Simulated evolution of distributed FSA behaviour-based arbitration

Pavel Petrovič, IDI, NTNU, 7491 Trondheim, ppetrovic@acm.org

The field of Evolutionary Robotics (ER) [1] deals with automatic design of robot controllers using Evolutionary Algorithms (EA). In most cases, the goal is to design the software controller for a mobile robot, where the controller is typically a program evolved using genetic programming (GP), or the connection weights and possibly topology of a feed-forward or recurrent neural network, or another architecture. The controller typically takes the sensory readings on the input and produces the control signals to the actuators on the output, possibly maintaining some internal state. However, the architectures of the evolved controllers are usually uniform, without any internal hierarchy or structure. Furthermore, the evolution is usually made responsible for generating the complete behaviour of the robot, thus specifying the control commands or actions and processing of the input to the least detail. Unfortunately, the physical interactions of mobile robots with their environments are of a very complex nature, made even more intricate by inaccuracy of the sensors and actuators. Thus even a very simple task, for example locating a recognizable object and delivering it to one of two landmark-marked locations based on its colour is already a quite difficult one to evolve from scratch. In addition, the reliability of the resulting behaviour of the robot is questionable; it is very insecure that the detailed behaviour will perform as desired. Still, the possibility to generate the controller automatically could be an important advantage of ER, if these difficulties were overcome.

Non-automatic design of controllers for mobile robots developed through decades from hierarchical deliberative architectures with functional decomposition to behaviour-based (BB) architectures with behaviour decomposition, where the reactive aspect decides the controller topology [2]. The common property of the BB architectures is the simultaneous execution of elementary behaviours that are coordinated using some mechanism. The problem of coordination of behaviours is referred to in the literature as action selection problem or behaviour arbitration, for an overview, see [3].

This work applies the methods of EA to the design of specific type of behaviour arbitration – distributed finite state automata (FSA) arbitration. The behaviours that form the controller are already designed (manually, or possibly automatically), and do not change during the evolution. The aim is to develop an efficient method for automatic design of controllers given a set of standard basic both low-level and high-level behaviours for the robot. By keeping the arbitration mechanism distributed, where each arbitration FSA is associated with its behaviour module, we achieve a relatively high degree of extensibility and modifiability: when a novel behaviour is added to an existing controller, also its arbitration FSA is added, and the previous functionality of the controller is affected only to the necessary extend. Finally, instead of using a single evolutionary run, the target behaviour of the robot is simplified stepwise either sequentially or qualitatively and evolved in steps – with the use of incremental evolution.

In the experiments, we use the LEGO RCX hardware platform. Its main advantages are high flexibility, availability, extensibility, large group of users with good support and many available tools, robustness, easy maintenance, attractivity, and price.

Disadvantages such as low processing power, limited sensors and actuators precision and variety, proprietary copyrights, limited controller hw extensibility can be dealt with since we use it only for testing the algorithms on the prototype robots.

The chosen BrickOS (former LegOS) software platform based on GNU C compiler and RCX ROM utilities produces the binary for the built in processor H8/300 and thus allows to utilize the resources of RCX to the best. Features as multiple threads, POSIX semaphores, dynamic memory, floating-point operations, random numbers, direct motor and sensor control, infrared communication, fast downloads, and other make LegOS the most suitable platform for research experiments. [5]

The experimental task for the robot is cargo delivery. The robot navigates in a rectangular environment with obstacles and using hints (light and dark line to be followed drawn on a white floor) locates cargo loading and unloading stations, where it repeatedly loads, delivers, and unloads cargo using high-lifting fork mechanism. The controller contains basic behaviours, such as line-follower, obstacle-avoidance, random explore, load cargo, and other. The task for the EA is to find a correct distributed FSA behaviour arbitration, which utilizes the basic behaviours in suitable cooperation so that the robot efficiently makes use of the hints present in the environment and navigates between the loading and unloading stations while avoiding obstacles and carrying the cargo.

