Artificial Evolution and Real Robots

Inman Harvey
School of Cognitive and Computing Sciences
University of Sussex
Brighton BN1 9QH, UK

Abstract

Artificial evolution as a design methodology allows the relaxation of many of the constraints that have held back conventional methods. It does not require a complete prior analysis and decomposition of the task to be tackled, as human designers require. However this freedom comes at some cost; there are a whole new set of issues relating to evolution that must be considered. Standard Genetic Algorithms may not be appropriate for incremental evolution of robot controllers. SAGA, Species Adaptation Genetic Algorithms, has been developed to meet these special needs.

The main cost of an evolutionary approach is the large number of trials that are required. Simulations — especially those involving vision in complex environments, or modelling detailed semiconductor physics — may not be adequate or practical.

Examples of evolved robots will be discussed, including a specialised piece of equipment allowed for the testing of a robot using simple vision in real time, and what is believed to be the first successful example of an evolved hardware controller for a robot.

1 Why Evolutionary Robotics?

Humans are naturally evolved creatures, and the selection criteria under which our ancestors were judged did not include the ability to design complex systems —- in fact, we are not very good at it. A common and useful trick to overcome our shortcomings is that of *divide and conquer*: a complex problem is decomposed into separate easier sub-problems.

However, the interactions between such sub-problems must be few in number, so that the human designers can temporarily ignore them while solving one sub-problem at a time. When it comes to designing such complex systems as a cognitive control system for a robot, there are at least three major problems.

• It is not clear *how* a robot control system should be decomposed.
• Interactions between separate sub-systems are not limited to direct connecting links, but also include interactions mediated *via the environment.*
• As system complexity grows, the number of potential interactions between sub-parts grows *exponentially.*

Classical approaches to robotics have often assumed a primary decomposition into Perception, Planning and Action modules. Many people now see this as a basic error. Brooks' subsumption approach [1] is explicitly claimed to be inspired by natural evolution. Initially simple behaviours are 'wired into' a robot, and thoroughly debugged, before adding the next behaviour. This incremental approach echoes the phylogenetic history of complex cognitive creatures, some of whose behaviours we are trying to emulate in robots. Nevertheless, each new layer of behaviour is wired in by hand design; despite the heuristics used to minimise interactions between layers, it seems that unpredictable interactions may become insuperable when the number of layers gets much bigger than 10.

So an obvious alternative approach is to explicitly use evolutionary techniques to incrementally evolve increasingly complex robot control systems, rather than attempt to figure out each evolutionary step by hand design. Unanticipated and elusive interactions between sub-systems, though tricky or perhaps impenetrable for human designers, need not directly bother an evolutionary process where the only benchmark is the behaviour of the whole system.

2 Artificial Evolution for Robots

Genetic Algorithms (GAs) are the most common form of algorithm which uses evolutionary ideas for search, optimisation and machine learning — the fields covered in [2]. However, recently concerns have been voiced to the effect that GAs, when originally proposed by Holland [3], were intended as algorithms for complex *adaptive* systems, and their use for function optimisation is perhaps not best suited to their strengths. Evolutionary robotics typically needs adaptive improvement techniques, rather than optimisation techniques, and this critical but little-understood distinction needs to be made clear.

Most published GA work, both applications and theoretical analysis, refers to optimisation problems which can be seen as search problems in some high-dimensional search space, of known (usually enormous) size. Each dimension typically corresponds to some parameter which needs to be set, which is coded for on a small section of the genotype, a 'gene'. What such optimisation problems share is the well-defined finite nature of the search space. This allows the choice of a genotype coding, such that a genotype, often binary, of fixed length can encode any potential solution within the space of possibilities. In robotics, a genotype specifies the characteristics of a control system.

The GA works with a population of such genotypes, each of which is evaluated in terms of how good is the potential solution that it encodes. Genotypes which happen to be fitter in the current population (which initially may be generated at random) are preferentially selected to be parents of the next generation. Offspring inherit genetic material from their parents; usually inheriting part of this from each of two parents. A small number of random mutations are applied to the genotypes of the offspring. This cycle of selection, reproduction with inheritance of genetic material, and variation, is repeated over many generations, with the population remaining the same size as old members are replaced by new ones, the offspring of those members demonstrated to be fitter.

A GA optimisation problem has typically been seen as starting with a population of random points effectively spanning and crudely sampling the whole search space. Successive rounds of selection, reproduction and mutation focus the population of sample points towards fitter regions of the space, homing in on an optimum or near-optimal region. One consequence of this approach has been the primary reliance on recombination as the genetic operator, which mixes and matches information from different samples in order to move towards regions of expected higher fitness; mutation is typically relegated to the rôle of a background genetic operator.

However, some domains — including much of evolutionary robotics — do not always fall into this convenient picture of a fixed-dimensional search space. Standard GA theory does not necessarily then apply.

