This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TEVC.2017.2722101, IEEE Transactions on Evolutionary Computation

Evolution of Repertoire-based Control for Robots with Complex Locomotor Systems

Miguel Duarte, Jorge Gomes, Sancho Moura Oliveira, *Member, IEEE*, and Anders Lyhne Christensen, *Senior Member, IEEE*

***Abstract*—The evolution of task-oriented control for robots with complex locomotor systems is currently out of reach for traditional evolutionary computation techniques, as the coordination of a high number of locomotion parameters in response to the robot's sensory inputs is extremely challenging. Evolutionary techniques have therefore mainly been applied to the optimization of specific locomotion patterns, such as forward motion. In this paper, we explore the Evolutionary Repertoire-based Control (EvoRBC) approach, which divides the synthesis of control into two steps: (i) the evolution of a repertoire of locomotion primitives using a quality diversity algorithm; and (ii) the evolution of a high-level arbitrator that leverages the locomotion primitives in the repertoire to solve a given task. We comprehensively study the main components of the EvoRBC approach using a four-wheel steering robot. We then conduct a set of experiments in simulation using a hexapod robot. Our results show that EvoRBC is robust to parameter variations, and for all the robots tested, it is able to evolve controllers for a maze navigation task and significantly outperforms the traditional evolutionary robotics approach.**

***Index Terms*—Evolutionary computation; neural networks; robot motion; legged locomotion; evolutionary robotics; quality diversity; hierarchical control**

I. Introduction

The field of evolutionary robotics (ER) focuses on methods for the automatic synthesis of control for robots based on the principles of natural evolution [1]. While ER techniques have been successfully used in a wide range of tasks, such as maze navigation, obstacle avoidance, and foraging, task-oriented control has so far only been achieved with relatively simple differential-drive robots [2], [3]. Many real-world robotics applications, however, rely on robots with complex locomotor systems, such as legged robots [4], multirotors [5], and soft robots [6]. These robots can operate in a wide range of scenarios, such as unstructured environments, rough terrain, and human-centric environments [7], [8].

Synthesizing control for robots with complex locomotor systems has proven to be a challenging endeavor, as many locomotion parameters must be well coordinated in order to achieve successful and efficient locomotion [9]. Evolutionary techniques are a promising approach to overcome this problem [10]. Traditionally, evolutionary algorithms have been used to produce one specific locomotion pattern [11], such as moving forward, or to optimize specific properties, such as speed or energy efficiency [12]. Recently, it has been shown that *quality diversity* (QD) algorithms [13] (also known as *illumination algorithms* [14]) can be used to produce a wide range of locomotion behaviors for complex robots in each evolutionary run [14]–[17]. The purpose of QD algorithms is to search for a diversity of optimized solutions – when applied to robot locomotion, the result of the process is a *repertoire* composed of a set of diverse locomotion behaviors.

Despite the success of traditional and QD evolutionary algorithms, these evolutionary approaches have so far only been used in robots with complex locomotor systems to generate locomotion behaviors. In such studies, the objective was to be able to move the robot, not to solve an actual task [15]. Going from locomotion-only to task-oriented control represents a significant challenge: the controller not only has to coordinate all the locomotion parameters of the robot, it also has to adapt them in response to the sensory inputs of the robot and the task objectives [18].

To enable the use of ER techniques for complex robots in tasks beyond mere locomotion, we recently proposed the *Evolutionary Repertoire-based Control* (EvoRBC) approach [19]. In EvoRBC, a QD algorithm is first used to evolve a comprehensive repertoire of low-level *primitives* (locomotion patterns). A high-level *arbitrator*, which selects individual primitives from the repertoire to execute at each control cycle, is then evolved using a traditional neuroevolutionary algorithm. EvoRBC thus separates the process of learning how to move from the task-learning process.

In this paper, we describe EvoRBC and thoroughly study it to determine its potential and limitations. We present two key contributions: (i) a systematic exploration of EvoRBC's key components to determine their impact on the performance of the approach, and (ii) a set of simulation-based validation experiments, where we apply EvoRBC to a hexapod robot with a high number of locomotion parameters. Our results represent a significant contribution to the field of evolutionary robotics, as they are the first instance of the successful evolution of task-oriented control for a robot with non-trivial locomotor systems. EvoRBC thus paves the way for the application of neuroevolution to robots too complex for existing approaches, including legged robots.

II. Related Work

A. Evolution of Control for Robots with Complex Locomotor Systems

Recently, the field of robotics has witnessed the introduction of many robots with different types of morphologies [9].

M. Duarte, S. M. Oliveira, and A. L. Christensen are with the University Institute of Lisbon (ISCTE-IUL), Lisbon, Portugal.

J. Gomes is with the University of Lisbon, Lisbon, Portugal.

All authors are with the BioMachines Lab, Lisbon, Portugal, and Instituto de Telecomunicações, Lisbon, Portugal.

1089-778X (c) 2017 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TEVC.2017.2722101, IEEE Transactions on Evolutionary Computation

Efficient gaits for such robots are generally challenging to optimize manually, since the robots often have non-trivial dynamics and many degrees of freedom. Evolutionary robotics has been widely used in robots with complex locomotor systems as a method for gait optimization [2], [10], which contrasts with the task-solving studies that have been conducted with simple differential-drive robots [2].

Examples of the application of evolutionary techniques for such robots include: a 12 DOF three-legged robot where each leg has three non-coplanar degrees of freedom [20]; tensegrity robots built from interconnected rods and cables with oscillatory dynamics [21]; modular robots with evolved morphologies which can then be built by combining off-the-shelf components and 3D-printed parts [22], [23]; and soft robots, which can be composed of materials with different properties [24].

One of the classic robots used in gait-optimization studies is the four-legged Sony AIBO [25]–[27]. Hornby et al. [11] presented one of the first instances of the application of evolutionary techniques to obtain forward gaits for a legged robot. The authors conducted the evolutionary process using a real robot situated on different surfaces, and obtained solutions that were better than their manually programmed solutions.

The evolution of gaits for insect-inspired legged robots, such as hexapods, has also been widely studied. Passault et al. [28] introduced an approach for control optimization of legged robots, where a corpus of 22 different robots, with 4 to 8 legs and 8 to 24 motors, was manually designed. An evolutionary process then optimized a controller that generated motor trajectories for a given robot model. The authors were able to evolve controllers capable of navigating according to a given speed vector. Moore and McKinley [12] compared a number of different multi-objective algorithms for the evolution of single-direction gaits using legged robots, while taking into consideration objectives such as distance traveled, efficiency, and vertical torso movement.

B. Quality Diversity Algorithms

Traditional evolutionary processes typically converge to a single class of solutions [29]. In contrast, the recently proposed *quality diversity* (QD) evolutionary algorithms [13] can be used to automatically generate a comprehensive repertoire of solutions in a single evolutionary run [15]. QD algorithms were proposed as an extension of evolutionary algorithms driven by behavioral novelty and diversity [30], [31]. QD algorithms differ from traditional evolutionary algorithms in the sense that they are not designed to find a single optimal solution for a given problem. Instead, the purpose of QD algorithms is to discover a repertoire of high-quality solutions located in different regions of the *behavior space* (or *feature space*). The quality of each solution is given by a *fitness metric* (denoted *quality metric* in [19]), which is used to rank solutions that occupy the same location in behavior space. QD algorithms are therefore not optimization techniques, but rather exploration or *illumination* techniques [14].

One of the first QD algorithms was *novelty search with local competition* (NSLC) [32]. In NSLC, a multi-objective algorithm combines the novelty objective with a local competition objective. The local competition objective rewards solutions that display a higher quality than the nearest neighbors in the behavior space, thereby encouraging the evolution of a diverse set of high-quality solutions. Unlike NSLC, the recent MAP-Elites algorithm [14], [17] divides the behavior space into discrete bins, and aims at finding high-quality solutions in each bin of the behavior space (the behavior repertoire). This local elitism in MAP-Elites promotes an increase in quality throughout the behavior repertoire, as existing solutions can be replaced by newer ones with higher quality. Other variants of NSLC and MAP-Elites have been proposed, combining key features from both [13].

QD algorithms have mainly been applied in the evolutionary robotics domain, including the generation of a diversity of viable robot morphologies [14], [32], generation of legged robot gaits [14]–[17], and robot controllers for solving mazes [13]. Some authors have proposed [15] and implemented [33], [34] path planning algorithms to control robots based on behavior repertoires generated by QD approaches. The path planner can choose the sequence of primitives to execute based on the expected trajectory of each locomotion primitive. These approaches rely on perfect knowledge about the environment and the location of the robot in it. The evolved arbitrators proposed in this paper, on the other hand, do not use such global information, but rely exclusively on information collected by the robot through its onboard sensors during task execution.

C. Hierarchical Evolved Behavior Control

The use of QD algorithms that can generate behavior repertoires opens interesting opportunities for evolving controllers that can automatically select and use these behaviors. EvoRBC takes inspiration from the *hierarchical control synthesis approach* [35], an engineering-oriented approach in which a goal task is recursively decomposed into simpler sub-tasks. Control is evolved for each sub-task, and the final controller is then composed hierarchically. Behavior nodes that have control over the robot's actuators are *primitives*, and behavior nodes that select and activate lower-level nodes are *arbitrators*.

The hierarchical control synthesis approach allows for the composition of increasingly complex controllers that leverage previously evolved primitives. In [36], for instance, a small differential-drive robot had to solve a double T-maze task. The behavior primitives were evolved neural networks (turn left, turn right, go forward), and the behavior arbitrator was a neural network that chose which behavior primitive should be active at each time step.

Although hierarchical control synthesis has been demonstrated to solve tasks beyond the complexity of what has been achieved with traditional ER approaches [35], the experimenter still has to decide what set of primitives should be made available to the arbitrator, and synthesize them individually. The proposed EvoRBC approach is an extension of the hierarchical behavior control approach that uses QD algorithms to automatically generate a large set of primitives.