Since the evolution requires many runs, it would be unfeasible to run the EA on-line on the robot controller. Therefore we have developed a simulator of the robot hardware and software. The simulator allows to run the same program that runs on the physical robot. It works with a continuous time and environment described in the project configuration files. The simulator allows to define active elements in the environment, such as lights that can be controlled, and provides simple scripting language for creating events that can be triggered either by changing the robot state or position, or by a (possibly periodical) timer. The result of each simulated run is a quantitative fitness value – estimation of how much the robot behaviour resembles the target behaviour. In order to achieve as high fidelity in simulation of the physical conditions as possible, random uniform noise is added to the sensor readings and robot movements. In addition, the speed of the motors is adjusted according to measurements made with a camera setup [4], see figure 1. Since the simulation is performed on a powerful computer (1.4GHz Athlon, 512MB RAM), the early runs show that the same program can run 10-times faster in simulation than on the real robot (the remaining time is used for a detailed simulation computation).

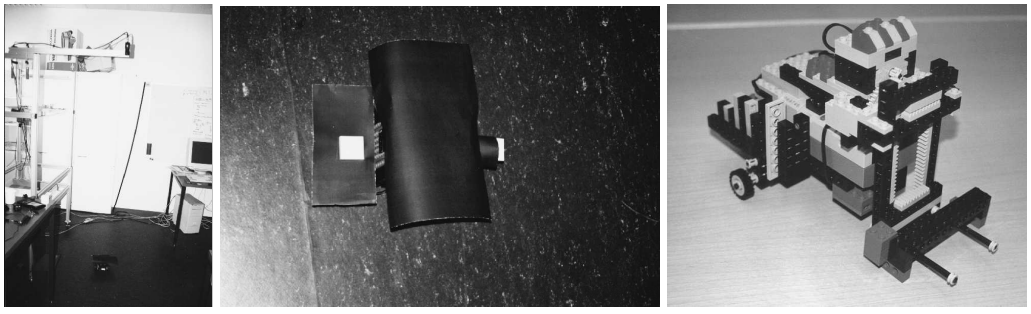

Figure 1. Camera setup for measuring the actual real-world outcome of the individual robot movements (left), the detail of the robot covered by black surface with 2 white marks detected by the calibrating software (centre), an experimental robot with high-lifting fork (right).

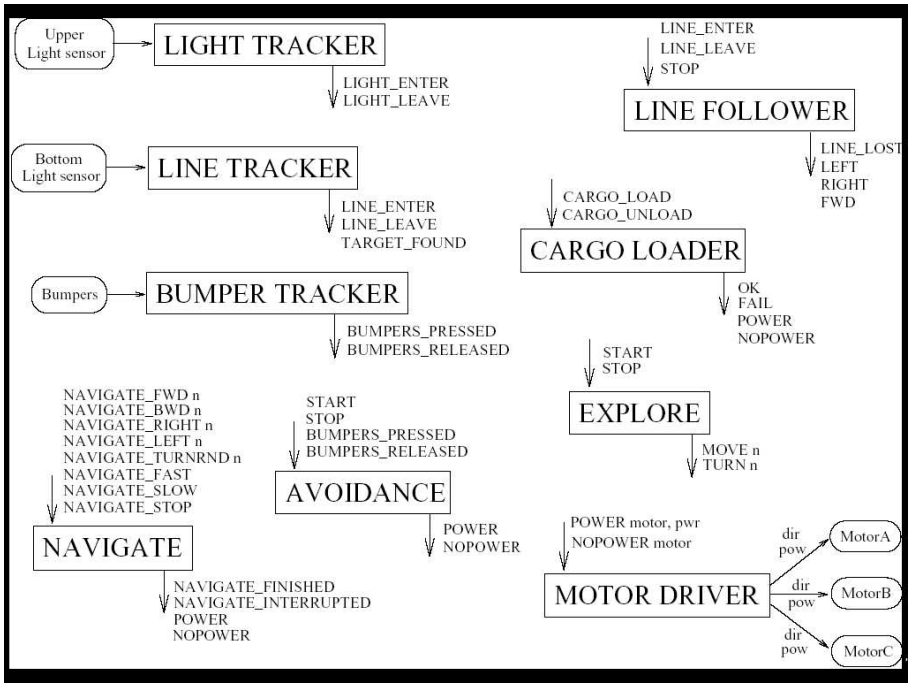

Figure 2. The architecture of the controller for the cargo task. Each module has associated an FSA arbitrator which translates the incoming and outgoing messages.

