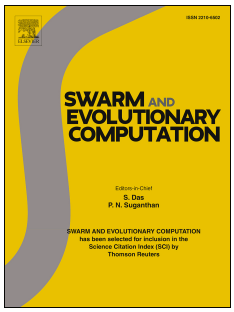
Accepted Manuscript



An approach to evolve and exploit repertoires of general robot behaviours

Jorge Gomes, Sancho Moura Oliveira, Anders Lyhne Christensen

PII: S2210-6502(17)30855-6

DOI: 10.1016/j.swevo.2018.06.009

Reference: SWEVO 420

To appear in: *Swarm and Evolutionary Computation BASE DATA*

Received Date: 25 October 2017

Revised Date: 18 May 2018

Accepted Date: 20 June 2018

Please cite this article as: J. Gomes, S.M. Oliveira, A.L. Christensen, An approach to evolve and exploit repertoires of general robot behaviours, *Swarm and Evolutionary Computation BASE DATA* (2018), doi: 10.1016/j.swevo.2018.06.009.

An Approach to Evolve and Exploit Repertoires of General Robot Behaviours

Jorge Gomes[a,c,f], Sancho Moura Oliveira[b,c,d], Anders Lyhne Christensen[b,c,d,e]

[a] *BioISI, Faculdade de Ciências da Universidade de Lisboa, Lisbon, Portugal*
[b] *Instituto de Telecomunicações, Lisbon, Portugal*
[c] *BioMachines Lab, Lisbon, Portugal*
[d] *Instituto Universitário de Lisboa (ISCTE-IUL), Lisbon, Portugal*
[e] *Maersk McKinney Moller Institute, University of Southern Denmark, Odense, Denmark*
[f] *Sonodot Ltd., London, United Kingdom*

Abstract

Recent works in evolutionary robotics have shown the viability of evolution driven by behavioural novelty and diversity. These evolutionary approaches have been successfully used to generate repertoires of diverse and high-quality behaviours, instead of driving evolution towards a single, task-specific solution. Having repertoires of behaviours can enable new forms of robotic control, in which high-level controllers continually decide which behaviour to execute. To date, however, only the use of repertoires of open-loop locomotion primitives has been studied. We propose EvoRBC-II, an approach that enables the evolution of repertoires composed of general closed-loop behaviours, that can respond to the robot's sensory inputs. The evolved repertoire is then used as a basis to evolve a transparent higher-level controller that decides when and which behaviours of the repertoire to execute. Relying on experiments in a simulated domain, we show that the evolved repertoires are composed of highly diverse and useful behaviours. The same repertoire contains sufficiently diverse behaviours to solve a wide range of tasks, and the EvoRBC-II approach can yield a performance that is comparable to the standard tabula-rasa evolution. EvoRBC-II enables automatic generation of hierarchical control through a two-step evolutionary process, thus opening doors for the further exploration of the advantages that can be brought by hierarchical control.

Keywords: Evolutionary computation; evolutionary robotics; novelty search; behaviour repertoires; hierarchical control

*Corresponding author
Email address: jmgomes@fc.ul.pt (Jorge Gomes)

1. Introduction

1.1. Motivation and Background

Evolutionary robotics is the field of research that employs evolutionary computation to generate robots that are adapted to their environment and task through a process inspired by natural evolution [1]. Historically, works on evolutionary robotics have mainly been focused on evolving robot controllers that are able to solve a single well-defined task [2, 3]. More recent works have, however, shown that diversity-driven evolutionary algorithms are a valuable tool for evolutionary robotics [4–6]. Evolutionary algorithms driven by behavioural novelty and diversity, such as the novelty search algorithm [7] and quality diversity algorithms [8], work by rewarding *behavioural novelty* instead of scoring solutions solely based on task performance. The behavioural novelty of a solution corresponds to its behavioural difference with respect to the solutions that have been evolved so far. Behavioural diversity based methods differ from the more traditional approaches that promote genetic diversity [9] in the sense that differences between candidate solutions are computed based on the actual behaviour of the robot in the environment and not in its genotype [4].

There are several potential benefits associated with the exploration of the behaviour space in evolutionary algorithms [5], including: (i) rewarding behavioural diversity can help mitigate premature convergence [7, 10, 11]; (ii) the evolutionary process can reveal many different ways of solving the given problem [8, 12]; and (iii) evolving a large set of diverse behaviours (*repertoires*) can enable new forms of control and the creation of new types of algorithms [13].

Novelty-driven algorithms have been used to evolve repertoires of robot behaviours in a number of different domains, for instance: the evolution of virtual walking creatures (including body plan and control policy) [12]; repertoires of morphological designs for walking soft robots [13]; repertoires of locomotion behaviours for legged robots [14–19] and four-wheeled steering robots [20]; repertoires of robotic arm behaviours [13, 19, 21]; repertoires of swarm behaviours [22, 23]; and repertoires of controllers for maze-navigation tasks [8, 17, 24]. The automatic generation of behaviour repertoires resembles self-exploration or babbling in developmental robotics [25], in which a robot experiments with a wide variety of motor commands, and learns the association between those motor commands and their consequence. The use of diverse *repertoires* can also be found in ensemble methods in machine learning, for instance in negative correlation learning [26, 27], where multiple neural networks in an ensemble are explicitly encouraged to behave differently from one another.

Since the advent of behaviour-based robotics [28], collections of behaviours have been extensively used to control robots. In general, a behaviour-based control system can be viewed as a structured network of behaviour modules, each of which achieves and/or maintains a specific goal [29]. These collections of behaviours are typically defined manually by the experimenter, based on a given task that the robot should solve. The basic behaviours from these collections can then be combined either manually or automatically through learning processes such as evolutionary algorithms [30, 31]. The automatic generation

of behaviour repertoires through evolution opens interesting opportunities for new forms of robot control. Instead of relying on a monolithic controller, the robots can be controlled by combining and selecting behaviours from the repertoire. Having access to a large set of behaviours allows for the choice of the behaviour that is most adapted to the context and the current situation of the robot. Cully et al. [14], for instance, study how a repertoire of hexapod locomotion behaviours can be used to make the robot fault tolerant: when the robot is impaired (for example, a broken leg), the controller can look for new locomotion behaviours that can cope with that fault. In a different application, Cully et al. [15] suggest that path planning algorithms can be used to control the robots based on an evolved behaviour repertoire. The path planner can choose the sequence of behaviours to execute based on the expected trajectory of each locomotion behaviour.

These forms of control can be seen as hierarchical control, in which the goal task (such as navigating towards the target) is decomposed into simpler sub-tasks (for example, walking forwards, turning right) [32–34]. Duarte et al. [35–37] have extensively studied this form of control, and developed the *hierarchical control synthesis approach*, where control is evolved for each sub-task, and the final controller is then evolved hierarchically in a bottom-up fashion. Behaviour nodes that have control over the robot's actuators are called *primitives*, and behaviour nodes that select and activate lower-level nodes are called *arbitrators*, following the terminology proposed by Lee et al. [32]. Such hierarchical controllers resemble stacking ensemble models [38] found in machine learning, which involve training a model (the aggregator) to estimate which base predictors perform well on the given data. The behaviour arbitrators are equated to the aggregator, while the behaviour primitives correspond to the base predictors.

Hierarchical control allows for the composition of increasingly complex controllers that leverage previously evolved primitives. Besides the achievement of control for increasingly more complex tasks, hierarchical control is also associated with other potential advantages [36], namely *reusability*: the sub-controllers can be used and combined to solve different tasks; and *incremental transfer*: the sub-controllers can be tested incrementally on real robotic hardware, and issues related to real-robot performance can be addressed independently from the rest of the control hierarchy.

In recent work, we proposed EvoRBC [18, 20], an evolutionary approach that combines the novelty-driven evolution of behaviour repertoires with the hierarchical control synthesis approach. EvoRBC divides the evolution of robotic control in two main steps:

1. The evolution of a repertoire of locomotion patterns (each a vector of parameters for the robot's locomotor system) using a quality diversity (QD) algorithm, for a given robot and according to a provided *fitness metric*.
2. The evolution of a high-level arbitrator (a neural network) using a neuroevolution algorithm, for a given task, using a provided *fitness function*,

and leveraging the repertoire evolved in the first step.

In a controller evolved following the EvoRBC approach, the high-level arbitrator receives the sensor readings from the robot, and outputs the mapping values. The mapping values are fed to a mapping function, which selects the corresponding low-level locomotion primitive. The locomotion parameters that are encoded in the primitive are then applied to the actuators of the robot. This process is repeated every control cycle during the robot's operation. EvoRBC abstracts the task-oriented control from the details of the robotic hardware itself, since the low-level interactions with actuators are removed from the task-learning process. Evolution only needs to optimise for a robot's intention (for instance, choose a direction and speed), and not learn how that intention should be realised in terms of tuning and coordinating multiple low-level locomotion parameters. EvoRBC therefore enables the evolution of task-oriented control for robots with complex locomotor systems by providing access to a variety of previously generated locomotion behaviours. EvoRBC has been evaluated in maze navigation tasks using a four-wheeled steering robot [20] and using a hexapod robot [18, 20]. Compared to the traditional evolutionary robotics approach, in which a monolithic controller is evolved, EvoRBC was able to achieve far more effective controllers using fewer evaluations in the evolutionary process.

1.2. Research Questions and Contributions

In EvoRBC, as well as in the vast majority of works that have evolved repertoires of robot behaviours, those behaviours are restricted to open-loop control and locomotion primitives [14, 16, 21]. While such an approach has its merits when the objective is to obtain controllers for robots with complex locomotor systems, it begs the question: can we leverage these principles to evolve repertoires of behavioural primitives that go beyond mere locomotion? In this paper, we propose EvoRBC-II, which extends the EvoRBC approach [18] to higher-level closed-loop behaviour primitives. Note that we use the term *open-loop* to refer to controllers that do not have access to the robot's sensor readings, while we used *closed-loop* to refer to controllers that take the robot's current sensor readings as input. Enabling higher-level closed-loop behaviour primitives raises a number of challenges, which we study and address in this paper. We pursue the following main research question, from where we derive three supporting research questions:

Main Research Question: How can repertoires of general primitives that rely on the robot's sensors be evolved, and how to take advantage of such repertoires to solve specific tasks?

Supporting Question 1: How can the evolution of general-purpose primitives, that can be used in a wide range of situations and tasks, be promoted?

Supporting Question 2: How can the behaviour of general-purpose closed-loop primitives be characterised?

Supporting Question 3: How can a repertoire of closed-loop primitives be leveraged to evolve solutions for specific tasks?

Supporting Question 1 concerns how to meaningfully evaluate the primitives during the repertoire evolution. In order for the primitives to be useful, they need to display robust behaviours, so as arbitrator evolution can reliably build on them. Previous work has been focused on the evolution of repertoires of locomotion primitives [15, 17, 18], where this problem is not present, as the primitives do not rely on sensor values and are always evaluated in the same conditions.

Supporting Question 2 concerns the characterisation of behaviour, an essential step for diversity-driven evolution. While in repertoires of locomotion behaviours this tends to be trivial – the displacement of the robot is used as behaviour characterisation [39], characterising general robot behaviour and interactions with the environment tends to be more challenging [8, 40]. An adequate behaviour characterisation is essential to evolve a diverse repertoire. Such diversity is valuable as it potentially enables the reusability of previously evolved repertoires of behaviours to rapidly solve new tasks.

Supporting Question 3 addresses the challenges in evolving a top-level arbitrator that must be able to find suitable primitives in the repertoire, and switch among them as needed during task execution. Since the primitives themselves can be quite capable, the top-level arbitrator should be able to take maximum advantage of those capabilities.

We explore these and other intertwined questions in this paper, and thoroughly evaluate the proposed approaches in nine simulated robotic tasks, based on canonical evolutionary robotics problems. This paper is organised as follows: the EvoRBC-II approach is described in Section 2. In Section 3, we describe the experimental setup that is used to assess the proposed approach. We evaluate EvoRBC-II in Section 4, including an analysis of the evolved repertoires and arbitrators, and comparisons with competing approaches. Our findings are summarised and discussed in Section 5, and we end with the main conclusions and future perspectives in Section 6.

2. Methods

The main contribution of this paper is EvoRBC-II, an extension of EvoRBC that enables the evolution and usage of repertoires of closed-loop behaviour primitives. As an example, while a primitive in the previously proposed EvoRBC approach [20] is restricted to open-loop locomotion, and thus encodes behaviours such as "*to turn to the right, apply 100% power to the left wheel and 50% to the right wheel*", a primitive in EvoRBC-II has access to the robot's sensory information, and can thus display behaviours such as "*go to the nearest target while avoiding obstacles*". Figure 1 summarises the two main steps of EvoRBC-II, which will be presented in detail in the following sections.

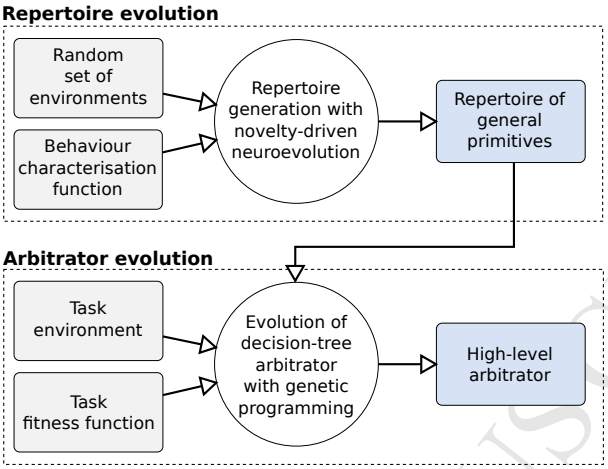


Figure 1: Evolution of control using the EvoRBC-II approach, showing the main inputs that must be provided by the experimenter (in gray), and the outputs of each step (in blue). Note that the same repertoire can potentially be used to evolve arbitrators for a broad range of different tasks.

