DOI: http://dx.doi.org/10.7551/978-0-262-32621-6-ch113

The Case for Engineering the Evolution of Robot Controllers

Fernando Silva[1,3], Miguel Duarte[1,2], Sancho Moura Oliveira[1,2],
Luís Correia[3] and Anders Lyhne Christensen[1,2]

[1]Instituto de Telecomunicações, Lisboa, Portugal
[2]Instituto Universitário de Lisboa (ISCTE-IUL), Lisboa, Portugal
[3]LabMAg, Faculdade de Ciências, Universidade de Lisboa, Portugal
fsilva@di.fc.ul.pt

Abstract

In this paper, we discuss how the combination of evolutionary techniques and engineering-oriented approaches is an effective methodology for leveraging the potential of evolutionary robotics (ER) in the synthesis of behavioural control. We argue that such combination can eliminate the issues that have prevented ER from scaling to complex real-world tasks, namely: (i) the bootstrap problem, (ii) deception, (iii) the reality gap effect, and (iv) the prohibitive amount of time necessary to evolve controllers directly in real robotic hardware. We present recent studies carried out in our research group involving real-robot and simulation-based experiments. We provide examples of how the synergistic effects of evolution and engineering overcome each other's limitations and significantly extend their respective capabilities, thereby opening a new path in the design of robot controllers.

Introduction

Evolutionary computation techniques have been widely studied as a means to design robot controllers and body morphologies (Floreano and Keller, 2010), a field of research entitled evolutionary robotics (ER). ER has the *potential* to automate the synthesis of control systems. The experimenter relies on a self-organisation process, in which evaluation and optimisation of controllers is holistic, thereby avoiding the need for manual and detailed specification of the desired behaviour (Doncieux et al., 2011). The general idea is to optimise a population of genomes, each encoding a number of parameters of the robots' control system. Optimisation of genomes is based on Darwin's theory of evolution, namely blind variations and survival of the fittest, as embodied in the neo-Darwinian synthesis. The mapping from genotype to phenotype can capture different properties of the developmental process of natural organisms, and the phenotype can assume various degrees of biological realism (Stanley and Miikkulainen, 2003). Thus, ER draws inspiration from biological principles at multiple levels.

After approximately two decades of ER research, controllers have been evolved for robots with varied functionality, from terrestrial robots to flying robots (Floreano et al., 2005). Although there has been a significant amount of progress in the field (Doncieux et al., 2011), it is arguably on a scale that has precluded ER techniques of being widely adopted. Evolved controllers are in most cases not yet competitive with human-designed solutions (Doncieux et al., 2011), and have only proven capable of solving relatively simple tasks such as obstacle avoidance, gait learning, and distinct searching tasks (Nelson et al., 2009). In effect, researchers have been consistently faced with a number of issues that must be addressed before ER becomes a viable approach, including: (i) the bootstrapping issues when solutions to more complex tasks are sought (Nelson et al., 2009), (ii) deception (Whitley, 1991), (iii) the *reality gap* (Jakobi, 1997), which occurs when controllers evolved in simulation become inefficient once transferred to the physical robot, and (iv) the prohibitively long time necessary to evolve controllers directly on real robots (Matarić and Cliff, 1996).

This paper is concerned with the synthesis of behavioural control for autonomous robots. We discuss the current limitations of ER, and we present directions for future research. We argue that it is conceivable to *engineer* the evolution of robotic controllers, i.e., to combine evolved solutions and human knowledge to better address the fundamental problems in ER. In effect, evolutionary algorithms are also engineered algorithms and, above all, the fitness function is usually the result of trial-and-error experiments involving a substantial amount of experimentation and human intervention. The key decision is therefore where to draw the line between human design and evolution. We argue that the role of evolution and of human expertise should be defined based on *when* the synthesis of controllers takes place, namely *offline* or *online*. We present recent research in our lab, and we show the synergistic effects and potential of this combined approach through a series of real robot and simulation-based experiments involving an e-puck robot (Mondada et al., 2009). By combining engineering-oriented approaches and evolutionary techniques, we successfully evolve controllers for three tasks: (i) a double T-maze rescue task, (ii) a two-room cleaning task, and (iii) a deceptive phototaxis task. The main conclusion is that the proposed methodology is a viable new technique for leveraging the potential of ER.

Background and Related Work

In traditional ER approaches, controllers are synthesised *offline*, in simulation, to avoid the time-consuming nature of performing all evaluations on real robotic hardware. When a suitable controller is found, it is deployed on real robots. One of the central issues with the simulate-and-transfer approach is the reality gap (Jakobi, 1997), a frequent phenomenon in ER experiments. Controllers evolved in simulation can become inefficient once transferred onto the physical robot due to their exploitation of features of the simulated world that are different or that do not exist in the real world.