1089-778X (c) 2017 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TEVC.2017.2722101, IEEE Transactions on Evolutionary Computation

III. The EvoRBC Approach

EvoRBC is an evolutionary approach that divides the evolution of robotic control in two main steps, see Figure 1: (i) the generation of a repertoire of primitives (locomotion patterns) for a given robot, using a *quality diversity* algorithm, and (ii) the evolution of a high-level arbitrator for a given task, which leverages the locomotion primitives evolved in the first step. In a controller evolved following the EvoRBC approach, the high-level arbitrator (an evolved neural network in this study), 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 (a vector that encodes specific values for each locomotion parameter of the robot) from the repertoire. The chosen primitive is then applied to the actuators of the robot. This process is repeated every control cycle during operation, see Figure 2. The control cycle frequency is fixed at 10 Hz for all robots used in our experiments.

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 optimize for a robot's *intention* (e.g. choose a direction and speed), and not learn *how* that intention should be realized in terms of tuning and coordinating multiple locomotion parameters. EvoRBC therefore has the potential to allow for the evolution of control for robots with complex locomotor systems by providing access to a variety of previously generated locomotion behaviors.

A. Repertoire Generation

The first step in the EvoRBC approach is the generation of a comprehensive repertoire of locomotion primitives for a given robot, where a primitive corresponds to a locomotion pattern. Considering the robot has N locomotion parameters, a primitive is given by a vector in $\mathbb{R}^N$. The locomotion parameters can be values that are directly applied to the actuators of the robot, such as in the case of encoding wheel speeds or wheel turning angles, or they can correspond to parameters of functions, such as the amplitude and phase of periodic functions for controlling robot legs [17].

We use the MAP-Elites algorithm [14] to generate the repertoire (see Appendix A for a detailed description of MAP-Elites and the parameters used in our experiments). Although more recent QD algorithms have shown promising results [13], we use MAP-Elites in our study because of its relative simplicity. The chromosome space is defined as the space of locomotion primitives ($\subset \mathbb{R}^N$). The behavior space for a ground-based robot is defined as a 2-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, the *primitive evaluation time*, see Figure 3.

A fitness score is assigned to each primitive based on a provided *fitness metric*. The fitness scores are used to rank different primitives that fall within the same *bin* in the behavior space. We use *bin size* to denote the side-length of each square-shaped bin in the repertoire, which determines the resolution of the repertoire: a smaller bin size will result in a larger number of bins in the same area, and vice-versa. As the use of different fitness metrics can result in significantly different repertoires, we experiment with three different fitness metrics, described below and summarized in Figure 4:

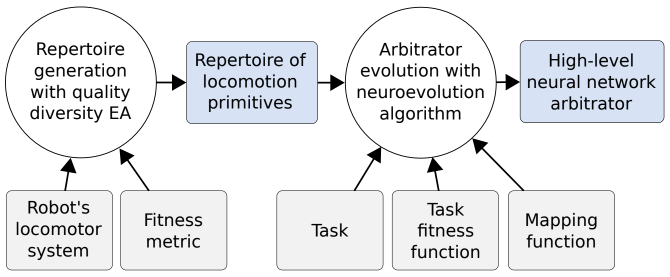

Repertoire generation with quality diversity EA
Repertoire of locomotion primitives
Arbitrator evolution with neuroevolution algorithm
High-level neural network arbitrator
Robot's locomotor system
Fitness metric
Task
Task fitness function
Mapping function

Fig. 1. Evolution of control using the EvoRBC approach.

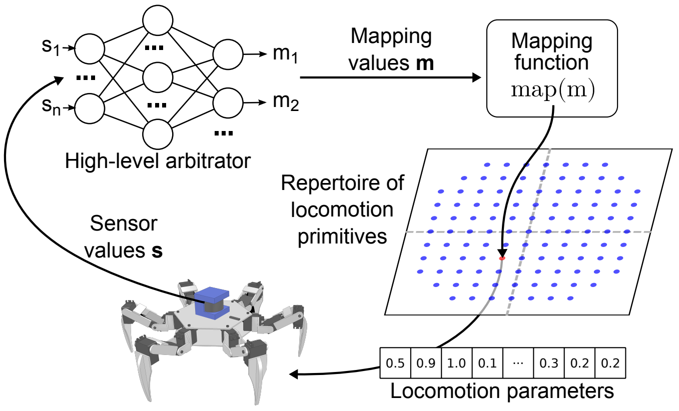


Fig. 2. Control cycle of a robot using the EvoRBC approach.

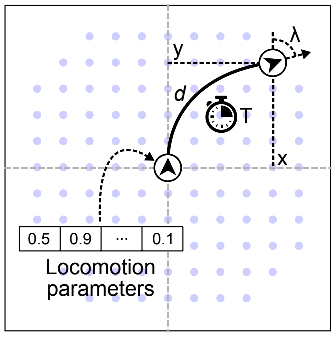


Fig. 3. Example of the evaluation of a locomotion primitive. The actuators of the robot are set according to the values encoded in the primitive, and the robot then moves for a fixed period of time (T). The behavior characterization is the displacement of the robot $\langle x, y\rangle$, while the final orientation (λ) or the distance traveled (d) are used to compute the fitness score. The light-blue dots represent other primitives with different displacement values.

- **Circular fitness metric:** Calculated based on the difference between the robot's final orientation (λ) and a *desired orientation* (ω) for that point in space $\langle x, y\rangle$. The desired orientation ω at each point is given by the tangent of the circle that intersects the robot and that point, with the center on the horizontal axis of the robot's initial position referential [16], see Figure 4 (left). For negative x displacements, π is added to ω to ensure that the robot faces forward while traveling backwards.

$$\text{circularFM}(i) = 1 - \frac{|\omega(x_i, y_i) - \lambda_i|}{\pi}$$
$$\omega(x, y) = \text{atan2}\left(\frac{y - x^2}{y}, 2x\right) + \begin{cases} \pi & y < 0 \\ 0 & \text{otherwise} \end{cases} \tag{1}$$

1089-778X (c) 2017 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TEVC.2017.2722101, IEEE Transactions on Evolutionary Computation

- **Radial fitness metric:** Similar to the circular fitness metric, but the desired orientation (ω) is the angle from the robot's starting position to the final position $\langle x, y\rangle$, see Figure 4 (middle):

$$\omega(x, y) = \operatorname{atan2}(x, y) + \begin{cases} \pi & y < 0 \\ 0 & \text{otherwise} \end{cases} \quad (2)$$

- **Distance fitness metric:** Promotes a repertoire of efficient primitives, in which each point in space is reached traveling the shortest distance, according to the following equation:

$$\text{distanceFM}(i) = 1 - \frac{d_i}{\sqrt{x_i^2 + y_i^2}}\ , \quad (3)$$

  where d_i is the total distance traveled by the robot, and $\langle x_i, y_i\rangle$ is the final position of the robot, assuming the robot started at the origin $\langle 0, 0\rangle$. As opposed to the previous two metrics, the distance fitness metric does not take the final orientation of the robot into consideration, see Figure 4 (right).

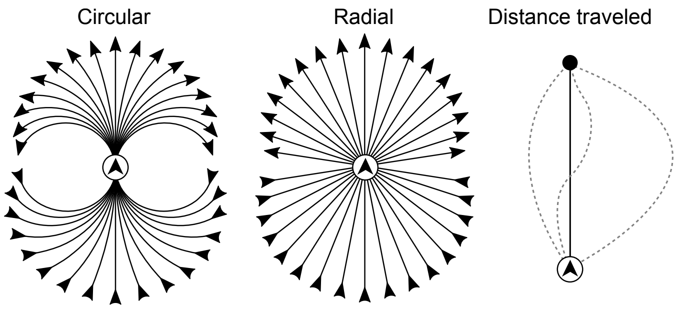


Fig. 4. Examples of ideal final orientations (arrows) or trajectories, for each of the proposed fitness metrics. See Section III-A for details.

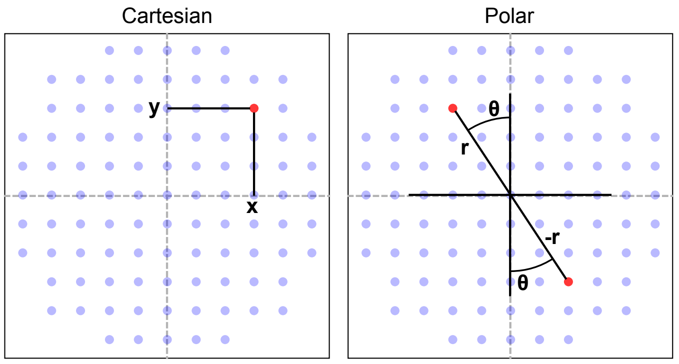


Fig. 5. The two mapping functions studied in this paper, *Cartesian* and *polar*. The arbitrator outputs are fed to the mapping function, which in turn selects the corresponding primitive from the repertoire.

B. Mapping Function

The *mapping function* maps the outputs of the arbitrator at each control cycle to a specific primitive in the repertoire. In the proposed implementation, a function φ maps the arbitrator outputs to a position in the behavior space and selects the primitive that is closest to that position:

$$\text{map}(\mathbf{m}) = \underset{i \in R}{\arg\min}\, \text{dist}(\mathcal{B}_i, \varphi(\mathbf{m})), \quad \mathbf{m} \in [0, 1]^{|\mathbf{m}|} \quad (4)$$

where $\mathbf{m}$ is the vector with the arbitrator outputs (in the range $[0, 1]$), R is the set of primitives in the repertoire, and $\mathcal{B}_i$ is the behavior characterization of primitive i.

We experimented with two different mapping functions (φ) in this study, see Figure 5:

- **Cartesian mapping:** Based on the Cartesian coordinate system, where the outputs are linearly mapped to $\langle x, y\rangle$ in the behavior space.
- **Polar mapping:** Based on polar coordinates, where the arbitrator outputs are used as distance r and angle θ.

Implementation details can be found in Appendix B. It should be noted that both of these mapping functions can be generalized to more dimensions.

C. Arbitrator Evolution

Once the repertoire is generated and the mapping function is defined, an arbitrator can be synthesized. The arbitrator is a high-level controller that receives the robot's sensory readings and determines which behavior primitive should be executed by outputting *mapping values*, see Figure 2. In the study presented in this paper, it is possible for an arbitrator to switch to a new primitive every control cycle. Other approaches could be considered, such as a mechanism where behavior primitives are always executed to completion, or for a predefined amount of time [37], [38].

