2. Learning Classifier System Overview

A LCS is a machine learning system based on reinforcement learning and GAs. Similar to an expert system, it utilizes a knowledge base of syntactically simple production rules that can be manipulated by a GA [11]. This learning and generalization [12] ability make LCSs convenient for solving complicated multi-step problems [13]. GAs are a class of computerized search procedures that are based on the mechanics of natural genetics [14]. In a LCS, the GA discovers new rules among a population of candidate rules based on the experience of existing rules. The use of a rule-based system allows the LCS to conveniently represent and refine complex control strategies [10]. The robust search ability of the GA enables effective discovery of new rules on the basis of performance-only feedback. The reinforcement learning technique determines the rule fitness and enables the system to learn from its environment based on a reward signal that implies the quality of its action.

LCSs have found such real-world applications as gas pipeline control (the first application) [15], Boolean function learning [16], function approximation [17], sequence prediction [18], letter recognition [19], modeling economic markets [2], and job-shop scheduling [20].

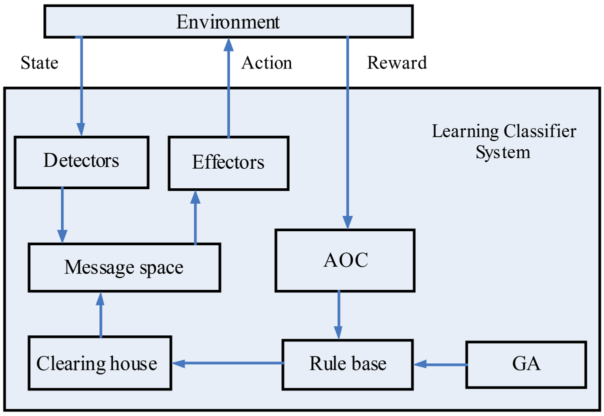

Figure 1. The structure of a LCS.

The structure of a LCS is depicted in Figure 1. The LCS senses its environment through its detectors and takes appropriate action with the help of its effectors. The environment generates a reward signal that can be used to guide the learning system by the reinforcement program. The apportionment of credit (AOC) is based on an economic analogy and consists of a bid competition among classifiers that match the current environmental input. Accordingly, matched classifiers bid a certain proportion of their strength and rule conflicts are resolved based on a probability distribution over the bids [10]. A winner classifier has to pay out all its debt through the clearing house, hence risking a certain percentage of its strength with the possibility of getting a reward.

3. The Proposed Bidding Strategy for Learning Classifier Systems

The following implementation has three major parts: the learning system, reservoir, and bank. The reservoir and bank are conceptually part of the clearing house but are portrayed on separate blocks for clarity. Figure 2 shows how the learning system interacts with all other elements in the system (environment, bank, and reservoir). It senses its environment through its detectors and takes action on the environment via its effectors. The environment will then generate a reward or punishment in response depending on the quality of the action taken. During auctions, matching classifiers that qualify for a loan will request a loan from a bank to boost their potential to win. The reservoir collects all the taxes from the classifiers for later use to subsidize new classifiers and the bank at times of bankruptcy. From the bank and real market points of view, the terms reward, tax, and strength of a classifier can be taken as analogous to money.

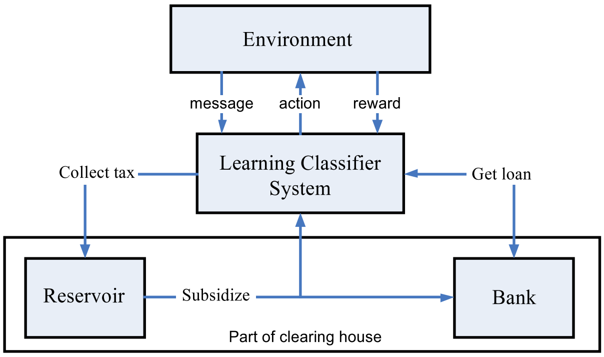

Figure 2. Block diagram of the proposed approach.

