

Adaptive Systems: Programming project suggestions.

These can be used by undergraduates for their programming task, or by postgraduates as a basis for their projects. Please refer to the course handout for details on assessment (including dates) and read the notes and advice on assessment on Anil's website.

Option 1: Genetic Algorithm search project

Solve the following problem using a genetic algorithm. You may program in C/C++, Java, etc. The problem is to find a path from START to FINISH on the 2D plane shown in Figure 1. The plane is littered with 'illegal zones'. The aim is to find the shortest possible path that does not cross any illegal zones. A path should consist of a sequence of straight line segments like those shown in the Figure.

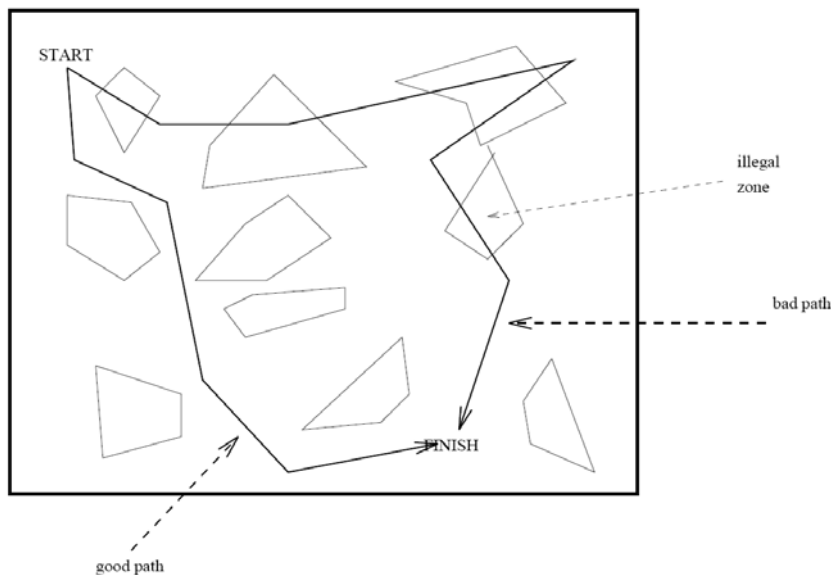


Figure 1:

The simplest representation of a solution for use with a GA is a string of values representing the coordinates of the start and end points of the line segment. It is suggested you use a fixed number of line segments (say 10), and possibly move on to a variable number later.

A suitable cost function to use would be:

$$C = \sum_i [L_i + P_i]$$

where L_i is the length of the i th segment of the path and P_i is the penalty incurred by the i th segment. The function is to be minimised. L_i is just the Euclidean distance:

$$L_i = \sqrt{(x_f - x_s)^2 + (y_f - y_s)^2}$$

where x_s, y_s, x_f, y_f are the (x,y) coordinates of the start and finish of a line segment. P_i is a function that penalises paths that cross illegal zones:

$$P_i = N_i \times pen$$

where N_i is the number of illegal zones the i th segment crosses, and pen is some penalty score.

Data describing the environment and a function for calculating N_i can be found in the attached file `lines.c`

Important: This is a relatively straightforward exercise, but not trivial. You will need to perform systematic comparisons and explorations of genetic operators, selective and encoding schemes, etc. You don't need to explore everything, but explore at least a few, and reach some conclusion (e.g. this genetic operator works better than that one in this problem.) You can also compare a GA with a non-evolutionary search method, such as random start hill-climbing, simulated annealing, etc.

Option 2: The Crazy Killer Cowboys problem

This is a variation on Option 1. This time the scenario is as follows. Some number (about 10, you can experiment) of crazy cowboys are placed at separate locations in a field full of rocks (the rocks could be the same size and position as the illegal zones in Option 1). Each cowboy will shoot dead any other cowboy he has a direct uninterrupted line of sight to (the rocks interrupt the line of sight, and the cowboys have a full 360 degrees of rotational freedom and shoot all other cowboys possible on a sweep round one full circle). Use a GA to find ways of positioning the cowboys such that there are a minimum of lethal interactions for some given distribution of rocks. The code given for Option 1 will come in very useful. You may want to attempt both options 1 and 2 to see if the same operators etc. are best in both problems.

