Parameter Adaptation within Co-adaptive Learning Classifier Systems

Chung-Yuan Huang[1,2] and Chuen-Tsai Sun[1]

[1] Department of Computer and Information Science
National Chiao Tung University
Hsinchu, Taiwan 300
{gis89802, ctsun}@cis.nctu.edu.tw
[2] Department of Computer Science and Information Engineering
Chung Kuo Institute of Technology
Hsinchu, Taiwan 303

Abstract. The authors propose a co-adaptive approach to controlling parameters for coevolution-based learning classifier systems. By taking advantage of the on-line incremental learning capability of such systems, solutions can be produced that completely cover a target problem. The system combines the advantages of both adaptive and self-adaptive parameter-control approaches. Using a coevolution model means that two learning classifier systems can operate in parallel to simultaneously solve target and parameter-setting problems. Furthermore, the approach needs very little time to become efficient in terms of latent learning, since it only requires small amounts of information on performance metrics during early run-time stages. Our experimental results show that the proposed system outperforms comparable models regardless of a problem's stationary/non-stationary status.

1 Introduction

Learning classifier systems (LCSs) [2] were introduced in the mid-1970s by John H. Holland, known as the father of genetic algorithms (GAs). Such systems take advantage of GAs and reinforcement learning (RL) [11] to build adaptive rule-based systems that learn gradually via on-line experiences [3, 6]. Depending on its architecture, a learning classifier system may be regarded as an extended GA application or a reinforcement learning algorithm. The reinforcement component of learning classifier systems is responsible for distributing credit among rules and resolving rule conflicts (i.e., distinguishing between appropriate and inappropriate rules). The genetic algorithm component is responsible for comparing performance in order to identify potentially better rules to replace unsuitable rules. Originally, learning classifier systems were not completely analyzable due to the complex nature of component interrelationships [3]. It wasn't until 1995, when Wilson proposed his eXtended Classifier System (XCS) [12] based on classifier prediction accuracy, that a number of new models and applications were presented, thus renewing interest in learning classifier systems. Wilson retained Holland's original ideas and main architecture, but made some fundamental changes that gave XCS at least four

K. Deb et al. (Eds.): GECCO 2004, LNCS 3103, pp. 774–784, 2004.

advantages [3, 12, 13]: a) easier analyzability; b) the ability to deal with complex problems (e.g., optimization issues) that had previously been considered unsolvable; c) adding a robust generalization mechanism capable of generating compact, complete, maximized, and accurate solutions [6]; and d) the capability to use various representations to specify classifiers [9, 10].

As with evolutionary computations (ECs), parameter settings determine whether learning classifier systems can generate optimal or near-optimal solutions and whether it can do so efficiently. All of the currently available approaches [1, 4, 5] to solving the parameter-setting problem associated with learning classifier systems have important drawbacks requiring improvement and modification [14-18]. Our proposed co-adaptive approach, which is based on the coevolution concept and Dyna architecture [11], takes advantage of the incremental on-line learning capability of learning classifier systems to produce solutions that completely cover a target problem.

The system simultaneously adapts parameters according to current learning performance and state. As shown in Fig. 1, the framework consists of two learning classifier systems. Main-LCS is responsible for solving the external target problem and Meta-XCS is responsible for adapting internal parameters. We used the Dyna architecture to acquire the parameter-control capability of Meta-XCS in a short time period. Dyna uses an internal world model to save real experiences that are obtained during learning and uses them for an intensive latent learning process that shortens training time and speeds up the construction of a complete set of solutions. In [8], Lanzi showed that a combination of Dyna and XCS (Dyna-XCS) was capable of greatly enhancing learning performance. For the present study we used Dyna-XCS to a) solve the slow-learning problem of the adaptive parameter control approach (which requires a long training period) and b) significantly enhance parameter-control stability.

