Evolution in Simulation – A Comparison

Joanne Walker and Myra Wilson

Computer Science Department,
University of Wales, Aberystwyth
e-mail jhw94@aber.ac.uk

5.4.2001

Abstract

Evolutionary techniques are often used in the design of mobile robot controllers. When carried out on real robots, evolution is time consuming so it is often done in simulated worlds. However, the problems surrounding the use of simulators are well known, and much work has gone into improving and validating these simulations. This paper describes work that tests a *methodology* of evolution in simulation described in previous work by Ram *et al.* The work is recapitulated on a different simulator for a different robot, and the results of the first set of experiments, from the first group of world types are presented. In comparing the success of this project with the previous, the indications are that the method can transfer successfully to a completely different robot.

1 Introduction

Evolutionary techniques are often used to help in the design and optimisation of mobile robot controllers as building and fine tuning behaviours by hand is a difficult task. The interactions between the robot and its environment are hard to predict, and as the complexity of systems and environments increase with more sophisticated robots, it becomes increasingly harder (Husbands *et al.*, 1996).

Researchers at Georgia Institute of Technology (Ram *et al.*, 1994; Pearce *et al.*, 1992) have used genetic algorithms (GA's) to optimise behaviours for a robot that navigated through an unconstrained environment to a goal. Evolution was carried out in a simulated set of worlds to produce a set of behaviours suited to different environments or niches.

Due to the intensive nature of evolutionary techniques, which typically need to run for a long time, they are usually carried out in simulation, as they were in (Ram *et al.*, 1994). However, the problems of working in simulation are well known, the simulated world and robot simply cannot perfectly match reality. In evolution this is partcularly significant as the evolutionary algorithm can take advantage of flaws in the simulation and evolve solutions based on these flaws (Mataric & Cliff, 1996). For this reason it is important to validate the results of evolution in simulation by testing them on a real robot.

The behaviours produced in (Ram *et al.*, 1994) were only tested in further simulated worlds, and although Ram *et al* state that the simulator used was one that has been proven to produce control systems that transfer well to a real robot in the past, the work still needs to be validated on a real robot.

The work reported in this paper is part of a larger project to validate and extend the work of Ram *et al.* This larger project hopes to use behaviours evolved in a standard, off the shelf simulation as the basis for further evolution on a real robot as it moves through new environments. The first stage, reported here, was to recapitulate the work of Ram *et al.* A genetic algorithm was used to evolve schema-based navigation behaviours, using the same methodology as reported in Ram *et al.* In this paper the results of the first set of simulation experiments are described.

Proc. TIMR 01 - Towards Intelligent Mobile Robots, Manchester 2001.
Technical Report Series, Department of Computer Science, Manchester University, ISSN 1361 - 6161.
Report number UMCS-01-4-1. http://www.cs.man.ac.uk/cstechrep/titles01.html

2 Background

Behaviour-based robotics was introduced as a new approach to the problem of controlling a mobile robot by Brooks in 1986 (Brooks, 1986b; Brooks, 1986a). Since then it has been widely used and many diverse architectures have been designed using the behaviour-based paradigm. One of these, designed by Arkin in 1989 (Arkin, 1989), is called the schema-based architecture where individual behaviours are encoded as schemas. The schema-based architecture has been incorporated into an overall control system called AuRA by workers at Georgia Institute of Technology (Arkin & Balch, 1997) and has been shown to work effectively on a wide number of robots and projects. One of these projects, the one this work follows, involved using a genetic algorithm to evolve the schemas (Ram *et al.*, 1994; Pearce *et al.*, 1992).

The advantage of using the schema-based architecture in evolution is that it is possible to manipulate the individual reactive behaviours and not just some abstraction of them. This is because the schemas are equations with a number of parameters and it is these parameters that are altered by the genetic algorithm (see Methodology in Section 3).