2.1. Repertoire Evolution

The first stage of EvoRBC-II is the evolution of a diverse repertoire of general primitives. The process is illustrated in Figure 2 and described in Algorithm 1.

2.1.1. Novelty Search

We rely on the novelty search algorithm [7] to generate a set of diverse primitives. Novelty search is an evolutionary approach in which individuals are rewarded for being behaviourally novel. The novelty of a primitive is computed by comparing its behaviour with the behaviour of the other individuals in the current population, and the individuals in an archive (Algorithm 1, step 11). The archive is composed of individuals evolved during the evolutionary run (steps 13–14): the most novel individuals from the population are added to the archive every generation, and the least novel individuals are removed from the archive if it has reached a predefined capacity. In Sections 2.1.2 and 2.1.3, we describe in detail how the behaviour of each primitive is obtained.

Once the novelty search-based evolutionary process has terminated, the archive becomes the repertoire of primitives. The repertoire is therefore a representative sample of the individuals evolved during the evolutionary process. In our study, each primitive is a neural network, and the evolutionary process is implemented using the NEAT neuroevolution algorithm [41]. NEAT evolves both the weights and the topologies of the networks, meaning that the experimenter does not have to specify the neural network architecture. The neural networks evolved by NEAT are initialised with a minimal structure (inputs connected to outputs), and grow new nodes and connections as needed. The combination of NEAT and novelty search has been used with success in a large number of previous works, see for instance [7, 42].

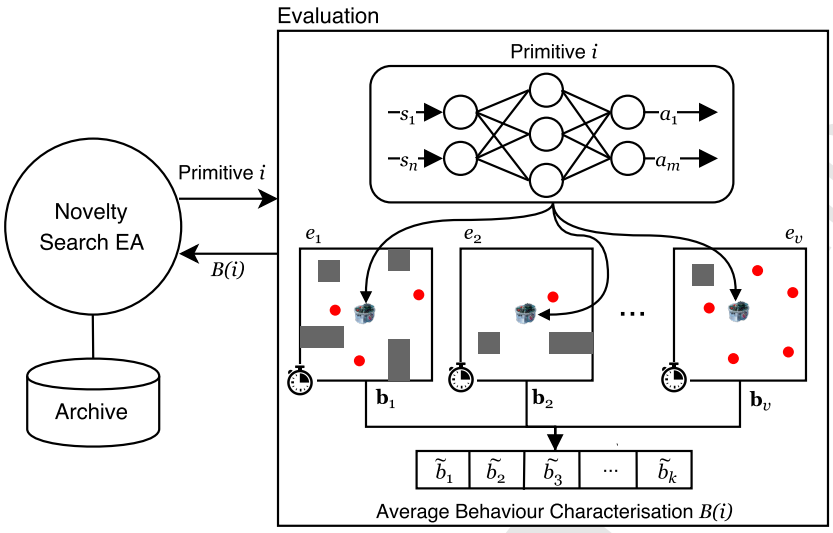


Figure 2: The primitives are neural networks with n inputs, one for each sensor ($s_{1..n}$), and m outputs, one for each actuator ($a_{1..m}$). Each primitive is evaluated in s simulations, corresponding to different randomly generated environments ($e_{1..v}$). During each simulation, the behaviour of the primitive is characterised according to the provided behaviour characterisation (see Section 2.1.3). The results from all simulations ($\mathbf{b}_{1..v}$) are aggregated to obtain the average behaviour characterisation $\mathcal{B}(i)$, a vector of length k, where k corresponds to the number of behaviour features comprising the characterisation. This behaviour characterisation is returned to the novelty search algorithm, where it is used to compute the novelty score of the primitive.

Algorithm 1 Repertoire generation with novelty search.

Let S be the maximum size of the archive, and s the number of individuals added per generation.
Let E be a set of randomly generated environments.
$\mathcal{A} \leftarrow \emptyset$ ▷ Archive
$\mathcal{P} \leftarrow$ `RandomInitialPopulation`() ▷ Population
for each generation **do**
  for each individual $i \in \mathcal{P}$ **do**
    for each environment $e \in E$ **do**
      $\mathbf{b}_e \leftarrow$ `Evaluate`(i, e) ▷ Simulate i in the environment e
    $\mathcal{B}(i) \leftarrow$ `GeometricMedian`$(\{\mathbf{b}_e : e \in E\})$ ▷ Combine the multiple observations
  for each individual $i \in \mathcal{P}$ **do**
    $\eta_i \leftarrow$ `ComputeNovelty`$(\mathcal{B}(i),\ \{\mathcal{B}(a) : a \in \mathcal{A}\} \cup \{\mathcal{B}(x) : x \in \mathcal{P} \wedge x \neq i\})$
    ▷ Compute novelty based on the archive and the other individuals in $\mathcal{P}$
  $\mathcal{A} \leftarrow \mathcal{A} \cup$ `SelectMostNovel`$(\mathcal{P}, s)$ ▷ Add the s most novel individuals to $\mathcal{A}$
  while $|\mathcal{A}| > S$ **do** ▷ The archive exceeds its capacity
    $\mathcal{A} \leftarrow \mathcal{A} \setminus$ `SelectLeastNovel`$(\mathcal{A}, 1)$ ▷ Remove the least novel individual
  $\mathcal{P} \leftarrow$ `Breed`$(\mathcal{P})$ based on the scores η ▷ Breed next generation
return $\mathcal{A}$

The original EvoRBC approach [20], as well as a number of other works that have generated repertoires of robot behaviours [13, 15], rely on the MAP-Elites algorithm for the repertoire generation [13]. MAP-Elites discretises the behaviour space, and aims to obtain a high-quality primitive in each of the resulting behaviour bins. In EvoRBC-II, we do not use MAP-Elites, but instead we have adopted the standard novelty search algorithm, to avoid having to specify the behaviour grid, which can become challenging when dealing with many behaviour dimensions. Moreover, previous work has shown that novelty search can achieve a performance comparable to MAP-Elites, with respect to the exploration of the behaviour space [8, 24]. Recently, new approaches for repertoire generation have been proposed, that overcome some of the MAP-Elites limitations [17, 19], which could be promising alternatives to the novelty search algorithm used in this paper.

2.1.2. Primitive Evaluation

The evaluation of each primitive (Algorithm 1, step 8) has the objective of characterising its general behaviour, that is, what a primitive does when placed in an arbitrary environment. To this end, each primitive is evaluated in a large number of independent trials (100 in our experiments), each simulating the robot in a randomly generated environment. In each simulation, the robot operates for a fixed amount of time executing the primitive, and the behaviour of the robot while running in the environment is characterised (see Section 2.1.3 for a discussion of the behaviour characterisation).

The behaviour characterisations obtained in all simulations are then aggregated to obtain the *average behaviour characterisation*, that should be representative of the primitive's typical behaviour. The average behaviour characterisation places the primitive in the behaviour space, and is used to compute the novelty score. We resort to the geometric median, a robust estimator of location [43], to determine the average characterisation from the multiple characterisations obtained in the different simulations. The geometric median is the equivalent of the median for multi-dimensional spaces, being robust to a large number of outliers (breakdown point of 50%). A robust average is useful in this application because of the stochasticity in the environments used for evaluation, which can lead to aberrant and unrepresentative behaviours (for example, an environment in which, by chance, the robot starts trapped in a corner, surrounded by obstacles).

2.1.3. Behaviour Characterisation

In novelty-driven evolution, the behaviour characterisation is acknowledged as being one of the most fundamental decisions, as it directly influences the direction of the evolutionary process [5]. Previous work has shown that behaviour characterisations are the most useful when they are *aligned* with the objectives of the task at hand. This alignment corresponds to the degree to which finding novelty tends also to lead to higher fitness solutions [8]. In the repertoire generation phase of EvoRBC-II, however, there is no task or objective, and there is thus no notion of *alignment*.

The behaviour characterisation used for the repertoire generation should instead capture how the robot interacts with the elements of the environment, with as little experimenter bias as possible. The behaviour characterisation should promote the discovery of the robot's behavioural possibilities, without biasing the search towards any specific types of behaviours. This type of behaviour characterisations are known as *generic* or *task-agnostic* characterisations, in the sense that they are not devised based on a specific task objective. Some generic characterisations that have been proposed in previous work, and that can be considered for the repertoire evolution, include:

State count: Count the number of times each sensory-effector state is observed in the robot during simulation [44]. The states can be obtained by discretisation of continuous sensor and effector values [45].

Sensor-effector trajectory: Measure the sensor and effectors values over a simulation [44]. The sensor-effector states can also be averaged over a time window [45].

SPIRIT: *Stochastic Policy Induction for Relating Inter-task Trajectories.* Each element in the behaviour vector represents the probability of the agent taking a particular action in a particular sensory state [46].

SDBC: *Systematically Derived Behaviour Characterisations.* The characterisation is automatically extracted given a formal description of the environment elements. It measures the spatial distances between the different elements of the environment, and the state of the agent or agents over time [47].

In this study, we rely on SDBC for its relative simplicity: it tends to produce behaviour characterisations with fewer dimensions than the other generic characterisations; and the behaviour characterisations are easily interpretable by the experimenter [47], which facilitates the analysis of the results.

2.2. Arbitrator Evolution

After having evolved a repertoire of general primitives, arbitrators can be synthesised for specific tasks. The arbitrator is a high-level controller that receives the robot's sensory readings and determines which behaviour primitive should be used to control the robot at each moment, see Figure 3 and Algorithm 2. The robot's sensor values are then fed to the selected primitive, which in turn outputs the values that are used to control the actuators of the robot. This cycle is repeated every control step, meaning that the arbitrator can switch to a new primitive every control cycle.

In the study presented in this paper, the arbitrator is a decision tree induced with genetic programming, in which the splits are based on the sensor values [32, 48]. The *classes* in the decision tree correspond to primitives chosen from the repertoire. This type of arbitrator contrasts with the neural-based arbitrators proposed in the original EvoRBC [18]. There are several motivations for

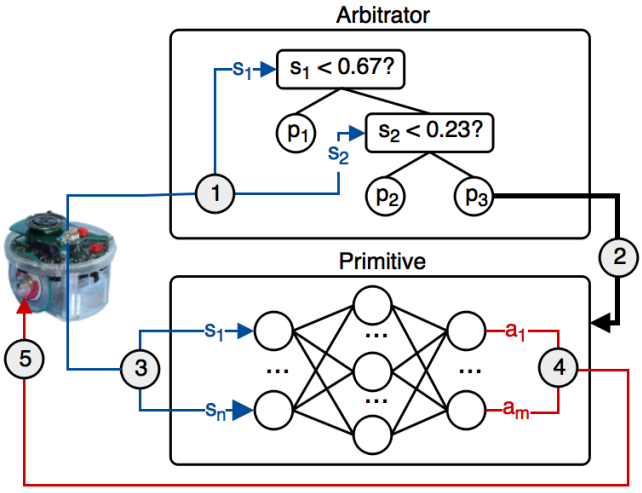


Figure 3: Execution cycle of a complete controller following the EvoRBC-II approach. (1) The robot's sensor values are fed to the high-level decision tree arbitrator. (2) The decision tree outputs one primitive. (3) The sensor values are fed to the selected primitive, outputting (4) the actuator values for the robot. (5) The actuator values are applied to the robot. This cycle repeats for every control step.

Algorithm 2 Control cycle of the robot following the EvoRBC-II approach.

```
1: procedure ControlStep(a, s)                ▷ Arbitrator a, sensor values s
2:     s' ← Normalise(s)                      ▷ Normalise the sensor values
3:     p ← Execute(a, s')                     ▷ Run arbitrator a with the inputs s'
4:     e ← Execute(p, s')                     ▷ Execute primitive p with the inputs s'
5:     ApplyToActuators(e)                    ▷ Actuate the robot
```

doing so [48]: (i) as each primitive is individually more capable, few primitives should be enough for solving most tasks, (ii) the arbitrator can make decisions based on only some of the sensors; (iii) the arbitrator becomes transparent and understandable by the experimenter, contrasting with neural networks that are mostly opaque; (iv) the arbitrator becomes mostly unaffected by the number of dimensions in the repertoire behaviour space and a potential lack of continuity; and (v) the controllers are faster to execute and have a smaller memory footprint, as there is no need for lookups in the repertoire during task execution, and only the primitives that are in the decision tree need to be stored.

The decision trees are evolved using tree-based strongly-typed genetic programming [49, 50], using the functions proposed by Gomes et al.[48]. There are two data types in the trees: P, a primitive of the repertoire; and C, a real value in the same range as the sensor values $[-1, 1]$. The return type of the tree is P, a primitive. The following functions have been defined to express the decision trees:

`Constant-Terminal`: $\varnothing \mapsto C$. Ephemeral random constant (ERC) terminal

that encodes a single real value, which it returns. The value can be modified by Gaussian mutation during evolution.