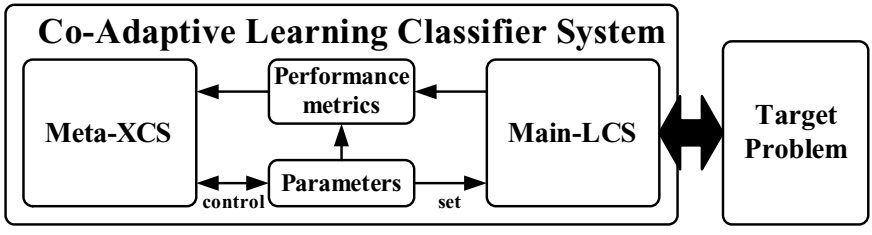


Fig. 1. Co-adaptive learning classifier system framework

To solve the parameter-control problem common to learning classifier systems, we established a framework in which Main-LCS and Meta-XCS operate in parallel. Solutions co-evolve as the systems cooperatively adjust parameters to a given target problem. There are two advantages to using coevolution to solve the parameter setting problem: many benefits of the self-adaptive parameter-control approach are maintained without expanding the target problem's original search space, and the premature convergence problem that often accompanies this approach is avoided.

2 XCS Overview

As with traditional learning classifier systems, XCS is a problem-independent and adaptive machine learning model. As shown in Fig. 2, XCS has four components: a finite population of classifiers, a performance component, a reinforcement component, and a rule discovery component. Stored classifiers control the system through a horizontal competition mechanism and perform tasks via vertical cooperation. The performance component governs interactions with the target problem. The input interface (called a detector) is used to transmit the current state of the target problem to the performance component and to determine dominant classifiers according to an exploration/exploitation criterion. Through the output interface (called an effector), any action advocated by dominant classifiers is performed and receives feedback. The reinforcement component (also known as the credit assignment component) uses an algorithm similar to Q-learning [11] to update the reinforcement parameters of classifiers advocated the action. Finally, the rule discovery component uses a genetic algorithm to search for better or more general classifiers and to discard existing incorrect or more specific classifiers.

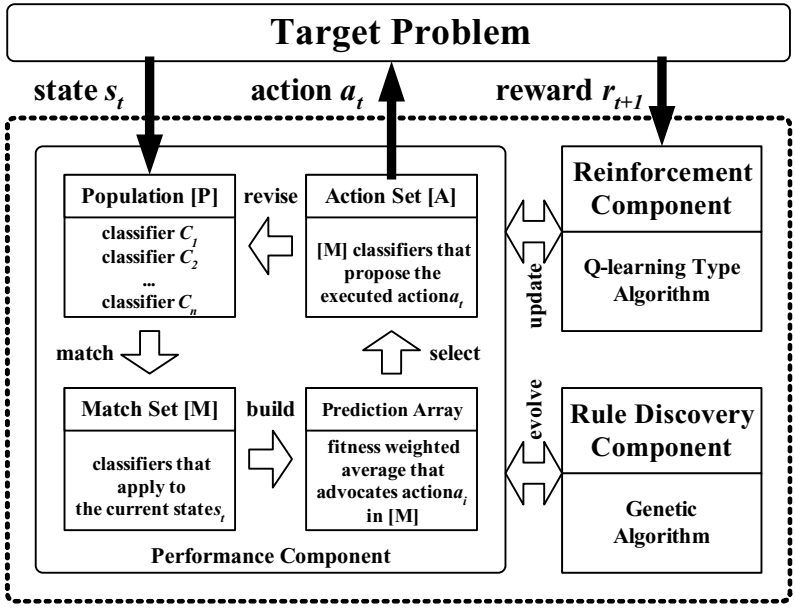


Fig. 2. XCS architecture

When running XCS, we use performance metrics to observe system performance and to state classifier populations. Kovacs [6] divided these performance metrics into two categories: performance measures and population state measures. Three well-known on-line metrics for measuring performance in research environments and real-world XCS applications are performance ρ, system error, and population size. Performance ρ and system error are used to measure XCS learning capability

according to results from target problem interactions. As one would expect, the population size performance metric (defined as the number of macro-classifiers in a classifier population) belongs to the category of population state measures. It is responsible for measuring XCS learning quality.