In other work using genetic algorithms to evolve behaviours, it has been an abstraction of the behaviours that has been evolved, rather than the individual behaviours themselves. For instance in the SAMUEL project, (Schultz & Grefenstette, 1992; Grefenstette *et al.*, 1990), a GA was used to learn the reactive behaviours defined by stimulus response rules in simulation, before being tested on a real robot. PRESTON (Wilson, 2000; Wilson, 1998; Wilson *et al.*, 1997), evolved complex, abstracted behaviours from combinations of low level behaviours using a form of genetic algorithm. The behaviours were evolved for navigation through a grid maze. PRESTON split the process of evolution into distinct phases wih some phases in simulation and some on a lego robot. Learning classifier systems (LCS's) are another approach. These use genetic algorithms to learn rules, or classifiers, and to evolve better classifiers. ALECSYS (Dorigo & Schnepf, 1993; Colombetti & Dorigo, 1992) is a system that combines LCS's and the behaviour-based approach in the control of a mobile robot. Evolution was again carried out variously on real robots and in simulation.

A large amount of research has been undertaken that looks at ways to evolve behaviours for specific robots , mainly in simulation. Much work has also been done to find ways to make the behaviours evovled in simulation transfer successfully to the real world. However, no work has been found which attempts to assess the effectiveness of the *method of evolution* when it is applied to different simulations for different robots. This work assesses a method of simulation in terms of how well it generalises to a different robot simulation.

3 Methodology

The methodology for this project followed that used in (Ram *et al.*, 1994) as closely as possible. The only variation was due to differences between the simulated worlds and environments, and any possible effects of these differences are discussed in this section.

3.1 Simulation

A three-dimensional simulated world was used for the evolution of the behaviours, see Figure 1. The Webots software (made by Cyberbotics, www.cyberbotics.com) simulates a khepera robot with infra-red sensors that can detect obstacles and light sources. The robot task was to move from a start position to a light source goal in as few steps and shortest distance possible whilst avoiding collisions with obstacles.

The robot was controlled by three behaviours - **Move-To-Goal, Avoid-Static-Obstacle** and **Noise** which were controlled by five parameters - goal gain, obstacle gain, obstacle sphere of influence, noise gain and noise persistence. The schemas are essentially equations, as shown below, from (Arkin, 1989).

Avoid-Obstacle

$$
\begin{aligned}
O_{magnitude} &= 0 \quad \text{for} \quad d > S \\
& \frac{S-d}{S-R} * G \quad \text{for} \quad R > d \leq S \\
& \infty \quad \text{for} \quad d \leq R
\end{aligned}
$$

$$0_{direction} = \quad \text{along a line away from the center of the obstacle}$$

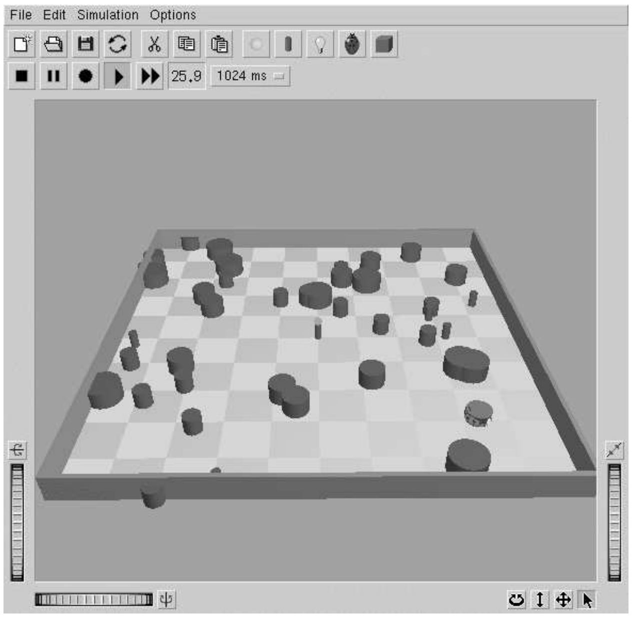

Figure 1: Webots - 3D simulator for the Khepera robot

where:

$$\begin{array}{rcl} S & = & \text{sphere of influence (from the center of the obstacle)} \\ R & = & \text{radius of the obstacle} \\ G & = & \text{gain} \\ d & = & \text{distance of robot to center of obstacle} \end{array}$$

Move-To-Goal

$$\begin{array}{rl} V_{magnitude} = & \text{fixed gain value} \\ V_{direction} = & \text{in direction towards perceived goal} \end{array}$$

Each schema produces a vector which describes the direction and the speed the robot should move in. These vectors are summed and normalised to give the final step the robot takes. This summing and normalisation means that the *ratio* of the evolved parameters is what is important rather than their actual values.