In *online evolution*, on the other hand, the evolutionary algorithm is executed on the robots themselves, while they perform their tasks. The main components of the evolutionary algorithm (evaluation, selection, and reproduction) are carried out autonomously by the robots without any external supervision. If the environmental conditions or task requirements change, the robots can modify their behaviour to cope with the new circumstances. However, the prohibitively long time required to evolve solutions on real robotic hardware is still a central impediment to large-scale adoption.

Besides the specific shortcomings of offline evolution and online evolution, there are two issues transversal to the two approaches: (i) the bootstrap problem (Gomez and Miikkulainen, 1997), and (ii) deception (Whitley, 1991). Bootstrapping issues occur when the task is too demanding to apply any meaningful selection pressure on a randomly generated population of candidate solutions. All individuals in the early stages of evolution may perform equally poorly, and evolution drifts in an uninteresting region of the search space. Deception occurs when the fitness function fails to build a gradient that leads to a global optimum, and instead drives evolution towards local optima. The more complex the task, the more susceptible is evolution to deception (Lehman and Stanley, 2011). Consequently to all these issues, ER techniques do not yet scale to tasks with the level of complexity found outside strictly controlled laboratory conditions (Nelson et al., 2009). The next sections review the current approaches introduced in ER for dealing with the problems discussed above.

Crossing the Reality Gap

Miglino et al. (1996) proposed three complementary approaches to cross the reality gap: (i) using samples from the real robots' sensors to enable more accurate simulations, (ii) introducing a conservative form of noise in simulated sensors and actuators to reduce the performance gap between the simulated and the real world, and (iii) continuing evolution for a small amount of time in real hardware if a decrease in performance is observed when controllers are transferred. The sensor sampling and the conservative noise methods have since become widespread. Continuing evolution in real hardware has not been frequently used, despite pioneering work in this direction (Nolfi et al., 1994).

Jakobi (1997) advocated the use of *minimal simulations*, in which the experimenter only implements features of the real world deemed needed for successful evolution of controllers. The remaining features are hidden in an "envelope of noise" to minimise the effects of simulation-only artifacts. It is not clear if Jakobi's approach scales well to complex tasks, since such tasks: (i) naturally involve more robot-environment interactions, and therefore more features, and (ii) require that the experimenter can determine the set of relevant features and build a task-specific simulation model.

Recently, Koos et al. (2013) introduced the *transferability approach*, in which controllers are evaluated based on their combined simulation and real-robot performance. To avoid testing each candidate solution in a real robot, a surrogate model is created and then updated periodically based on the results of real-robot experiments. The transferability approach has been shown to work when a solution can be found in relatively few generations (100 or less), but it can become unfeasible once the task requires several hundreds or thousands of generations with long evaluations. Furthermore, the difficulties in automatically evaluating controllers in real hardware represent an additional challenge.

Overcoming the Bootstrap Problem and Deception

Over the years, different approaches have been proposed to solve increasingly more complex tasks. In *incremental evolution*, the experimenter decomposes a task to bootstrap evolution and circumvent deception. There are numerous ways to apply incremental evolution (Mouret and Doncieux, 2008), such as dividing the task into sub-tasks that are solved sequentially, or making the task progressively more difficult through environmental complexification (Christensen and Dorigo, 2006). Although incremental evolution can be seen as an approach in which engineering and evolution are combined, it is typically performed in an unstructured manner. The experimenter has to perform a manual switch between the execution of each component of the evolutionary setup, such as different sub-tasks, which can significantly affect the global performance of solutions evolved (Mouret and Doncieux, 2008). In addition, if the components of the setup are highly integrated, incremental evolution can be difficult to apply successfully (Christensen and Dorigo, 2006).

Lehman and Stanley (2011) introduced *novelty search*, in which the idea is to maximise the novelty of behaviours instead of their fitness, i.e., to search directly for novel behaviours as a means to circumvent convergence to local optima. A number of studies outlined that novelty search is unaffected by deception, less prone to bootstrapping issues, and can evolve simpler solutions than those evolved by traditional fitness-based optimisation (Lehman and Stanley, 2011). Novelty search is, however, significantly dependent on the *behaviour characterisation* (Kistemaker and Whiteson, 2011), and can be challenging to apply when such a metric is not easy to define. That is, although novelty search

operates independently of fitness, its effectiveness is dependent on a similar form of human knowledge, despite recent studies involving generic characterisations (Gomes and Christensen, 2013).

Recently, a human-in-the-loop approach for avoiding deception was introduced by Celis et al. (2013). The approach allows non-expert users to guide evolution away from local optima by indicating intermediate states that the robot must go through during the task. A gradient is then created to guide evolution through the states. This approach was demonstrated in a deceptive object homing task, and it is still unknown if it generalises to different types of tasks.

Evolution in Physical Hardware

The first example of online evolution in a real, neural network-driven mobile robot was performed by Floreano and Mondada (1996). The authors successfully evolved navigation and homing behaviours for a Khepera robot. The studies were a significant breakthrough as they showed the possibility of online evolution of robot behaviour. Researchers then focused on the challenges posed by evolving controllers directly on physical robots, with a special focus on the prohibitively long time required (Matarić and Cliff, 1996). Afterwards, Watson et al. (2002) introduced *embodied evolution*, in which the use of multirobot systems was motivated by an anticipated speed-up of evolution due to the inherent parallelism in such systems.

