but also affects the performance [Kharbat et al., 2004].

Most significantly, rule compaction is found to be more efficient under roulette wheel selection due to its inherent bias towards the most fit individuals within a population.

Acknowledgments

Faten Kharbat would like to acknowledge the support of Zarqa Private University (ZPU), Zarqa, Jordan for funding this research

Bibliography

Baker, J. (1987) Reducing Bias and Inefficiency in the Selection Algorithm. In J. Grefenstette (ed) *Genetic Algorithms and their applications: proceeding of the second International Conference of Genetic Algorithms*. Lawrence Erlbaum Associates, pp14-21.

Bull, L. (2005) Two Simple Learning Classifier Systems. In L. Bull and T. Kovacs (eds) *Foundations of Learning Classifier Systems*. Springer.

Bull, L. (2004) Lookahead and Latent Learning in a Simple Accuracy-based Learning Classifier System. In X. Yao et al. (eds) *Parallel Problem Solving from Nature - PPSN VIII*. Springer Verlag pp 1042-1050.

Bull, L. (2002) On Accuracy-based Fitness. *Soft Computing* 6(4): 154-161.

Bull, L. and Hurst, J. (2002) ZCS Redux. *Evolutionary Computation* 10(2): 185-205.

Bull, L. (2001) Simple Markov Models of Genetic Algorithms in Classifier Systems: Multiple-step tasks. In P-L. Lanzi, W. Stolzmann and S.W. Wilson (eds) *Advances in Learning Classifier Systems: Proceedings of the Third International Workshop*. Springer, pp29-36.

Butz, M., Goldberg, D., and Tharakunnel, K. (2003a) Analysis and improvement of fitness exploration in XCS: bounding models, tournament selection, and bilateral accuracy. *Evolutionary Computation* 11(3):239-277.

Butz, M., Sastry, K. and Glodberg, E. (2003b) Strong, Stable, and Fitness Pressure in XCS due to Tournament Selection. Technical Report IlliGAL no 2003027, University of Illinois at Urbana-Champaign.

Butz, M., Sastry, K., Goldberg, D. (2002) Tournament selection in XCS. Technical Report IlliGAL no 2002020, University of Illinois at Urbana-Champaign.

Butz, M., Kovacs, T., Lanzi, P., and Wilson, S. (2001) How XCS Evolves Accurate Classifiers. In L. Spector , E. Goodman , A. Wu , W. Langdon, H. Voigt , M. Gen, S. Sen, M. Dorigo, S. Pezeshk, M. Garzon and E. Burke (eds) *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO 2001)*. Morgan Kaufmann, pp.927-934.

Butz, M., and Pelikan, M. (2001) Analyzing the evolutionary pressures in XCS. *Proceeding of the Genetic and Evolutionary Computation Conference (GECCO 2001)*. Morgan Kaufmann, pp.935-942.

Butz, M., and Wilson, S. (2001) An Algorithmic Description of XCS. In P-L. Lanzi, W. Stolzmann and S. W. Wilson (eds) *Advances in Learning Classifier Systems*. Springer-Verlag, pp.253-272.

Eiben,A., and Smith, J. (2003) *Introduction to Evolutionary Computing*. Springer.

Goldberg, D., and Deb, K. (1991) A comparative analysis of selection schemes used in genetic algorithms. *Foundations of Genetic Algorithms*. Morgan Kaufmann, pp. 69-93.

Holland, J. (1986) Escaping Brittleness: The Possibility of General-Purpose Learning Algorithms Applied to Rule-Based Systems. In R.S.Mishalski, J.G. Carbonell, and T.M. Mitchell *Machine Learning II.* Kaufman, pp.593-623.

Holland, J., and Reitman, J. (1978) Cognitive Systems Based on Adaptive Algorithms. *In* .A. Waterman and F. Hayes-Roth *Pattern-directed Inference Systems*. Academic Press, pp313-29.

Holland, J. (1976) Adaptation. In R. Rosen and F.M. Snell (Eds) *Progress in Theoretical Biology IV.* Academic Press, pp263-93.

Holland, J. (1975) *Adaptation in Natural and Artificial Systems*. University of Michigan Press.

Kharbat, F., Bull, L. & Odeh, M. (2004) Further Investigation of Accuracy-based Fitness Using a Simple Learning System. In *Proceedings of the International Arab Conference on Information Technology (ACIT2004)*, pp311-318.

Watkins, C., (1989). *Learning from delayed rewards*. Dissertation, King's College, Cambridge, UK.

Wilson, S.W. (1995) Classifier Fitness Based on Accuracy. *Evolutionary Computing* **3**: 149-175.