The evolution of the arbitrator follows a traditional evolutionary robotics approach, with each individual encoding an instance of an arbitrator. Given a repertoire and a mapping function, each arbitrator is evaluated in the task environment and assigned a fitness score, which is then used for selection in the evolutionary process. In this study, the arbitrator is an evolved artificial neural network, where the activations of the output neurons are used as mapping values. In theory, any other evolvable policy that can map inputs to outputs could also work as arbitrator.

IV. Exploratory Experiments

In this section, we present a set of experiments designed to explore the configuration parameters and implementation options of EvoRBC to assess the robustness of the approach.

A. Four-wheel Steering Robotic Platform

The exploratory experiments were conducted with a simulated square-shaped four-wheel steering (4WS) robotic platform [39] with up to 5 locomotion parameters. Each wheel can be turned independently and the wheel speed is common to all wheels. A total of four different robot types were defined by pairing or fixing some of the locomotion parameters, see Figure 6. The robots can be divided into two *categories*: (i) 2WS robots that can only turn the front wheels, while the rear wheels remain fixed and aligned with the robot, and (ii) 4WS robots that can turn both the front and rear wheels. Details regarding the locomotion dynamics of the robotic platform can be found in Appendix C.

1089-778X (c) 2017 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TEVC.2017.2722101, IEEE Transactions on Evolutionary Computation

The locomotion of four-wheel steering robots can be challenging since there are large numbers of combinations that result in wheels being dragged instead of rolling (imagine, for instance, turning the front wheels in opposite directions). In order to achieve efficient locomotion and to minimize wheel slip, the wheels must be properly aligned with one another, ideally following the Ackermann steering principle.[1]

B. Maze Task

We used a maze navigation task (see Figure 7, left), where the robot must navigate a maze as quickly as possible. The task setup is based on [41], where the robot is equipped with six rangefinders that measure the distance to the nearest obstacle (up to a range of 1 m), and four Boolean radar sensors (unlimited range, $90°$ opening angle) that detect when the goal is within the respective circular sector (see Figure 7, right). The normalized readings of these sensors are the input to the controller. The simulation terminates when the end-point is reached, or when 120 s elapse. The fitness function is defined as follows:

$$F = \begin{cases} 2 - t/T & \text{target reached} \\ 1 - d_f/d_i & \text{otherwise} \end{cases} \quad (5)$$

where t is the elapsed time, T is the allowed time (120 s), and d_f and d_i are the final and initial distances from the robot to the target, respectively. Each controller is evaluated in all five mazes, and the average fitness is used for selection in the evolutionary process.

C. Experimental Methodology

We conducted a total of 30 independent runs for each evolutionary setup. For each run, we evolved a behavior repertoire and an arbitrator. We followed the procedure described in Section III-A for the generation of the repertoire, using the MAP-Elites algorithm. Regarding the arbitrators, we evolved neural networks with the NEAT [42] algorithm, which evolves both the topology and the connection weights of the networks. The common evolution parameter values are listed in Table I in Appendix A. These experiments were implemented in the JBotEvolver [43] open-source framework.

In order to conduct a fair comparison between EvoRBC and standard NEAT, we have assigned similar computational budgets to both approaches. In EvoRBC, the repertoire generation process uses approximately 1 M evaluations, each lasting for 2 s of simulated time. We thus ran the standard NEAT setups for an additional number of generations corresponding to 2 M seconds of simulated time (25 generations, rounded). Details regarding the computational budget and corresponding runtime of both approaches can be found in Appendix E.

D. Comparison with Standard Evolution

We compared the performance of EvoRBC with a standard evolutionary approach (*standard*), where the neural network under evolution receives the robot's sensor values and has one output neuron for each locomotion parameter of the robot. We evolved controllers for the maze task (Section IV-B) using four different robot types of varying locomotion complexity (Section IV-A). The arbitrator evolved using EvoRBC and the controller evolved using standard evolution were both optimized with the same neuroevolution algorithm (NEAT), with the same parameter values, except for the number of output neurons (see Table I in Appendix A). For the EvoRBC setups, we used the polar mapping function, the circular fitness metric, a bin size of 4 cm and primitive evaluation time of 2 s.

The repertoires generated in EvoRBC varied significantly in terms of the behavior space regions that were reached, depending on the robot type and its locomotion capabilities, see Figure 8. The ability to control two wheels on the same axis independently (2WS-3 and 4WS-5 robot types) allows for sharper turns by following the Ackermann steering principle. Furthermore, the ability to turn the back wheels (4WS robots) allows for a larger and more symmetrical range of motion. Despite these differences, MAP-Elites is able to evolve high-quality repertoires for all robot types.

In Figure 9, we compare the performance of standard evolution with EvoRBC in terms of fitness scores achieved in the maze task. With standard evolution, performance degrades for each robot category (2WS and 4WS) as more locomotion parameters are available to the controller (Mann-Whitney U, adjusted $p < 0.001$)[2]. These results highlight how challenging it is to coordinate multiple actuators in order to produce effective locomotion. In the 2WS-2 and 4WS-3 robots, a

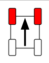
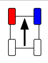
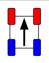
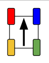
				
Robot type	2WS-2	2WS-3	4WS-3	4WS-5
Front wheel angles	Together	Independent	Together	Independent
Back wheel angles	Fixed	Fixed	Together	Independent

Fig. 6. Different robot types use in the exploratory experiments. *Fixed:* The wheels do not turn. *Together:* The wheels are paired, with their angle given by a single parameter. *Independent:* The wheels turn independently from one another, with one parameter for each. In addition to the wheel angle parameters, there is a single wheel speed parameter.

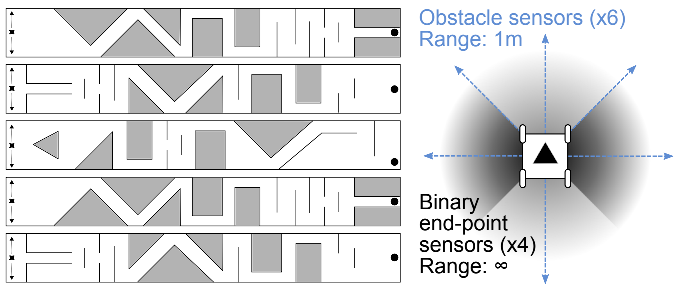


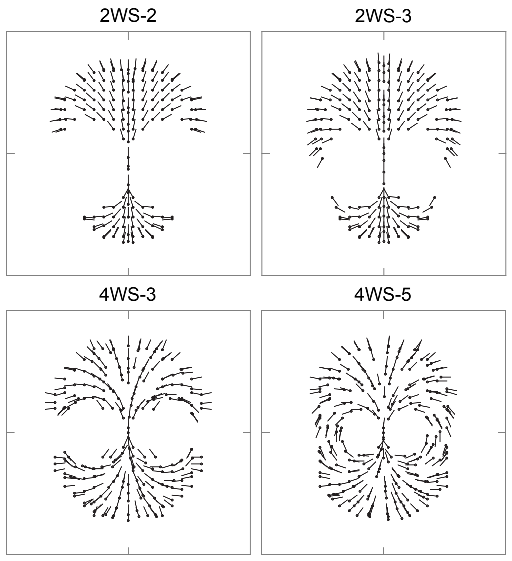
Fig. 7. Left: mazes used in the exploratory experiments. The robot starts in a random position on the left-hand side of the maze and must reach the target on the right-hand side (black circle). The mazes have a size of $11.0\,\text{m} \times 1.4\,\text{m}$, and the robot has $10\,\text{cm} \times 10\,\text{cm}$. Right: sensory configuration of the robot.

[1]The Ackermann steering geometry is a arrangement of linkages in a steering system which solves the problem of tracing circles with different radii by inside and outside wheels [40].

[2]In all the results presented in this paper, we use the two-sided Mann-Whitney U test when comparing two samples. When multiple comparisons within the same set of results were made, the p-values were adjusted with the Holm-Bonferroni correction.

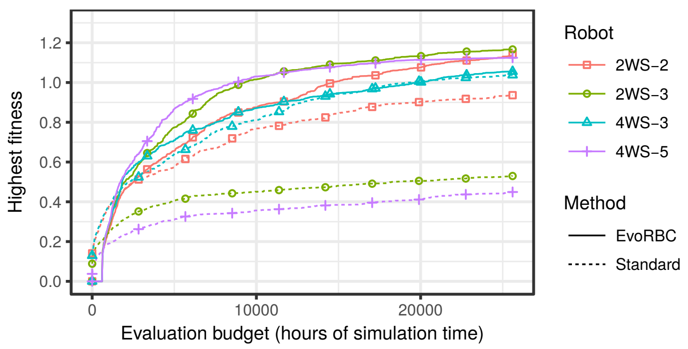
1089-778X (c) 2017 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TEVC.2017.2722101, IEEE Transactions on Evolutionary Computation



	2WS-2	2WS-3	4WS-3	4WS-5
Fitness	0.93 ± 0.05	0.91 ± 0.07	0.97 ± 0.08	0.99 ± 0.05
Size	152.4 ± 0.8	178.6 ± 1.2	226.8 ± 0.5	252.0 ± 3.1

Fig. 8. Top: examples of repertoires generated for each type of robot. Each box has a side-length of 100 cm, and each behavior primitive in the repertoire is represented by a marker, which is composed of a circle ($\langle x, y\rangle$ displacement) and a line (final orientation λ). Bottom: mean fitness (maximum is 1) and number of primitives generated by MAP-Elites for each robot (± standard deviation).