Over the past decade, different approaches to online evolution have been proposed (Silva et al., 2012). Notwithstanding, few studies have been conducted on real robots. Researchers have focused on developing different evolutionary approaches and evaluating them mainly through online evolution in simulation. Despite the algorithmic advances, the strikingly long time that the online evolutionary process still requires during complex experiments renders the approach infeasible.

Engineering the Evolution of Controllers

The main objective of our ongoing work is to enable ER techniques to scale to more complex tasks by minimising the current issues in the field. We propose the *systematic* use of more practical, engineering-oriented approaches in which the significant potential of evolution in controller design is leveraged by human knowledge.

An engineering methodology in ER has not yet been agreed upon (Trianni and Nolfi, 2011). For instance, while different studies have combined evolved control and preprogrammed control, it is usually done in an ad-hoc manner, see Groß et al. (2006) for example, or by imposing hard behaviour-based architectures in which the role of evolution is minimal, see Urzelai et al. (1998). Contrary to such approaches, we argue that there is a *context-dependent compromise* between engineering and evolution. When conducting evolution offline, the experimenter has complete control over the experimental conditions and can modify and correct the selection pressures. Furthermore, the experimenter can take a methodical approach to find a suitable fitness function, an appropriate controller structure, or explore different evolutionary algorithms. That is, evolution is put at the service of engineering.

Complementarily, when evolving controllers online, the evolutionary algorithms run autonomously from the start and execute without any kind of human supervision. However, the experimenter can seed evolution with a bias towards certain types of solutions or behaviours, thereby inserting specific human knowledge into the evolutionary search. That is, the experimenter can give evolution *direct access* to task-related competences that are engineered before online evolution is conducted. If the structure and the parameters of these competences are under evolutionary control, they can be optimised during task execution, and evolution can progressively complexify controllers by using these building blocks as a substrate. In this way, engineering is put at the service of evolution.

At first sight, one may argue that the above perspectives imply an antagonistic relationship between offline evolution and online evolution. However, depending on the task complexity and requirements, offline evolution and online evolution may complement each other. In relatively simple tasks, it may be indifferent to conduct evolution offline or online. As the complexity of the task increases, the issues of each approach are exacerhated: (i) the more complex the controller, the more difficult it is to ensure successful transfer from simulation to reality, and the more time-consuming is evolution directly on real hardware, and (ii) in both cases, the more prone is evolution to bootstrap issues and deception. One solution is to exploit the benefits of each approach to bypass each other's limitations. Offline evolution can be used as an *initialisation* procedure in which approximate, yet effective solutions are engineered and deployed to real robots. During task execution, online evolution can serve as a *refinement* procedure that enables robots to adapt to changing or unforeseen circumstances.

In the following sections, we describe two complementary approaches for engineering ER: the hierarchical controller approach for offline evolution and the macro-neurons approach for online evolution, and we discuss how our approaches can mitigate the current issues in ER.

Engineering Offline Evolution

The hierarchical controller approach relies on a systematic hierarchical decomposition of the task, and structured composition of controllers that can be either evolved or preprogrammed (Duarte et al., 2014). We divide the task into simpler sub-tasks when evolution is unable to find a solution to a given task. Sub-controllers are evolved or preprogrammed to solve each sub-task, and the complete controller is composed in a hierarchical, bottom-up manner as shown

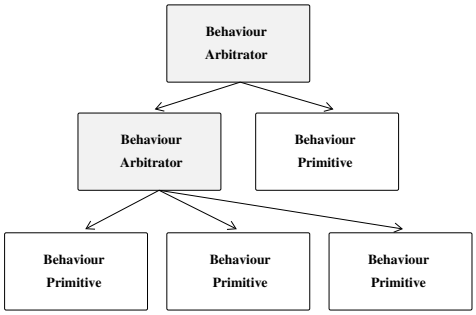


Figure 1: Representation of a hierarchical controller. Behaviour arbitrators determine which sub-controller to execute, and behaviour primitives control the actuators.

in Fig. 1. Each node in the hierarchy is either a *behaviour arbitrator* or a *behaviour primitive* (Lee, 1999). Behaviour primitives are at the bottom of the controller hierarchy and control a number of actuators of the robot, while behaviour arbitrators determine which primitive to execute at a given time. The logic in each node is independent of the logic in other nodes. Thus, evolved nodes can be synthesised by different evolutionary processes.