3 Co-adaptive Learning Classifier Systems

Because of its ability to deal with complex problems, we decided to use XCS to optimize other learning classifier system parameters, especially its ability to represent various parameter-control strategies [3, 9, 10]. Specifically, we used XCS to capture relations and changing parameter patterns between parameter-control strategies and to observe their effects on a given target problem. Its principal features include a) the combined advantages of adaptive and self-adaptive parameter-control approaches that allow for the use of a coevolution model to simultaneously solve a target problem and parameter-setting problem, and b) reduced time requirements for becoming efficient via a latent learning process that uses small pieces of information about performance metrics in the early stages of a run.

3.1 The Model

The architecture of the co-adaptive learning classifier system is shown in Fig. 3. Its four principal components are the Main-LCS, Meta-XCS, performance-metrics, and parameter components. As with ordinary learning classifier systems, the Main-LCS component is responsible for interacting with and solving the target problem. Meta-XCS integrates Dyna architecture with XCSμ [7, 8], which is especially useful in stochastic environments where the results of actions are affected by uncertainty. The Meta-XCS component learns parameter control and adjustment strategies in a short time period. The performance-metrics component is responsible for collecting, recording, and evaluating Main-LCS performance and regularly transmitting performance metrics information to the Meta-XCS component. The last component stores all parameters that need to be adjusted by the Meta-XCS component and assigns updated parameters to the Main-LCS component.

3.2 Meta-XCS Component

After a certain number of Main-LCS runs, the Meta-XCS component receives a message (s_t) transmitted by the parameter and performance-metrics components. In addition to the current parameter settings affecting the Main-LCS component, s_t also contains measures of the Main-LCS component's performance and population states. Based on the information in s_t, XCSμ in the Meta-XCS component executes parameter-control action a_t and instructs the parameter component to update the corresponding parameter. Next, the Meta-XCS component receives a s_{t+1} message and r_{t+1} feedback in the form of a reward or penalty from the performance-metrics component. The Meta-XCS component uses r_{t+1} for reinforcing learning and for storing the $<s_t, a_t, s_{t+1}, r_{t+1}>$ information within the Dyna architecture. During intervals

between parameter-control actions, the Meta-XCS component uses these records for latent learning. The cycle continues until the target problem is solved by the Main-LCS component or a user-requested criterion is met.