3.1 Classifier Format

As shown in Figure 3, the classifier in our implementation has seven parameters: condition, action, debt, strength, due date, experience, and accuracy. Similar to previous implementations, the condition is a string from the ternary alphabet (0, 1, or #) and the action is binary (0 or 1). The hash symbol (#) in the condition matches both 0 and 1 inputs. The debt parameter represents the loan amount a classifier takes from a bank. A classifier that takes a loan but is unable to pay out its debt will incur an interest charge from the bank each time it belongs to the match set. The higher the debt value, the more likely the classifier will be picked for deletion by the GA. The due date parameter indicates how much time (in terms of number of times the classifier belonged to the match set) is left for the classifier to pay out its debt. The value for the due date is derived in Section 3.3.3.

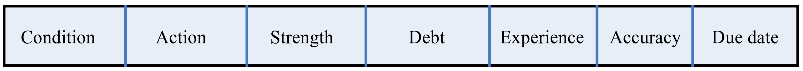

Figure 3. Classifier format.

3.2 Learning System

The learning system consists of four major components: the auction, clearing house, reinforcement program, and the GA.

3.2.1 Auction

All classifiers in the match set participate in auctions by bidding a fixed proportion of their strength. The bidding approach does not distinguish whether the classifier is specific or general, experienced or new, accurate or bad. The bid amount depends merely on the value of its current strength. In our approach, we have introduced a new modified bidding strategy to the existing LCS implementation where classifiers in the match set can request a loan from a bank based on clearly defined loan-granting criteria discussed later. Once the loan is granted, the value can be added to the classifier's current strength value and used during auctions to issue a bid. This improves a classifier's potential to win an auction and try its action, which may in turn result in receiving a reward at earlier iterations. In other terms, a classifier gets a chance to take action without waiting until bad classifiers weaken due to continuous taxation.

The potential bid (PB) of a classifier during auction is given by

$$\mathrm{PB} = C_{\mathrm{bid}} * \mathrm{Strength}. \tag{1}$$

And its effective bid (EB) is computed by adding a random noise to its potential bid as

$$\mathrm{EB} = \mathrm{PB}(1 + \mathrm{rand}() * \mathrm{EBID}) \tag{2}$$

where EBID is a constant (like 10%) used during simulations and C_{bid} is the bid tax constant (see Table 1 later). In our case, the value for the strength in equation (1) includes a loan taken from a bank, if needed, and is calculated using

$$\mathrm{Strength} = (\mathrm{Current\ strength}) + (\mathrm{Loan\ from\ bank}). \tag{3}$$

3.2.2 Clearing House

The clearing house is the part of the learning system where all classifiers clear out their taxes. Accordingly, all classifiers in the population set pay an existence tax and classifiers in the match set pay an additional overhead tax, while a winner classifier also has to pay the bid amount.

3.2.3 Reinforcement Program

The learning system continuously interacts with its environment through its detectors and effectors. The reinforcement program determines the rule's fitness by generating a signal in the form of a reward or punishment. It guides the learning system to evolve to optimal or near optimal solutions. A reward is given to a classifier provided its proposed action is correct, while absence of a reward is considered a punishment for a wrong action. In our implementation, general classifiers are more favored over specific ones as the amount of reward (RD) generated is made proportionate to the number of hash symbols in the classifier's condition:

$$\begin{aligned} \mathrm{RD} = {} & \mathrm{R} * \\ & (\mathrm{ONH} * (1 - \mathrm{abs}(\mathrm{Numhashes} - \mathrm{ONH}) / \mathrm{ONH}) + 1) \end{aligned} \tag{4}$$

where R is the minimum reward that can be given for a classifier with correct action and ONH is the optimum number of hashes in the condition of a classifier. This scheme enables general classifiers to dominate specific classifiers at a faster rate, hence speeding up overall system convergence to perfect solutions.

3.2.4 Genetic Algorithm

In a LCS, the GA discovers new rules among a population of candidate rules based on the experience of existing rules. Each GA operation brings two new classifiers to the existing population of classifiers. GAs diversify the population by adding some degree of randomness in picking parents for reproduction. However, applying a GA at each iteration will not cause major variations in the system. In addition, because the GA is applied to the whole population, applying it iteration-wise will be costly from a computational point of view. Instead, we applied it on an interval basis where the interval is determined by observing the performance of the system. A roulette wheel selection strategy is used to select two parents for reproduction. The new offspring inherit some portion of their parents' strengths and also a subsidy from the reservoir system to make them competent with the relatively more experienced classifiers in the system during auctions.