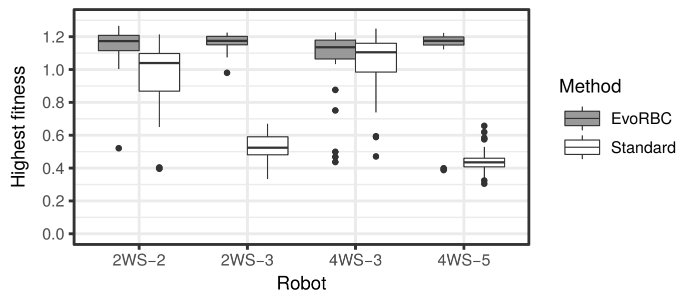


Fig. 9. Results for the experiments where EvoRBC is compared to standard neuroevolution. Top: highest fitness score found so far over the evolutionary run, averaged over 30 evolutionary runs for each robot type and approach. A fitness of 1.0 or higher indicates that the robot successfully reached the end-point of the maze. The initial period with a fitness of 0.0 for EvoRBC corresponds to the repertoire generation phase. Bottom: boxplot of the fitness scores of the highest-performing individuals for each setup. Each box summarizes the outcome of 30 independent evolutionary runs, with the whiskers representing the highest and lowest value within 1.5 IQR, and the dots indicating outliers.

controller might be able to solve the task by varying just one locomotion parameter – the angle of the front wheels. The other parameters (angle of back wheels and speed) can be fixed by the neural network, thus making it easier to coordinate the locomotion parameters. The 2WS-3 and 4WS-5 robots, however, control the wheels on the same axis independently, which significantly increases the difficulty of controlling the robot. Even with the 2WS-3 robot, where the controller only has to coordinate three parameters (the angle of the two front wheels and the speed), standard evolution fails to achieve high-performance solutions.

The results for EvoRBC, on the other hand, show that the approach was not significantly affected by the locomotion complexity, achieving similar performance across all robot types (Kruskal-Wallis test, $p = 0.068$). EvoRBC outperformed standard evolution in all robot types (Mann-Whitney, adjusted $p < 0.001$), except for the 4WS-3 robot ($p = 0.42$), where performance was similar. The results also highlight that the advantages of EvoRBC are not merely a consequence of reducing number of outputs in the neural controller. This can be seen in the case of the 2WS-2 robot, where EvoRBC has a significantly higher performance than standard evolution, despite the fact that the controller evolved with standard evolution has the same number of outputs (two) as the controller evolved with EvoRBC.

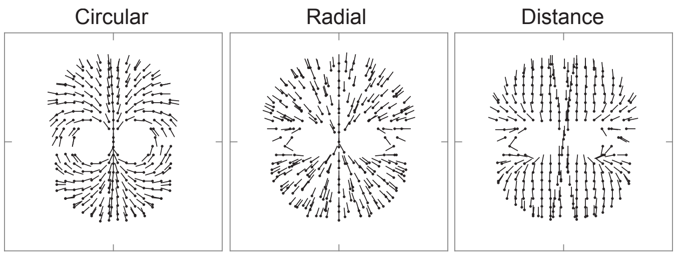
E. Fitness Metric

We assessed the performance of EvoRBC with the three fitness metrics presented in Section III-A: *circular*, *radial*, and *distance*. In these experiments, we used the 4WS-5 robot, the polar mapping function, a bin size of 4 cm and primitive evaluation time of 2 s. Figure 10 shows the results of the repertoire generation with the three different fitness metrics. Although the different fitness metrics resulted in the generation of different primitives (see Figure 10, top), MAP-Elites evolved high-quality and comprehensive repertoires in all experimental setups, as demonstrated by the number of primitives in the repertoire and their average fitness (see Figure 10, bottom).

The results of using these repertories for solving the maze task show that the circular fitness metric achieved the highest performance, followed by the radial fitness metric, and finally by the distance fitness metric (Mann-Whitney, $p < 0.001$), see Figure 11. Based on the visual inspection of the evolved solutions, we found that the lower performance of the arbitrators evolved with the *radial* and *distance* is caused by the near-absence of primitives where the robot's orientation is aligned with the movement direction. This means that with the radial and distance fitness metrics, the evolved arbitrators navigate the maze by moving sideways or diagonally with minor changes in orientation. The robot's sensors are therefore not consistently aligned with the movement direction, making it difficult to adopt simple maze-solving techniques, such as following either walls on the left or walls on the right.

The results show that using the circular fitness metric leads to movement patterns that approximate those typically

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TEVC.2017.2722101, IEEE Transactions on Evolutionary Computation



Fitness metric	Average fitness	Primitives
Circular	0.985 ± 0.05	252.0 ± 3.1
Radial	0.982 ± 0.06	262.1 ± 1.4
Distance	0.975 ± 0.05	262.3 ± 2.3

Fig. 10. Top: examples of repertoires generated with each fitness metric. Each box has a side-length of 100 cm, and each behavior in the repertoire is represented by a marker, which is composed of a circle ($\langle x, y \rangle$ displacement) and a line (final orientation λ). Bottom: average fitness (maximum is 1) and number of primitives generated by MAP-Elites with each fitness metric ($\pm$ standard deviation).

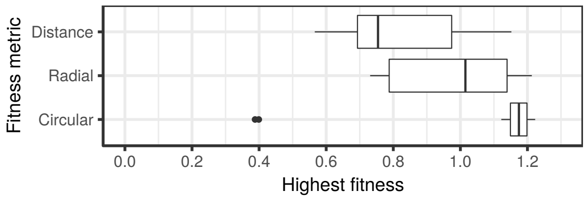


Fig. 11. Boxplots of the highest fitness scores achieved in the maze task, with each of the three proposed fitness metrics for repertoire generation.

found in differential drive robots (except for turning on the spot), making it a good starting point for the types of tasks typically studied in ER. We further tested combinations of the distance fitness metric with the other two fitness metrics during the repertoire generation, namely *distance+circular* and *distance+radial*, but the performance in the maze task showed no statistical difference when compared to only using the circular or radial fitness metrics (Mann-Whitney, $p = 0.54$ in both cases).

F. Bin Size and Primitive Evaluation Time

We tested different combinations of bin size (side length in cm: 1, 2, 4, 10, 20, and 40) and primitive evaluation time (in seconds: 0.5, 1.0, 2.0, 3.0, 5.0, and 10.0) to determine the effect of these parameters in the generated repertoires and in the task-solving performance. We used the 4WS-5 robot for this set of experiments, the circular fitness metric, and the polar mapping function.

Increasing the evaluation time enables the robot to travel larger distances, thus resulting in a larger repertoire, see Figure 12 (left). Increasing the bin size results in repertoires with fewer primitives, which, in extreme cases, can lead to repertoires that only represent a small subset of the robot's capabilities. Some combinations were not tested since they resulted in repertoires with only one primitive (large bin size and short evaluation time), or repertoires with a very high number of primitives (small bin size and long evaluation time).

In terms of task performance, the results show that decreasing the bin size always yielded a superior or similar

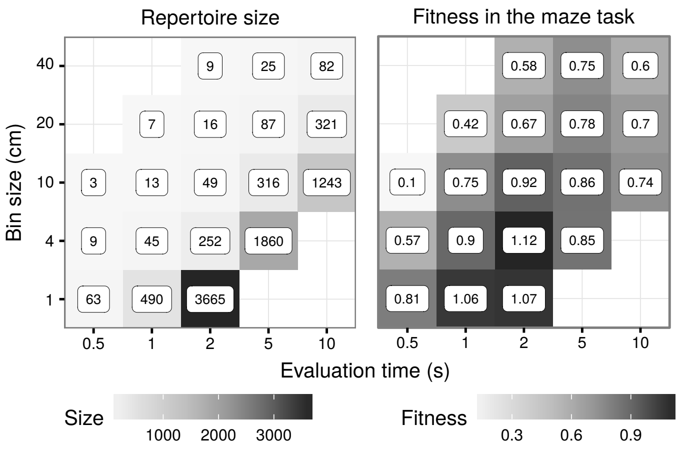


Fig. 12. Results for each combination of bin size and primitive evaluation time. Left: average number of primitives in the repertoires. Right: average fitness obtained in the maze navigation task.

performance (assuming a significance level of $p \leq 0.05$, Mann-Whitney), see Figure 12 (columns). Smaller bin sizes can be beneficial for the task-solving effort, as the arbitrator gains access to a more fine-grained repertoire of primitives. However, there are two potential drawbacks associated with smaller bin sizes: (i) the generation of the repertoire can be more demanding, as the number of iterations of the QD algorithm necessary to optimize all individuals increases; and (ii) the amount of storage necessary for the repertoire increases polynomially in its dimensionality.

Regarding the evaluation time for primitives, the results show that for each bin size, there is no linear correlation between evaluation time and fitness in the maze task, see Figure 12 (rows). Decreasing the evaluation time too much will lead to repertoires with a limited number of primitives, which might be insufficient to solve the task at hand. Increasing the evaluation time too much, on the other hand, can preclude the generation of primitives that are necessary to solve the task, such as quick and sharp curves, as the only way for the robot to end up in the regions near the origin is by moving slowly. Figure 13 illustrates the effects of increasing the evaluation time while keeping the bin size fixed, and vice-versa.

Our results show that generating repertoires with a bin size of 4 cm and an evaluation time of 2 s allows for the evolution of arbitrators with the highest fitness scores on average, for this particular robot and task. Two other combinations yielded high fitness scores, with no statistically significant differences: (time=1 s, bin size=1 cm) and (time=2 s, bin size=1 cm).