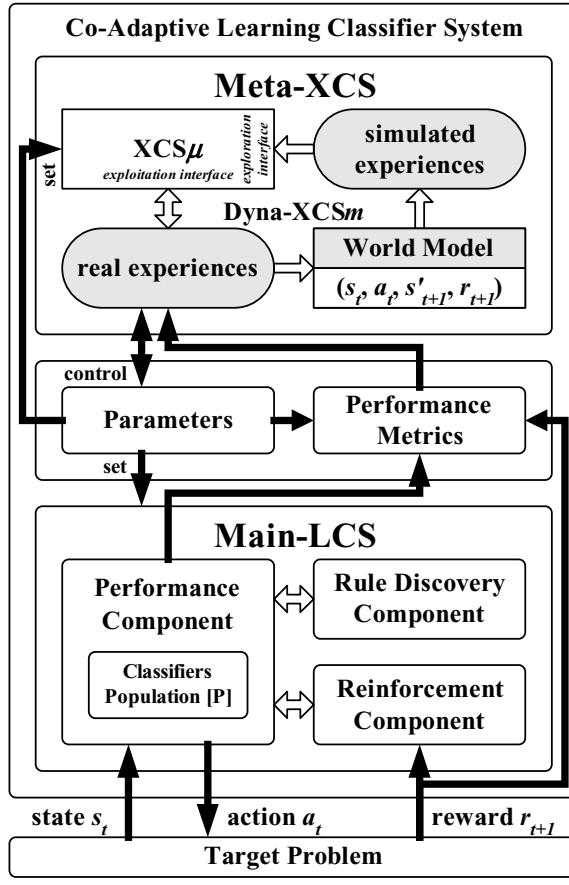


Fig. 3. Detail of co-adaptive learning classifier system architecture

3.3 Meta-XCS Dyna

Dyna uses the XCSμ exploration interface to perform latent learning, and the parameter-control operation is executed through the XCSμ exploitation interface. Theoretically, the latent learning and practical parameter control operations of Meta-XCS can be processed simultaneously, but in practice, a higher priority is assigned to the parameter-control operation in the Meta-XCS component in order to decrease potential conflicts and to meet the hardware restrictions of sequential processing. Therefore, latent learning is delayed until parameter-control operations are fully executed. However, we believe that the arrangement takes advantage of system idling time to improve parameter control and learning performance.

4 Experiments

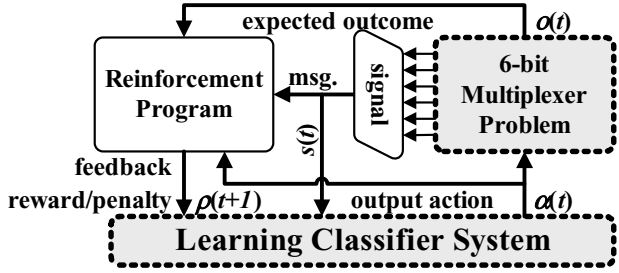
In learning classifier systems, the mutation operator plays an important role in learning performance and target problem solution quality. During the early stages of a run, the mutation operator provides novelty by moving classifiers within the search space. A faster mutation rate helps speed up the rule discovery component. As an essential background operator during the later stages, mutation ensures some probability of finding a better solution. A slower mutation rate helps fine-tune existing classifiers without disturbing runs or decreasing learning performance. However, it is difficult to predetermine the optimal mutation rate for a given target problem or to dynamically adjust the mutation rate during every stage of a run.

4.1 Test Environment: 6-bit Multiplexer Problems

Given the restrictions just described, we experimented with a co-adaptive parameter-control approach in the form of the 6-bit multiplexer problem (6-MP)—a version of the well-known benchmark single-step problem for machine learning in general and learning classifier systems in particular [12]. As shown in Fig. 4, the input message signal transmitted to learning classifier systems consists of a string of six binary digits in which the first (version A) or last (version B) two bits (called *address bits*) represent a binary index and the remaining bits represent data bits. The expected outcome is the value of the indexed data bit. For example, the expected outcome of "11*1110*" in version A is 0, since the first two bits (11) represent index 3—the fourth bit following the address. The expected outcome of "*0100*01" in version B is 1, since the second bit preceding the address is indexed.

6-MP is considered challenging because of its non-linear characteristic, yet it yields many useful generalizations that help in comparing learning performance in various models. During each cycle, the 6-MP produces signals by randomly setting all six bits. Expected outcomes are computed as single bits from the generated signals, which are transmitted as input messages to the learning classifier system on request and returned as output actions that are compared with expected outcomes. A positive feedback score of 1,000 means that a reward was returned to the learning classifier system for reinforcement; a feedback score of 0 means that a penalty was returned. During the run, the 6-MP continues to produce 6-bit messages with similar probabilities as the classifier system tries to learn the correct mapping relations between signals and expected outcomes—thus developing an optimal solution.

With the exception of the mutation rate, the default parameter for our experiments was $N = 800$, $\beta = 0.2$, $\alpha = 0.1$, $\varepsilon_0 = 10$, $\nu = 5$, $\theta_{GA} = 25$, $\chi = 0.8$, $\theta_{del} = 20$, $\delta = 0.1$, $\theta_{sub} = 20$, $P_{\#} = 0.33$, $p_I = 10$, $\varepsilon_I = 0$, $F_I = 0.01$, $p_{explr} = 0.5$, *doGAsubsumption* = true, *doActionSetSubsumption* = true. All results discussed in this report represent an average of ten runs. Learning classifier system performance metrics were recorded for each trial and computed as average moving window numbers in the last 50 trials.