3.2.5 Deletion Criteria

Each GA operation brings two new offspring to the population. In a fixed population size approach, a criteria for selecting two classifiers for deletion has been an important issue. A simple but crude approach is to select the classifier with minimum strength for deletion. This, however, has several shortcomings as strength alone does not indicate the quality of a classifier. Instead, we considered the due date, strength, accuracy, and experience parameters of the classifier as criteria for deletion. A deletion point (DP) is computed for each classifier in the population using the formula

$$\mathrm{DP} = \left(\frac{1}{(1+\mathrm{acc})^{\mathrm{exp}}} + \frac{1}{S_t} \right) * 100 - D \tag{5}$$

where acc is the accuracy, exp is the experience, S_t is the strength, and D is the due date value of the classifier.

Accordingly, two classifiers with maximum DP values are selected and replaced by the new offspring. Unlike traditional LCSs, however, we do not lose the strength value of the outgoing classifiers. Instead, the strength value is added to the reservoir, provided that the classifier has no debt. But if the classifier picked for deletion already has a debt, the debt amount is returned to the bank. At times, when a classifier's strength is less than the amount of its debt, the reservoir will subsidize the bank by paying the remaining amount to the bank, hence putting the bank at no risk all the time. Subsidizing the bank during bankruptcy guarantees that the bank will always have some capital to lend to classifiers at any time.

3.3 Bank

Nature has given some degree of inherent intelligence to living things. The essence behind machine learning has been to introduce this kind of intelligence to inanimate things. To this end, auction-based competition in a LCS is crafted in direct analogy to how human beings participate in auctions in real life. In real life, however, people also propose a business plan and ask for a loan to implement their ideas, taking some risk at times of business failures. Before people decide to participate in an auction, they make a survey of the overall market activity by reviewing previous auction histories. In so doing, they can get an estimate of the potential of possible competitors in the current market that helps them decide how much to borrow, if needed. In the same analogy, we have introduced loan and bid history concepts to the traditional LCS implementation. The role of the bank is to grant a loan to classifiers that satisfy the loan criteria discussed in Section 3.3.2.

3.3.1 Bid History

The bid history is a global variable that keeps track of the average bid in previous auctions. It serves as a guide to the classifiers that participate in an auction. Each classifier that belongs to the current match set compares its strength value to the average bid history of the previous few auctions. Based on that information, the classifier decides how much to loan from a bank. The whole intention of a classifier is to build up its strength by winning an auction to get a reward from its environment. Hence, if a classifier believes its strength suffices to win an auction, there is no point of taking a loan. So, the average bid history serves as a benchmark for classifiers to decide how much of a loan to request from a bank or to bid without any loan.

3.3.2 Loan Criteria

Loan criteria are set to distinguish classifiers that qualify for a loan from a bank. To request a loan, a classifier must belong to the current match set and its strength should be less than the average bid history. The maximum loan amount a classifier can request is limited to 75% of its current strength. This is to make sure that it will be able to pay out its debt within a few cycles. Based on its experience (the number of times the classifier has belonged to the match set), accuracy, and the amount of the loan it has requested, a bank will decide to either grant or deny the loan. When the loan is granted, the due date and debt parameters of the classifier and the capital of the bank are updated accordingly. A classifier that belonged to the match set several times, but has never received the chance to fire, has a higher chance of getting a loan. Similarly, a classifier that has won several times, but its action has been incorrect most of the time, is least likely to qualify for a loan. In other words, the classifier is considered bad because it has already received the chance to fire. Subsequently, there is no point in favoring this kind of classifier. Since the main purpose of the loan is to give good classifiers a chance to test their action as early as possible, a classifier with no or little experience also has a high chance of getting a loan.