The evolution of behaviour primitives is based on the concept of an *appropriate* fitness function, which: (i) enables evolution to bootstrap, (ii) leads to controllers that consistently and efficiently solve the task in simulation, and (iii) evolves controllers that are able to maintain their performance levels in real robotic hardware. Provided an appropriate fitness function can be defined for a given task, we evolve a behaviour primitive composed of a single ANN. Otherwise, we recursively divide the task into sub-tasks until appropriate fitness functions have been found for each sub-task. Behavioural primitives are manually programmed when: (i) a sub-task cannot be further divided and an appropriate fitness function cannot be found, or (ii) if a particular robot-environment interaction is too difficult to accurately simulate. After the synthesis of behaviour primitives, sub-controllers are created by evolving or programming behaviour arbitrators in a bottom-up fashion. Each behaviour arbitrator receives a number of sensory inputs and is responsible for delegating control to the level below. Sub-controllers are then combined with other sub-controllers until the hierarchical controller is completed. Each time a new sub-controller has been synthesised, its performance on real robotic hardware can be evaluated, which allows to address transfer-related issues in an incrementally manner during the development of the control system.

An important aspect of our approach is that, as we move up the controller hierarchy and attempt to synthesise controllers for increasingly complex tasks, appropriate fitness functions may be increasingly difficult to define. In such cases, the fitness function can be *derived* based on the task decomposition and constructed to reward the arbitrator for activating a valid sub-controller for the current sub-task, rather than for solving the complete task. Thus, while previous studies have hierarchically decomposed controllers based on different techniques, from genetic programming to neuroevolution, see Duarte et al. (2014) for a review, our approach is distinct in a number of aspects. Firstly, we synthesise hybrid controllers in which preprogrammed control and evolved control can be seamlessly integrated, thus compounding the benefits of ER in the design of controllers, and preprogrammed behaviours that would otherwise be difficult or infeasible to evolve. Secondly, we use derived fitness functions to circumvent the otherwise increase in fitness function complexity. Finally, we bypass bootstrapping and deception-related issues due to the hierarchical task decomposition.

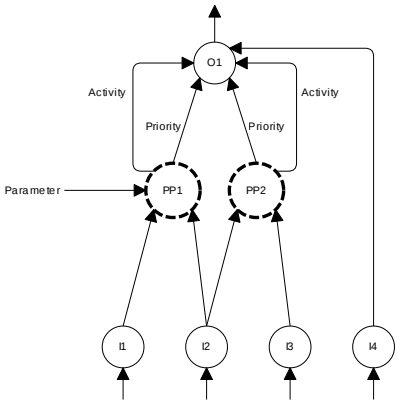
Engineering Online Evolution

This section introduces the macro-neurons approach for online evolution of neural network-based controllers (Silva et al., 2014). In this approach, neural networks use standard neurons as elementary components, and higher level units representing behaviours called the macro-neurons.

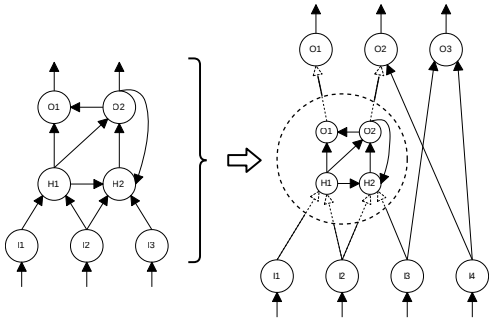
Each macro-neuron M is defined by (I, O, f, P), where I and O are respectively the set of input connections and of output connections, f is the function computed by the macro-neuron, and P is the set of parameters that can be optimised through evolution. Each connection $I_i \in I$ contains a weight $w_i \in w$ and transmits to M an input value $x_i \in x$. The computation of M is given by $f(w, x) = y$, where y is the output vector of M, and each $y_j \in y$ is transmitted to other neurons via the corresponding connection $O_j \in O$. Depending on the type of the macro-neuron M, the set P contains different elements. If M is an evolved ANN, then P refers to the connections and neurons that can be modified by evolution; if M is preprogrammed, P contains the parameters of the behaviour, if any.

In our approach, the macro-neurons are prespecified in the neural architecture before online evolution is conducted. The construction of ANNs using macro-neurons is shown in Fig. 2. Figure 2a illustrates how different preprogrammed macro-neurons are specified. Each macro-neuron transmits two values to each output neuron: (i) an *activity* value representing the signal to be sent to the actuators controlled by the output neurons, and (ii) a *priority* value, which represents the effective need of the behaviour to execute at a given time. Priority and activity values are used to better resolve conflicts when different preprogrammed macro-neurons compete for control (Silva et al., 2014). Complementarily, Fig. 2b shows how an evolved ANN is represented as a macro-neuron. The connections from the macro-neuron to the output neurons enable evolution to arbitrate and shape the output values of different macro-neurons.