G. Mapping Function

In a final set of exploratory experiments, we tested the two different mapping functions described in Section III-B: *polar* and *Cartesian*. We used the 4WS-5 robot, a bin size of 4 cm, primitive evaluation time of 2 s, and the circular fitness metric. Evolution was able to evolve high-quality arbitrators with both mappings – with the *Cartesian mapping*, solutions achieved on average a highest fitness score of 1.10 ± 0.16, while with the *polar mapping*, solutions achieved an average score of 1.12 ± 0.20.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TEVC.2017.2722101, IEEE Transactions on Evolutionary Computation

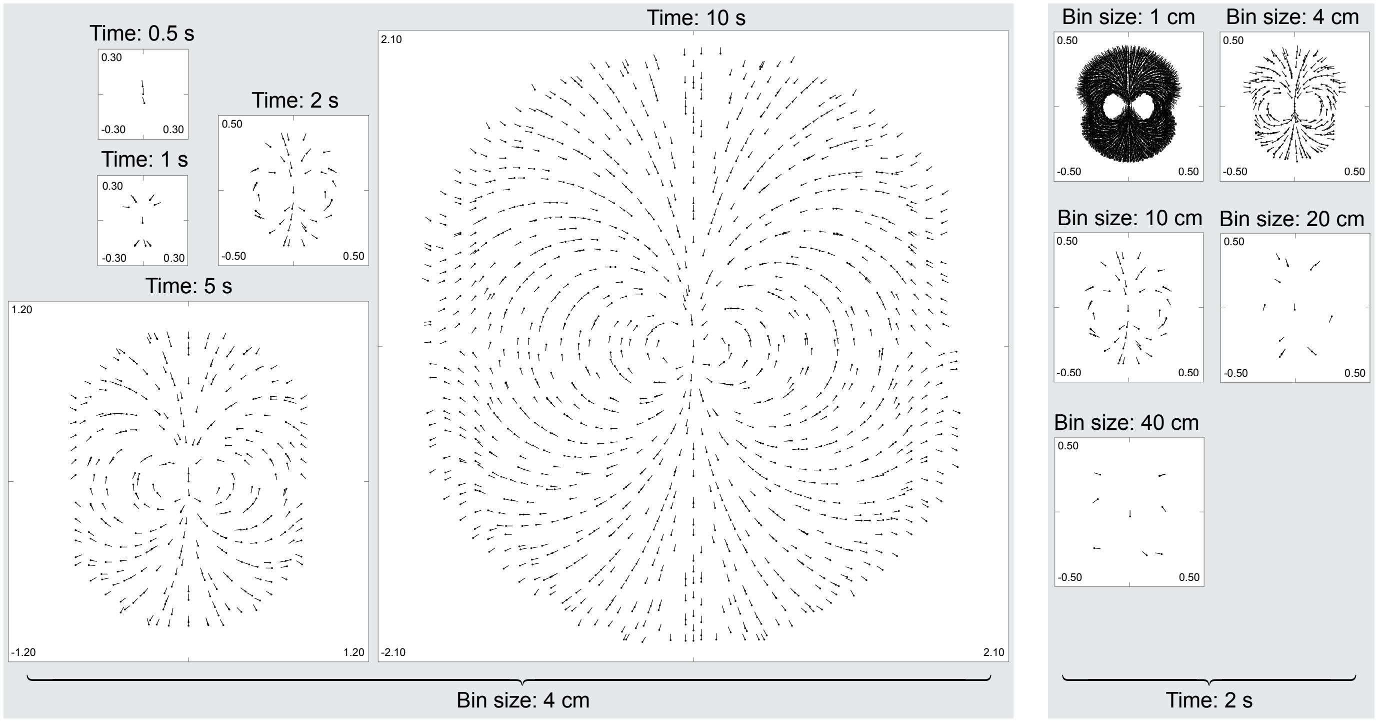


Fig. 13. Examples of repertoires generated with different combinations of bin size and primitive evaluation time. The values in the corners of each box represent the distance of the box limits from the center, in meters.

In Figure 14, we show the activation patterns obtained during the task execution of the highest-performing individuals found in each run. The different mapping functions cause different activation patterns to be evolved: while the arbitrators evolved with the *polar* mapping choose, on average, a wider range of primitives, the arbitrators with the *Cartesian* mapping tend to focus on a narrower set of primitives, that can be obtained by saturating the output neurons. It should be noted that there are no primitives in the corners of the repertoire, but since the mapping chooses the closest primitive in behavior space, selecting the left/right corners corresponds to selecting a turn left/right behavior, as shown in Figure 14 (left column).

It is clear that evolution tends to produce controllers that rely on the simple solution of fixing one of the parameters (ρ in the polar mapping and y in the Cartesian mapping) while changing only the remaining parameter (α and x, respectively), see Figure 14 (top) for examples. In 23 out of the 30 evolutionary runs (both with the polar and the Cartesian mapping), the highest-fitness solutions displayed such patterns. These activation patterns enable the controller to select a wide variety of forward motion primitives that result in the robot moving at its maximum speed, which correspond to the behaviors that are key to solving the maze task.

V. Experiments With a Hexapod Robot

In this section, we apply EvoRBC to a hexapod robot. This robotic platform allow us to assess EvoRBC's potential to evolve high-level task control for robots which have typically only been used to study gait evolution in the field of evolutionary robotics [27], [28].

A. Experimental Setup

We used the hexapod robot model available in the V-REP simulator [44] and the Bullet physics engine[3] to conduct our experiments. We actuated all 18 joints of the robot, see Figure 15. The angle of each actuated joint at each time step is dictated by a time-dependent periodic function, as done in previous works with legged robots [10], [17], [45]. Each joint moves with a proportional controller until the given angle is reached, respecting the torque and velocity limits of the joint. The locomotion primitives for the hexapod are therefore parameters of the periodic functions that control the positions of the joints over time.

The actuation of last joint of each leg of the robot is calculated automatically in order to always remain perpendicular to the robot's body [16]. Each locomotion primitive has a total of 37 parameters that define the periodic function: one global parameter (the wave's smoothness ρ), and three additional parameters per controllable joint in each leg (an offset ϵ, an amplitude α, and a phase ϕ). Additional details can be found in Appendices A and D.

Due to the high computational cost of running the V-REP simulator and Bullet physics engine, we used a single $2\,\mathrm{m} \times 8\,\mathrm{m}$ maze, with similar characteristics as those used in the exploratory experiments (see Figure 15, left). We generated a single locomotion repertoire the hexapod and conducted five evolutionary runs to evolve arbitrators, where each candidate solution was evaluated in one simulation sample. When evolving an arbitrator, we set a time limit to complete the maze of 30 seconds. The fitness function for the evolution of the arbitrator

[3]http://bulletphysics.org

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TEVC.2017.2722101, IEEE Transactions on Evolutionary Computation

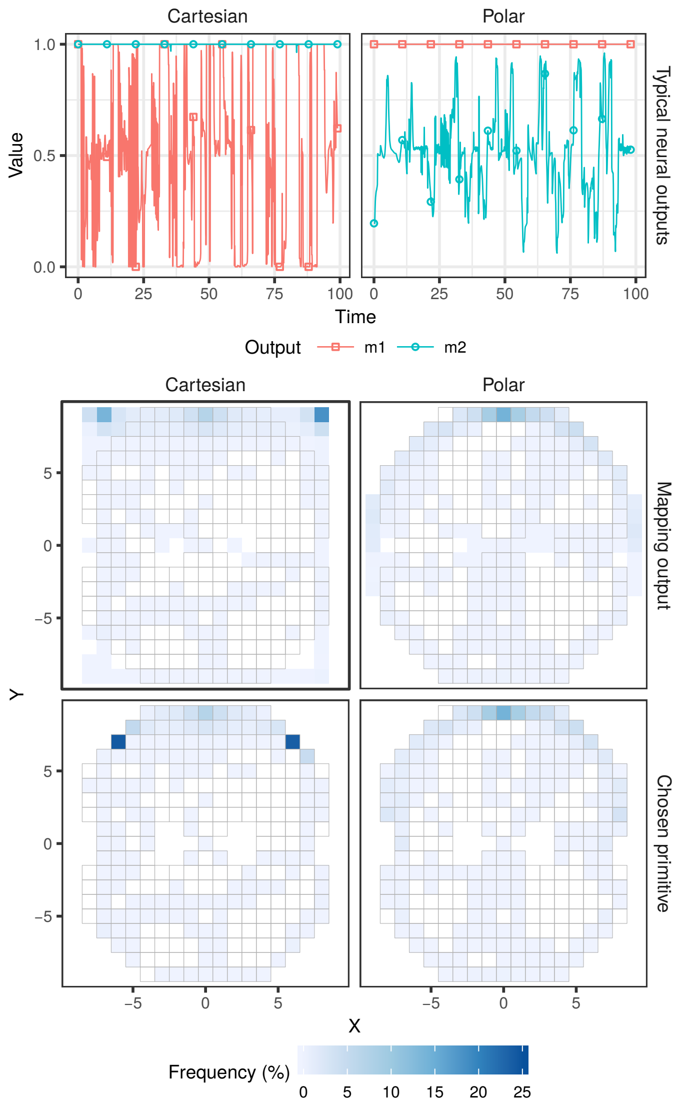


Fig. 14. Output patterns and behavior selection using the Cartesian (left column) and polar (right column) mapping functions. The first row shows an example of the output activations of the highest-performing arbitrator for each mapping function in a single simulation sample. The middle and bottom rows show the aggregated distribution, for the highest-performing arbitrators of all 30 evolutionary runs, of the mapping outputs and the actual chosen primitive from the repertoire, respectively. The cells delimited with grey lines represent repertoire bins populated with a locomotion primitive.

is the same as used in the exploratory experiments, see Eq. 5.

We added a 2D laser scanner on top of the hexapod robot to provide sensory inputs to the arbitrator during task execution, see Figure 15. The sensor was configured with a field of view of $[-90°, 90°]$, a range of 2 meters, and a total of 25 evenly distributed measuring points. The measuring points were aggregated in groups of five, with each group returning the minimum distance measured. The robot can sense its relative orientation to the end-point of the maze with an additional, single-value sensor.

B. Repertoire Generation

MAP-Elites was used to generate the repertoire using similar parameter values as in the exploratory experiments, see

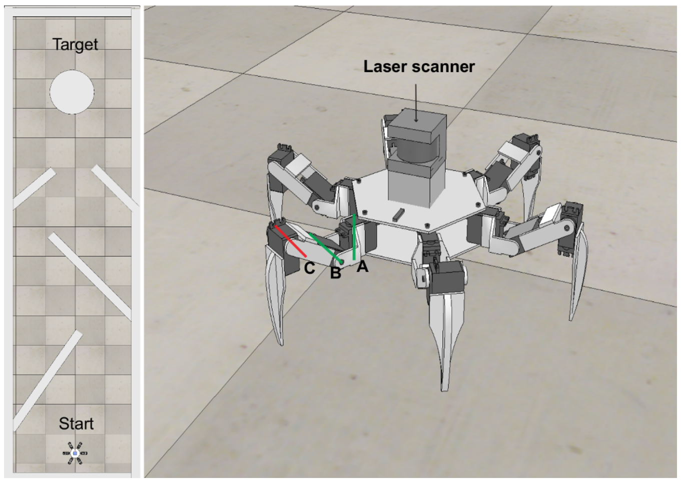


Fig. 15. Left: top view of the maze used in the task, with the robot shown for scale. Right: the simulated hexapod robot used in the validation experiments. The joints which are controlled by the locomotion parameters are marked in green. The joints marked in red are linearly dependent on the positions of the other joints in the leg, so that the feet are always perpendicular to the hexapod's body.