3.3.3 Due Date Calculation

The due date parameter of the classifier indicates the time left for a classifier to pay out all of its debt. At first, its value is initialized to zero. When a classifier receives a loan, the due date parameter value is updated using equation (10). Each time a classifier belongs to the match set, its due date value is decreased by one. A negative due date indicates that the classifier is unable to pay back its loan on time and will have a bad credit history thereafter. The worse the credit history of a classifier, the more likely it will be picked for replacement by new classifiers coming from the GA operation.

Assuming a classifier has a strength S_t immediately prior to getting a loan and assuming the worst-case scenario that it fails to get a reward every time it fires, its strength value after n iterations is given by

$$S_n = S_t * (1 - K)^n; \tag{6}$$

adding interest to its debt each time it belongs to the match set, its total debt at the end of m intervals is given by

$$A_m = A * (1 + I)^m \tag{7}$$

where S_n is strength after n iterations, S_t is the strength at the time of requesting a loan, A_m is the amount of debt after a classifier belongs to the match set m times, A is the loan granted, I is the interest rate, K is the tax constant, n is the number of iterations, and m is the number of times the classifier has been in the match set.

The more hashes (#) a classifier has in its condition, the higher its chance of belonging to the match set. Hence, without loss of generality, we can establish a relation between n and m as

$$m = \left(\frac{\text{number of hashes}}{\text{condition length}} \right) * n. \tag{8}$$

To solve for the due date parameter, we equate equations (6) and (7):

$$S_t * (1 - K)^n = A * (1 + I)^{p*n}. \tag{9}$$

Solving equation (9) for n, we obtain the expression for the due date as

$$\text{due date} = \frac{\log\left(\frac{S\,t}{A}\right)}{\log\left(\frac{(1+I)^p}{1-K}\right)} \tag{10}$$

where p = number of hashes / condition length.

3.4 Reservoir

The approach followed is a closed system where money enters into it only in the form of reward from the environment. Otherwise, all the money circulates between the bank, reservoir, and classifiers. All forms of taxes (existence, overhead, and bid) collected from the classifiers are stored in the reservoir for later use to subsidize the bank to avoid bankruptcy and to support new classifiers received through the GA. Unlike existing LCS implementations, when classifiers are selected for replacement by new offspring from the GA, their strength value will not be discarded. Instead, any value in excess of their debt is accumulated in the reservoir. At times when a classifier's debt exceeds its strength, the reservoir will subsidize the bank by paying the difference.

4. Simulation Results

The proposed algorithm is applied for solving two well-known multiplexer problems: 6mux and 11mux. In our terminology, *N*mux stands for an *N* input multiplexer problem. In an *N*mux problem, the first *b* inputs represent the control inputs and the remaining $N - b$ inputs are the data lines. The optimum number of hashes in the condition of a classifier for an *N*mux is given by $2^b - 1$. Accordingly, the optimum number of hashes for 6mux is 3 and for 11mux is 7.

Environmental inputs for both scenarios are generated and simulations are carried out for various values of the parameters given in Table 1. The results for the specific values of the parameters given are depicted in Figures 4 through 11. The fraction correct (Figures 4 and 8) shows the percentage of correctly identified environmental inputs at each epoch. One epoch stands for one complete presentation of all the environmental inputs to the LCS. For instance, for the 6mux (see Figure 4), the learning system identifies about 90% of the environmental inputs after the 40th epoch. For the 11mux (Figure 8), the same level of correctness is achieved at about the 15th epoch. The number of possible environmental inputs is given by 2^x (where x is the number of inputs, 6 or 11 in our case).

To contrast the performance of the proposed approach, similar simulations without a loan were also conducted. As can be clearly seen from Figures 4 and 8, significant improvements in convergence are obtained by introducing a loan. Oscillations in the performance plot (see Figure 4) arise due to the randomness in the GA during deletion and insertion of new classifiers. Figures 5 and 9 show a moving average plot for the bid history. The bid history plot basically shows the strength that a classifier needs to have in order to win an auction. It gives an estimate of the average bid value in the current market.