In the simulation, populations of robots were used to test the population of chromosomes in each generation. This paper reports on the results from the first set of experiments where the robots were trained in worlds with 1% obstacle coverage (or clutter). There were nine worlds in total which differed in terms of the changeability of the environment and the robot strategy:

Fixed The amount and locations of the obstacles remain the same for the duration of evolution

Varying The amount of clutter is fixed but the locations of the obstacles change between runs

General The amount and locations of obstacles change between runs

Each of the world types were run with three different optimising criteria which determined whether the robots prioritised safety, speed in reaching the goal, or distance travelled to the goal. A range of environments was chosen so that the system would be able to work more effectively in new environments after the training session.

3.2 Genetic Algorithm

The genetic algorithm used in (Ram *et al.*, 1994) differed from a typical GA in a number of ways. The most significant difference was that a floating point representation was used rather than the usual bit string. The chromosomes were made up of five numbers, whose values corresponded directly to the value of the five schema parameters. The genetic operators reproduction, crossover and mutation were used. Reproduction simply copied a proportion of the best individuals (chosen by the roulette method) replacing the worst individuals (also chosen by roulette). In crossover, two chromosomes were copied and a single crossover point chosen. At the crossover point, the children's gene values were calculated as:

$$newGene = average + difference * random$$

Using the average of the two genes at the crossover point, the difference between them, and a random number between 0 and 1. Mutation occurred on single genes, and again the new value of the gene was given by an equation:

$$newGene = oldGene + oldGene * random * mutationDelta$$

where the mutationDelta was set to 0.5.

Two changes were deliberately made to the methodology used in (Ram *et al.*, 1994): the mutation rate was reduced, and the number of generations run was increased. For an explanation of these changes, see the Results and Discussion (Section 4).

The optimisation results were compared to those reported by Ram *et al* to show how well Ram *et al's* methodology applies to a different robot and simulator. The final evolved parameter values were compared to see whether it is possible to predict the schema parameters for a given environment, without evolving them.

4 Results

4.1 Comparison of optimisation

When the GA was configured exactly the same as reported in (Ram *et al.*, 1994), no improvement in chromosome fitness was seen over the course of the GA. The mutation rate they used was very high, at 0.5 (i.e. half of all chromosomes were mutated in each generation). When this was changed to a more typical rate of 0.05 optimisation improved significantly. In fact the percentage improvement in fitness values between the first and final generations was better than that seen by Ram *et al.*

Although the optimisation in seen these experiments is greater than that seen by Ram *et al*, the GA took more generations to reach a steady, low fitness: up to 250 generations, compared to about 30. This difference in optimisation rate could be due to their having more experience of the robot simulator and the schema architecture, meaning that they could initialise the GA with better values, giving it an immediate advantage. This would also explain why these experiments saw a much better improvement in fitness over the GA. In starting off with worse parameter values, there was much more room for improvement.

The levels of optimisation seen varied significantly between the different world types. Although some experiments showed clear optimisation over the 500 generations, others saw minimal improvement in fitness, if any, and distinct patterns governing the success of the GA were evident. When comparing worlds that were the same apart from the amount the changeability of the environment, robots in fixed worlds tended to optimise less well than those in general and varying worlds. In Figure 2, it is clear that the optimisation in the fixed world is less successful than that in the other two worlds. This may be because the solutions evolved for the fixed worlds were more brittle than those in the more changeable ones.

The strategies - safe, fast or direct - which were evolved for made an enormous difference to the level of optimisation seen. Robots evolved for safety and speed both optimised much better than those evolved for directness, as can be seen from Figure 3 which shows an example of a direct robot improving less than the safe and fast ones.

Further analysis considered the individual parts of the fitness function - the safety, speed and directness. It was found that directness was the most difficult thing to optimise. In robots optimised for safety and speed the directness often did not improve at all. See Figure 4 for an example of this in a robot optimised for speed. Only in robots specifically trying to optimise directness did it improve and then only marginally. This was also at the expense of safety and speed which both failed to optimise at all in direct worlds.