$$o_a(t) = f_{6\text{-}mp}(b_0,b_1,b_2,b_3,b_4,b_5) = \overline{b_0}\,\overline{b_1}\,b_2 + \overline{b_0}\,b_1\,b_3 + b_0\,\overline{b_1}\,b_4 + b_0\,b_1\,b_5 \quad \textit{(version A)}$$

$$o_b(t) = f_{6\text{-}mp}(b_0,b_1,b_2,b_3,b_4,b_5) = \overline{b_4}\,\overline{b_5}\,b_0 + \overline{b_4}\,b_5\,b_1 + b_4\,\overline{b_5}\,b_2 + b_4\,b_5\,b_3 \quad \textit{(version B)}$$

Fig. 4. The 6-bit Multiplexer Problem (6-MP)

4.2 Experiment 1: Stationary 6-bit Multiplexer Problem

We determined average variation in performance metrics at fixed mutation rates of 0.01, 0.05, and 0.09 (Fig. 5), and used our experimental results to observe parameter-setting problems that often occur in XCS. At a high mutation rate (0.09) the population size metric was often poor while performance ρ and system error metrics were good among the three models. At a low mutation rate (0.01) the population size metric was often good but the performance ρ and system error metrics very poor. When applying the co-adaptive learning classifier system to solve the 6-MP, issues tied to learning performance and population state were avoided. As shown In Fig. 5, the performance ρ and system error metrics for the co-adaptive learning classifier system were comparable to or better than those from the XCS at a fixed mutation rate of 0.09. Regarding the population size metric, the co-adaptive learning classifier system metric was similar to or outperformed the XCS metric at a fixed mutation rate of 0.01. After 10,000 trials, approximately 21 classifiers remained for forming an optimal solution.

As shown in Fig. 6, during the early stages of a run the co-adaptive learning classifier system increases its own mutation rate in order to produce classifiers that can solve the 6-MP, and in the later stages decreases the mutation rate in order to fine-tune existing classifiers and to arrive at an optimal solution. This variation in mutation rate continually oscillates between faster and slower. From our observations, it appears as though the co-adaptive learning classifier system tests the influence of a lower mutation rate during the early stages of a run and tests the influence of a higher mutation rate in the later stages. Mutation rates are abandoned if they negatively impact classifier populations or learning performance; in such cases, the system returns to the original rate.

4.3 Experiment 2: Non-stationary 6-bit Multiplexer Problem

Our second experiment was similar to the first except that the second made use of 6-MP versions A and B (Figs. 7 and 8). For this experiment we ran 20,000 trials—10,000 that were similar to the first experiment and 10,000 in which the input signal bit sequence was abruptly changed from version A to B. In the version B run, the two address lines were moved from the initial ($b_0 b_1$) to final input signal bit ($b_4 b_5$) position. Whenever the bit sequence underwent a sudden change during the second 10,000 trials, the co-adaptive learning classifier system had to re-generalize the existing classifiers and rebuild an appropriate solution. The goal was to determine whether or not the co-adaptive learning classifier system could quickly reestablish a proper mutation rate following an abrupt change in the problem environment, recover the original learning performance and population size state, and still rebuild an optimal solution.

Details of co-adaptive learning classifier system performance are presented in Fig. 7. Regardless of the performance metric, the system outperformed XCS at the fixed mutation rates of 0.01, 0.05, or 0.09. At a fixed mutation rate of 0.09, the performance ρ and system error metrics for the co-adaptive learning classifier system outperformed those from the XCS. At a fixed mutation rate of 0.01, the population size metric for the co-adaptive learning classifier system was similar to that from the XCS. An optimal solution aimed at the 6-MP version B was rebuilt after 20,000 trials.

Fig. 8 has two mutation rate peaks, the first before 2,500 trials and the second between 10,000 and 12,400 trials. Each peak reflects the time required by the co-adaptive learning classifier system to learn from the beginning. According to these experimental results, the co-adaptive learning classifier system is capable of handling non-stationary problems at a high performance level.