Table I. We used the circular fitness metric (Eq. 1) with an additional component that takes into account the stability of the robot. If the robot tilts more than $T = 45°$ during the primitive evaluation, the primitive is discarded and is not added to the repertoire (higher tilt values are typically associated with the robot falling or toppling over). Otherwise, it receives a fitness according to the following equation:

$$\text{circularTiltFM}(i) = 1 - \frac{t}{T} + \text{circularFM}(i) \qquad (6)$$

where t is the maximum tilt observed (in degrees) during the primitive execution, and T the maximum tilt allowed $(45°)$.

The evolved repertoire had an average fitness of 1.69 ± 0.24, a total of 336 primitives, and can be seen in Figure 16. The repertoire is less symmetrical than the ones generated for the four-wheeled robot, with some regions being more thoroughly explored than others. The results obtained in the repertoire generation step reveal that MAP-Elites was unable to fully explore the behavioral capabilities of the hexapod, as it is morphologically symmetrical and should therefore be able to make the same movements to either side. The elitist nature of MAP-Elites introduces the potential pitfall of allowing strains of locomotion primitives to populate large regions of the repertoire [13], therefore making it difficult for truly novel primitives to be generated.

Many of the locomotion primitives in the edges of the repertoire displayed low fitness values, mostly due to a deviation of their final orientations when compared to the ideal orientation calculated by the circular fitness metric. The presence of these low-fitness primitives show that evolution was only able reach these locations using non-circular movement patterns, and further iterations of the QD algorithm were not able to refine those primitives.

1089-778X (c) 2017 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TEVC.2017.2722101, IEEE Transactions on Evolutionary Computation

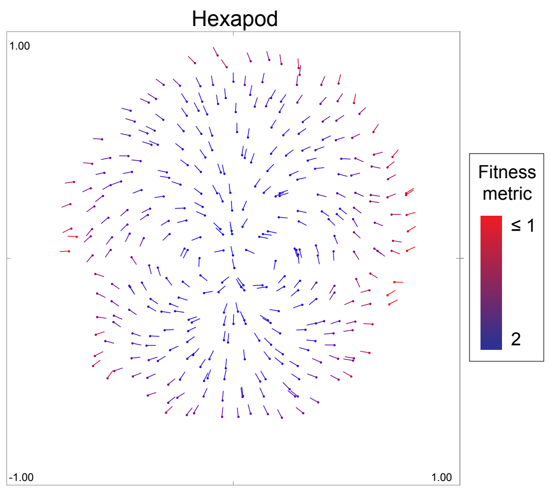


Fig. 16. The repertoire generated for the hexapod. The color of each marker represents the fitness achieved during repertoire generation for each locomotion primitive.

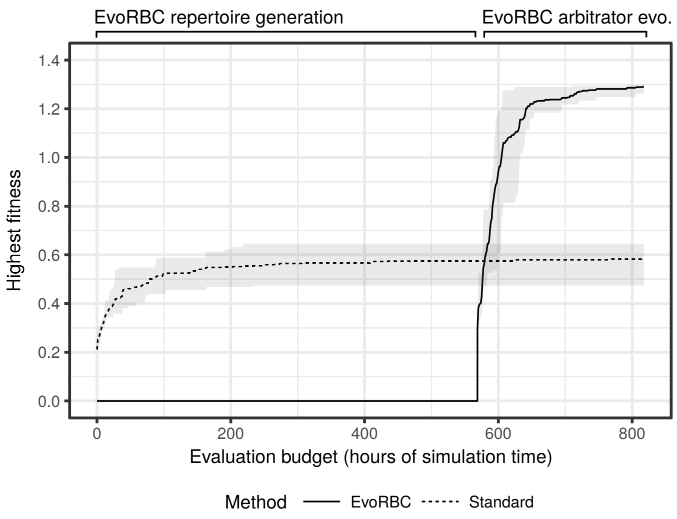


Fig. 17. Highest fitness scores achieved over the evolutionary run, averaged over five runs for each method. The shaded area shows the minimum and maximum for the five runs. The initial period with a fitness of 0.0 for EvoRBC corresponds to repertoire generation phase.

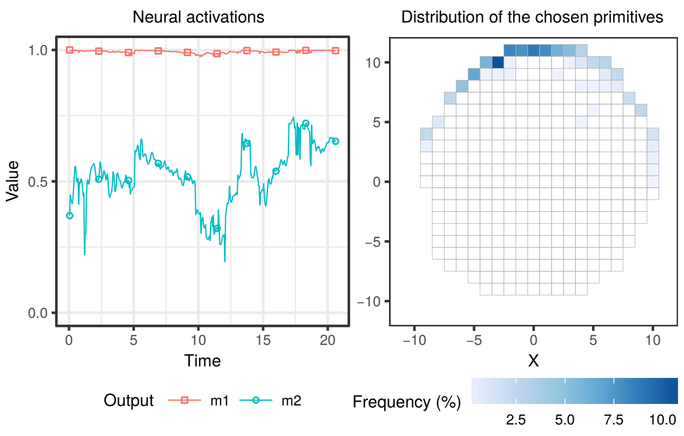


Fig. 18. Left: the output activations of the highest-performing arbitrator in a simulation. Right: aggregated distribution of the chosen primitives from the repertoire for the highest-performing arbitrators of all five evolutionary runs.

C. Maze Task Results

The results of the fitness achieved in the maze task can be seen in Figure 17, and videos of the highest-performing controllers evolved in each evolutionary run can be found in the Supplemental Material. EvoRBC was able to successfully reach the end-point of the maze in all five evolutionary runs, achieving fitness scores of 1.28±0.02. The evolutionary process needed only 25 generations on average to find controllers that reached the end-point of the maze, and continued to refine the arbitrators until the end of the evolutionary process. The evolved arbitrators were able to effectively navigate the robots through the maze quickly, and often without colliding any of the walls.

In Figure 18 (left), we show the neural outputs measured during the task execution of the highest-performing arbitrator. It is possible to observe that the arbitrator frequently switches between a wide range of primitives, varying only one mapping parameter (the angle) while keeping the other (the distance) fixed at the maximum value, in order to choose high-speed primitives. This result is consistent with the experiments presented in Section IV-G, where we observed similar activation patterns with the polar mapping. This activation pattern was evolved in all evolutionary runs, see Figure 18 (right). The highest-performing arbitrators change the active locomotion primitive 234 times on average during the task execution, resulting in an average time per primitive of just 1.9 control cycles. Even though each primitive is active for only a relatively short amount of time, the arbitrator tends to switch to other primitives that are nearby in the behavior space, resulting in a smooth transition that does not disrupt the robot's locomotion (see supplementary video).

We further compared the performance of EvoRBC with standard NEAT, where a neural network directly selects the values of each locomotion parameter (see Figure 17). The standard NEAT setups was allowed to run for 455 additional generations for a fair comparison (see Section IV-C and Appendix E). Unsurprisingly, we found that standard NEAT was unable to reach the same level of performance as EvoRBC, as it is extremely challenging to effectively coordinate the many different locomotion parameters. Standard NEAT was able to produce relatively slow forward and turning gaits, allowing the robot to progress up to roughly the mid-point of the maze.

VI. Discussion

We first conducted a set of exploratory experiments using a four-wheel steering robotic platform with up to five locomotion parameters (Section IV). We used this robotic platform to conduct a comprehensive analysis of EvoRBC's performance under different configurations. Based on these results, we then applied EvoRBC to a legged robotic platform (Section V), a hexapod with 37 locomotion parameters. We compared EvoRBC with the standard ER approach, where the evolved neural network controller receives sensor readings as inputs, and outputs locomotion parameters.

1089-778X (c) 2017 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TEVC.2017.2722101, IEEE Transactions on Evolutionary Computation

Our results showed that EvoRBC is an effective approach to the evolution of task-oriented control for robots with different locomotor system complexities. In the experiments with the 4WS robotic platform, EvoRBC outperformed the standard ER approach even for the simplest robot, with only two locomotion parameters. For the more complex robot types, including the experiments with the hexapod robot, EvoRBC was able to consistently achieve significantly higher fitness scores than the standard ER approach.

A. EvoRBC Configuration

Our results showed that EvoRBC is relatively robust to variations in its main parameters. Moreover, we were able to successfully apply many of the same parameter values in completely different robots, thus confirming the general applicability of the proposed approach. Based on a comprehensive set of experiments with the 4WS robotic platform, we devised a number of guidelines for the configuration of EvoRBC:

Fitness metric: Using different fitness metrics resulted in repertoires composed of clearly different sets of locomotion primitives. Repertoires generated with the *circular* fitness metric resulted in primitives in which the robot's orientation changes while curving, which facilitated the evolution of control for navigation tasks.

Bin size: The bin size should be relatively small in order to obtain a high-resolution repertoire. Below certain values however, the performance gains become insignificant, as the neighboring primitives become very similar. Based on our exploratory experiments, we chose a value corresponding to $1/5^{\text{th}}$ of the distance the robot can travel in one second.

Primitive evaluation time: We found that relatively short evaluation times (1 s or 2 s in the 4WS robot and 2 s in the hexapod robot) yielded the best results with respect to the fitness scores achieved by the evolved arbitrators. With these values, we obtained a diverse set of primitives that covered the locomotion capabilities of the robots. It should, however, be noted that this parameter might need to be empirically adjusted for robots with significantly different locomotion speeds.