To understand the interaction between the different objects in our model (bank, learning system, and the reservoir), simulations showing the bank and reservoir capital are also portrayed. The bank starts with initial capital and its only source of income is interest collected from classifiers that take a loan. As can be seen in Figures 6 and 10, the capital increases with more iterations. This is an indication that classifiers are borrowing money from the bank during auctions and are able to repay their loans. For the 6mux (Figure 6), the bank's capital becomes stable after the 120th epoch. As for the 11mux, there is no variation in the bank's capital after about the 30th epoch (Figure 10). These results are expected because, as the system evolves to optimal classifiers, their strength becomes high enough as a result of continuous reward from the environment. In other words, classifiers can issue a bid and win without the need of a loan from the bank. The reservoir capital (Figure 7) first decreases and then increases in almost a linear fashion. This is due to the fact that initially classifiers take a loan from the bank and then may be deleted by the GA before they repay their loan. The learning system achieves 100% convergence for the 6mux (Figure 4) and about 98% level of convergence is achieved for the 11mux (Figure 8).

Parameter	6mux	11mux	Meaning
Pop Size	100	300	Number of classifiers
Iterations	12000	55000	Number of iterations
Cexs	0.001	0.001	Existence tax
Ctax	0.005	0.005	Overhead tax
Cbid	0.1	0.1	Bid tax
Px	0.8	0.8	Probability of crossover
Pm	0.01	0.01	Probability of mutation
GA_TH	10	25	GA threshold

Table 1. List of simulation parameters with their optimum values.

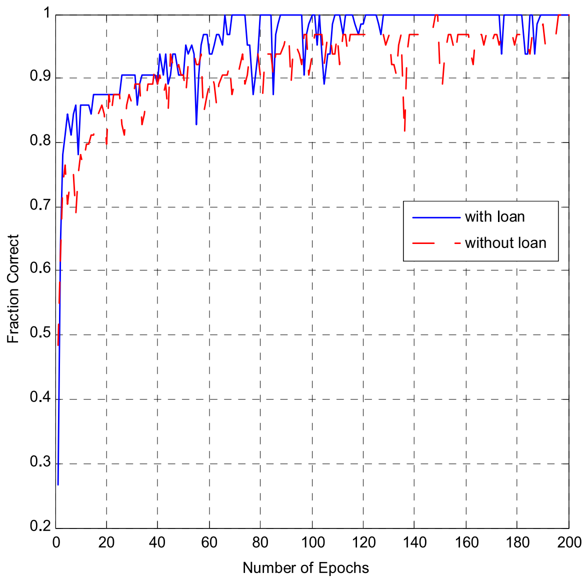

Figure 4. Performance plot for the 6mux problem.

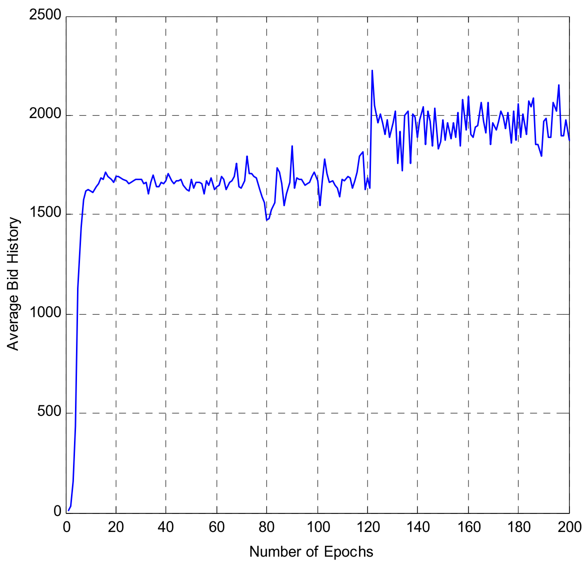

Figure 5. Bid history plot for the 6mux problem.