5 Conclusions

Previous studies of evolutionary computation and learning classifier systems describe the search for robust or optimal parameter sets for target problem solutions as a time-intensive trial-and-error task requiring large amounts of computation resources. Different parameter values are essential for inducing an optimal balance between exploration and exploitation at different run stages. In response to the common problem of setting parameters for practical applications, we extended the original learning classifier system with a parameter-control approach in order to enhance performance and learning stability.

Our proposal for a co-adaptive approach to learning classifier system parameters is based on a coevolution process and a Dyna architecture. The approach takes advantage of the on-line learning capabilities of learning classifier systems; solutions produced in this manner cover entire target problems. Results from our experiments show that the co-adaptive approach was successful in terms of setting parameters according to target problem properties. In both stationary and non-stationary problem experiments, the system outperformed the models it was tested against.

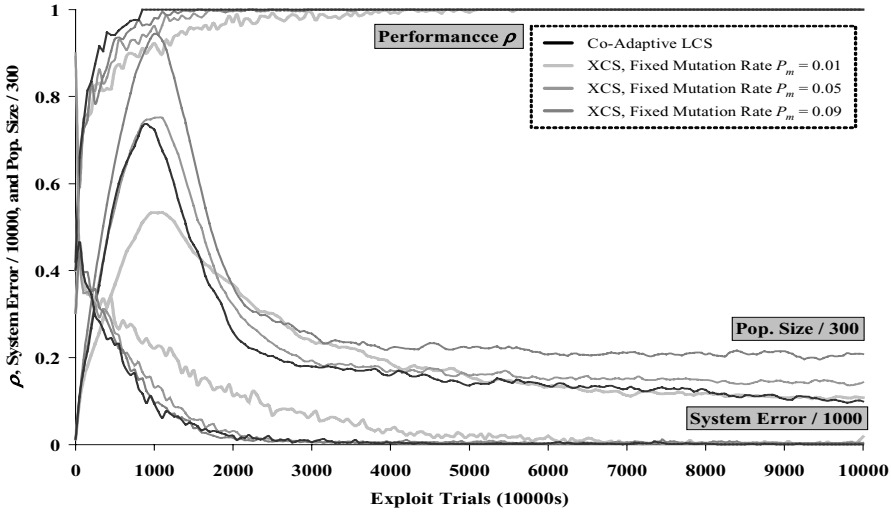


Fig. 5. Performance metrics (performance ρ, system error, and population size) of the co-adaptive learning classifier system and comparative models with fixed mutation rates of 0.01, 0.05, and 0.09 in the stationary 6-MP (version B)

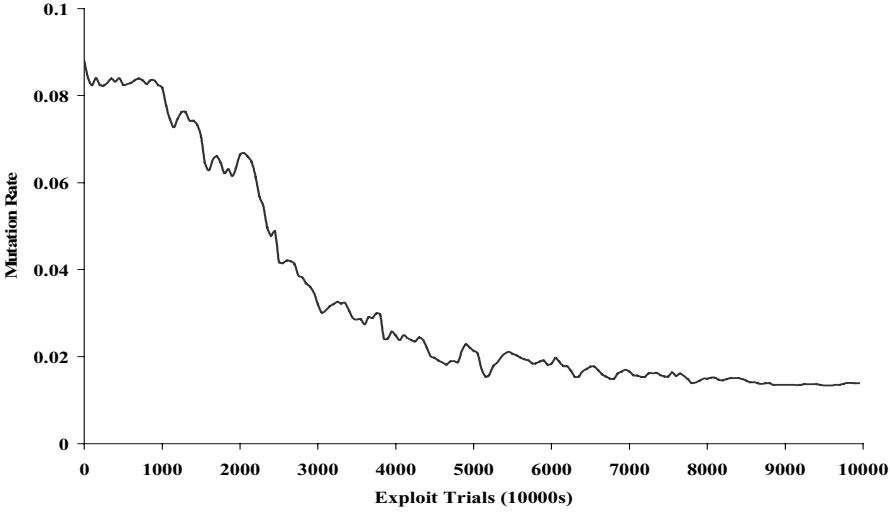


Fig. 6. Mutation rate adaptation for the co-adaptive learning classifier system in the stationary 6-MP (version A)

