A Comparison of Hybrid Incremental Reuse Strategies for Reinforcement Learning in Genetic Programming

Scott Harmon, Edwin Rodríguez, Christopher Zhong, and William Hsu

Department of Computing and Information Sciences
Kansas State University
{sjh4069,edwin,czh9768,bhsu}@cis.ksu.edu

Easy missions is an approach to machine learning that seeks to synthesize solutions for complex tasks from those for simpler ones. ISLES (Incrementally Staged Learning from Easier Subtasks) [1] is a genetic programming (GP) technique that achieves this by using identified goals and fitness functions for subproblems of the overall problem. Solutions evolved for these subproblems are then reused to speed up learning, either as automatically defined functions (ADF) or by seeding a new GP population. Previous positive results using both approaches for learning in multi-agent systems (MAS) showed that incremental reuse using easy missions achieves comparable or better overall fitness than single-layered GP. A key unresolved issue dealt with hybrid reuse using ADF with easy missions. Results in the keep-away soccer (KAS) [2] domain (a test bed for MAS learning) were also inconclusive on whether compactness-inducing reuse helped or hurt overall agent performance. In this paper, we compare reuse using single-layered (with and without ADF) GP and easy missions GPs to two new types of GP learning systems with incremental reuse.

In our research we performed six experiments. The first experiment used standard, conventional GP without any enhancement. We will refer to this as single-layered GP. The second used using standard GP enhanced with ADF. We will refer to this as single-layered ADF. The rest of the experiments used double-layered (two stages of evolution). The third used ISLES with Standard GP in the first and second stage. We will refer to this as ISLES - SGP/SGP. The fourth used ISLES with Standard GP in the first stage and ADF in the second stage. We will refer to this as ISLES - SGP/ADF. The fifth used ISLES with ADF in the first stage and Standard GP in the second stage. We will refer to this as ISLES - ADF/SGP. The sixth and final experiment used ISLES with ADF in the first and second stage. We will refer to this as ISLES - ADF/ADF.

Each experiment was done using ECJ [3] and a KAS simulator created by S. Gustafson [1]. For both single-layered experiments, the target concept was to minimize the number of turn overs. For all of the experiments with ISLES, the first stage goal was to maximize the number of successful passes between two teammates in the absence of takers. The second stage goal was to minimize the number of turnovers from keepers (3 keepers) to takers (1 taker).

We took the average of ten runs for each experiment. The population size for all the experiments was 4000. For the single-layered experiments, we stopped at the 101th generation. For the ISLES experiments we stopped the first stage at the 10th generation and the second stage at the 90th generation. When going from the first stage to the second stage, we used the individuals from the first stage to seed the population of the second stage. For ISLES -SGP/ADF that involved putting the GP trees into the ADF portions and initializing the main trees to random sequences. For ISLES - ADF/SGP, it involved taking the ADF trees and placing then into the main trees of the GP (discarding the main trees of the ADF). For the others a simple transfer of the individuals was performed.

The results we obtained show evidence of the possible existence of a well defined spectrum of behaviors that ranges from techniques that use compactness inducing reuse (such as ADF) to those that use totally dynamic reuse (ISLES). This spectrum is defined by a trade off between high fitness and space efficiency. Our preliminary results allow us to hypothesize that for problems that surpass a certain threshold of complexity, compactness inducing techniques, like ADF, will tend to yield solutions that are far more efficient in terms of space. However, this space saving comes at the cost of suboptimality in terms of fitness. This conjecture is very interesting and poses several questions: is this phenomenon problem independent? Is this phenomenon independent of the design of the system, parameter tuning, inductive bias and other characteristics of the underlying evolutionary engine? Unfortunately, given the premature nature of our results and lack of diversity in terms of problems studied (as well as some uncertainty introduced by the small amount of repetitions), it is impossible, for now, to answer these questions. However, despite the weakness of these preliminary results in terms of support for our conjecture, the issue posed herein is a very interesting one and we believe that it should beintegrated into the current efforts of our research community.

The ideas presented in this work have a potential effect on other well developed areas such as Schema Theory and Code Bloat Theory. Our long term goal is to develop a theory that characterizes the behavior of GP in the presence of different types of reuse. This theory should have both a qualitative and quantitative value. In this sense it should not only give insight about how reuse affect the way GP searches for problems, but also serve as a tool to determine under what conditions reuse is beneficial, what kind of reuse should be applied or if it should be avoided altogether.

References

1. Hsu, W.H., Gustafson, S.M.: Genetic programming and multi-agent layered learning by reinforcements. In: GECCO 2002: Proceedings of the Genetic and Evolutionary Computation Conference, New York, Morgan Kaufmann Publishers (2002) 764–771
2. McAllester, D., Stone, P.: Keeping the ball from CMUnited-99. In: RoboCup-2000: Robot Soccer World Cup IV. Springer Verlag, Berlin (2001)
3. Luke, S.: Issues in Scaling Genetic Programming: Breeding Strategies, Tree Generation, and Code Bloat. PhD thesis, Department of Computer Science, University of Maryland, A. V. Williams Building, University of Maryland, College Park, MD 20742 USA (2000)