In evolutionary robotics a genotype will specify the control system (possibly more, see below) of a robot which is expected to produce appropriate behaviours when tested in its environment. However the evaluation of fitness is in terms of the robot's *behaviour*; for all except toy problems there is unlikely to be any obvious way to predict in advance the necessary complexity of control system for a given behaviour. Hence it is often appropriate to choose a genetic encoding which allows for, and encodes the characteristics of, a variable number of components. This has the added benefit of making incremental evolution possible: initially simple robots are evolved under a selection criterion based on simple tasks, and then the same robot population is allowed to increase in complexity in response to a gradual and continuing increase in task complexity. Such incremental evolution calls for *GAs as adaptive improvers* rather than *GAs as optimisers.*

3 SAGA

The conceptual framework of SAGA was introduced to deal with the dynamics of a GA when genotype lengths are allowed to increase [4]. It was shown, using concepts of epistasis and fitness landscapes drawn from theoretical biology [5], that progress through such a genotype space will only be feasible through relatively gradual increases in genotype length. A general trend towards increase in length is associated with the evolution of a *species* rather than global search — the population will be largely genetically converged.

Evolutionary search can be thought of as searching around the current focus of a species for neighbouring regions which are fitter (or in the case of neutral drift, not less fit) while being careful not to lose gains that were made in achieving the current *status quo*. The population can be visualised as moving around on a mountainous fitness landscape, where altitude represents fitness, and movements measured in horizontal directions loosely represent movements in genotype space. Selection is a force which tends to move a population up hills, and keep them centred around a local optimum; mutation produces offspring exploring outwards from the current population.

To increase exploration mutation rates should be increased; but if they are too high then the population disperses completely, losing the current local optimum, and the search becomes random. For any given selection pressure, there is a maximum rate of mutation which simultaneously allows the population to retain a hold on its current hill-top, while maximising search along relatively high ridges in the landscape, potentially towards higher peaks [6]. In SAGA, this means that rank-based or tournament selection should be used to maintain a constant selective pressure (the expected number of offspring of any individual should depend on its current ranking within the population, rather than the ratio of its fitness to the average fitness); and mutation rates should be maintained at a rate of about 1 mutation per genotype [7].

4 What building blocks?

We are relying on evolution for the design of a control system, but we must choose appropriate building blocks for it to work with. There is good reason to believe that the primitives manipulated by the evolutionary process should be at the lowest level possible. Any high level semantic groupings inevitably incorporate the human designer's prejudices. Primitives that are equivalent to a programming language give rise to a rugged fitness landscape with steep precipices. A program taken as a linear string of characters *can* be treated as a genotype, but typically a single mutation in a working program is fatal — a 'precipice'. Genetic programming [8] relies on recombination rather than mutation, but typically relies both on clever, domain-specific, choice of primitives, and on enormous population sizes which are difficult when evaluating robots.

With Brooks [1], we dismiss the classical Perception, Planning, Action decomposition of robot control systems. Instead we see the robot — body, sensors, motors and control system or 'nervous system' — as a dynamical system coupled (via the sensors and motors) with a dynamic environment [9]. This coupled interaction generates the robot behaviour which is to be evaluated. The control system is itself a dynamical system, and hence its genetic specification should be at the level of the primitives of a dynamical system.

One convenient form of dynamical system is an (artificial) neural net (NN). If this takes the form of a feedforward net, from sensors, perhaps via hidden nodes, to motors, then such a control system would have no internal state, and be capable only of generating reactive behaviour. However if a recurrent net is used, with temporal specifications to determine the timescales on which internal feedback is propagated, then non-reactive behaviour is also possible. Dynamic recurrent NNs (DRNNs), with temporal delays on links between nodes, are a class of dynamical systems capable in principle of replicating to an arbitrary degree of accuracy the dynamical behaviour of any other dynamical system with a finite number of components. Such DRNNs are equivalent (only trivial transformations are needed) to Brook's subsumption architectures using Augmented Finite State Machine (AFSMs). The temporal properties can derive either from within each AFSM (Brooks), or through the time-delays on links between them (DRNNs).

One significant difference from subsumption architecture is the deliberate introduction of internal noise at the nodes of DRNNs, with two effects. First, it makes possible new types of feedback dynamics, such as self-bootstrapping feedback loops and oscillator loops, which would not initiate themselves without the noise. Second, it helps to make more smooth the fitness landscape on which the GA is operating; a mutation which deletes a link or a node is comparable to a lot of noise, and hence the change in behaviour due to such a mutation is more closely correlated in the presence of noise than it would otherwise have been.

Thus genotypes need to specify a finite number of nodes, together with their thresholds or other details of a non-linear activation function transforming summed node inputs into node outputs; and links and connections between nodes, specifying weights and time-delays on the links. This can be generalised to include weight-changing rules. A specified subset of the nodes are designated as input nodes, receiving sensory inputs; similarly there is a set of output or motor nodes. Other nodes ('hidden') can be arbitrary in number, and genetically specified links are not necessarily restricted to feedforward ones.