`Primitive-Terminal`: $\varnothing \mapsto P$. ERC terminal that encodes one primitive of the repertoire, which it returns. The primitive can be changed during evolution through multivariate Gaussian mutations on the behaviour characterisation vector of that primitive. The new primitive is the closest one in the repertoire with respect to the mutated vector.

`If-Sensor-Lower-i`: $(C, P, P) \mapsto P$. If the current value of sensor i is lower than C, the function returns the first P, else it returns the second P. Note that a P can be other `If-Sensor-Lower-*` functions.

The concrete implementation of the genetic programming algorithm for our experiments, along with its parameters, is further detailed in Section 3.4.

3. Experimental Setup

In order to assess the versatility and reusability of the evolved repertoires, and the potential of the EvoRBC-II approach, we defined an experimental environment that allowed us to implement a broad range of tasks. In this section, we describe the experimental setup used to assess the EvoRBC-II approach.

3.1. Environment

The experiments are conducted in a 2D simulated environment modeled in the MASON simulator [51]. The environment, with size 150×150 cm and bounded by impassable walls, see Figure 4, can contain the following elements:

Obstacles: Non-overlapping impassable rectangular shapes with sizes ranging from 10×10 to 30×30 cm.

POIs: *Points of Interest*. Point-sized, non-collidable objects that can be sensed by the robots. Three types of POI are considered: (i) *Static*, POIs remain at their respective locations during the entire simulation; (ii) *Dynamic*, POIs can move inside the environment, even over obstacles; and (iii) *Perishable*, POIs are static and disappear when a robot moves over them.

Robot: A differential-drive robot represented by a circular shape with a diameter of 8 cm. The robot is equipped with two types of sensors (depicted in Figure 4): (i) Obstacle sensors, six ray-based sensors that return the distance to the nearest intersection with an obstacle or wall (if any); and (ii) POI sensors, eight cone-based sensors, evenly distributed around the robot, each returning the distance to the closest POI in the respective circular sector (if any). The movement of the robot is controlled by two actuators that dictate the left wheel and right wheel speed, respectively. See Table 1 for additional parameters.

Table 1: Experimental parameters.

Parameter	Value	Parameter	Value
Robot and simulation common parameters			
Environment size	150×150 cm		
Obstacle sensor angles	0°,±45°, ±90°,180°	Obstacle sensor range	25 cm
		POI sensors	8 cones
POI sensor range	100 cm	Max. wheel speed	±10 cm/s
Max. wheel acceleration	±10 cm/s^2	Wheel axle length	8 cm
Max. turning speed	≈ 14.3°/s	Control cycle	10 steps/s
Repertoire generation – Novelty Search			
Novelty k-nearest	15	Add archive criterion	novelty
Archive growth	25 inds/gen	Maximum archive size	1000
General repertoire evolution – NEAT			
Population size	500	Generations	500
Target species count	10	Recurrency allowed	no
Survival threshold	20%	Crossover probability	20%
Mutation prob.	25%	Add link prob.	5%
Add node prob.	3%	Mutate bias prob.	30%
Evaluation samples	100		
Tabula rasa evolution – NEAT			
Population size	200	Generations	1000
Target species count	5	Recurrency allowed	yes
Evaluation samples	10	Evaluation simulation time	50 s
(Remaining parameters are the same as above)			
Arbitrator evolution – GP			
Population size	2000	Generations	100
Initialisation	PTC2 [52]	Init. max size	10
Subtree crossover prob.	50%	Point mutation prob.	20%
ERC mutation prob.	30%		
Node selection terminal	50%	Node selection non-term.	50%
Crossover/mutation tries	10	Max tree depth	17
Max tree size	∞	Tournament size	7
Constant ERC range	[-1,1]	Constant ERC mut. σ	0.2
Primitive ERC range	[0,1]	Primitive ERC mut. σ	0.1
Locomotion repertoire evolution – MAP-Elites			
Iterations	500	Batch size	1000
Initial batch size	10000	Gene range	$[0, 1]$
Gene mutation prob.	50%	Genome size	2
Mutation σ	0.1	Mutation type	Gaussian
Evaluation time	2 s	Bin size	1×1 cm
Behaviour char.	Displacement	Fitness metric	Circular

The source code used for all the experiments and analysis described in this paper can be found at `https://github.com/jorgemcgomes/mase/releases/tag/swevo_revision`.

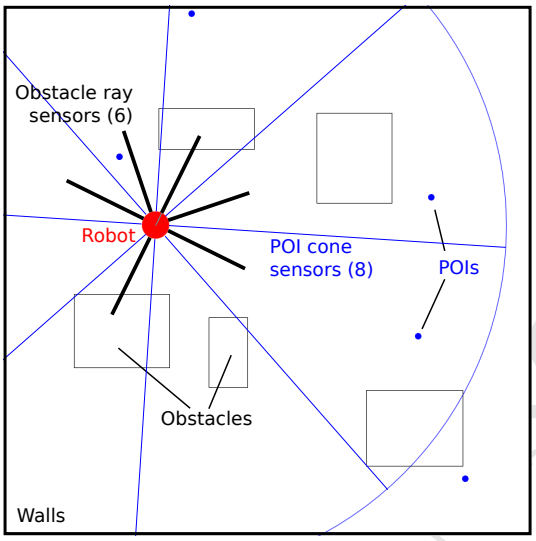


Figure 4: An example of an environment with walls, five obstacles, and five POIs. The blue lines depict the range of the robot's four POI sensors, and the thick black lines depict the range and angle of the three obstacle sensors.

3.2. Evaluation Tasks

Based on the environment described in Section 3.1, we defined a total of nine single-robot evaluation tasks, inspired by the tasks commonly found in evolutionary robotics studies [2, 3]. We chose a diverse set of tasks that could be defined using the environment elements described in Section 3.1, and solved by the specified robot. A brief description of the tasks and the respective fitness functions is presented next, and key environment parameters can be found in Table 2.

Foraging. Foraging task in an open environment, with N perishable POIs placed randomly in the environment. The objective is for the robot to *collect* as many POIs as possible during a simulation. The fitness function corresponds to the number of POIs collected (n):

$$F_F = n/N \tag{1}$$

Foraging-O. Same as the Foraging task, but with obstacles placed randomly in the environment.

Phototaxis. In this task, a single static POI is randomly placed in the environment, and the objective is for the robot to reach it as quickly as possible. The task ends when the robot reaches the static POI (time t) or the allowed time (T) elapses. The fitness function rewards getting closer to the POI (with d_f being the final distance from the robot to the POI, and D the environment's

Table 2: Key environment parameters for the nine evaluation tasks, and environment parameters used for the repertoire generation (last line). *POI speed* is given relative to the robot's maximum speed.

Task	# Obs.	Obs. side (cm)	# POI	POI type	POI speed	Time (s)
Foraging	0	–	15	Perishable	0	50
Foraging-O	5	[15,30]	15	Perishable	0	50
Phototaxis	0	–	1	Static	0	50
Phototaxis-O	5	[15,30]	1	Static	0	50
Exploration	5	20	0	–	–	100
Maze	25	[10,20]	0	–	–	50
Avoidance	0	–	15	Dynamic	25%	50
Prey	[1,5]	[15,30]	1	Dynamic	75%	50
Tracking	5	[15,30]	1	Dynamic	50%	50
Repertoire	[1,10]	[10,30]	[1,10]	Static	0	50

diagonal length), or reaching the POI as quickly as possible:

$$F_{Ph} = \begin{cases} 1 - (t/T)/2 & \text{POI reached} \\ 0.5 - (d_f/D)/2 & \text{otherwise} \end{cases} \quad (2)$$

Phototaxis-O. Same as the Phototaxis task, but with obstacles placed randomly in the environment.

Exploration. In an environment with randomly placed obstacles, the objective is for the robot to explore as much of the environment as possible within the given time. The environment is divided into a regular grid with C cells of size 30×30 cm to quantify the explored space. The fitness function corresponds to the number of cells visited by the robot (c):

$$F_E = c/C \quad (3)$$

Maze. The robot starts in the centre of the environment, with several obstacles randomly placed around it, and must reach the environment's boundaries without colliding with obstacles. The task ends if the robot collides with an obstacle, reaches the boundaries, or the allowed time (T) elapses. The fitness function rewards the robot for reaching the boundaries as quickly as possible (time t), or getting closer to them (d_c – final distance from the centre):

$$F_M = \begin{cases} 1 - (t/T)/2 & \text{robot escaped} \\ d_c/D & \text{otherwise} \end{cases} \quad (4)$$

Avoidance. A number of *dynamic* POIs are placed in the environment, and the robot must maintain a safety distance (25 cm) from all POIs. Each dynamic POI moves in a straight line, with a speed of 25% of the robot's maximum speed, and takes a new random direction whenever the environment boundaries

are reached. The fitness reward is inversely proportional to the amount of time (t_c) the robot is close (less than 25 cm) to one or more POIs:

$$F_A = 1 - t_c/T \tag{5}$$

Prey. In this task, one dynamic POI is always moving towards the robot at 75% the robot's maximum speed, in an environment with randomly placed obstacles. The robot must avoid the dynamic POI. If the POI gets near the robot, the simulation ends. The fitness function rewards the robot for surviving as long as possible (t):

$$F_P = t/T \ , \tag{6}$$

Tracking. One dynamic POI with random movement (similar to the movement in the Avoidance task), with a speed of 50% of the robot's maximum speed, is placed in an environment with obstacles. The robot must stay as close as possible to the POI during the simulation:

$$F_T = 1 - F_P \tag{7}$$

3.3. Behaviour Characterisation

The behaviour characterisation used for repertoire generation was devised according to the SDBC [47] methodology, as discussed in Section 2.1.3. The characterisation is composed of a total of seven behaviour features, systematically extracted based on the defined environment:

1. Mean distance of the robot to the walls. ($\mu = 26.3, \sigma = 15.9$)
2. Mean distance of the robot to all obstacles. ($\mu = 55.0, \sigma = 18.2$)
3. Mean distance of the robot to the closest obstacle. ($\mu = 24.8, \sigma = 21.9$)
4. Mean distance of the robot to all POIs. ($\mu = 66.4, \sigma = 20.0$)
5. Mean distance of the robot to the closest POI. ($\mu = 34.2, \sigma = 24.1$)
6. Robot's mean linear speed. ($\mu = 0, \sigma = 1$)
7. Robot's mean turn speed. ($\mu = 0, \sigma = 14.3°$)

The behaviour features are scaled based on the mean (μ) and standard deviation (σ) of the feature values, as proposed by [47]. The rationale is that all features should have similar ranges, so that all features have a similar weight in the behaviour distance calculation. The μ and σ values for each feature were computed a priori based on a large number of randomly generated environments.

3.4. Evolutionary Setup

Evolution of Repertoires of General Behaviours. For the repertoire generation, each primitive is evaluated in a total of 100 randomly generated environments, with the parameters shown in Table 2 (Repertoire).[1] It should be noted that all primitives are evaluated in the same set of environments in all generations. We

[1] Examples of environments that were used for evaluation can be seen in Figures 4 and 7.

use the NEAT neuroevolution algorithm [41] driven by pure novelty search [7], as described in Section 2.1.1. The parameters of the evolutionary algorithm are shown in Table 1. The repertoire of primitives is simply the content of the novelty search archive after the last generation. This means that the generated repertoires always have a number of primitives equal to the maximum archive size, 1000 individuals.

Evolution of the Task-oriented Arbitrator. The strongly-typed GP is implemented in the ECJ framework [53]. The population is initialised with the PTC2 method [52], breeding is performed based on tournament selection, and each new tree is bred through one of three possible genetic operators, chosen randomly: Koza-style subtree crossover [54], point mutation [54], and Ephemeral Random Constant mutation (mutates all ERC nodes in the randomly chosen subtree). To limit bloat, evolved trees are pruned after every fitness evaluation to remove all the `Primitive-Terminal` nodes that were never executed during the simulations, as proposed by Iba et al. [55]. All parameters are listed in Table 1.

Tabula-rasa evolution. Tabula-rasa evolution, the traditional evolutionary robotics approach [3, 6], is used as comparison to EvoRBC-II. In tabula-rasa, the controller receives the sensory readings from the robot and outputs the actuator values. The controllers are neural networks evolved with fitness-driven NEAT [41], which has been used with considerable success in a large number of previous evolutionary robotics studies [4]. The parameters are listed in Table 1.

4. Results

4.1. Repertoire Diversity

We begin by analysing the evolved repertoires and the primitives contained in them. We first analyse whether the final repertoire captures the entire diversity of evolved behaviours, that is, whether the behaviour space covered by the repertoire matches the behaviour space covered by the entire evolutionary run. As discussed in Section 2.1, the repertoire should be general and cover as much of the behaviour space as possible, so that the top-level arbitrator has a wide diversity of behaviours to choose from, and the same repertoire can be used in a wide array of tasks. To establish a baseline, we additionally compare the evolved repertoire with a repertoire composed of randomly generated primitives. This repertoire is composed of 1000 fully connected Multi-Layer Perceptrons, with a random number of hidden neurons ranging from zero to ten, and random connection weights. This means the networks have a complexity similar to those that can be found in the evolved repertoires.