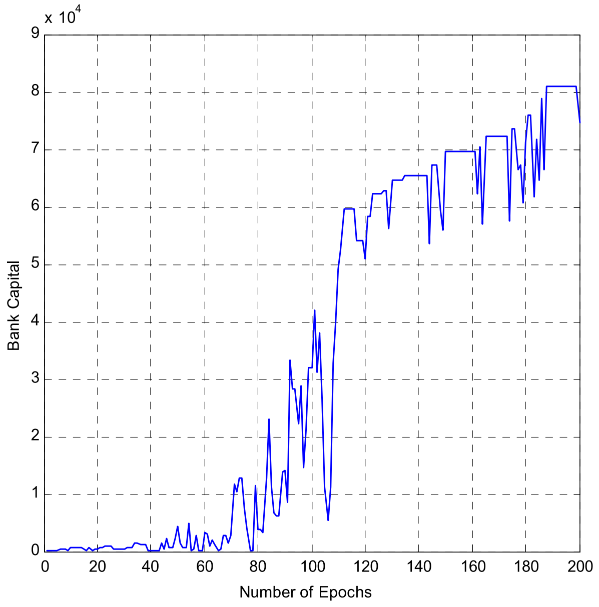

Figure 6. Bank capital for the 6mux problem.

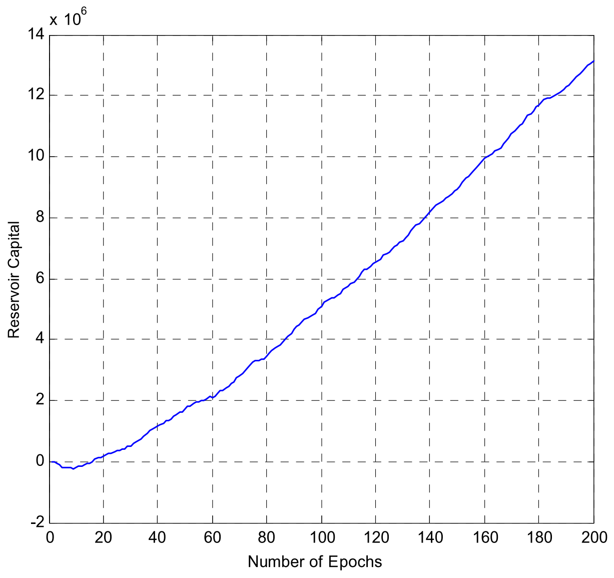

Figure 7. Reservoir capital for the 6mux problem.

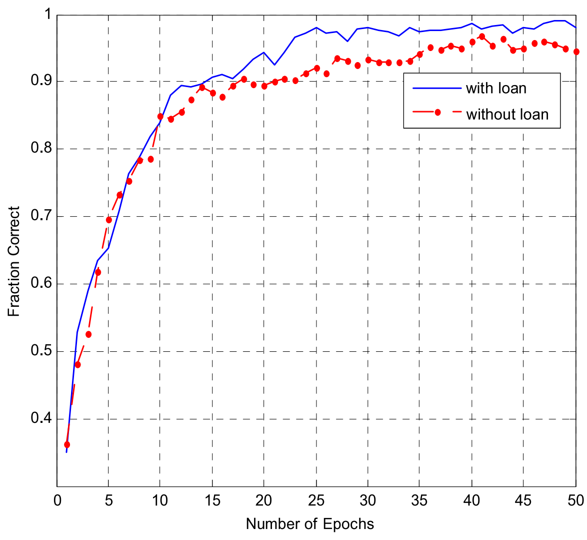

Figure 8. Performance plot for the 11mux problem.

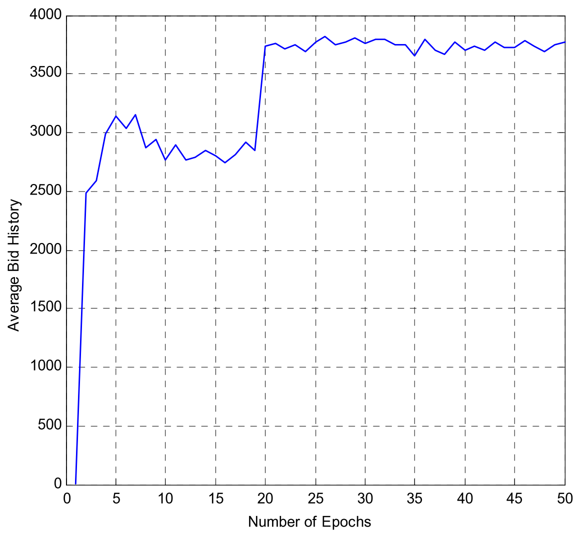

Figure 9. Bid history plot for the 11mux problem.