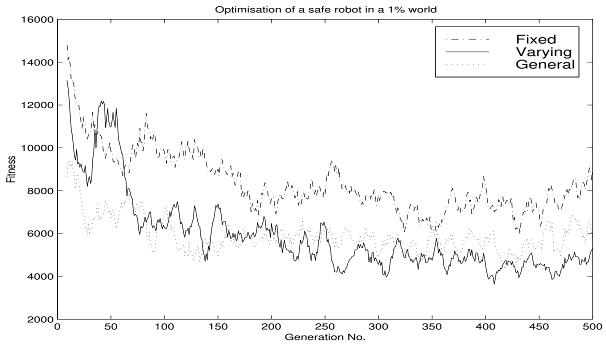

Figure 2: COMPARISON BETWEEN THE MOVING AVERAGES OF THE FITNESS OF THE BEST CHROMOSOMES IN THREE DIFFERENT WORLD TYPES

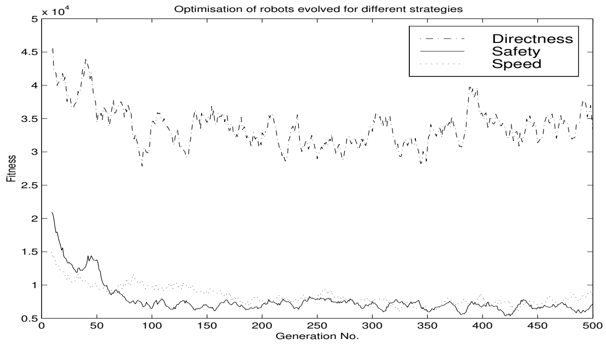

Figure 3: COMPARISON BETWEEN THE MOVING AVERAGES OF THE FITNESS OF THE BEST CHROMOSOMES FOR ROBOTS EMPLOYING THREE DIFFERENT STRATEGIES, IN A FIXED 1% WORLD

4.2 Comparison of parameter values

The final schema parameter values evolved by the GA were found to be significantly different to those reported in (Ram *et al.*, 1994), suggesting that it would not be straightforward to predict them without doing any evolution. The main difference was that the noise gain parameter, which amplifies the random wandering behaviour of the robot, is very high compared to Ram *et al's* findings. This disparity could be due to implementation differences between the simulated worlds. For instance, Ram *et al's* robot seems to have been able to see the goal from all parts of its world, whereas the robot in these experiments cannot; this could make a random wandering behaviour important when the robot moves out of range of the light source. This comparison between final evolved parameters shows that even small differences in the environment, or the *simulation of the environment*, may make large differences to the evolution of behaviours. This means that as well as being difficult to predict good parameter values, it is important that the results of evolution in simulation are tested on a real robot.

5 Discussion and Conclusions

The results gained in these experiments vary from those reported in (Ram *et al.*, 1994) in two ways:

1. the success of the GA varies significantly depending on the world type and strategy the robot is being optimised for
2. the actual parameter values evolved are completely different

Together with the fact that some of the GA parameters had to be changed to see any improvement in fitness, this shows that the method used in (Ram *et al.*, 1994) to evolve behaviours does not transfer

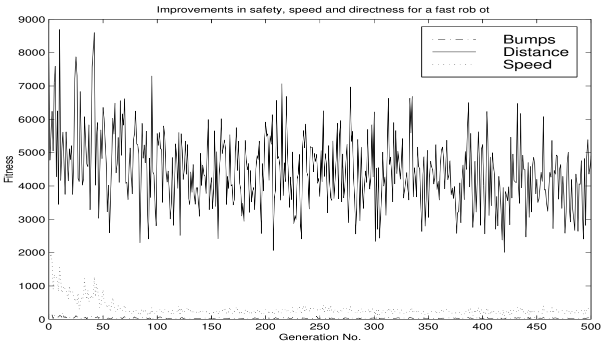

Figure 4: COMPARISON BETWEEN THE IMPROVEMENTS IN SAFETY, SPEED AND FITNESS FOR ROBOTS OPTIMISING SPEED IN A 1% GENERAL WORLD

perfectly to another robot and simulation. However, the method has had much success. In most of the worlds, the GA did improve the fitness of the behaviours. To show whether this provides a basis for a generic methodology for evolving behaviours in simulation would need more experimentation on more simulators, although the indications are that it could.