In the experiments described in the following section, the macro-neurons are used in combination with odNEAT (Silva et al., 2012), an online neuroevolutionary algorithm that



(a) Preprogrammed macro-neurons



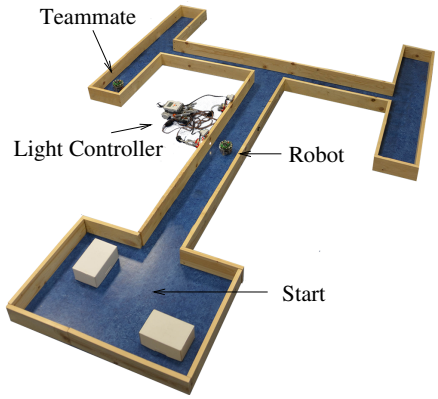
(b) Evolved macro-neuron

Figure 2: Examples of the integration of different types of macro-neurons in neural architectures. (a) Two preprogrammed macro-neurons. (b) An evolved ANN-based macro-neuron inserted into a larger controller.

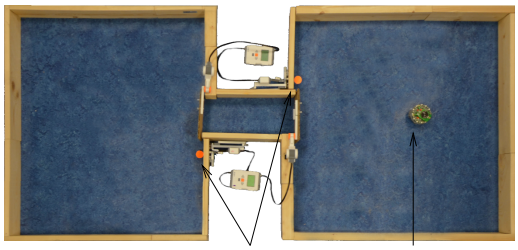
evolves the weights and the topology of ANNs in single and multirobot systems (Silva et al., 2012, 2014). Thus, the evolutionary process can: (i) adapt the preprogrammed behaviours by adjusting their parameters, see PP1 in Fig. 2a, (ii) modulate the execution of macro-neurons by increasing or decreasing the strength of connections, including those related to the priority and activity values, and (iii) optimise evolved ANN-based macro-neurons and the entire network by augmenting their structure and by adjusting the connection weights. By combining ANNs and macro-neurons, we compound: (i) the ANNs' robustness and tolerance to noise, (ii) the benefits of each type of macro-neuron, which can be synthesised by distinct evolutionary processes or manually designed to shortcut complex evolutionary processes, (iii) higher level bootstrapping, which can enable robots to adapt to complex and dynamic tasks in a timely manner.

Experimental Results and Discussion

In this section, we assess the viability of our approaches in both real-robot and simulation-based single robot experiments. In our experiments, we use an e-puck (Mondada et al., 2009), a 7.5 cm in diameter differential drive robot capable of moving at a maximum speed of 13 cm/s. The experiments were introduced in Duarte et al. (2012, 2014) and



(a) Real double T-maze environment



(b) Real two-room cleaning environment

Figure 3: The environments in which the hierarchical controller is assessed: (a) double T-maze with size of 2 x 2 meters, and (b) the two-room cleaning environment. The rooms are connected by a corridor blocked by two doors. Each room has one button that can be pushed to open the doors.

Silva et al. (2014). We review our previous results, and we argue the importance and effectiveness of combining engineering and evolution in the synthesis of robotic controllers.

Offline Design of Hierarchical Controllers

In this section, we apply the hierarchical controller approach to solve two tasks: (i) a rescue task in a double T-maze (Duarte et al., 2012), and (ii) a dust cleaning task (Duarte et al., 2014). The two environments are shown in Fig. 3. In the rescue task, the robot must exit a room with a number of obstacles, solve the T-maze, find the teammate, and safely guide the teammate back to the initial room. Two rows of flashing lights in the main corridor of the double T-maze give the robot information by indicating the branch leading to the teammate. In the dust cleaning task, dust spots appear in two rooms that are connected by a corridor. A new dust spot is randomly placed in one of the rooms every 10s, up to a maximum of five dust spots in the environment at any given time. Each room has one button that can be pushed to open the doors that give access to the corridor.

We first tried to evolve a monolithic controller for the complete rescue task using: (i) a standard $(\mu + \lambda)$ evolution strategy that optimised the weights of fixed-topology continuous-time recurrent neural networks with one hidden layer of fully-connected neurons, and (ii) the prominent

NEAT algorithm (Stanley and Miikkulainen, 2002), which evolves both the neural network's weights and topology. While the controllers evolved by the respective algorithms successfully solved initial parts of the task, none of them was able to complete the entire rescue task in simulation. We therefore divided the rescue task into three sub-tasks: (i) exit the room, (ii) navigate through the double T-maze *and* find the teammate, and (iii) return to the initial room while guiding the teammate. We decomposed the control system into three main sub-controllers: "Exit Room" primitive, "Solve Maze" arbitrator, and "Return to Room" arbitrator. Both the "Solve Maze" and the "Return to Room" arbitrators had access to three locomotion behaviour primitives: "Follow Wall", "Turn Left" and "Turn Right". A top-level arbitrator was evolved to select which sub-controller to activate at any given time. The controllers achieved an average success rate of 85%. The highest scoring hierarchical controller solved the task 93% of the times in simulation and of 92% in real robotic hardware.