Figure 5 shows all the primitives evolved up to a certain generation in a given evolutionary run, in comparison with the final repertoire, and a randomly generated repertoire. As described in Section 2.1, the evolved repertoire corresponds to the novelty archive at the end of the evolutionary run. For the sake of visualising the behaviour space, the seven behaviour features were reduced to two using Robust Principal Component Analysis [56]. All the behaviour

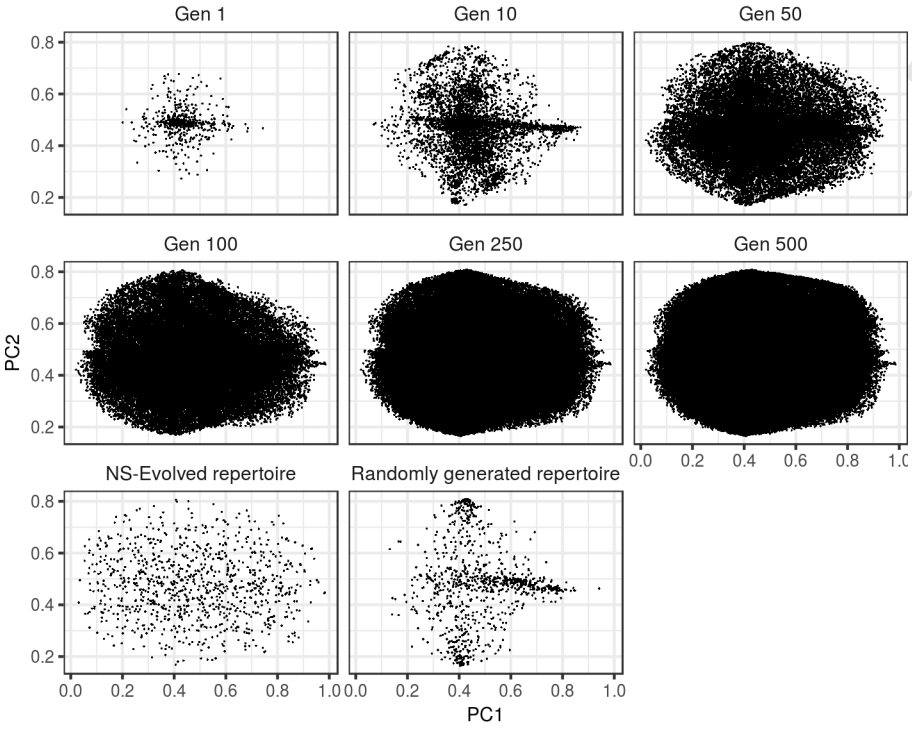


Figure 5: Behaviour space covered in one evolutionary run of the repertoire generation process, by all primitives evolved up to a given generation (Gen #), the final repertoire, and a repertoire composed of randomly generated neural networks. The behaviour space is reduced to two dimensions using RPCA [56]. Each dot corresponds to one primitive, located according to the two first principal components.

data from all evolutionary runs and randomly-generated repertoires was used to calculate the principal components, with the two first principal components accounting for 75% of the variance in the data. In Figure 6, we show the behaviour characterisations that can be found in each behaviour region.

The results from Figure 5 show that the repertoire is representative of all primitives evolved during the evolutionary run (*Gen 500* in Figure 5). Around 250 generations were needed to fully cover the *reachable* behaviour space, which suggests that the chosen number of generations for the repertoire generation (500) was sufficient. The results also show that novelty search is an effective exploration technique, achieving a substantially wider diversity of primitives than the randomly generated repertoire. The visual inspection of the primitives in the repertoire confirms the diversity measured by the behaviour characterisation, see Figure 7. For instance, some primitives navigate towards a POI and stay close to it (a), others visit multiple POIs (b), stay close to obstacles (c), try to get away from everything (d), roam around the environment (e), or move

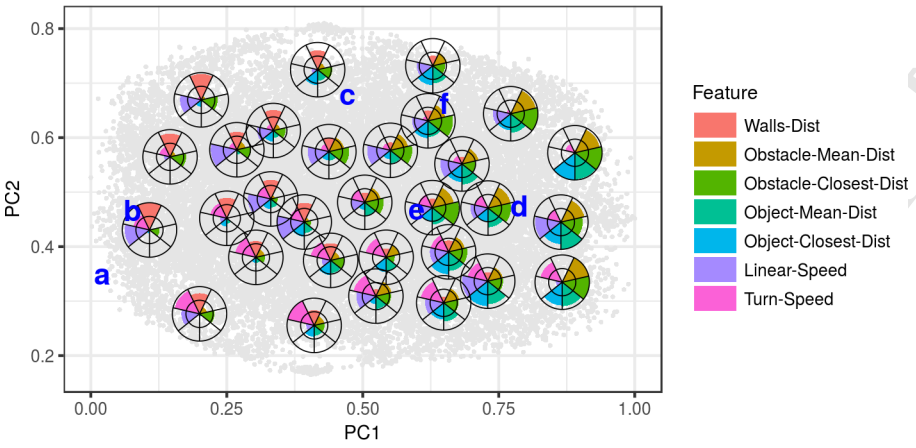


Figure 6: Behaviour characterisations that can be found in different regions of the behaviour space. Each circular sector corresponds to a behaviour feature. The bigger the radius of the segment, the higher the feature value, and vice-versa. The letters indicate the position in behaviour space of the primitives shown in Figure 7.

away from POIs (f). Overall, the analysis of the repertoire reveals that the evolutionary process was able to evolve a highly diverse set of controllers, with clearly distinguishable behaviours.

4.2. Repertoire Quality and Versatility

We evaluate the primitives of each repertoire in the tasks presented in Section 3.2 to assess the quality and potential of the repertoire primitives. We evolved 30 repertoires with novelty search, and randomly generated another 30 repertoires. For each repertoire, we then evaluated all the primitives in the nine tasks, thus obtaining a fitness measure of how well each primitive performs in the given task. Note that there is no arbitrator involved yet: each primitive is used to control the robot in the given task from the start to the end of the simulation. The results are shown in Figure 8.

The best controllers evolved by tabula-rasa (TR) are able to achieve significantly higher fitness scores than the primitives in the NS-evolved repertoires (NS) ($p < 0.001$, Mann-Whitney)[2]. This result was expected, given that tabula-rasa evolves controllers specifically for solving each of the tasks, while in the *NS* setup we are using previously-evolved general primitives to attempt to solve specific tasks. Nevertheless, it is noteworthy that, in some tasks, such as Phototaxis, Avoidance, or Exploration, the fitness achieved by the highest-performing

[2]Throughout the paper, we use the Mann-Whitney U test (nonparametric) for comparing two independent samples, and the Kruskal-Wallis H test (nonparametric) for comparing more than two samples. The p values are adjusted using the Holm-Bonferroni method when multiple comparisons are made. The significance level is set to $p = 0.05$.

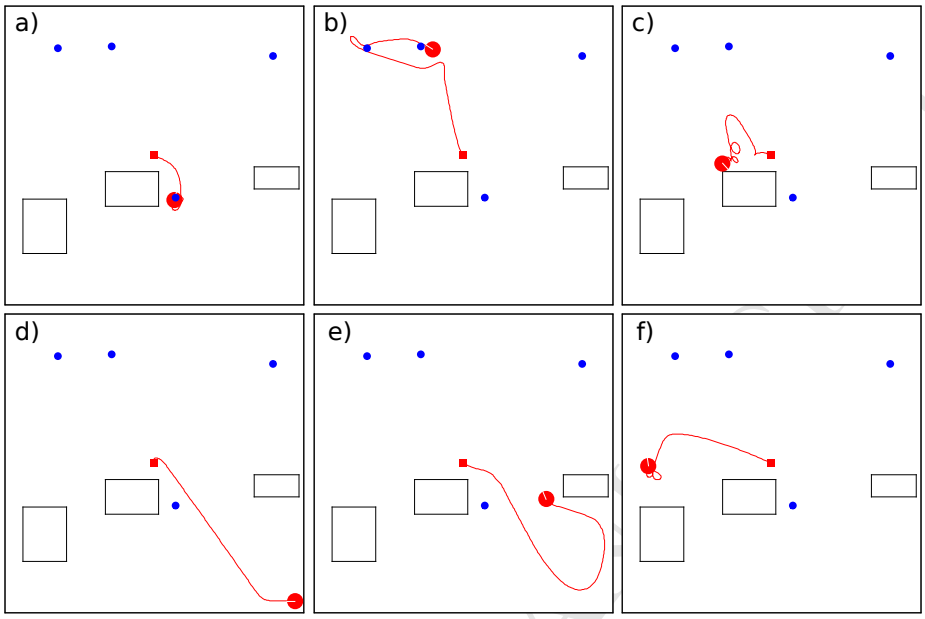


Figure 7: Robot trajectories described by six different primitives, in one randomly generated environment. The robot is depicted in red, and the POIs are shown in blue. The squares represent the initial position of the robot, and the filled circle its final position.

primitive in each of those tasks is relatively close to the fitness achieved by tabula-rasa.

In the case of Phototaxis, for instance, a fitness of 0.83 (TR) means that the robot reached the POI, and took on average 17 s to do so, while a fitness of 0.73 (NS) means that it took on average 27 s. In the case of Avoidance, a fitness of 0.58 (TR) means that the robot managed to avoid dynamic POIs during 58% of the time, while a fitness of 0.51 (NS) means that it avoided POIs during 51% of the time. In the case of Exploration, a fitness of 0.72 (TR) means that the robot visited on average 18/25 regions, while a fitness of 0.61 (NS) means that it visited on average 15/25 regions. These relatively small differences, mean that the robot controlled by a single primitive was able to solve some tasks reasonably well, although not with the same level of performance as the controllers evolved by tabula-rasa.

The results in Figure 8 additionally show that the evolved repertoires contain primitives that yield significantly higher fitness scores in all tasks ($p < 0.001$), when compared to repertoires composed of randomly generated primitives. This confirms that the wider diversity discovered by novelty search, as seen in Section 4.1, translates into a larger number of potentially useful primitives.

To gain insight over the composition of the evolved repertoires, we measured the fitness achieved by each primitive of each repertoire in each task. We then

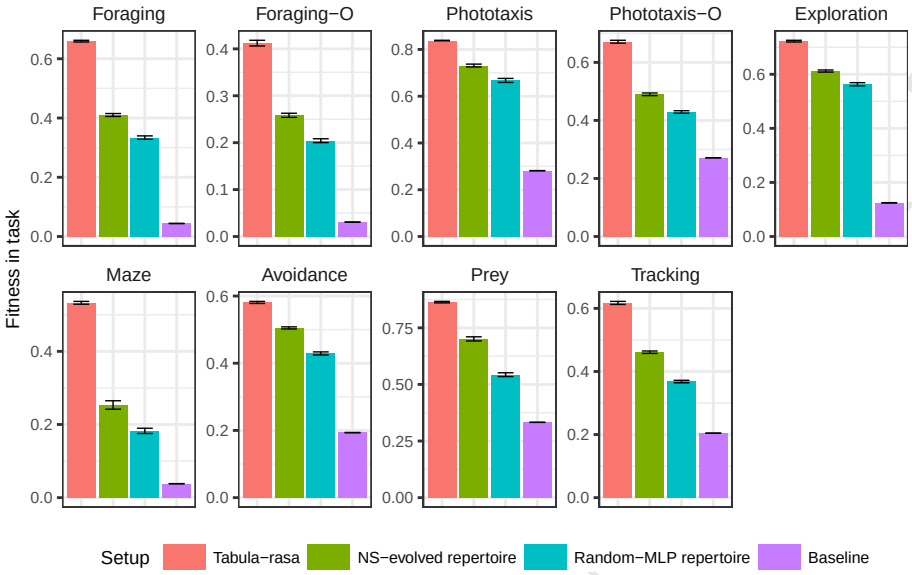


Figure 8: Highest fitness scores achieved by tabula-rasa, by the repertoires evolved with novelty search, and by the repertoires composed of randomly-generated networks. The baseline corresponds to the fitness obtained on average by a randomly generated neural network controller.

averaged the fitness scores achieved by the primitives in each behavior region, taking into account all the primitives in the 30 NS-evolved repertoires. The results are shown in Figure 9. First, it is possible to observe that most of the regions of the repertoire contain primitives that perform relatively well in some task. This shows that a large part of the repertoires is composed of potentially useful primitives. Second, the primitives that work best for some task tend to be focused around a certain behaviour region, which suggests that the behaviour space is moderately continuous, and there is a real and consequential diversity of primitives. Third, the best-performing primitives for each task are aligned with our intuition of how each task should be solved. For instance, for solving the Phototaxis(-O), Foraging(-O), and Tracking tasks, the best primitives are from a behaviour region characterised by low distance to POIs (see Figure 6, left); for the Avoidance and Prey tasks the best primitives belong to a region characterised by high distance to POIs (right); for the Exploration and Maze tasks, the primitives that perform best belong to a central region characterised by low turn speeds.[3] This suggests that the behaviour characterisation is ade-

[3]The minimum value of turn-speed corresponds to turning sharply to the left, and the maximum value corresponds to turning sharply to the right. Low turn-speeds thus correspond to the intermediate values. The same logic applies to the linear-speed, where the minimum is moving backwards, the maximum is moving forwards, and the intermediate values represent low speeds.