The fact that the parameter values differ significantly between these results and those reported in (Ram *et al.*, 1994) suggest that it would not be possible to predict the values needed for different robots in a given world.

Before this methodology could become the basis for a general approach to behaviour evolution, more generality of the method needs to be shown, both in simulated experiments, and on real robots. It has also been shown that small differences in the world and the robot make large differences to the path evolution takes, so that is difficult to successfully predict the behaviour parameters without any evolution.

6 Further Work

The next stage in this project is to extend the simulation experiments to cover the remaining world types used in (Ram *et al.*, 1994). This will mean training the robot in 10% and 25% clutter environments and comaparing the results to those from the 1% world. It is also important that the results of the evolution are validated by testing them on a real robot. This testing also show how much further work is needed to adapt to the real world. A method for adapting to new environments as they are encountered will then be developed.

References

Arkin, R.C. 1989. Motor schema–based mobile robot navigation. *International journal of robotics research*, **8**(4), 92 – 112.

Arkin, R.C., & Balch, T. 1997. Aura: Principles and practice in review. *Journal of experimental and theoretical artificial intelligence*, **9**, 175 – 189.

Brooks, R.A. 1986a. *Achieving artificial intelligence through building robots.* A.I. Memo 899. Massachusetts Institute of Technology Artificial Intelligence Lab.

Brooks, R.A. 1986b. A robust layered control system for a mobile robot. *Ieee journal of robotics and automation*, **2**(1), 14–23.

Colombetti, M., & Dorigo, M. 1992. Learning to control an autonomous robot by distributed genetic algorithms. *In: From animals to animats 2, proceedings of the 2nd international conference on the simulation of adaptive behaviour.* Cambridge, Massachusettes: Bradford Books, MIT Press.

Dorigo, M., & Schnepf, U. 1993. Genetics-based machine learning and behaviour based robotics: a new synthesis. *Ieee transactions on systems, man, and cybernetics*, **23**(1), 141 – 154.

Grefenstette, J., Ramsey, C., & Shultz, A. 1990. Learning sequential decision rules using simulation models and competition. *Machine learning*, **5**(4), 355 – 381.

Husbands, P., Harvey, I., Cliff, D., Thompson, A., & Jakobi, N. 1996. The artificial evolution of control systems. *Pages 41 – 49 of: Proceedings of the 2nd international conference on adaptive computing in engineering design and control.*

Mataric, M.J., & Cliff, D. 1996. Challenges for evolving controllers for physical robots. *Robotics and autonomous systems*, **19**(1), 67–83.

Pearce, M., Arkin, R.C., & Ram, A. 1992. The learning of reactive control parameters through genetic algorithms. *In: Ieee/rsj international conference on intelligent robots and systems.*

Ram, A., Arkin, R.C., Boone, G., & Pearce, M. 1994. Using genetic algorithms to learn reactive control parameters for autonomous robotic navigation. *Adaptive behaviour*, **2**(3), 277–304.

Schultz, A., & Grefenstette, J. 1992. Using a genetic algorithm to learn behaviours for autonomous vehicles. *In: Proceedings of the aiaa guidance, navigation and control conference.*

Wilson, M.S. 1998. Evolution of behaviour sequences for real robots. *Pages 105 – 115 of: Proceedings of the 7th european workshop on learning robots.*

Wilson, M.S. 2000. Preston: A system for the evaluation of behaviour. *Chap. 9 of:* Demmiris, J., & Birk, A. (eds), *Interdisciplinary approaches to robot learning*, vol. 24. Vol 24 of Series In Robotics and Intelligent Systems.

Wilson, M.S., King, C., & Hunt, J. 1997. Evolving hierarchical robot behaviours. *Robotics and autonomous systems*, **22**, 215 – 230.