To solve the two-room cleaning task, we decomposed the control system into two main sub-controllers: an evolved "Change Room" arbitrator and an evolved "Clean" primitive. The "Change Room" arbitrator was given access to an evolved "Open Door" arbitrator and to an evolved "Enter Corridor" primitive. The "Open Door" arbitrator had access to an evolved "Go To Button" primitive and to a preprogrammed "Push Button" primitive. Thus, all arbitrators and primitives were evolved, except for the "Push Button" primitive. Pushing a button to open the doors requires fine sensorimotor coordination, since the buttons are difficult to detect and hit. As this is a difficult interaction to model correctly in simulation, and therefore a behaviour to evolve and transfer successfully, the "Push Button" primitive was preprogrammed. In the complete task, the hierarchical controllers were evaluated according to the number of dust spots they cleaned in five minutes of real and simulated time. The controllers displayed high performance levels as they cleaned an average of 18.74 dust spots in simulation, and an average of 18.44 dust spots on the real e-puck robot.

We successfully synthesised controllers to solve two tasks with different requirements. One of the main ideas behind our approach is that ER techniques should not be applied blindly. We proposed an engineering methodology that exploits the knowledge acquired from negative results when a suitable controller cannot be evolved, and enables the decomposition of the task into simpler sub-tasks on an as-needed basis. By taking a systematic approach that combines evolution with engineering, we were able to overcome three fundamental issues: (i) the bootstrap problem, (ii) deception, and (iii) the reality gap, as the controllers maintained their performance levels in real robotic hardware. The bootstrapping problem and deception are naturally bypassed by dividing a complex task into simpler sub-tasks. The success in crossing the reality gap is due to the hand-design of sub-controllers when necessary and to the iterative tests of evolved sub-controllers (Duarte et al., 2014), in which the experimenter can address transfer-related issues locally in the controller hierarchy.

Additionally, it should be noted that by recursively focusing on controllers for simpler sub-tasks, the experimenter can more easily encourage the evolution of robust solutions that operate effectively in a large number of environmental conditions and that maintain their performance levels on real robots. In this way, solutions evolved can be made more general and therefore better sustain conditions not seen during evolution. As a final remark, it is worth discussing the role of human knowledge in our approach. In standard ER experiments, evolutionary setups are often found in an ad-hoc manner. The experimenter has to determine a suitable fitness function, the controller type and structure, the evolutionary algorithm, and the parameters associated with the evolutionary algorithm through a trial-and-error process. All these components are hand-designed, and usually involve a substantial amount of experimentation and human intervention. Contrary to unregulated trial-and-error methods, we follow a structured approach in which human knowledge is used to actively eliminate the factors that limit evolution and guide it towards classes of controllers relevant to the task.

Online Evolution with Macro-neurons

To assess the macro-neurons approach, we study a single robot deceptive and dynamic version of the phototaxis task with three light sources (Silva et al., 2014). The task environment is shown in Fig. 4. The robot has a constant virtual energy consumption value. The light sources are sensed by the robot within a 25 cm range. One source is beneficial to the robot as it increases the energy level, one source is neutral, and the remaining source is detrimental as it decreases the energy level. The sources are static, but they switch their type in a clockwise manner at five minute intervals. Deceptiveness is introduced by the fact that the three light sources are indistinguishable to the robot's light sensors. Thus, the robot must discriminate between the different sources based on the temporal correlation between its energy sensor readings and proximity to a given source.

We conducted experiments using two types of macro-neurons: evolved ANNs, synthesised offline using NEAT (Stanley and Miikkulainen, 2002), and preprogrammed behaviours. We synthesised three basic primitives of each type: (i) a move forward behaviour, (ii) a turn left behaviour, and (iii) a turn right behaviour. We conducted four sets of experiments: (i) evolution without macro-neurons, (ii) and (iii) evolution with access respectively to the preprogrammed and the evolved macro-neurons, and (iv) an hybrid approach involving a preprogrammed "Move Forward" macro-neuron and two evolved "Turn" behaviours.

The experimental setups involving macro-neurons enabled an efficient synthesis of controllers, with the advan-

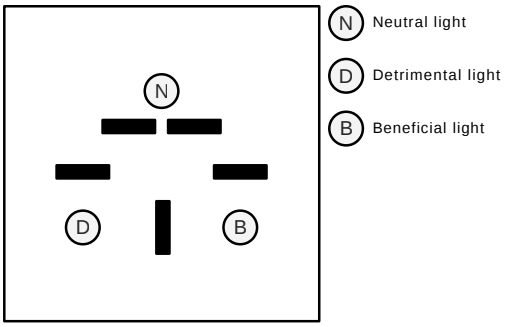


Figure 4: The task environment. The arena measures 3 x 3 meters. The dark areas denote obstacles, while the circular areas represent the different sources. The distance between the sources is set at 1.5 meters.

tage being in favour of evolved macro-neurons. Given the deceptiveness and complexity of the task, evolution without access to macro-neurons required an average of 9.50 hours of simulated time to evolve controllers that solve the task. The three types of controllers with macro-neurons required between 1.78 hours and 2.91 hours, thereby reducing the evolution time between 53% and 80%. In addition, the use of macro-neurons yielded competitive or superior solutions in terms of the fitness score (Silva et al., 2014).