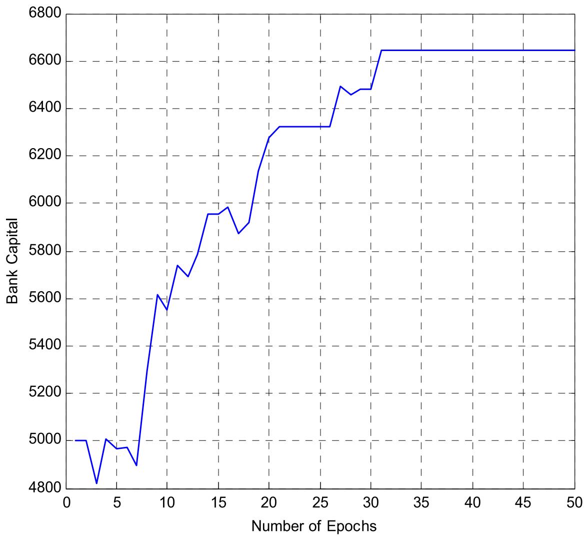

Figure 10. Bank capital for the 11mux problem.

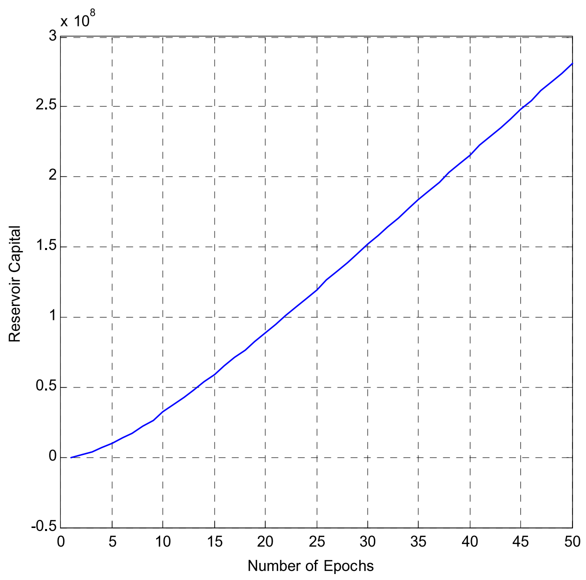

Figure 11. Reservoir capital for the 11mux problem.

5. Conclusion

This paper has shown the feasibility of introducing loan and bid history to the bidding strategy in existing learning classifier systems (LCSs). In existing LCSs, it usually takes a higher number of iterations before getting a satisfactory result. As the number of inputs gets larger (e.g., for 11mux), the problem gets worse and convergence becomes a nightmare. Based on clearly defined criteria, classifiers are allowed to consider a loan from a single central bank to boost their strength during auctions. A boost in the bidding potential of classifiers that matches most of the environmental niches, by introducing a loan from the start, speeds up convergence of the system by allowing them to fire earlier. The impact of a loan is two-fold depending on how good the classifier is. After receiving a loan, a good classifier will get a reward and pay its debt right away as the reward value is generally much larger than the loan amount. On the other hand, a bad classifier that takes a loan but does not get a reward will be weakened further due to interest and taxation. This enables optimal classifiers to dominate in the population at a faster rate. The results obtained showed a significant improvement in the performance of the system as a result of introducing a loan. In our future work, we will consider a decentralized approach where a loan exchange among classifiers in the system is allowed, instead of using just a single central bank for loaning.

References

[1] J. Holland, "Adaptation," *Progress in Theoretical Biology*, **4**, 1976 pp. 263–293.

[2] J. H. Holland and J. S. Reitman, "Cognitive Systems Based on Adaptive Algorithms," in *Pattern Directed Inference Systems* (D. A. Waterman and F. Hayes-Roth, eds.), New York: Academic Press, 1978 pp. 313–329.

[3] J. H. Holland, "Adaptive Algorithms for Discovering and Using General Patterns in Growing Knowledge Bases," *International Journal of Policy Analysis and Information Systems*, **4**(3), 1980 pp. 245–268.