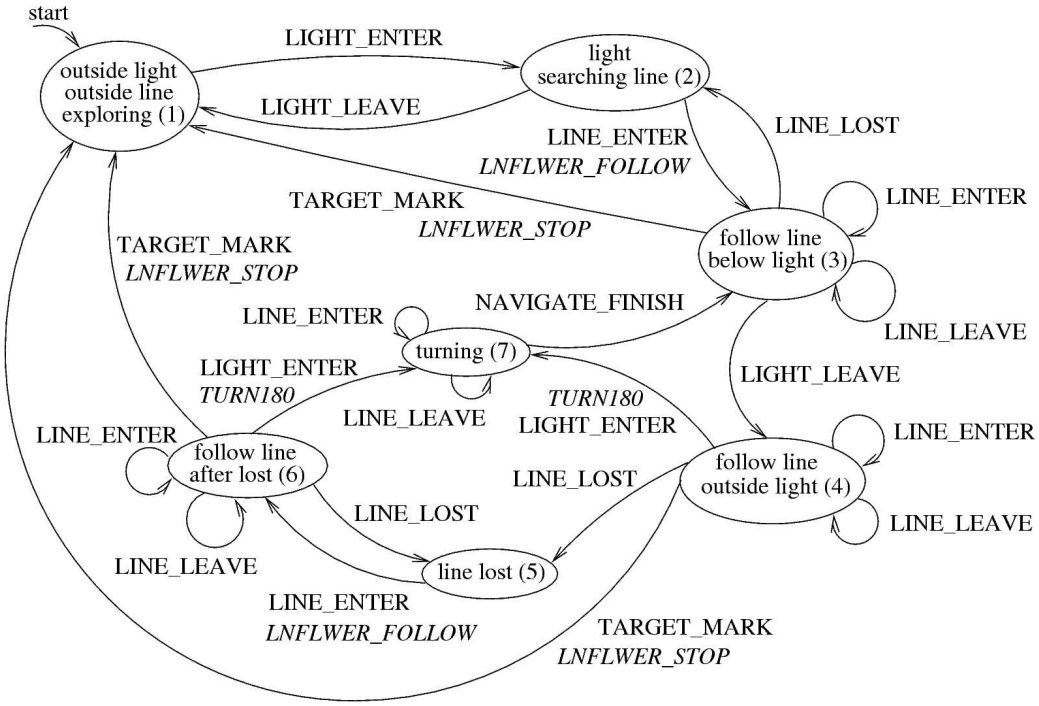

Figure 3. An example of the FSA arbitrator for the line follower module. This arbitrator was designed manually for the optimal performance. The labels on the transitions represent the incoming messages, in *italics* are shown the messages sent down to the module or out to other modules.

The arbitrators translate the messages sent to or by a module. The individuals in the EA population are sets of FSAs. Special EA operators for mutation, crossover, and initialization for this genotype representation were designed.

The main focus of the work lies in the analysis of the incremental evolution method, comparison of various task-decomposition strategies, population reinitialization, and controller architectures suitable for incremental evolution. The work is a PhD project in progress.

References.

[1] Nolfi S. and Floreano D. Evolutionary Robotics: The Biology, Intelligence, and Technology of Self-Organizing Machines. Cambridge, MA: MIT Press/Bradford Books, 2000.

[2] Arkin R.C. Behavior-Based Robotics. Cambridge, MA: MIT Press/Bradford Books, 1998.

[3] Pirjanian P. Behavior Coordination Mechanisms State-of-the-art. Technical Report IRIS-99-375, Institute of Robotics and Intelligent Systems, School of Engineering, University of Southern California, October 1999.

[4] Miglino O. Lund H.H. Nolfi S. Evolving Mobile Robots in Simulated and Real Environments. Artificial Life, vol. 2, nr. 4, pp. 417-434, 1995.

[5] BrickOS home: http://brickos.sourceforge.net.