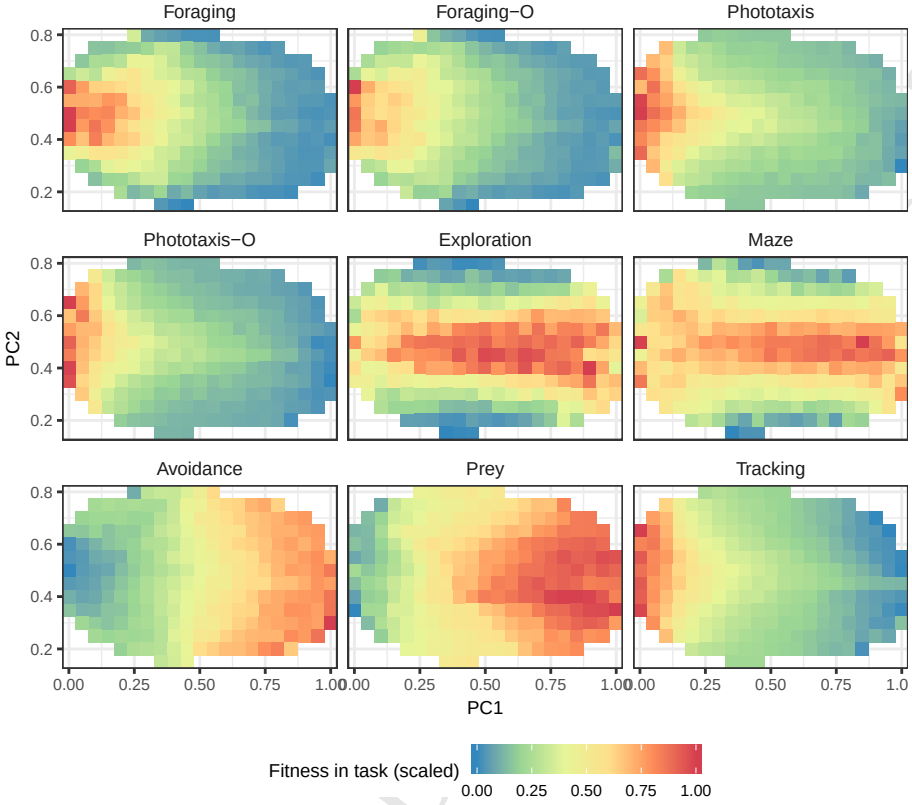


Figure 9: Mean fitness of the primitives in the NS-evolved repertoires, in each of the test tasks. The behaviour space was discretised for calculating the mean fitness of the primitives in each behaviour region.

quately capturing the primitives' behaviour, as the characterisations are aligned with our intuition of how each task should be solved.

4.3. Repertoire Evolution Environment

One of the inputs that the experimenter must provide in EvoRBC-II is the set of environments used for evaluation of primitives during the repertoire generation. In this section, we study the impact of this choice, and what is the relation between the environments used for repertoire generation, and the environment of the tasks for which the arbitrator is evolved. Besides the environment used in the previous experiments, referred to as *Base*, we evolve repertoires using five other environments, described in Table 3.

Each environment was used to evolve 30 different repertoires in independent evolutionary runs. Each of these repertoires was then assessed by testing the primitives directly in the tasks, following the methodology used in Section 4.2. The fitness scores of the best-performing primitives from the different repertoires

Table 3: Different environments used for repertoire generation. In all cases, 100 environments with randomly generated positions for obstacles, POIs, and the robot, are used to evaluate each primitive.

Environment	# Obstacles	Obs. side (cm)	# POI	POI type	Time (s)
Base	[1,10]	[10,30]	[1,10]	Static	50
Fixed	5	[10,30]	5	Static	50
Few	[1,5]	[10,30]	[1,5]	Static	50
No-obstacles	0	–	[1,10]	Static	50
No-POIs	[1,10]	[10,30]	0	–	50
Only-walls	0	–	0	–	50

are presented in Figure 10. The results show that the choice of evolution environments has a significant impact in all tasks ($p < 0.001$, Kruskal-Wallis test), except in Phototaxis ($p = 0.11$). The *Base* environment yields fitness scores similar to the *Fixed* and *No-Obstacles* environments across all tasks (Mann-Whitney test, $p < 0.05$). The *Few* environment yields significantly lower fitness scores in three tasks, and higher fitness scores in one task (Tracking). The *No-POIs* and the *Only-walls* environments consistently and significantly yield the worse results, except in the Phototaxis task and also the Exploration task in the case of *No-POIs*.

The significant performance drop observed with *No-POIs* and *Only-walls* is understandable, given that seven out of nine tasks require the interaction with POIs. The primitives in these repertoires are not adapted to process POI sensory information, and can therefore display erratic behaviour when acting in environments with POIs. Although six out of nine tasks have obstacles, the *No-Obstacles* environment actually yields relatively good fitness scores. This can potentially be explained by the fact that the robot's sensors make no distinction between obstacles and the walls that bound the environment. Since all environments have walls, the primitives in the repertoire are likely adapted to deal with that sensory information. The fact that *Fixed* (fixed number of obstacles and objects) and *Few* yielded a performance similar to *Base* is ascribed to the fact that all elements, as well as the robot, are initially placed in random positions. This alone appears to be enough to expose the robot to a sufficiently wide range of scenarios.

These results show that it is important that the environment used for repertoire generation contains the elements that the robot might face in solving the tasks. That way, the primitives can be adapted to deal with such sensory information. Other than that, our results suggest that the environment used for the evolution of the repertoire is moderately robust to changes in its parameters and configuration.

4.4. Arbitrator Evolution

In this section, we assess the performance of the complete EvoRBC-II approach in solving the evaluation tasks presented in Section 3.2. We evaluate

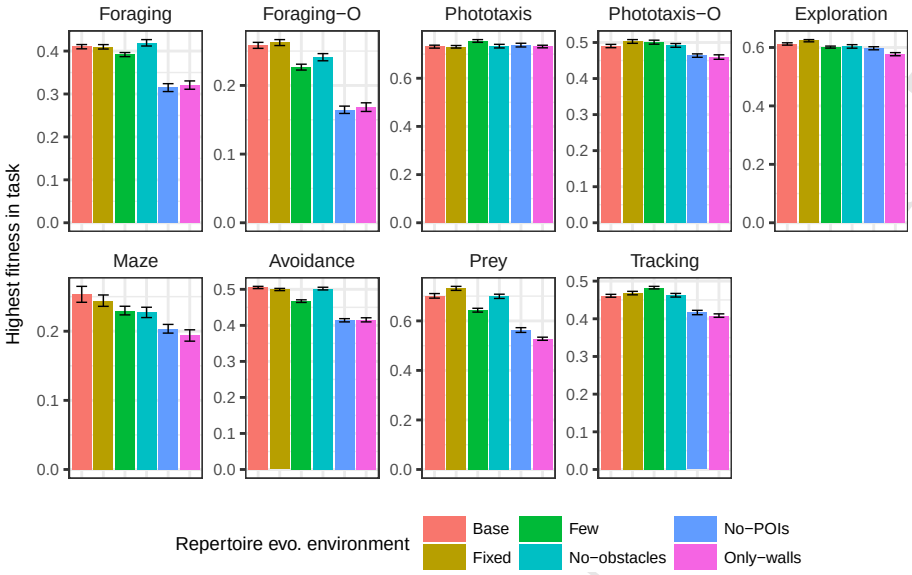


Figure 10: Fitness scores achieved in each task, using repertoires evolved in the different environments.

EvoRBC-II using ten different repertoires evolved with novelty search (see Section 4.1), and compare it with the fitness scores obtained by using the primitives from those repertoires directly in the tasks (as done in Section 4.2), and with the traditional tabula-rasa evolutionary approach. For EvoRBC-II, a total of 100 evolutionary runs were conducted for each task (10 repertoires $\times$ 10 arbitrator evolution runs), while for tabula-rasa a total of 30 evolutionary runs was conducted per task. For a fair comparison, the arbitrator evolution in EvoRBC-II and tabula-rasa were allowed the same total number of individual evaluations in each evolutionary run, see Table 1. Note that an absolutely fair comparison is hard to achieve, as EvoRBC-II uses additional resources for the repertoire evolution, but the same repertoire can be used to solve many tasks, and the repertoire evolution uses a significantly different evolutionary setup. The results of the fitness scores achieved in each task are presented in Figure 11.

Comparing EvoRBC-II with the primitives from the repertoires, EvoRBC-II achieves significantly superior fitness scores in all tasks (Mann-Whitney, $p < 0.001$). This shows that the combination of multiple primitives was always preferable to any single primitive in the repertoire, and that the evolutionary process was capable of selecting from the repertoire the most suitable primitives for solving each task. Comparing EvoRBC-II with tabula-rasa, EvoRBC-II achieves slightly inferior fitness scores (around 5% less) across all tasks, except in the Maze task where the difference is more pronounced. In the Foraging-O task, for instance, the different in fitness scores between tabula-rasa (TR) and EvoRBC-II means that the controllers evolved by TR were able to capture on av-

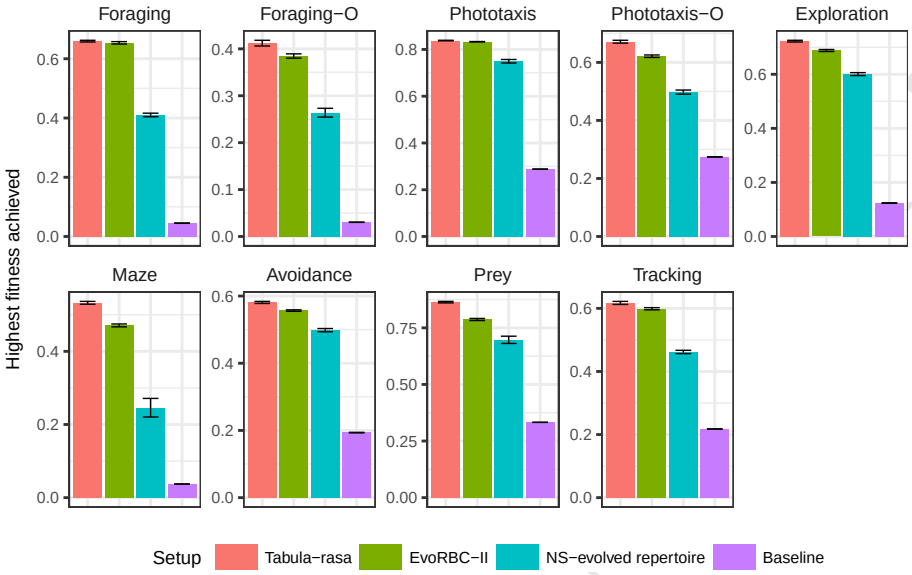


Figure 11: Mean of the highest fitness scores achieved in each run, for each evaluation task. The fitness scores are scaled to $[0, 1]$, according to the minimum and maximum fitness scores ever observed for each task.

erage 0.4 more items (6.2/15 vs 5.8/15); in Phototaxis-O, the controllers evolved by TR reached the POI five seconds earlier on average (33/50 s vs 38/50 s); in the Prey task, the controllers evolved by TR survived on average four seconds longer (43/50 s vs 39/50 s); and in the Exploration task, the controllers evolved by TR visited on average one region more (18/25 vs 17/25).

These results show that the EvoRBC-II approach is highly versatile: EvoRBC-II was able to achieve relatively good fitness scores in all tasks, and the results suggest that the repertoires contained the necessary primitives to reasonably solve all of the tasks. The fact that EvoRBC-II did not outperform tabula-rasa with respect to fitness scores is justified by the simplicity of the tasks: none of the tasks is particularly challenging nor deceptive, and hence the tabula-rasa approach can rapidly converge to good solutions in any of the tasks. As discussed in Section 3.2, these nine tasks are used to study the versatility of the EvoRBC-II approach, not its capability of outperforming competing evolutionary approaches with respect to fitness scores.

4.5. Arbitrator's Behaviour and Repertoire Usage

We analysed the highest-fitness controllers evolved in each evolutionary run, for each task, in order to understand how the EvoRBC-II controllers are solving the task. For each task, we simulated each of these controllers in 10 simulation runs, and recorded the primitives that were selected during the task execution. The results in Figure 12 show the primitives that were used for solving each task, averaged over all the evolutionary runs of EvoRBC-II for each task.

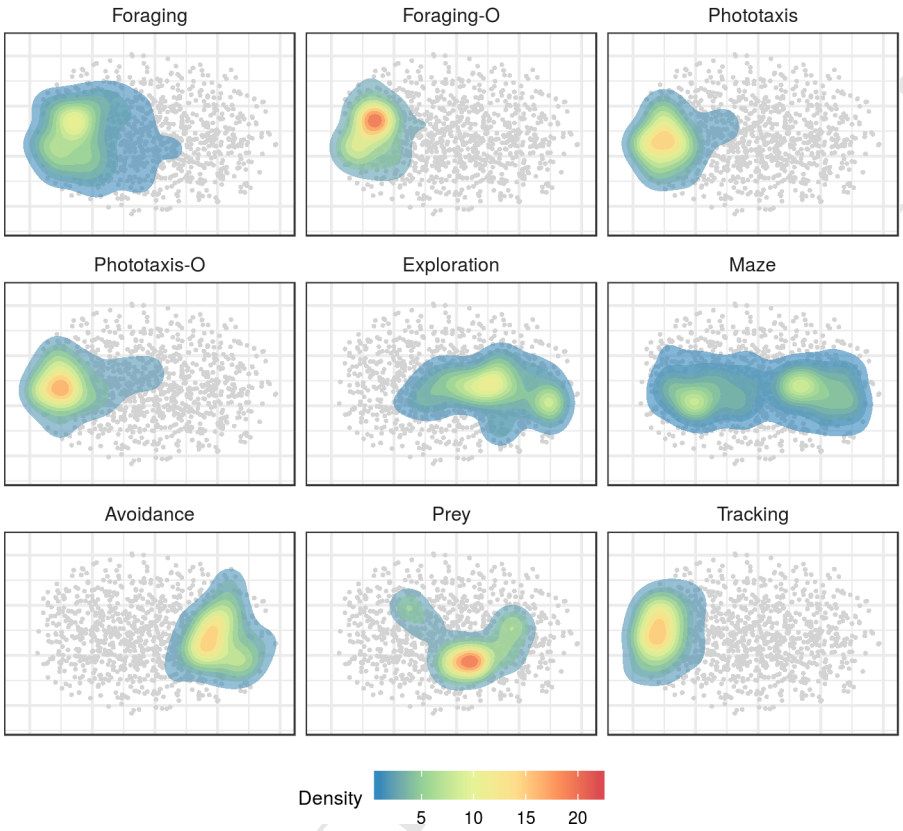


Figure 12: Primitives that were used for solving each task, mapped according to their behaviour characterisation. Higher density means that primitives from that region were selected more often. The behaviour space is reduced to two dimensions using RPCA. See Figure 6 for an example of behaviour characterisations that can be found in each region.

