Reactive and Memory-Based Genetic Programming for Robot Control

Björn Andersson, Per Svensson, Peter Nordin, and Mats Nordahl

Complex Systems Group, Institute of Physical Resource Theory, Chalmers University of Technology, S-412 96 Göteborg, Sweden, Email: nordin,tfemn@fy.chalmers.se

Abstract. In this paper we introduce a new approach to genetic programming with memory in reinforcement learning situations, which selects memories in order to increase the probability of modelling the most relevant parts of memory space. We evolve maps directly from state to action, rather than maps that predict reward based on state and action, which reduces the complexity of the evolved mappings. The work is motivated by applications to the control of autonomous robots. Preliminary results in software simulations indicate an enhanced learning speed and quality.

1 Introduction

The field of autonomous mobile robotics attracts an accelerating interest. Autonomous robots that use legs instead of wheels are useful both in environments created for humans and in more natural terrain. Humanoid robots are particularly interesting, since man is the standard for almost all interactions in the world, and most environments, tools and machines are adapted to our abilities, motion capabilities and geometry.

An autonomous mobile robot system demands high performance both of its mechanical components and control software. The many degrees of freedom in a light mobile robot create new problem domains in control and navigation where conventional methods often fall short. Control of autonomous robots is a promising application area for evolutionary algorithms.

We have attempted to use genetic programming (GP) to control autonomous robots. In particular, this article describes a new approach, called reactive GP with memory, to using genetic programming in a reinforcement learning [8] situation, where an explicit world model is used as part of the learning algorithm.

In the work on robots, this algorithm is part of a larger software structure for learning, control and planning based on genetic programming. This hierarchical structure consists of three layers: a reactive layer, a model building layer, and a reasoning layer. These layers represent different levels of control and consciousness.

– The first layer is a reactive layer based on on-line evolution of machine code.

This assumes that all fitness feedback is obtained directly from the physical robot, which means that the evolved programs often spend most of their time
waiting for feedback from the physical environment during their evaluation. This limits the speed of learning, and the constant movement also shortens the life span of the hardware. On the other hand the method is simple, and operates without any need to introduce a priori assumptions about the problem domain. This layer is used for reactive behaviours like balancing.

- The second layer is a model building layer, which works with memories of past events. The system tries to evolve a model of the environment as well as the robot itself. The model could for example map sensor inputs and actions to a predicted reward. The best model at a certain instant is then used to determine the action which results in the best predicted fitness given current sensor inputs. In this way, the genetic programming system can run at full speed without having to wait for feedback from the environment, instead fitting the programs based on memories of past events. The model building layer is also used for basic control tasks.

- The third layer, the reasoning layer, is a symbolic processing layer for higher brain functions that require reasoning, such as navigation, safety, and energy supply. This layer is built on genetic reasoning, a method where evolution is used as inference engine [5].

The algorithm introduced in this article is primarily relevant for the model building layer. In earlier work, we have used genetic programming to evolve a model of registered memory events [6]. This model was then used to find the action with the best predicted outcome. The problem with this approach is the balance between exploration and exploitation, in that the GP system spends a lot of effort trying to model the entire memory space, while we are primarily interested in the parts with high predicted rewards. So while a symbolic model of the memories may seem to correspond very well to the actual memories it may not accurately model the important region of best reward, where we would prefer the system to spend most of its time. This region is often minute compared to other recorded reward signals. However, if the structure of the best reward is lost then it is usually impossible for the agent to make reasonable decisions. The effect of the phenomenon in realistic applications is that we either fail to map the important part of the memory space, or at least spend too much time mapping its less interesting regions.

In this paper we present novel work on how to select memories in order to increase the probability of modelling the relevant parts of memory space. The basic principle is to increase the probability of storing memory instances with a favorable reward value. The approach also differs from previous work on genetic programming with memory by evolving maps directly from state to action, rather than maps that predict reward based on state and action. This reduces the complexity of the evolved mappings. Preliminary results in software simulations indicate an enhanced learning speed and quality.