Chapter 5

Efficient Evolution of Modular Robot Control via Genetic Programming

Tüze Kuyucu
Doshisha University, Japan

Ivan Tanev
Doshisha University, Japan

Katsunori Shimohara
Doshisha University, Japan

ABSTRACT

In Genetic Programming (GP), most often the search space grows in a greater than linear fashion as the number of tasks required to be accomplished increases. This is a cause for one of the greatest problems in Evolutionary Computation (EC): scalability. The aim of the work presented here is to facilitate the evolution of control systems for complex robotic systems. The authors use a combination of mechanisms specifically designed to facilitate the fast evolution of systems with multiple objectives. These mechanisms are: a genetic transposition inspired seeding, a strongly-typed crossover, and a multiobjective optimization. The authors demonstrate that, when used together, these mechanisms not only improve the performance of GP but also the reliability of the final designs. They investigate the effect of the aforementioned mechanisms on the efficiency of GP employed for the coevolution of locomotion gaits and sensing of a simulated snake-like robot (Snakebot). Experimental results show that the mechanisms set forth contribute to significant increase in the efficiency of the evolution of fast moving and sensing Snakebots as well as the robustness of the final designs.

DOI: 10.4018/978-1-4666-4225-6.ch005
1. INTRODUCTION

Genetic Programming (GP) is a methodology specifically developed for the evolution of computer programs that perform predefined tasks (Koza, 1994). It is a powerful representation for automatically creating computational systems via simulated evolution, and it has produced many human-competitive results (Koza, 2003). The use of GP for the automated design of artificial systems is a popular and promising method; it provides a valuable approach to achieving designs that are too complex or alien for human designers (such as the automated control of a racing car (Schichel & Sipper, 2011)).

Several examples of real-life designs obtained via GP can be found in literature, however, as we try to tackle harder problems with GP the inherent weaknesses in GP design become more apparent. The exponential growth of the search space with increasing complexity of the target design, and poor locality are only a couple of these weaknesses that present a challenge to GP design of real-life systems. Hence the design of systems with the use of GP is generally a time consuming process. The expensive evaluation times of real-life problems, such as physics, 3D graphics, and electronic circuit simulations, when multiplied by the large population sizes and the number of evolutionary generations required to obtain a result makes GP design a slow process. Furthermore, the designs obtained via GP can often end up overly fit, low quality solutions that fail to generalize. If used in real-world applications, a design must be reliable in terms of accurate functionality, and ensuring this can be a costly process in GP; multiple GP runs may be required until designs that are reliable can emerge. Therefore, the convergence of GP runs to a desired solution within a given time is important. Situations where the aim is to come up with designs with multiple features can especially present a challenge; the GP may fail to identify and preserve sub-solutions, and successfully build upon these partial solutions in order to provide the optimal design.

Poor scalability is a common problem in the simultaneous evolution of multiple features of a target system, this is because of the growing search space of evolution which increases faster than linearly with the increase of the number of simultaneously evolved features. In order to tackle the scalability problem, several approaches to the evolutionary design of systems with multiple features or objectives have been devised. However, achieving good scalability in the evolution of complex systems still remains to be one of the great challenges (McConaghy, Vladislavleva, & Riolo, 2010).

On the other hand, ensuring the evolution of reliable results in a timely manner is another persisting challenge present in GP design. Due to the opportunistic nature of evolution, the resulting designs achieved via GP can often be something other than what is desired. The design approach used by evolutionary designs exploit all the available properties of an evolutionary platform and environment. Due to this, the results can sometimes perform unexpectedly (good or more often bad (Bird & Layzell, 2002; Kuyucu, Trefzer, Greensted, Miller, & Tyrrell, 2008)).

In this paper we introduce mechanisms that will help to tackle these two great challenges. We show the effects of these new mechanisms by evolving designs for a challenging application, which will be described in Section 2. The computational effort of GP could be improved in several ways, such as, incorporating a domain-specific knowledge into the key attributes of GP (e.g., genetic representation, genetic operations, etc.), imposing problem-specific syntax constrains (i.e., grammar) on the evolved genetic programs, employing probability-distribution models, etc. These approaches are usually intended to steer the simulated evolution towards the most promising areas in the explored (presumably rugged) fitness landscapes. Another approach of improving computational effort of GP stems from the assumption that, the main genetic operations (crossover and mutation), due to their randomness, can be damaging the already existing building blocks of