Mapping function: There were no significant differences in performance between using the polar and the Cartesian mapping functions. We did, however, observe differences in the behavior selection strategy. Arbitrators evolved with the polar mapping chose, on average, a wider range of primitives than those evolved with the Cartesian mapping.

B. Future Work

The next step is to validate EvoRBC on real robots, where the role of transferability from simulation to reality is crucial. The transferability of controllers of complex robots can be problematic, due to the difficulty in accurately simulating real-world physics [46], [47]. A number of approaches have been studied to overcome this *reality gap*, including in legged robots, such as injecting noise [48] in simulation, or automatically tuning the simulator for better accuracy [49]. Cully and Mouret [16] have shown that it is possible to generate a repertoire of transferable locomotion behaviors by combining simulated evaluations with periodic transfers on the real robot. Generating repertoires of transferable locomotion primitives can be a key step towards the successful transfer of complete EvoRBC controllers.

We are also assessing the generality of the evolved arbitrators and primitive repertoires. First, it should be studied whether the same repertoire can be used to solve different tasks, by only evolving the arbitrator. Our initial exploratory experiments [19] suggested that this is possible. The experiments in this paper (Section IV), however, show that certain parameters (the fitness metric and the evaluation time) influence the evolved primitives, which could limit the generality of the repertoire. Second, we will study if the same arbitrator can be used with different repertoires, evolved for different robots. In previous experiments [19], we showed that this could be possible if the different robots had very similar morphologies. There are a number of factors impeding the arbitrator's generality, such as different locomotion capabilities (speed, turning angle, etc), different consequences when transitioning from one primitive to the other, and different sensory capabilities. We will study how to evolve general arbitrators capable of coping with these differences.

A parallel line of research is to explore how repertoires composed of more high-level behaviors, not just locomotion primitives, can be automatically synthesized and combined with evolved arbitrators. Previous works have shown that quality diversity and novelty-based techniques can be successfully used to evolve repertoires of behaviors for simple robots far beyond locomotion [13], [31], [50], which could potentially be leveraged by a high-level arbitrator [35]. We will also explore the use of more recent quality diversity techniques [13], [51], [52] for the generation of behavior repertoires.

VII. Conclusion

In this paper, we explored EvoRBC, an approach for the evolution of control for robots with complex locomotor systems. EvoRBC relies on two main steps: (i) the evolution of a repertoire of locomotion *primitives* using quality diversity techniques, and (ii) the evolution of a high-level *arbitrator* for a given task, which leverages the previously evolved locomotion primitives.

We first explored how to optimally configure EvoRBC, from the evolution of the repertoire to the evolution of the arbitrator, showing that EvoRBC is relatively robust to variations in its main parameters. We then validated the approach using a simulated legged robot (a hexapod). We were able to successfully employ EvoRBC to produce controllers for all the tested robots to solve a maze navigation task. Overall, EvoRBC was able to consistently achieve solutions for problems on which the standard evolutionary robotics methodology failed.

Based on the results presented in this paper, we contend that EvoRBC has considerable potential to enable the application of evolutionary techniques to robots with complex locomotor systems. We have shown that not only it is possible to evolve effective gaits for such robots, it is also possible to evolve task-oriented controllers that leverage those gaits. The experiments with the hexapod robots, in particular, represent a significant advance in the state of the art of evolving control for robots with complex locomotor systems.

1089-778X (c) 2017 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TEVC.2017.2722101, IEEE Transactions on Evolutionary Computation

APPENDIX

A. Evolutionary Setup

The evolutionary parameters for the repertoire generation and the arbitrator evolution can be found in Table I, and the MAP-Elites algorithm can be found in Algorithm 1.

TABLE I
PARAMETERS USED FOR THE GENERATION OF THE REPERTOIRE AND FOR THE EVOLUTION OF THE ARBITRATOR.

Repertoire generation – MAP-Elites			
Iterations (I)	1000	Batch size (S_B)	1000
Gene range	$[-1, 1]$	Gene mutation prob.	25%
Mutation type	Gauss.	Mutation σ	0.5
4-wheeled robots		*Hexapod robot*	
Initial batch (S_G)	10k	Initial batch (S_G)	25k
Bin size	4×4 cm[†]	Bin size	8×8 cm
Fitness metric	Circular[†]	Fitness metric	Circular+Tilt
Eval. time (T)	2 s[†]	Eval. time (T)	2 s
Arbitrator evolution – NEAT			
Population size	150	Survival threshold	20%
Target species count	5	Mutation prob.	25%
Mutate bias prob.	30%	Crossover prob.	20%
Add connection prob.	5%	Add node prob.	3%
4-wheeled robots		*Hexapod robot*	
Evaluation samples	5	Evaluation samples	1
Evaluation time	120 s	Evaluation time	30 s
Generations	1000	Generations	200
Mapping function	Polar[†]	Mapping function	Polar

[†] Unless stated otherwise.

Algorithm 1 Batch variant of the MAP-Elites algorithm [14]. The generation and evaluation of new individuals can be parallelized in each batch.

```
1: $\mathcal{X} \leftarrow \emptyset, \mathcal{P} \leftarrow \emptyset$    ▷ Map of solutions and their performance
2: for $i \in 1..I$ do    ▷ Repeat for $i$ iterations
3:   $B \leftarrow \emptyset$    ▷ Current batch
4:   if $i = 1$ then    ▷ Initial batch of size $S_G$
5:     for $j \in 1..S_G$ do
6:       $B \leftarrow B \cup \{\texttt{RandomSolution}()\}$    ▷ Fill with random solutions
7:   else
8:     for $j \in 1..S_B$ do    ▷ Batch of size $S_B$
9:       $a \leftarrow$ Randomly select a solution $\in \mathcal{X}$
10:      $B \leftarrow B \cup \{\texttt{Mutate}(a)\}$    ▷ Fill with mutations of existing individuals
11:  for $a \in B$ do
12:    $(p, \mathbf{b}) \leftarrow \texttt{Evaluate}(a)$    ▷ Obtain performance $p$ and behavior $\mathbf{b}$ of $a$
13:    $\mathbf{b}' \leftarrow \texttt{Discretize}(\mathbf{b})$    ▷ Find the corresponding bin
14:    if $\mathcal{X}[\mathbf{b}'] = \emptyset$ or $\mathcal{P}[\mathbf{b}'] < p$ then
15:      $\mathcal{X}[\mathbf{b}'] \leftarrow a$    ▷ Replace if $a$ has higher performance or if the bin is empty
16:      $\mathcal{P}[\mathbf{b}'] \leftarrow p$
```

B. Mapping Functions

The *Cartesian* mapping scales the mapping parameters according to the dimension of the repertoire, with respect to the behavior space, thus ensuring that all the primitives in the repertoire can be selected. Given the minimum (x_{min}, y_{min}) and maximum (x_{max}, y_{max}) values of x and y found in the repertoire, respectively, the *Cartesian* mapping is defined by:

$$\text{Cartesian}(m_1, m_2) = \langle m_1(x_{max} - x_{min}) + x_{min}, \\ m_2(y_{max} - y_{min}) + y_{min} \rangle \tag{7}$$

The *polar* mapping function uses two parameters: a distance r, and an angle θ, relative to the robot's frame-of-reference. This mapping function allows the arbitrator to indicate how fast the robot should move, and how much it should turn, allowing for similar control mechanics as those typically found for simple differential-drive robots. The radius of the repertoire (τ) is defined as the maximum distance from the center $\langle 0, 0 \rangle$ to any behavior in the repertoire:

$$\tau = \max_{i \in R} \sqrt{x_i^2 + y_i^2} \tag{8}$$

$$\text{polar}(m_1, m_2) = \tau \times (2m_1 - 1) \times \\ \langle \cos(\pi m_2 - {}^{\pi}/_{2}),\ \sin(\pi m_2 - {}^{\pi}/_{2}) \rangle \tag{9}$$

C. Four-wheeled Robot Locomotion

The locomotion dynamics of the robotic platform are based on the calculation of the instantaneous center of rotation (ICR) [40], see Figure 19. In an ideal turning configuration, there is no wheel slip in any wheel ($\beta_{a..d} = 0$), meaning that $ICR = I_{ac} = I_{bd}$. In our setup, robot configurations with too much wheel slip ($> 10°$) result in the robot not moving.

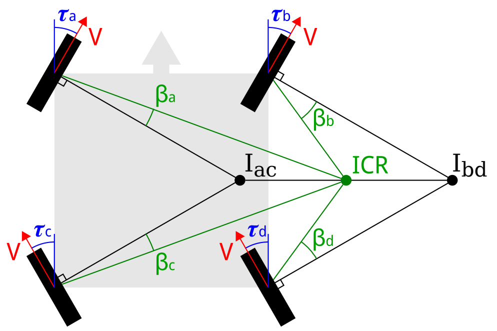


Fig. 19. Model of the four-wheeled robotic platform's dynamics. $\tau_{a..d}$: wheel angle, $\tau_i \in [-45°, 45°]$ and $\Delta\tau_i \leq 90°/s$. V: linear wheel speed, $V \in [-20, 20]$ cm/s and $\Delta V \leq 40$ cm/s. I_{ac}, I_{bd}: intersection points of the left and right wheels' normals, respectively. ICR: instantaneous center of rotation, defined as the mean point between I_{ac} and I_{bd}. $\beta_{a..d}$: wheel slip angle.