As Figure 12 shows, the arbitrators evolved for different tasks tend to use primitives from different regions of the repertoire. For instance, for solving the *Phototaxis* and *Phototaxis-O* tasks, the arbitrators tend to use primitives characterised by very low distance to POIs (see Figure 6 for the correspondence between the regions in 2D space and the behaviour characterisations). For the *Prey* task, on the other hand, they rely on primitives characterised by large distances to POIs. For *Exploration*, the arbitrators use primitives characterised by high linear speeds and some distance to obstacles. Additionally, it should be noted that most of the regions of the repertoire were used by some task. Comparing these results with the results shown in Figure 9, where we analysed the fitness of each individual primitive in the tasks, we can see that the arbitrator tends to select primitives that perform relatively well in the task. This observation confirms that the evolutionary process of the arbitrator is taking advantage of the primitives available in the repertoire, selecting and combining

Table 4: Usage of the repertoire by the highest-fitness arbitrators evolved in each evolutionary run, averaged over 1000 simulations (100 evolutionary runs × 10 simulations). *Tree splits*: mean number of splits in the arbitrator's decision tree. *Prims. used*: mean number of different primitives used per simulation. *Prim. duration*: mean number of consecutive control cycles each primitive was selected. *Most used*: total time occupied by the most used primitive in a given simulation. The standard deviation across the multiple controllers is shown in parentheses.

Task	Tree splits		Prims. used		Prim. duration		Most used (%)	
Foraging	11.8	(5.1)	6.5	(2.2)	13.5	(49.3)	49.5	(14.6)
Foraging-O	9.4	(6.2)	5.1	(2.1)	31.3	(63.1)	69.6	(17.5)
Phototaxis	3.1	(1.6)	2.5	(0.8)	31.0	(29.1)	75.0	(14.4)
Phototaxis-O	6.9	(3.6)	4.2	(1.6)	54.9	(56.6)	70.5	(14.4)
Exploration	7.1	(4.6)	4.7	(2.2)	89.1	(168.1)	79.4	(14.9)
Maze	8.9	(4.0)	5.0	(1.9)	13.3	(18.4)	59.7	(13.3)
Avoidance	6.5	(3.7)	4.3	(1.8)	35.4	(63.7)	67.0	(13.8)
Prey	3.2	(2.2)	2.4	(0.9)	140.4	(111.1)	87.5	(11.3)
Tracking	9.5	(4.1)	6.1	(2.1)	17.2	(15.8)	59.7	(13.6)

suitable primitives for solving each task.

To analyse how the arbitrators perform during task execution, we extracted a number of different metrics on how the arbitrators select and switch between different primitives, see Table 4. The data shows that the evolved trees tend to be relatively small in size, with the average tree size varying modestly from task to task (Kruskal-Wallis, $p < 0.0001$). The arbitrators tend to use a relatively low number of primitives, around 3–6 primitives (out of a total of 1000 primitives in the repertoire), also with significant differences depending on the task (Kruskal-Wallis, $p < 0.0001$). Note that the number of primitives used is typically smaller than the number of tree splits, which means that the same primitive often appears in different branches of the tree. The results also reveal that even when a larger number of primitives is used, there is always a single primitive that dominates the others. That is, most of the simulation tends to be spent on a single primitive, switching momentarily to different ones as needed.

The amount of time each primitive is continuously executed varies largely from task to task ($p < 0.0001$). While in some tasks (such as Foraging, Maze, and Tracking) the arbitrator frequently switches primitive, in other tasks (such as Exploration and Prey) each primitive is continuously executed for a long time. Frequent switching between primitives can have different causes, for instance: (i) there might not be any single primitive in the repertoire to control the robot as needed in the given situation, and thus the arbitrator has to *combine* multiple primitives; or (ii) the task itself requires frequent switching between significantly different actions.

4.6. Analysis of Selected Solutions

In this section, we visually inspect some of the best arbitrators evolved by EvoRBC-II. We selected four tasks, and for each of these, we chose one of

highest-fitness arbitrators with the smallest tree size[4]. The arbitrator trees and the trajectories of the robot being controlled with those arbitrators are shown in Figure 13. All four inspected arbitrators have decision trees with typical sizes, as shown in Table 4. Note that these four depicted controllers do not necessarily represent the entire universe of the best evolved solutions (9 tasks × 100 best-of-run solutions). For a more general and quantitative analysis, please refer to Section 4.5.

The behaviours illustrated in Figure 13 confirm that the strategy adopted by the arbitrator varies widely depending on the task, as the results in Section 4.5 have shown. For the Phototaxis task, for instance, a single primitive (#371) was used from the beginning to the end. Although the decision tree has three other primitives, these are only used in specific circumstances (when the robot has the POI on its back, for instance). For the Phototaxis-O task, the arbitrator used a larger number of primitives, but three primitives alone (#614, #291, and #303) accounted for ≈90% of the execution time. For the Prey task, primitive #993 accounted for 93% of the execution time, with the two other primitives occupying the remaining time. The Foraging task was one of the tasks where the arbitrators switched primitives more often on average, and had the largest trees. In the simulation shown in Figure 13d, however, primitive #574 (which appears in four different branches) alone accounted for 56% of the simulation time, with the remaining time being occupied approximately equally by primitives #101, #553, and #274.

Based on our visual inspection, on both these trials and others not reported in this paper, the solutions evolved by EvoRBC-II were qualitatively similar the solutions evolved by tabula rasa, with respect to the intelligibility of the robot's movement, and perceived quality of behaviour. One notable and expected difference is that while TR-evolved behaviours tend to be smoother and more continuous, the controllers evolved by EvoRBC-II exhibit more discrete changes of behaviour. These discrete changes typically do not cause the behaviour to appear erroneous, and actually tend to make the behaviours more intelligible. Note that in some of the illustrated trials, while the robot's behaviour might appear sub-optimal, it can actually be hard to do significantly better given the robot's limited sensory capabilities and the random configuration of the environments. For instance, in Phototaxis-O (Figure 13b), the robot is not within range of the target at the beginning of the trial, and it thus has to search. In the Foraging task (Figure 13d), it is practically impossible to capture all the items in the allowed time, and it is thus likely that the evolved behaviour purposely leaves some items behind.

4.7. Comparison with Open-Loop Primitives

In the original version of EvoRBC [18], the repertoires were composed of open-loop locomotion primitives, contrasting with the repertoires composed of

[4]Among the top 10% of the best-of-run arbitrators, for each task.

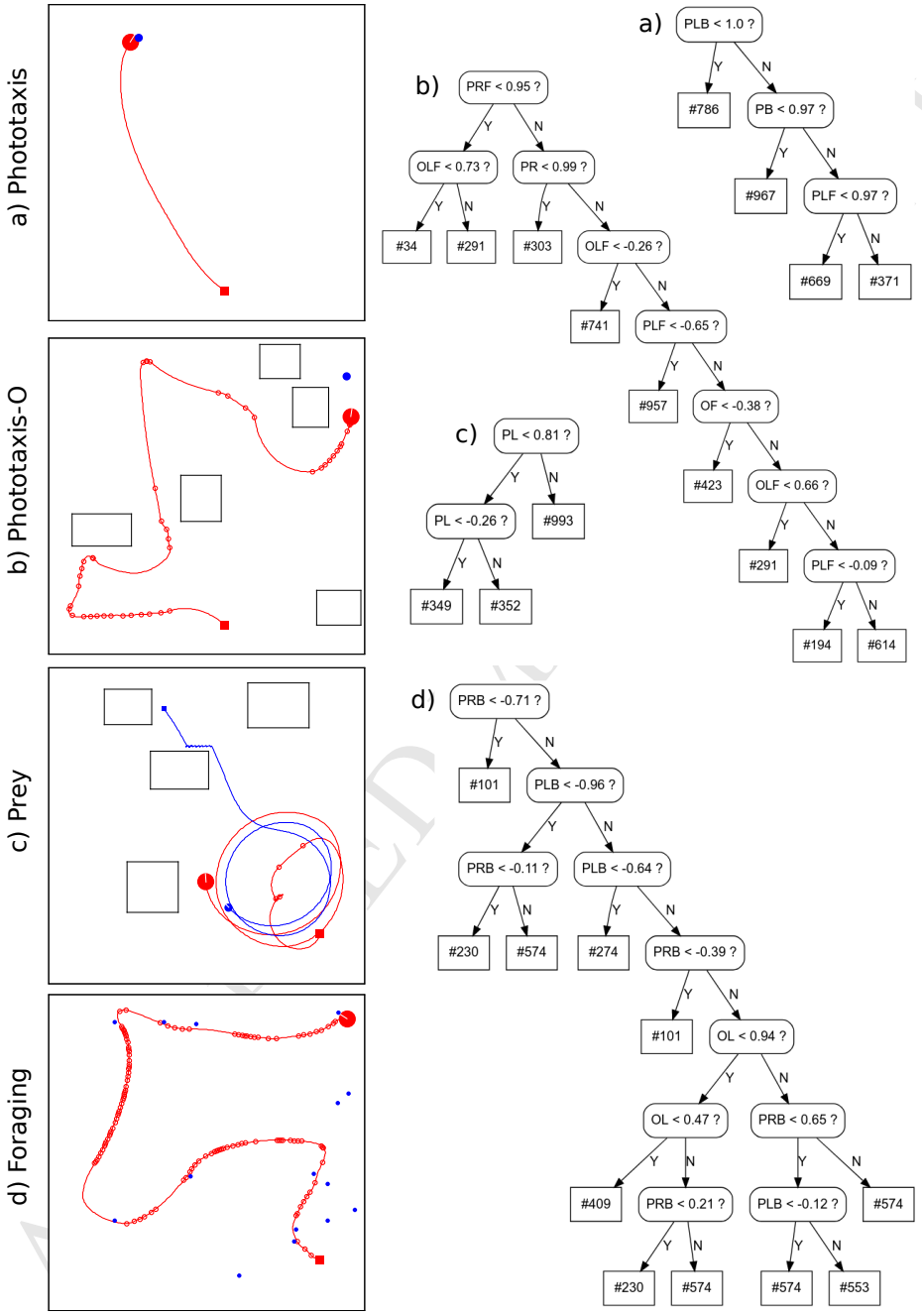


Figure 13: The red features depict the agent, while the blue features depict POIs. The square indicates the initial position of the robot, and the circle its final position. The small circles along the trajectory depict the instant the controller switched primitive. The sensor values used in the tree splits are named as following: the first letter indicates whether it is an obstacle sensor (O) or POI sensor (P), and the remaining letters indicate the sensor's direction – front (F), right-front (RF), right (R), right-back (RB), back (B), left-back (LB), left (L), or left-front (LF).

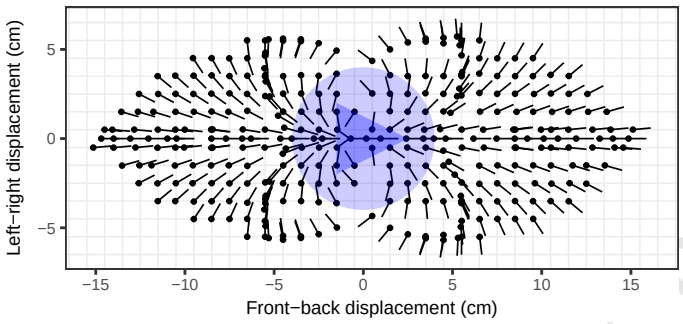


Figure 14: One of the evolved repertoires, with size 315 and average fitness of 0.982. Each locomotion primitive in the repertoire is represented by a marker, which is composed of a small, filled circle (displacement) and a line (final orientation). The blue circle and triangle depict the robot in its initial position and orientation. The grid corresponds to the grid used by the MAP-Elites algorithm.

general closed-loop primitives proposed in this paper. In this section, we establish a comparison between the two approaches. It should be noted that the original EvoRBC used a neural network as top-level arbitrator. For a more informative and fair comparison, in this study we also use decision trees as the arbitrator for EvoRBC, as done in [48]. This means that both EvoRBC and EvoRBC-II use the exact same algorithm and parameters for the evolution of the arbitrator (see Section 3.4), with the only difference residing on the type of repertoires used.