Our current results suggest that approaches such as the macro-neurons may be a viable solution to speed-up online evolution in real robotic hardware. The key idea is that the experimenter can compensate for the absence of control he or she has during online evolution experiments by biasing evolution towards desired classes of behaviour. In effect, macro-neurons need not only to represent task-oriented behaviours. The macro-neurons can also represent prespecified survival-oriented behaviours that enable, for instance: (i) a group of robots to coordinate and share the access to battery charging stations, a task that been found to be highly deceptive (Gomes and Christensen, 2013), and (ii) to self-preserve by minimising collisions and hardware damage. By giving evolution access to these fundamental building blocks of distinct complexity and with different functions, bootstrapping is made easier because partial solutions to the task are already available. Additionally, evolution can focus on combining the engineered building blocks with evolved behaviours to synthesise increasingly sophisticated action patterns. New competences can be integrated in a scalable manner by gradually expanding the behavioural repertoire of the robot. Thus, rather than attempting to develop a purely automatic and potentially less efficient online evolutionary algorithm, the experimenter can take advantage of his or her knowledge to determine what are the basic components to solve the task. Each of the components can then be used by evolution in the search for a complete controller.

Intuitively, seeding evolution with specific behavioural properties may restrict the search space, and therefore potentially represents a trade-off between the adaptation time and the generality of behaviours that can be evolved. Nonetheless, recent experiments (Silva et al., 2014) have shown that evolution may be able to successfully adapt and reuse macro-neurons that are less optimised or even unsuited to the task. Additional experiments with different tasks are required to successfully answer this question.

Conclusions and Future Work

In this paper, we have argued that the combination of engineering-oriented and evolutionary approaches can minimise the current issues in ER, namely: (i) the bootstrap problem, (ii) deception, (iii) the reality gap, and (iv) the long time required for online evolution experiments. There are multiple reasons why our proposed methodology represents a valuable design tool, one of the most important being that the experimenter can influence how human knowledge and evolution are combined. In this way, the advantages of engineering-oriented and evolutionary approaches can be united to more easily overcome each other's limitations.

We presented two methods that combine the strengths of evolution and engineering: (i) the hierarchical controller approach, and (ii) the macro-neurons approach. The incorporation of evolution and engineering resulted in an effective synergy that enabled us to successfully evolve controllers for three tasks with a number of different traits. An important methodological advantage of our approaches is that they can be combined if deemed necessary. Hierarchical controllers of distinct complexity and functionality can also be encapsulated in a macro-neuron and adapted online. This versatility moves engineering and evolution from the space of offline or online synthesis of controllers to the space of offline *approximation* and online *refinement* of solutions. Thus, the key contribution of this paper is that our methodology is a flexible and viable approach for scaling evolutionary robotics to more complex tasks, without burdening the experimenter with the responsibility of performing a manual and detailed specification of the desired behaviour.

We are currently assessing our methodology in a varied set of tasks that have proven challenging for existing techniques, such as those that require a fine sensorimotor coordination. Because in more complex tasks, the division of a task into sub-tasks may not be intuitive, we are working towards having the evolutionary algorithm to perform the task decomposition itself. We are also extending our methods to multirobot systems to take advantage of the properties of decentralisation and robustness that pertain to self-organising systems. The main objective of our research is to reduce the current gap between ER and "mainstream" robotics.

Acknowledgements This work was partially supported by the Fundação para a Ciência e Tecnologia (FCT) under the grants SFRH/BD/76438/2011, SFRH/BD/89573/2012, PEst-OE/EEI/LA0008/2013, PEst-OE/EEI/UI0434/2014, and EXPL/EEI-AUT/0329/2013.

References

Celis, S., Hornby, G. S., and Bongard, J. (2013). Avoiding local optima with user demonstrations and low-level control. In *Proceedings of the IEEE Congress on Evolutionary Computation*, pages 3403–3410. IEEE Press, Piscataway, NJ.

Christensen, A. L. and Dorigo, M. (2006). Incremental evolution of robot controllers for a highly integrated task. In *Proceedings of the Ninth International Conference on the Simulation of Adaptive Behavior*, pages 473–484. Springer, Berlin, Germany.

Doncieux, S., Mouret, J.-B., Bredeche, N., and Padois, V. (2011). *Evolutionary robotics: Exploring New Horizons*, volume 341 of *Studies in Computational Intelligence*, chapter 1, pages 3–25. Springer, Berlin, Germany.

Duarte, M., Oliveira, S., and Christensen, A. L. (2012). Hierarchical evolution of robotic controllers for complex tasks. In *Proceedings of the IEEE International Conference on Development and Learning and on Epigenetic Robotics*, pages 1–6. IEEE Press, Piscataway, NJ.

Duarte, M., Oliveira, S. M., and Christensen, A. L. (2014). Evolution of hybrid robotic controllers for complex tasks. *Journal of Intelligent and Robotic Systems, in press*.

Floreano, D. and Keller, L. (2010). Evolution of adaptive behaviour by means of Darwinian selection. *PLoS Biology*, 8(1):e1000292.