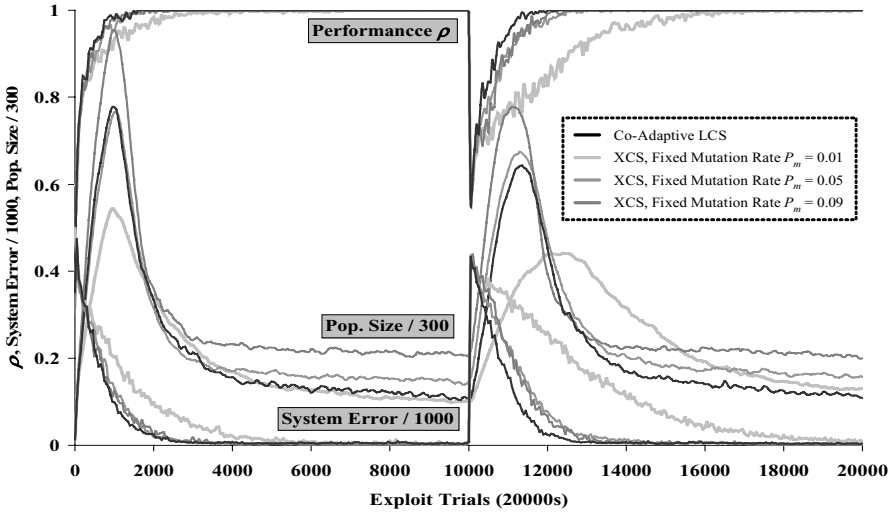


Fig. 7. Performance metrics (performance ρ, system error, and population size) of the co-adaptive learning classifier system and comparative models with fixed mutation rates of 0.01, 0.05, and 0.09 in the non-stationary 6-MP (versions A and B)

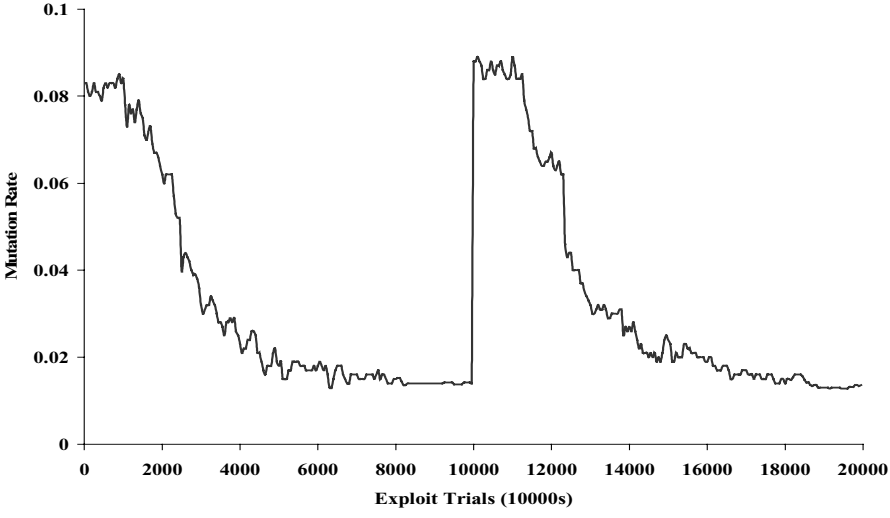


Fig. 8. Mutation rate adaptation for the co-adaptive learning classifier system in the non-stationary 6-MP (versions A and B)