Option 3: Development of a Robot Control System

The aim of this option is to develop a control system for a simulated robot. You *must* base the work on an adaptive technique covered in the course. You may write your own simulator (perhaps adapting the robot model described in the file `low-inertia-robot.pdf`) or use something in the public domain (but check it out with Anil or myself first!) *If you do use someone else's code then it must be clearly identified and acknowledged.*

The control system should at least generate simple wall and obstacle avoidance behaviours in an environment such as that shown in Figure 2. If you have time you could move on to more complex navigation behaviours, investigate how behaviours change with different sensor noise properties, or experiment with the use of light-sources, etc. A more challenging task is to evolve a robot to discriminate between small and large obstacles by approaching the former but not the latter using sensorimotor coordination.

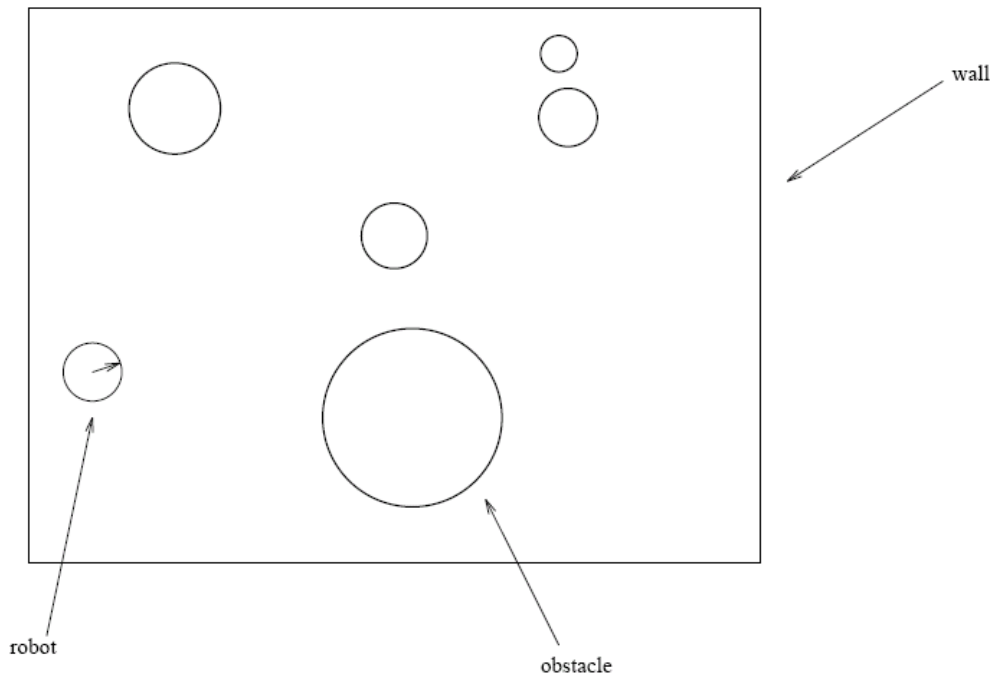


Figure 2: Example environment for Option 3.

Option 4: Study of a simulated Homeostat

Implement a version of Ashby's Homeostat and explore its behavioural aspects (e.g. replicate the adaptation experiments), or examine factors such as the influence of number of units and connectivity patterns on probability of adaptation, time to adaptation, etc. Explore different kinds of connectivity – e.g. random connectivity, ordered connectivity, small-world networks, sparse connectivity, and so on.

The simplest implementation of a single unit for the Homeostat is:

$$\frac{dy_i}{dt} = \sum w_{ji}y_j + b_i$$

where y_i is the internal state variable of the unit (which must remain within certain bounds) and w_{ji} is the connection strength from unit j to unit i , and b_i is a non-homogeneous term. Whenever the internal variable escapes from the viability zone, the parameters for the unit are randomized (w_s and b_s). You could choose variations on this idea (non-random changes, evolved rules of change, etc.)

NB: start by using symmetrical boundary values and parameter ranges, except perhaps for the self-weight w_{ii} which should be negative to induce stability.

