Extended Rule-Based Genetic Network Programming

Xianneng Li
Graduate School of Information, Production and Systems, Waseda University
sennou@asagi.waseda.jp

Kotaro Hirasawa
Graduate School of Information, Production and Systems, Waseda University
hirasawa@waseda.jp

ABSTRACT

Recent advances in rule-based systems, i.e., Learning Classifier Systems (LCSs), have shown their sequential decision-making ability with a generalization property. In this paper, a novel LCS named eXtended rule-based Genetic Network Programming (XrGNP) is proposed. Different from most of the current LCSs, the rules are represented and discovered through a graph-based evolutionary algorithm GNP, which consequently has the distinct expression ability to model and evolve the "if-then" decision-making rules. Experiments on a benchmark multi-step problem (so-called Reinforcement Learning problem) demonstrate its effectiveness.

Categories and Subject Descriptors

I.2 [**Artificial Intelligence**]: Problem Solving, Control Methods, and Search

General Terms

Algorithms

Keywords

learning classifier systems, genetic network programming, XrGNP

1. INTRODUCTION

Learning Classifier System (LCS) [1] is a research branch of evolutionary algorithm (EA) which merges it with different fields of machine learning, i.e., reinforcement learning (RL) and supervised learning, within one whole.

In this paper, a novel LCS named eXtended rule-based Genetic Network Programming (XrGNP) is proposed. Different from the existing LCSs, the rules are formulated based on a graph-based EA – Genetic Network Programming (GNP) [2, 3] – using its distinguished directed graph structures. XrGNP can be considered as a mixture of two msot famous LCSs styles, i.e., Pittsburg-style and Michigan-style, since each individual of XrGNP includes a set of rules as Pitt-style and its rule base is used in Mich-style. However, thanks to the unique and fixed structures of GNP, XrGNP can efficiently model and recombine the rules within its chromosomes and never cause the bloat problem [2] in variable-length GA based Pitt-style and S-expression based LCSs. On the other hand, each chromosome is evaluated as the standard EA, therefore, eventually XrGNP has much simpler fitness design than the Mich-style LCSs.

Copyright is held by the author/owner(s).
GECCO'13 Companion, July 6–10, 2013, Amsterdam, The Netherlands.
ACM 978-1-4503-1964-5/13/07.

2. EXTENDED RULE-BASED GNP

XrGNP and the existing LCSs share much conceptual similarity, however, the major distinction arises from many microscopic parts, including the knowledge representation, knowledge discovery, credit assignment and action selection.

2.1 Knowledge representation

XrGNP originates from a graph-based EA named GNP [2, 3], which shows higher expression ability than that of GA and GP. GNP develops a directed graph structure consisting of judgment/processing nodes to model the complex systems. By separating judgments and processing, GNP can efficiently evolve the compact programs by only selecting the necessary judgments and processing. Such a property of *selection by necessity* exactly makes GNP an efficient knowledge generator. The rules are defined by a sequence of node transitions. A rule consists of a set of successive judgment functions with their results, and a processing function indicating the action.

2.2 Knowledge discovery

The rules are extracted from the *elite* individuals in each generation and saved in the rule-pool, which indicates that the rule-pool is incrementally extracted during evolution, allowing it to potentially obtain much more knowledge. In comparison with the classical LCSs, the evolution of GNP allows XrGNP to perform in more efficient way to recombine the rules since the directed graph omits the # "don't care" symbol in its genotype, which reduces the No. of alphabets to be considered in each attribute.

XrGNP proposes a *niching* GNP (NGNP) to allow more diverse rules discovered. NGNP applies the fitness sharing to find individuals with not just high quality but also distinct structures. In every generation, the raw fitness f_p of individual p is adjusted by: $f'_p = (f_p)^\beta / m_p$, where $\beta \geq 1$ is the scaling factor and m_p is the niche count estimating the crowding of p comparing with the others: $m_p = \sum_{q=1}^{N} \mathcal{S}(d_{pq})$. Here, N is the population size, d_{pq} represents the distance between individual p and q, and $\mathcal{S}$ is the sharing function measuring the similarity between two individuals:

$$\mathcal{S}(d_{pq}) = \begin{cases} 1 - \left(\frac{d_{pq}}{\sigma}\right)^\mu & \text{if } d_{pq} \leq \sigma, \\ 0 & \text{otherwise.} \end{cases} \quad (1)$$

The niche radius σ denotes the threshold of acceptable distance and μ is a constant for regulating the shape of $\mathcal{S}$. To measure the similarity between two individuals, Levenshtein distance is selected as the metric.

With NGNP, XrGNP is allowed to not just discover rules from the single elite individual e, but also from the other individuals with *high raw fitness* and *long distance* between them and e, hence more diverse rules found.

2.3 Credit assignment

XrGNP performs the similar procedure of Mich-style to utilize the knowledge, in which the rules compete, complement and cooperate. A novel RL-based credit assignment method is proposed. This method is inspired by a recent work [4, 5, 3] which has shown the successful ability to identify the quality of node connections of GNP. The state and action space of RL is defined by the branch and nodes of GNP. Therefore, a state-action pair (s, a) corresponds to a node connection from branch s to node a. The updating algorithm of Q values is based on Sarsa Learning (Sarsa) [6] which follows the true experience of the agents to update the Q values. As a result, the good state-action pairs will be rewarded higher Q values and the bad ones will be assigned lower Q values, which allows us to explicitly judge the quality of each (s, a).