The locomotion repertoire is evolved using the methodology described in [18]. Each locomotion primitive is a vector of length two, corresponding to the two locomotion parameters of the robot (left and right wheel speeds). We use the MAP-Elites algorithm [13] to evolve the repertoire (parameters in Table 1). The behaviour space is defined as a two-dimensional space where each point $\langle x, y \rangle$ represents the displacement of the robot after executing the primitive for a fixed amount of time (1 s). A fitness score is assigned to each primitive based on the *Circular fitness metric* used in [15, 18, 20]. This metric favours the evolution of locomotion primitives where the robot follows circular trajectories. The fitness is calculated based on the difference between the robot's final orientation (λ) and a desired orientation (ω) for that point in space $\langle x, y \rangle$. An example of an evolved locomotion repertoire is shown in Figure 14.

The fitness scores achieved by each approach for each task are shown in Figure 15a. EvoRBC-II performs slightly better than EvoRBC in Phototaxis and Avoidance, the opposite is observed in Foraging, Phototaxis-O, and Maze, and no sigificant differences were found in the remaining tasks. In all cases, the differences are negligible in terms of absolute fitness scores achieved. This similarity in the fitness scores achieved contrasts with the large difference in the primitives that compose each repertoire. To understand why this is the case, we analysed the arbitrator trees of the best controllers of each evolutionary run, see Figure 15b–d.

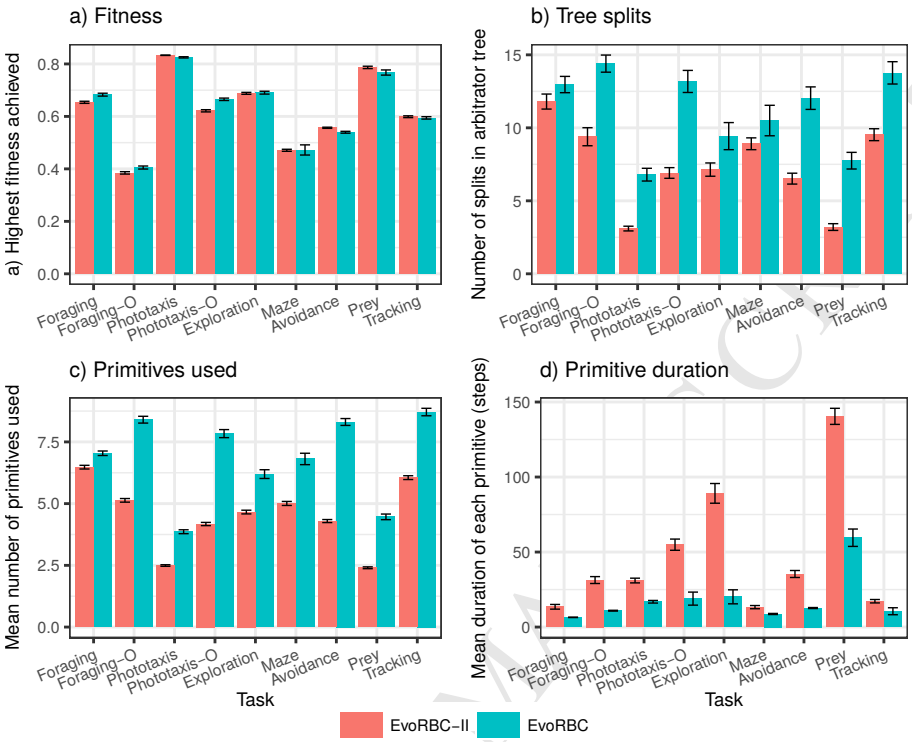


Figure 15: Comparison of EvoRBC-II (using general closed-loop primitives) with EvoRBC (using locomotion open-loop primitives). a) Fitness achieved in each evaluation task. b) Average size of the best decision trees (measured in terms of tree splits) evolved for solving each task. c) Number of different primitives used by the best controllers during task execution. d) Average time each primitive was kept active in the best controllers.

EvoRBC consistently produces significantly larger trees than EvoRBC-II ($p < 0.05$, except in Foraging task, Figure 15b), that use a larger number of primitives ($p < 0.001$, Figure 15c), and that switches between them more frequently ($p < 0.05$, except in Phototaxis and Maze tasks, Figure 15d). These results suggest that EvoRBC is able to solve the tasks because it compensates with more capable arbitrators. That is, the complexity shifts from the lower level to the higher level. With EvoRBC-II, on the other hand, the top-level arbitrator tends to remain significantly more simple, since the more complex sensor-actuator mappings are handled by the primitives at the lower level.

It is important to note that EvoRBC and EvoRBC-II have significantly different purposes. EvoRBC is suited for evolving control for robots with complex locomotor systems, while EvoRBC-II is suited for evolving higher-level hierarchical control. The primitives in the locomotion repertoires used by EvoRBC would be incapable of solving a task without any arbitrator, as we have shown possible with the EvoRBC-II repertoires (see Section 4.2). On the other hand, EvoRBC-II could struggle when dealing with robots with complex locomotor

systems, as the evolution of the repertoire of general behaviours could be hindered by the difficulty in moving the robot at all. A combination of the two approaches is possible and should be studied in future work, as we discuss below in Section 5.5.

5. Discussion

In this paper, we proposed EvoRBC-II, an approach that divides the evolution of robot control into two stages: the evolution of a repertoire composed of general closed-loop behaviours (primitives); and the evolution of a higher-level arbitrator that during task execution dictates which primitive from the repertoire should be used to control the robot. We relied on nine different simulated tasks, based on tasks commonly used in evolutionary robotics studies, to assess the performance of EvoRBC-II. Each evolved repertoire was used to evolve arbitrators for each of those nine tasks. We compared EvoRBC-II with the traditional direct evolution (tabula-rasa), and also studied the capability of the primitives from the repertoires to solve the tasks directly, without any arbitrator. In this section, we discuss the main findings of our study, and outline directions for future work.

5.1. Repertoire Evolution

We proposed and studied the evolution of repertoires of general and closed-loop behaviours. This contrasts with previous work (including the original EvoRBC) that has focused on repertoires of open-loop locomotion primitives [15, 17, 18]. To evolve repertoires of general primitives, we proposed an approach where each primitive is a neural network controller that receives the robot's sensor values and outputs the actuator values. Each primitive is evaluated by letting the robot act in the environment while executing the primitive. A large number of randomly generated environments are used for each evaluation (100 environments in our experiments), in order to assess the general behaviour of the primitive. The repertoire is evolved using the novelty search algorithm and a generic (task-agnostic) behaviour characterisation.

Our results showed that novelty search was an effective strategy for exploring the behaviour space, and that the evolved repertoires contained a wide diversity of primitives. Primitives from different regions of the repertoire exhibited clearly distinguishable behaviours. To assess the potential usefulness of the primitives in the evolved repertoires, we evaluated the primitives from the repertoires directly in the nine tasks, without any arbitrator involved. Our results showed that (i) a large portion of the repertoires contained primitives that were useful to solve some task, and (ii) for some tasks, single primitives could achieve a fitness score close to the fitness achieved by the specialised solutions evolved by tabula-rasa. Overall, we showed that the evolved repertoires were composed of a wide diversity of potentially useful primitives.

5.2. Hierarchical Control

In order to leverage the evolved repertoires, we relied an approach in which the arbitrators are decision trees, as recently proposed in [48], where the splits are based on sensory readings, and the terminals are primitives from the repertoire. The decision trees are induced using genetic programming. The decision tree arbitrators offer a number of advantages over the neural-based arbitrators proposed in the original EvoRBC: (i) they are transparent and understandable by the experimenter; (ii) they are only weakly affected by the dimensionality of the repertoire; and (iii) it is easier for the arbitrator to keep a certain primitive active for longer, which is desirable since a single primitive can have considerable capabilities.

Our results showed that the same repertoire could be used to solve a wide variety of tasks, nine in total, closely matching the performance of tabula-rasa evolution in all the evaluation tasks. The evolved arbitrators relied on primitives from different regions of the repertoire to solve different tasks, thus showing that the arbitrators were capable of selecting suitable primitives for solving each task. We also showed that the arbitrators were able to switch between different primitives during task execution as needed, which ranged from keeping the same primitive active for most of the simulation, to frequently switching between a few primitives. The number of different primitives used by an arbitrator to solve the task was typically low (around 3–6), and the respective decision trees tended to be small (from 3 to 10 splits, depending on the task). All this means that the arbitrator trees can easily be understood by the experimenter, and the decision process can be followed, contrasting with the neural network-based approach used in the original EvoRBC. Overall, we showed that the evolutionary process was able to find arbitrators that were well adapted to the complexity and specificities of each of the tasks.

The value of having hierarchical control based on higher-level primitives was confirmed by replacing the repertoires of closed-loop primitives with repertoires of simple open-loop locomotion primitives. The results showed that the arbitrators that relied on closed-loop primitives were significantly less complex than the arbitrators that relied on locomotion primitives, and switched less often between primitives. This confirms that the arbitrators evolved by EvoRBC-II can rely on the primitives in the repertoire for dealing with potentially complex sensor-actuator mappings.

5.3. Parameters and Configuration

In its current form, the following main decisions are required by to apply the EvoRBC-II approach:

Repertoire evolution environment. Our results showed that the environments used to evolve the primitives in the repertoire can have a significant impact on the usefulness of those primitives, but overall, we observed a high degree of robustness to the choice of environments. Some evaluation tasks used environments that were never used in the repertoire evolution, and EvoRBC-II still

managed to solve those tasks effectively. We found that the main requirement is that the environments used for repertoire evolution should contain all elements that the robot will encounter while solving tasks. This way, the primitives in the repertoire are adapted to deal with the sensory inputs they might encounter.

Behaviour characterisation. The behaviour characterisation is key in driving the repertoire evolution. In this study, we relied on Systematically Derived Behaviour Characterisations (SDBC), which allowed us to obtain a relatively short behaviour characterisation (seven features), with minimal intervention or bias from the experimenter. The behaviour characterisation was not fine-tuned for the experiments. In future work, we will study if and how other generic characterisations (see Section 2.1.3) are suitable for the EvoRBC-II aproach.

Evolutionary parameters. The repertoire evolution relied on the NEAT neuroevolution algorithm and novelty search, using *default* parameters, commonly used in the literature [42]. The use of NEAT means that only the number of inputs (sensors) and outputs (actuators) had to be specified, without the need to specify the neural architecture any further. Regarding the arbitrator evolution, the genetic programming algorithm used to induce the decision-tree arbitrators used a minimal function set (sensor-based splits, and primitive terminals). The genetic operators and parameters were also based on previous work, with only two noteworthy differences: (i) there is a relatively high probability of mutating the primitive terminals (meaning replacing the primitive with a different one from the repertoire), which is needed in order to ensure that suitable primitives are found and placed on the trees; and (ii) the trees are pruned after evaluation to remove all primitive terminals that were never executed during the simulations, in order to control bloat and thus keep the trees intelligible.

5.4. Applicability

For tasks as simple as the one used in this study, the advantages of EvoRBC-II are naturally limited in terms of the fitness scores achieved in those tasks. However, the sole fact that EvoRBC-II was able to practically solve all evaluation tasks, closely matching the performance of direct evolution, is a significant accomplishment. It is especially noteworthy that the same repertoire contained primitives that allowed the solution of all nine evaluation tasks. Our study highlights the versatility of the EvoRBC-II approach, and its potential to scale to more complex problems.

As studied in previous work [35, 36], hierarchical behaviour decomposition has the potential to offer significant advantages in evolutionary robotics:

1. It can allow the achievement of more complex behaviours [36], by bootstrapping the evolutionary process with pre-existing behaviour primitives.
2. The primitives can be forced to meet certain criteria, and as the arbitrators can only use the available primitives, the robot controller will also likely meet those criteria. For instance, the primitives can be optimised for energy efficiency, safety of operation, transferability, and so on.

3. One particular instance of this capability is enabling incremental transfer. Behaviour decomposition can facilitate the transfer of evolved control to real robots, one of the biggest challenges in evolutionary robotics [3, 57, 58]. The primitives from the repertoire can be assessed in the real robots independently, and the repertoire can be filtered based on the transferability of the primitives. If the primitives are able to transfer successfully, the same will likely be observed in the hierarchical controller that uses those primitives [59].
4. Hierarchical control allows for behaviour reuse, meaning that previously evolved controllers can potentially be reused to rapidly solve new tasks or face new challenges [14, 59]. We have shown, for instance, that by using only a single primitive from the repertoire, we could achieve relatively high fitness scores in some tasks.
5. Hierarchical controllers can be more intelligible and understandable by the experimenter [36, 59]. Such transparency is highly beneficial for use cases where the experimenter wants to modify or verify the robot's behaviour.

EvoRBC-II is an approach that allows for these advantages to be explored, while mitigating the main disadvantage of hierarchical decomposition: the need to manually decompose the behaviour into suitable sub-tasks [36].

5.5. Future Work

The promising results obtained with EvoRBC-II open up interesting avenues of research and further development. As mentioned in the previous section, the evolution of the repertoire could additionally have the objective of generating robust primitives, that is, primitives that would likely transfer successfully from simulation to the real robots. This could be accomplished by using Novelty Search with Local Competition (NSLC) or MAP-Elites instead of pure novelty search [19], with the fitness objective being the robustness or transferability of the primitive [15].

While we obtained relatively good results with decision trees as arbitrators, the use of neural networks for the arbitrator should not be disregarded. As we show in [48], there are several different arbitrator architectures that can be considered, and we have obtained good results in the past when using neural networks for arbitrating among locomotion behaviours [18]. Alternative arbitrators and alternative methods for synthesising arbitrators can potentially be better suited for different kinds of sensory inputs (high-dimensional input from camera sensors, for instance), and should thus be studied further.