This project is not difficult to implement, so it will be evaluated mainly on the quality and interest of the explorations undertaken, and the correctness and insightfulness of the

conclusions. And use the original source: *Design for a brain: the origin of adaptive behaviour*, by W. R. Ashby, 2nd ed, Chapman, 1960. You can download the pdf from <http://www.archive.org/details/designforbrainor00ashb>

Option 5: Minimally cognitive agents

Replicate, modify, and extend Randall Beer's experiments on minimal cognition. See:

Beer, R.D. (1996). *Toward the evolution of dynamical neural networks for minimally cognitive behavior*. In P. Maes, M. Mataric, J. Meyer, J. Pollack and S. Wilson (Eds.), *From animals to animats 4: Proceedings of the Fourth International Conference on Simulation of Adaptive Behavior* (pp. 421-429). MIT Press.

Slocum, A.C., Downey, D.C. and Beer, R.D. (2000). *Further experiments in the evolution of minimally cognitive behavior: From perceiving affordances to selective attention*. In J. Meyer, A. Berthoz, D. Floreano, H. Roitblat and S. Wilson (Eds.), *From Animals to Animats 6: Proceedings of the Sixth International Conference on Simulation of Adaptive Behavior* (pp. 430-439). MIT Press.

Both available from <http://mypage.iu.edu/~rdbeer/pubs.html>

You will have to use continuous time recurrent neural networks.

$$\tau_i \dot{y}_i = -y_i + \sum_j w_{ji} z_j + I_i; \quad z_j = \frac{1}{1 + \exp[-(y_j + b_j)]}$$

where, using terms derived from an analogy with real neurons, y_i represents the cell potential, τ_i the decay constant, b_i the bias, z_i the firing frequency, w_{ij} the strength of synaptic connection from node i to node j and I_i the degree of sensory perturbation of the sensory node. Parameters are best evolved using real-valued genotypes. In case of doubt, consult Owen or Anil for appropriate mutation operators. The above equations must be integrated numerically. Use the Euler method with a time step of half the minimum neural decay constant, or less.

Option 6: Design a homeostatic robot

Implement a simple robotic agent (writing your own simulator, or using something in the public domain) based on the principle of homeostatic adaptation. *If you do use someone else's code then it must be clearly identified and acknowledged.* Devise internal variables for this robot that are essential for survival (e.g., energy level, different kinds of nutrients, etc.) and a habitat where the robot would be able to keep these variables within bounds. Use artificial evolution by selecting for internal stability (suppose the internal variables are inherently unstable, unless the robot does something), and implement a mechanism by which plastic reconfiguration happens when the variables enter a danger zone. Test the robot in its 'natural' environment (the one that has been used during evolution) and then test it in an environment full of different kinds of disturbances. Alternatively, you can choose not to use evolution, but let your robot achieve adaptation directly by including plastic mechanisms (such as random

change in activation functions in a Braitenberg architecture) that maintain the essential variables bounded.

This may be a complicated project. Do not attempt to test radical disturbances first, but explore adaptation to simple perturbations (slight change in sensor to motor angle, obstacles if there weren't any originally, etc.) Be sure that if you see any adaptation that it is due to the reconfiguring mechanism and not to some inherent robustness of the design (i.e. perform appropriate control experiments).

Option 7: A simulation of Grey Walter's tortoises

Implement a simulation of Grey Walter's robots. (Discuss your robot model with Owen at an early stage – there are some unrecorded features of the robots that can be critical in producing some behaviours.) Try to reproduce the different experiments Walter is reported to have carried out, and try to account for any discrepancies between your results and his.

W. Grey Walter, *An imitation of life*, Scientific American, May 1950, 182(5), 42 - 45.

W. Grey Walter, *The living brain*, Duckworth, London, 1953.

Holland, O., (1996), *Grey Walter: the pioneer of real artificial life*, Proceedings of the ALife V Conference, Nara, Japan, MIT Press

Holland, O. (2003) Exploration and high adventure: the legacy of Grey Walter. *Phil. Trans. R. Soc. Lond. A* (2003) 361, 2085{2121