References

1. Eiben, A.E., Hinterding, R., Michalewicz, Z.: Parameter Control in Evolutionary Algorithms. IEEE Transactions on Evolutionary Computation, Vol. 3, No. 2 (1999) 124-141
2. Holland, J.H.: Adaptation in Natural and Artificial Systems. MIT Press (1992); University of Michigan Press, Ann Arbor (1975)
3. Holmes, J.H., Lanzi, P.L., Stolzmann, W., Wilson, S.W.: Learning Classifier Systems: New Models, Successful Applications. Information Processing Letters, Vol. 82 (2002) 23-30
4. Hurst, J., Bull, L.: A Self-Adaptive Classifier System. In: Lanzi, P.L., Stolzmann, W., Wilson, S.W. (eds.) Advances in Learning Classifier Systems. Paris: Springer-Verlag (2001) 70-79
5. Hurst, J., Bull, L.: A Self-Adaptive XCS. In: Lanzi, P.L., Stolzmann, W., Wilson, S.W. (eds.) Advances in Learning Classifier Systems. Berlin: Springer-Verlag (2002) 50-73
6. Kovacs, T.: What Should a Classifier System Learn and How Should We Measure It? Journal of Soft Computing, Vol. 6, Nos. 3-4 (2002) 171-182
7. Lanzi, P.L., Colombetti, M.: An Extension to the XCS Classifier System for Stochastic Environments. In: Banzhaf, W., Daida, J., Eiben, A.E., Garzon, M.H., Honavar, V., Jakiela, M., Smith, R.E. (eds.) Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-99). Morgan Kaufmann (1999) 353-360
8. Lanzi, P.L.: An Analysis of Generalization in the XCS Classifier System. Evolutionary Computation, Vol. 7, No. 2 (1999) 125-149
9. Lanzi, P.L.: Extending the Representation of Classifier Conditions Part I: From Binary to Messy Coding. In: Banzhaf, W., Daida, J., Eiben, A.E., Garzon, M.H., Honavar, V., Jakiela, M., Smith, R.E. (eds.) Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-99). Morgan Kaufmann (1999) 337-344
10. Lanzi, P.L.: Extending the Representation of Classifier Conditions Part II: From Messy Coding to S-Expressions. In: Banzhaf, W., Daida, J., Eiben, A.E., Garzon, M.H., Honavar, V., Jakiela, M., Smith, R.E. (eds.) Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-99). Morgan Kaufmann (1999) 345-352
11. Sutton, R.S., Barto, A.G.: Reinforcement Learning: An Introduction. MIT Press (1998)
12. Wilson, S.W.: Classifier Fitness Based on Accuracy. Evolutionary Computation, Vol. 3, No. 2 (1995) 149-175
13. Wilson, S.W.: Generalization in the XCS Classifier System. In: Koza, J.R., Banzhaf, W., Chellapilla, K., Deb, K., Dorigo, M., Fogel, D.B., Garzon, M.H., Goldberg, D.E., Iba, H., Riolo, R.L. (eds.): Genetic Programming 1998: Proceedings of the Third Annual Conference San Francisco. Morgan Kaufmann (1998) 665-674
14. Glickman, M., Sycara, K.: Reasons for Premature Convergence of Self-Adaptating Mutation Rates. Proceedings of the 2000 Congress on Evolutionary Computation CEC00 California, USA. IEEE Press (2000) 62-69
15. Liang, K.H., Yao, X., Newton, C.S.: Adapting Self-Adaptive parameters in Evolutionary Algorithms. Applied Intelligence, Vol. 15 (2001)
16. Rudolph, G.: Self-Adaptive Mutations May Lead to Premature Convergence. IEEE Transactions on Evolutionary Computation, Vol. 5, No. 4 (2001) 410-414
17. Swain, A.K., Morris, A.S.: Performance improvement of self-adaptive evolutionary methods with a dynamic lower bound. Information Processing Letters, Vol. 82, No. 1 (2002) 55-63
18. Herrera, F., Lozano, M.: Adaptive Genetic Operators Based on Coevolution with Fuzzy Behaviors. IEEE Transactions on Evolutionary Computations, Vol. 5, No. 2, (2001) 149-165