For robots with complex locomotor systems, the evolution of a repertoire of locomotion primitives, as done by Duarte et al. [18], could be a first step before the generation of repertoires of general behaviours. This would introduce an additional layer of hierarchical control: (i) a repertoire of locomotion primitives would be evolved for the given robot; (ii) a repertoire of general closed-loop behaviours would be evolved using the locomotion repertoire; and (iii) the arbitrator evolved for solving the given task could use behaviours from both repertoires. A similar architecture has been recently proposed by Cully et al. [60].

6. Conclusion

We proposed EvoRBC-II, an evolutionary approach that divides the evolutionary process into two steps: the evolution of a repertoire of general behaviour primitives, and the evolution of a higher-level arbitrator for a specific task, that switches between primitives of the given repertoire. Our contribution is twofold: first, we studied for the first time how to evolve repertoires of general-purpose behaviours, that can respond to the robot's sensory inputs. We showed that such repertoires can be obtained by relying on generic (task-agnostic) behaviour characterisations, and by evaluating each individual in a large number of randomly generated environments, which help assess the general robot behaviour. Second, we proposed an approach, based on EvoRBC [18], that allows for these repertoires of general behaviours to be leveraged by the top-level arbitrator to solve specific robotics tasks. We proposed a new architecture for the arbitrator: decision trees where the splits are based on sensory readings and the terminals are primitives from the repertoire.

Our results show that the evolved repertoires are highly diverse and contain many potentially useful behaviours. The behaviours contained in the repertoires were sufficient to solve all the nine evaluation tasks, with a performance close to that of tabula-rasa evolution. We could see that different regions of the behaviour space were used to solve different tasks, which showed that the evolved arbitrators could find and select suitable primitives in the repertoire, thus taking full advantage of the repertoire. The proposed approach opens doors for the further exploration of the advantages associated with hierarchical control, in particular with respect to transferability of controllers to real robots, and the rapid solution of new tasks by reusing previously evolved components.

Acknowledgments

This work was supported by Fundação para a Ciência e Tecnologia, Portugal, with grant UID/MULTI/04046/2013 (BioISI centre grant), and grant UID/EEA/50008/2013 (awarded to Instituto de Telecomunicações). This work used the EGI infrastructure with the support of NCG-INGRID-PT (Portugal) and BIFI (Spain).

References

[1] F. Silva, L. Correia, A. L. Christensen, Evolutionary robotics, Scholarpedia 11 (7) (2016) 33333.

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

[3] F. Silva, M. Duarte, L. Correia, S. M. Oliveira, A. L. Christensen, Open issues in evolutionary robotics, Evolutionary Computation 24 (2) (2016) 205–236.

[4] J.-B. Mouret, S. Doncieux, Encouraging behavioral diversity in evolutionary robotics: An empirical study, Evolutionary Computation 20 (1) (2012) 91–133.

[5] S. Doncieux, J.-B. Mouret, Beyond black-box optimization: a review of selective pressures for evolutionary robotics, Evolutionary Intelligence 7 (2) (2014) 71–93.

[6] S. Doncieux, N. Bredeche, J.-B. Mouret, A. E. G. Eiben, Evolutionary robotics: What, why, and where to, Frontiers in Robotics and AI 2 (2015) 4.

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

[8] J. K. Pugh, L. B. Soros, K. O. Stanley, Quality diversity: A new frontier for evolutionary computation, Frontiers in Robotics and AI 3 (2016) 1–40.

[9] B. Sareni, L. Krahenbuhl, Fitness sharing and niching methods revisited, IEEE Transactions on Evolutionary computation 2 (3) (1998) 97–106.

[10] J. Lehman, K. O. Stanley, R. Miikkulainen, Effective diversity maintenance in deceptive domains, in: Genetic and Evolutionary Computation Conference (GECCO), ACM Press, 2013, pp. 215–222.

[11] J. Gomes, P. Mariano, A. L. Christensen, Novelty-driven cooperative coevolution, Evolutionary Computation 25 (2) (2017) 275–307.

[12] J. Lehman, K. O. Stanley, Evolving a diversity of virtual creatures through novelty search and local competition, in: Genetic and Evolutionary Computation Conference (GECCO), ACM Press, 2011, pp. 211–218.

[13] J. Mouret, J. Clune, Illuminating search spaces by mapping elites, CoRR abs/1504.04909.
URL http://arxiv.org/abs/1504.04909

[14] A. Cully, J. Clune, D. Tarapore, J.-B. Mouret, Robots that can adapt like animals, Nature 521 (7553) (2015) 503–507.

[15] A. Cully, J.-B. Mouret, Evolving a behavioral repertoire for a walking robot, Evolutionary Computation 24 (1) (2016) 59–88.

[16] K. Chatzilygeroudis, A. Cully, J.-B. Mouret, Towards semi-episodic learning for robot damage recovery, in: Workshop on AI for Long-Term Autonomy at the IEEE International Conference on Robotics and Automation (ICRA), 2016.

[17] V. Vassiliades, K. Chatzilygeroudis, J. B. Mouret, Using centroidal Voronoi tessellations to scale up the multi-dimensional archive of phenotypic elites algorithm, IEEE Transactions on Evolutionary ComputationIn press.

[18] M. Duarte, J. Gomes, S. M. Oliveira, A. L. Christensen, Evolution of repertoire-based control for robots with complex locomotor systems, IEEE Transactions on Evolutionary ComputationIn press.

[19] A. Cully, Y. Demiris, Quality and diversity optimization: A unifying modular framework, IEEE Transactions on Evolutionary Computation.

[20] M. Duarte, J. Gomes, S. M. Oliveira, A. L. Christensen, Evorbc: Evolutionary repertoire-based control for robots with arbitrary locomotion complexity, in: Genetic and Evolutionary Computation Conference (GECCO), ACM Press, 2016, pp. 93–100.

[21] S. Kim, S. Doncieux, Learning highly diverse robot throwing movements through quality diversity search, in: Proceedings of the Genetic and Evolutionary Computation Conference Companion (GECCO), ACM Press, 2017, pp. 1177–1178.

[22] S. A. Engebråten, J. Moen, O. Yakimenko, K. Glette, Evolving a repertoire of controllers for a multi-function swarm, in: International Conference on the Applications of Evolutionary Computation, Springer, 2018, pp. 734–749.

[23] D. S. Brown, R. Turner, O. Hennigh, S. Loscalzo, Discovery and exploration of novel swarm behaviors given limited robot capabilities, in: Distributed Autonomous Robotic Systems, Springer, 2018, pp. 447–460.

[24] V. Vassiliades, K. Chatzilygeroudis, J.-B. Mouret, Comparing multimodal optimization and illumination, in: Genetic and Evolutionary Computation Conference (GECCO) Companion, ACM Press, 2017, pp. 97–98.

[25] Y. Demiris, A. Dearden, From motor babbling to hierarchical learning by imitation: a robot developmental pathway, in: International Workshop on Epigenetic Robotics: Modeling Cognitive Development in Robotic Systems.

[26] Y. Liu, X. Yao, T. Higuchi, Evolutionary ensembles with negative correlation learning, IEEE Transactions on Evolutionary Computation 4 (4) (2000) 380–387.

[27] X. Yao, M. M. Islam, Evolving artificial neural network ensembles, IEEE Computational Intelligence Magazine 3 (1) (2008) 31–42.

[28] R. C. Arkin, Behavior-based Robotics, 1st Edition, MIT Press, 1998.

[29] M. J. Matarić, Behavior-based robotics as a tool for synthesis of artificial behavior and analysis of natural behavior, Trends in cognitive sciences 2 (3) (1998) 82–86.

[30] S. Luke, C. Hohn, J. Farris, G. Jackson, J. Hendler, Co-evolving soccer softbot team coordination with genetic programming, in: Robot Soccer World Cup, Springer, 1997, pp. 398–411.

[31] B. G. Woolley, G. L. Peterson, Genetic evolution of hierarchical behavior structures, in: Proceedings of the 9th annual conference on Genetic and evolutionary computation, ACM, 2007, pp. 1731–1738.

[32] W.-P. Lee, J. Hallam, H. H. Lund, Applying genetic programming to evolve behavior primitives and arbitrators for mobile robots, in: IEEE International Conference on Evolutionary Computation, IEEE, 1997, pp. 501–506.

[33] W.-P. Lee, Evolving complex robot behaviors, Information Sciences 121 (1) (1999) 1–25.

[34] T. Larsen, S. T. Hansen, Evolving composite robot behaviour-a modular architecture, in: Proceedings of the Fifth International Workshop on Robot Motion and Control (RoMoCo), IEEE Press, 2005, pp. 271–276.

[35] M. Duarte, S. Oliveira, A. L. Christensen, Hierarchical evolution of robotic controllers for complex tasks, in: Development and Learning and Epigenetic Robotics (ICDL), 2012 IEEE International Conference on, IEEE Press, 2012, pp. 1–6.

[36] M. Duarte, S. M. Oliveira, A. L. Christensen, Evolution of hybrid robotic controllers for complex tasks, Journal of Intelligent & Robotic Systems 78 (3) (2015) 463–484.

[37] M. Duarte, Engineering evolutionary control for real-world robotic systems, Ph.D. thesis, University Institute of Lisbon (ISCTE-IUL) (2016).

[38] Y. Ren, L. Zhang, P. N. Suganthan, Ensemble classification and regression-recent developments, applications and future directions, IEEE Computational intelligence magazine 11 (1) (2016) 41–53.

[39] A. Cully, J.-B. Mouret, Behavioral repertoire learning in robotics, in: Genetic and Evolutionary Computation Conference (GECCO), ACM Press, 2013, pp. 175–182.

[40] S. Doncieux, J.-B. Mouret, Behavioral diversity with multiple behavioral distances, in: IEEE Congress on Evolutionary Computation (CEC), IEEE Press, 2013, pp. 1427–1434.

[41] K. Stanley, R. Miikkulainen, Evolving neural networks through augmenting topologies, Evolutionary Computation 10 (2) (2002) 99–127.

[42] J. Gomes, P. Mariano, A. L. Christensen, Devising effective novelty search algorithms: A comprehensive empirical study, in: Genetic and Evolutionary Computation Conference (GECCO), ACM Press, 2015, pp. 943–950.

[43] A. Beck, S. Sabach, Weiszfeld's method: Old and new results, Journal of Optimization Theory and Applications 164 (1) (2015) 1–40.

[44] S. Doncieux, J.-B. Mouret, Behavioral diversity measures for evolutionary robotics, in: IEEE Congress on Evolutionary Computation (CEC), IEEE Press, 2010, pp. 1–8.

[45] J. Gomes, A. L. Christensen, Generic behaviour similarity measures for evolutionary swarm robotics, in: Genetic and Evolutionary Computation Conference (GECCO), ACM Press, 2013, pp. 199–206.

[46] E. Meyerson, J. Lehman, R. Miikkulainen, Learning behavior characterizations for novelty search, in: Genetic and Evolutionary Computation Conference (GECCO), ACM Press, 2016, pp. 149–156.

[47] J. Gomes, P. Mariano, A. L. Christensen, Systematic derivation of behaviour characterisations in evolutionary robotics, in: International Conference on the Synthesis and Simulation of Living Systems (ALife), MIT Press, 2014, pp. 212–219.

[48] J. Gomes, A. L. Christensen, Comparing approaches for evolving high-level robot control based on behaviour repertoires, in: IEEE Congress on Evolutionary Computation (CEC), IEEE Press, 2018, in press.

[49] D. J. Montana, Strongly typed genetic programming, Evolutionary computation 3 (2) (1995) 199–230.

[50] J. R. Koza, Concept formation and decision tree induction using the genetic programming paradigm, in: International Conference on Parallel Problem Solving from Nature, Springer, 1990, pp. 124–128.

[51] S. Luke, C. Cioffi-Revilla, L. Panait, K. Sullivan, G. Balan, Mason: A multiagent simulation environment, Simulation 81 (7) (2005) 517–527.

[52] S. Luke, Two fast tree-creation algorithms for genetic programming, IEEE Transactions on Evolutionary Computation 4 (3) (2000) 274–283.

[53] D. R. White, Software review: the ecj toolkit, Genetic Programming and Evolvable Machines 13 (1) (2012) 65–67.

[54] J. R. Koza, Genetic programming: on the programming of computers by means of natural selection, Vol. 1, MIT press, 1992.

[55] H. Iba, M. Terao, Controlling effective introns for multi-agent learning by genetic programming, in: Conference on Genetic and Evolutionary Computation (GECCO), Morgan Kaufmann, 2000, pp. 419–426.

[56] M. Hubert, P. J. Rousseeuw, K. Vanden Branden, ROBPCA: a new approach to robust principal component analysis, Technometrics 47 (1) (2005) 64–79.

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

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

[59] M. Duarte, J. Gomes, V. Costa, S. M. Oliveira, A. L. Christensen, Hybrid control for a real swarm robotics system in an intruder detection task, in: European Conference on the Applications of Evolutionary Computation (EvoApps), Springer, 2016, pp. 213–230.

[60] A. Cully, Y. Demiris, Hierarchical behavioral repertoires with unsupervised descriptors, in: Conference on Genetic and Evolutionary Computation (GECCO), ACM Press, 2018, in press.