Floreano, D. and Mondada, F. (1996). Evolution of homing navigation in a real mobile robot. *IEEE Transactions on Systems, Man, and Cybernetics B*, 26(3):396–407.

Floreano, D., Zufferey, J.-C., and Nicoud, J.-D. (2005). From wheels to wings with evolutionary spiking circuits. *Artificial Life*, 11(1-2):121–138.

Gomes, J. and Christensen, A. L. (2013). Generic behaviour similarity measures for evolutionary swarm robotics. In *Proceedings of the Fifteenth Genetic and Evolutionary Computation Conference*, pages 199–206. ACM Press, New York, NY.

Gomez, F. and Miikkulainen, R. (1997). Incremental evolution of complex general behavior. *Adaptive Behavior*, 3-4:317–342.

Groß, R., Bonani, M., Mondada, F., and Dorigo, M. (2006). Autonomous self-assembly in swarm-bots. *IEEE Transactions on Robotics*, 22(6):1115–1130.

Jakobi, N. (1997). Evolutionary robotics and the radical envelope-of-noise hypothesis. *Adaptive Behavior*, 6(2):325–368.

Kistemaker, S. and Whiteson, S. (2011). Critical factors in the performance of novelty search. In *Proceedings of the Thirteenth Genetic and Evolutionary Computation Conference*, pages 965–972. ACM Press, New York, NY.

Koos, S., Mouret, J.-B., and Doncieux, S. (2013). The transferability approach: Crossing the reality gap in evolutionary robotics. *IEEE Transactions on Evolutionary Computation*, 17(1):122–145.

Lee, W.-P. (1999). Evolving complex robot behaviors. *Information Sciences*, 121(1-2):1–25.

Lehman, J. and Stanley, K. O. (2011). Abandoning objectives: Evolution through the search for novelty alone. *Evolutionary Computation*, 19(2):189–223.

Matarić, M. and Cliff, D. (1996). Challenges in evolving controllers for physical robots. *Robotics and Autonomous Systems*, 19(1):67–83.

Miglino, O., Lund, H. H., and Nolfi, S. (1996). Evolving mobile robots in simulated and real environments. *Artificial Life*, 2(4):417–434.

Mondada, F., Bonani, M., Raemy, X., Pugh, J., Cianci, C., Klaptocz, A., Magnenat, S., Zufferey, J.-C., Floreano, D., and Martinoli, A. (2009). The e-puck, a robot designed for education in engineering. In *Proceedings of the Ninth Conference on Autonomous Robot Systems and Competitions*, pages 59–65. IPCB, Castelo Branco, Portugal.

Mouret, J.-B. and Doncieux, S. (2008). Incremental evolution of animats' behaviors as a multi-objective optimization. In *Proceedings of the Tenth International Conference on the Simulation of Adaptive Behaviour*, pages 210–219. Springer, Berlin, Germany.

Nelson, A., Barlow, G., and Doitsidis, L. (2009). Fitness functions in evolutionary robotics: A survey and analysis. *Robotics and Autonomous Systems*, 57(4):345–370.

Nolfi, S., Floreano, D., Miglino, O., and Mondada, F. (1994). How to evolve autonomous robots: Different approaches in evolutionary robotics. In *Proceedings of the Fourth International Workshop on the Synthesis & Simulation of Living Systems*, pages 190–197. MIT Press, Cambridge, MA.

Silva, F., Correia, L., and Christensen, A. L. (2014). Speeding up online evolution of robotic controllers with macro-neurons. In *Proceedings of the Sixteenth European Conference on the Applications of Evolutionary Computation*. Springer, Berlin, Germany. In press.

Silva, F., Urbano, P., Oliveira, S., and Christensen, A. L. (2012). odNEAT: An algorithm for distributed online, onboard evolution of robot behaviours. In *Proceedings of the Thirteenth International Conference on the Simulation & Synthesis of Living Systems*, pages 251–258. MIT Press, Cambridge, MA.

Stanley, K. and Miikkulainen, R. (2002). Evolving neural networks through augmenting topologies. *Evolutionary Computation*, 10(2):99–127.

Stanley, K. and Miikkulainen, R. (2003). A taxonomy for artificial embryogeny. *Artificial Life*, 9(2):93–130.

Trianni, V. and Nolfi, S. (2011). Engineering the evolution of self-organizing behaviors in swarm robotics: A case study. *Artificial Life*, 17(3):183–202.

Urzelai, J., Floreano, D., Dorigo, M., and Colombetti, M. (1998). Incremental robot shaping. *Connection Science*, 10(3-4):341–360.

Watson, R., Ficici, S., and Pollack, J. (2002). Embodied evolution: Distributing an evolutionary algorithm in a population of robots. *Robotics and Autonomous Systems*, 39(1):1–18.

Whitley, L. (1991). Fundamental principles of deception in genetic search. In *First Workshop on Foundations of Genetic Algorithms*, pages 221–241. Morgan Kaufmann, San Mateo, CA.