5 Experiments

Using these ideas, a series of experiments were performed at Sussex with the gantry-robot using low bandwidth vision in a noisy real-world domain. A sequence of simple navigational tasks of increasing complexity was presented to the robot, and artifical evolution used to develop the control system and the visual morphology appropriate for success at these tasks. This work is reported elsewhere [10], and is believed to be the first example of artificial evolution for a robot using real vision.

The work used a genetically specified dynamical system as the control system, which is conceptualised as a DRNN, but in practice been implemented on a computer. There is a related approach of evolving control systems directly onto hardware, which has been taken within our group by Thompson [11].

This work is *intrinsic* hardware evolution, in that for each genetically specified piece of hardware, the actual hardware is tested *in situ*; as contrasted with extrinsic hardware evolution, where simulations of the hardware are evaluated during evolution. The actual low-level physics of the hardware can be utilised, and the realtime dynamics operate at their proper timescales. A human designer usually constrains and controls through clocking such features as switching transients; but by using artificial evolution, such design constraints can be relaxed in an unclocked system.

Thompson used artificial evolution to design a real EHW circuit as an on-board controller for a two-wheeled autonomous mobile robot required to display simple wall-avoidance behaviour in a wide corridor. The robot's only sensors were two sonars mounted on the left and right sides which fire simultaneously five times a second; the output from each sonar to the control system changes when the echo returns. In a conventional system the time of flight would be processed to estimate the range of obstacles, but in Thompson's implementation the pulses are fed directly into the hardware control system, termed a Dynamic State Machine (DSM). This is related to a Finite State Machine, except that each internal signal may (under genetic control) be clocked or unclocked. The global clock frequency, where used, is also under genetic control.

The DSM accepts pulses directly from the sonars, and outputs signals directly (no post-processing) to the motors for the left and right wheels which guide the robot. During evolution a population of DSM specifications (instantiated one at a time on the real hardware) is evaluated at the task of navigating to the centre of the corridor. Success was achieved within 35 generations; for full details see [11]. One successful evolved DSMs was found to be using just 32 bits of RAM and 3 flip-flops, excluding the clock generation. This minimal hardware produced the appropriate sensorimotor coupling between sonar echo signals and motor pulses, to guide the robot in its task. This is believed to be the first ever artificial evolved hardware controller for a robot.

6 Conclusions

Evolutionary robotics allows the relaxation of conventional design constraints, but new theoretical issues need to be studied. The SAGA framework allows for incremental evolution, and robot control systems should be treated as a class of dynamical system. Examples have been given of simple evolved robot behaviours in noisy real world conditions, including the use of evolvable hardware.

Acknowledgements

I acknowledge funding from the E.P.S.R.C., and discussions with Phil Husbands and Adrian Thompson.

References

[1] Brooks, R. A. "A robust layered control system for a mobile robot," *IEEE J. Rob. Autom.*, Vol. 2 pp. 14–23, 1986.

[2] Goldberg, D. E. *Genetic Algorithms in Search, Optimization and Machine Learning.* Addison-Wesley, Reading MA, 1989.

[3] Holland, J. *Adaptation in Natural and Artificial Systems.* University of Michigan Press, Ann Arbor, USA, 1975.

[4] Harvey, I. "Species adaptation genetic algorithms: The basis for a continuing SAGA," In Varela, F., Bourgine, P. (eds.), *Toward a Practice of Autonomous Systems*, pp. 346–354. MIT Press/Bradford Books, Cambridge, MA, 1992.

[5] Kauffman, S. "Adaptation on rugged fitness landscapes," In Stein, D. L. (ed.), *Lectures in the Sciences of Complexity*, pp. 527–618. Addison Wesley, 1989.

[6] Eigen, M., McCaskill, J., and Schuster, P., "Molecular quasi-species," *Journal of Physical Chemistry*, *92*, pp. 6881–6891, 1988.

[7] Harvey, I. "Evolutionary robotics and SAGA: the case for hill crawling and tournament selection," In Langton, C. (ed.), *Artificial Life III*, pp. 299–326. Addison Wesley, 1993.

[8] Koza, J. R. "Genetic programming," Tech. Report STAN-CS-90-1314, Dept. of Comp. Sc., Stanford University, 1990.

[9] Beer, R. D., and Gallagher, J. C. "Evolving dynamic neural networks for adaptive behavior," *Adaptive Behavior*, *1*(1), pp. 91–122, 1992.

[10] I. Harvey, P. Husbands, and D. Cliff. "Seeing the light: Artificial evolution, real vision." In D. Cliff, P. Husbands, J.-A. Meyer, and S. Wilson, (eds.), *From Animals to Animats 3*, pp. 392–401. MIT Press/Bradford Books, Cambridge MA, 1994.

[11] Thompson, A., Harvey, I., and Husbands, P. "Unconstrained evolution and hard consequences," In Sanchez, E., and Tomassini, M. (eds.), *Towards Evolvable Hardware.* Springer-Verlag, 1996.