Finally, the credit assignment to the rules of XrGNP is performed using the Q values, in which the credit of each rule is the average of the Q values of its node connections.

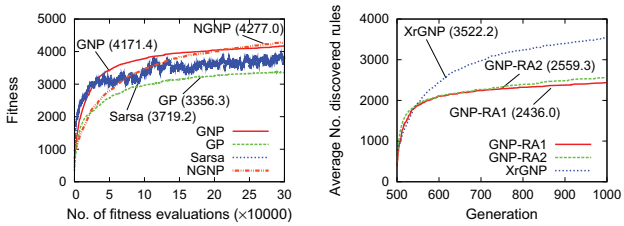
2.4 Action selection

The action selection is carried out by calculating the average matching degrees. First, the match set MS by grouping all the rules whose conditions match the environment d is built. The average matching degree of each action a is calculated by: $m_a(d) = \sum_{r \in MS_a} credit(r)/|MS_a|$. where MS_a is the match set of a. The final action a^* is the one with the highest average matching degree among all actions.

3. SIMULATIONS

XrGNP is applied to a benchmark multi-step problem – Tileworld [3]. Tileworld consisting of a grid of cells on which various objects exist is a widely-used testbed to testify the performance of intelligent agents. In Tileworld, each agent is capable of judging the cells around it and take the actions for movement. The objective of this problem is to find the appropriate controllers that allow the agents to push the tiles into the holes 1) as many as possible and 2) using as fewer steps as possible, or 3) push the tiles towards the holes as close as possible if there are remaining tiles in a limited steps. The details of this problem can be found in [3]. The reported results are the average over 30 independent runs.

The fitness curves of each algorithm are plotted in Fig. 1.(a). To evaluate the performance of knowledge discovery, a simple metric is reported: the average number of extracted rules per generation. Two variants of GNP-RA (an early version of XrGNP with only standard GNP) are selected for comparison with the proposed XrGNP, including the GNP-RA1 only extracting rules from the elite individual e and GNP-RA2 not just discovering rules from e but also the second best individual. Consequently, the effectiveness of NGNP for finding the high-quality diverse individuals can be verified. The results in Fig. 1.(b) confirm that with NGNP, XrGNP performs in the most efficient way to discover rules.



(a) Fitness curves (b) Discovered rules

Figure 1: Results of evolution

Table 1: Generalization ability

	Tileworld		
	Fitness±std.dev.	Droped Tiles	t-test
GNP	552.0 ± 860.1	3.8 ± 5.1	**3.3e-06**
GP	219.0 ± 830.0	1.6 ± 4.6	**2.1e-09**
Sarsa	324.4 ± 594.8	2.3 ± 3.3	**5.7e-08**
GNP-RA	1291.6 ± 701.2	7.3 ± 4.7	**2.8e-03**
XrGNP	1989.6 ± 601.2	10.3 ± 3.9	—
w/o niching	1439.2 ± 801.1	8.3 ± 3.9	**1.5e-02**
w/o credit	1533.2 ± 641.9	8.6 ± 3.9	**4.3e-02**

The final objective is to perform the generalization ability of XrGNP. The knowledge base evolved from the training environments is applied to Tileworld with new environments. The final results are reported in Table 1. From the results it is found that XrGNP performs the highest generalization ability than the others. The t-test results show the statistical difference between XrGNP and the others.

4. CONCLUSIONS

A mixture of Pitt- and Mich-style LCSs has been successfully proposed in this paper. By formulating its chromosome as a directed graph, XrGNP shows the powerful knowledge discovery ability. The proposed system is successfully applied to solve the multi-step RL problems including the benchmark one and a real-world case.

5. REFERENCES

[1] John H. Holland and Judith S. Reitman. Cognitive systems based on adaptive algorithms. *SIGART Bull.*, (63):49–49, June 1977.

[2] K. Hirasawa, M. Okubo, H. Katagiri, J. Hu, and J. Murata. Comparison between genetic network programming (GNP) and genetic programming (GP). In *Proc. of the IEEE Congress on Evol. Comput.*, pages 1276–1282, 2001.

[3] X. Li, S. Mabu, and K. Hirasawa. A novel graph-based estimation of distribution algorithm and its extension using reinforcement learning. *IEEE Trans. Evol. Comput.*, 2013. (early access).

[4] X. Li, B. Li, S. Mabu, and K. Hirasawa. A novel estimation of distribution algorithm using graph-based chromosome representation and reinforcement learning. In *Proc. of the IEEE Congress on Evol. Comput.*, CEC '11, pages 37–44, 2011.

[5] X. Li, S. Mabu, and K. Hirasawa. Use of infeasible individuals in probabilistic model building genetic network programming. In *Proc. of the Conf. on Genetic and Evol. Comput.*, GECCO '11, pages 601–608, 2011.

[6] R. S. Sutton and A. G. Barto. *Reinforcement Learning: An Introduction*. MIT Press, 1998.