[4] J. H. Holland, "Escaping Brittleness: The Possibilities of General-Purpose Learning Algorithms Applied to Parallel Rule-Based Systems," in *Machine Learning: An Artificial Intelligence Approach* (R. S. Michalski, J. G. Carbonell, and T. M Mitchell, eds.), St. Louis, MO: Morgan Kaufmann, 1986.

[5] S. Wilson and D. Goldberg, "A Critical Review of Classifier Systems," in *Proceedings of the Third International Conference on Genetic Algorithms and Their Applications (ICGA '89)*, Fairfax, VA (J. D. Schaffer, ed.), San Francisco, CA: Morgan Kaufmann, 1989 pp. 244–255.

[6] S. Wilson, "ZCS: A Zeroth Level Classifier System," *Evolutionary Computation*, **2**(1), 1994 pp. 1–18.

[7] S. Wilson, "Classifier Fitness Based on Accuracy," *Evolutionary Computation*, **3**(2), 1995 pp. 149–175.

[8] P. L. Lanzi, D. Loiacono, S. W. Wilson, and D. E. Goldberg, "XCS with Computed Prediction for the Learning of Boolean Functions," in *IEEE Congress on Evolutionary Computation (CEC '05)*, Endinburgh, Scotland, IEEE, 2005 pp. 588–595.

[9] A. Homaifar, D. E. Goldberg, and C. C. Carroll, "Boolean Function Learning with a Classifier System," in *Proceedings of the Applications of Artificial Intelligence VI at the International Society of Optical Engineering and the Computer Society of the IEEE (SPIE '88)*, Orlando, FL (M. M. Trivedi, ed.), Bellingham, WA: Society for Photo-Optical Instrumentation Engineers, 1988 pp. 264–272.

[10] R. E. Smith and D. E. Goldberg, "Reinforcement Learning with Classifier Systems," in *IEEE Proceedings of AI, Simulation, and Planning in High Autonomy Systems*, Tuczon, AZ, IEEE 1990 pp. 184–192.

[11] J. H. Holland, K. J. Holyoak, R. E. Nisbett, and P. R. Thagard, *Induction: Processes of Inference, Learning, and Discovery*, Cambridge, MA: MIT Press, 1986.

[12] M. V. Butz, T. Kovacs, P. L. Lanzi, and S. W. Wilson, "Toward a Theory of Generalization and Learning in XCS," *IEEE Congress on Evolutionary Computation*, **8**(1), 2004 pp. 28–46.

[13] M. V. Butz, D. E. Goldberg, and P. L. Lanzi, "Gradient Descent Methods in Learning Classifier Systems: Improving XCS Performance in Multistep Problems," *IEEE Transactions on Evolutionary Computation*, **9**(5), 2005 pp. 452–473.

[14] D. E. Goldberg, *Genetic Algorithms in Search, Optimization, and Machine Learning*, Reading, MA: Addison-Wesley, 1989.

[15] D. E. Goldberg, "Computer-Aided Gas Pipeline Operation Using Genetic Algorithms and Rule-Learning," Ph.D. dissertation, University of Michigan, 1983.

[16] S. W. Wilson, *Classifier System Learning of a Boolean Function*, Cambridge, MA: Rowland Institute for Science, 1986.

[17] M. V. Butz, P. L. Lanzi, and S. W. Wilson, "Function Approximation with XCS: Hyperellipsoidal Conditions, Recursive Least Squares, and Compaction," *IEEE Transactions on Evolutionary Computation*, **12**(3), 2008 pp. 355–376.

[18] G. Robertson and R. Riolo, "A Tale of Two Classifier Systems," *Machine Learning*, **3**(2–3), 1988 pp. 139–159.

[19] P. Frey and D. Slate, "Letter Recognition Using Holland-Style Adaptive Classifiers," *Machine Learning*, **6**(2), 1991 pp. 161–182.

[20] M. R. Hilliard et al., "A Classifier Based System for Discovering Scheduling Heuristics," in *Genetic Algorithms and Their Applications: Proceedings of the Second International Conference on Genetic Algorithms (ICGA '87)*, Cambridge MA (J. J. Grefenstette, ed.), Hinsdale, NJ: Lawrence Erlbaum Associates, 1987 pp. 231–235.