D. Hexapod Robot Locomotion

Each joint is controlled based on a target angle p. The joint moves with a proportional controller until the target angle is reached, respecting the torque and velocity limits of the joint. The value of p for each joint j over time t is defined by a periodic function as follows:

$$p_j(t) = \epsilon_j + \alpha_j \cdot \tanh(\rho \cdot \sin(2 \cdot \pi \cdot (t \cdot \omega + \phi_j))) \tag{10}$$

where t is the time elapsed since the beginning of the simulation, $\epsilon_j \in [-1, 1]$ is the offset, $\alpha_j \in [0, 1]$ is the amplitude, $\rho \in \left[\frac{\pi}{8}, \frac{\pi}{8} + 2\pi\right]$ is the smoothness of the wave (ranging from a sine wave to a near-square wave), ω is the frequency, and $\phi_j \in [0, 1]$ is the phase. The smoothness parameter (ρ) is common to all joints, the frequency (ω) is fixed to 1 Hz, and the other parameters are set individually for each joint. Considering J joints under control, a locomotion primitive $\mathcal{P}$ is therefore defined by:

$$\mathcal{P} = [\rho, \epsilon_1, \alpha_1, \phi_1, \cdots, \epsilon_J, \alpha_J, \phi_J] \tag{11}$$

The joints $A_1 \cdots A_6$ (see Figure 15) move in the range $[-70°, 70°]$, and in the neutral pose they are at $0°$. The joints

1089-778X (c) 2017 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TEVC.2017.2722101, IEEE Transactions on Evolutionary Computation

$B_1 \cdots B_6$ move in the range $[-90°, 90°]$, and are at $0°$ in the neutral pose. The joints $C_1 \cdots C_6$ are not controlled by a periodic function under evolutionary control. Instead, their value is set based on other joints according to: $C_i = -B_i + 90°$.

E. Computational Complexity and Runtime

Evolving an EvoRBC arbitrator has a computational complexity comparable to evolving a standard NEAT network. The primitive selection step in EvoRBC has a time complexity of $\mathcal{O}(1)$, since the mapping function provides the coordinates in the repertoire space in $\mathcal{O}(1)$ time. Although additional search in the repertoire might be needed if the chosen position in the repertoire does not contain a primitive, this can be avoided by filling all accessible positions of the repertoire a priori based on the nearest neighbors. The only significant difference is the additional memory used by the behavior repertoire, which in practice tends to be negligible. As an example, the repertoire used by the hexapod robot is less than 50 KB.

We validated our claim that both approaches are computationally comparable by measuring their respective runtimes, see Table II. In practice, the evolution of the arbitrator in EvoRBC can actually be faster than standard NEAT evolution, due to differences in neural network sizes. Using the Hexapod experiments as an example, EvoRBC evolves a network with 6 inputs and 2 outputs (the mapping parameters), while standard NEAT evolves a network with 6 inputs and 37 outputs (the locomotion parameters). This causes large differences in network sizes for the highest-performing controllers: on average, a total of 37.6 neurons and synapses for EvoRBC, and 277.6 for NEAT. Since larger networks take longer to execute, this results in faster runtimes for the evolution of the behavior arbitrator (see the runtime at 200 generations in Table II).

TABLE II
RUNTIMES FOR EVORBC AND STANDARD NEAT.

Process	Total sim. time (h)	Runtime (h)
4WS-5 robot†		
Repertoire generation	561	0.02
Arbitrator evolution (1000 gens)	25000	0.88
EvoRBC total	25561	0.90
Standard NEAT (1000 gens)	25000	0.87
Standard NEAT (1025 gens)	25625	0.90
Hexapod robot‡		
Repertoire generation	570	12.5
Arbitrator evolution (200 gens)	250	4.4
EvoRBC total	820	16.9
Standard NEAT (200 gens)	250	5.6
Standard NEAT (655 gens)	819	18

† Experiments in Section IV-D. Timed on a AMD FX-8350 @ 4.0GHz (4 cores / 8 threads). ‡ Experiments in Section V. Timed on a Intel Core i7-4790S @ 3.2GHz (4 cores / 8 threads)

ACKNOWLEDGEMENTS

This work was supported by Fundação para a Ciência e Tecnologia (FCT), Portugal, under the grant UID/EEA/50008/2013. Miguel Duarte was also supported with the grant SFRH/BD/76438/2011, and Jorge Gomes with the grant SFRH/BD/89095/2012.

REFERENCES

[1] F. Silva, L. Correia, and A. L. Christensen, "Evolutionary Robotics," *Scholarpedia*, vol. 11, no. 7, p. 33333, 2016.

[2] A. L. Nelson, G. J. Barlow, and L. Doitsidis, "Fitness functions in evolutionary robotics: A survey and analysis," *Robotics and Autonomous Systems*, vol. 57, no. 4, pp. 345–370, 2009.

[3] F. Silva, M. Duarte, L. Correia, S. M. Oliveira, and A. L. Christensen, "Open issues in evolutionary robotics," *Evolutionary Computation*, vol. 24, no. 2, pp. 205–236, 2016.

[4] D. J. Todd, *Walking machines: an introduction to legged robots*. Springer Science & Business Media, 2013.

[5] G. Cai, B. M. Chen, and T. H. Lee, *Unmanned Rotorcraft Systems*. Springer, 2011.

[6] J. Hiller and H. Lipson, "Automatic design and manufacture of soft robots," *IEEE Transactions on Robotics*, vol. 28, no. 2, pp. 457–466, 2012.

[7] J. R. Rebula, P. D. Neuhaus, B. V. Bonnlander, M. J. Johnson, and J. E. Pratt, "A controller for the LittleDog quadruped walking on rough terrain," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*. IEEE Press, 2007, pp. 1467–1473.

[8] J. Lim, I. Lee, I. Shim, H. Jung, H. M. Joe, H. Bae, O. Sim, J. Oh, T. Jung, S. Shin, K. Joo, M. Kim, K. Lee, Y. Bok, D.-G. Choi, B. Cho, S. Kim, J. Heo, I. Kim, J. Lee, I. S. Kwon, and J.-H. Oh, "Robot system of DRC-HUBO+ and control strategy of team KAIST in DARPA robotics challenge finals," *Journal of Field Robotics*, vol. 34, no. 4, pp. 802–829, 2017.

[9] S. Mintchev and D. Floreano, "Adaptive morphology: A design principle for multimodal and multifunctional robots," *IEEE Robotics Automation Magazine*, vol. 23, no. 3, pp. 42–54, 2016.

[10] D. Gong, J. Yan, and G. Zuo, "A review of gait optimization based on evolutionary computation," *Applied Computational Intelligence and Soft Computing*, vol. 2010, pp. 1–12, 2010.

[11] G. S. Hornby, S. Takamura, J. Yokono, O. Hanagata, T. Yamamoto, and M. Fujita, "Evolving robust gaits with AIBO," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, vol. 3. IEEE Press, 2000, pp. 3040–3045.

[12] J. M. Moore and P. K. McKinley, "A comparison of multiobjective algorithms in evolving quadrupedal gaits," in *International Conference on Simulation of Adaptive Behavior*. Springer, 2016, pp. 157–169.

[13] J. K. Pugh, L. B. Soros, and K. O. Stanley, "Quality Diversity: A new frontier for evolutionary computation," *Frontiers in Robotics and AI*, vol. 3, p. 40, 2016.

[14] J.-B. Mouret and J. Clune, "Illuminating search spaces by mapping elites," *arXiv preprint arXiv:1504.04909*, 2015.

[15] A. Cully and J.-B. Mouret, "Behavioral repertoire learning in robotics," in *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO)*. ACM Press, 2013, pp. 175–182.

[16] A. Cully and J.-B. Mouret, "Evolving a behavioral repertoire for a walking robot," *Evolutionary Computation*, vol. 24, no. 1, pp. 59–88, 2016.

[17] A. Cully, J. Clune, D. Tarapore, and J.-B. Mouret, "Robots that can adapt like animals," *Nature*, vol. 521, no. 7553, pp. 503–507, 2015.

[18] S. Nolfi and D. Floreano, *Evolutionary robotics: The biology, intelligence, and technology of self-organizing machines*. MIT Press, Cambridge, MA, 2000.

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

[20] G. Klaus, K. Glette, and M. Høvin, "Evolving locomotion for a 12-DOF quadruped robot in simulated environments," *Biosystems*, vol. 112, no. 2, pp. 102–106, 2013.

[21] A. Iscen, A. Agogino, V. SunSpiral, and K. Tumer, "Controlling tensegrity robots through evolution," in *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO)*. ACM Press, 2013, pp. 1293–1300.

[22] E. Samuelsen and K. Glette, "Real-world reproduction of evolved robot morphologies: Automated categorization and evaluation," in *Applications of Evolutionary Computation*. Springer, 2015, pp. 771–782.

[23] J. Auerbach, D. Aydin, A. Maesani, P. Kornatowski, T. Cieslewski, G. Heitz, P. Fernando, I. Loshchilov, L. Daler, and D. Floreano, "RoboGen: Robot generation through artificial evolution," in *Proceedings of the International Conference on the Synthesis and Simulation of Living Systems (ALIFE)*. MIT Press, 2014, pp. 136–137.

1089-778X (c) 2017 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TEVC.2017.2722101, IEEE Transactions on Evolutionary Computation

[24] N. Cheney, R. MacCurdy, J. Clune, and H. Lipson, "Unshackling evolution: evolving soft robots with multiple materials and a powerful generative encoding," in *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO)*. ACM Press, 2013, pp. 167–174.

[25] S. Chernova and M. Veloso, "An evolutionary approach to gait learning for four-legged robots," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, vol. 3. IEEE Press, 2004, pp. 2562–2567.

[26] R. A. Téllez, C. Angulo, and D. E. Pardo, "Evolving the walking behaviour of a 12 DOF quadruped using a distributed neural architecture," in *International Workshop on Biologically Inspired Approaches to Advanced Information Technology*. Springer, 2006, pp. 5–19.

[27] S. Kamio and H. Iba, "Adaptation technique for integrating genetic programming and reinforcement learning for real robots," *IEEE Transactions on Evolutionary Computation*, vol. 9, no. 3, pp. 318–333, 2005.

[28] G. Passault, Q. Rouxel, R. Fabre, S. N'Guyen, and O. Ly, "Optimizing morphology and locomotion on a corpus of parametric legged robots," in *Conference on Biomimetic and Biohybrid Systems*. Springer, 2016, pp. 227–238.

[29] M. Laumanns, L. Thiele, K. Deb, and E. Zitzler, "Combining convergence and diversity in evolutionary multiobjective optimization," *Evolutionary Computation*, vol. 10, no. 3, pp. 263–282, 2002.

[30] J. Lehman and K. O. Stanley, "Abandoning objectives: Evolution through the search for novelty alone," *Evolutionary Computation*, vol. 19, no. 2, pp. 189–223, 2011.

[31] J.-B. Mouret and S. Doncieux, "Encouraging behavioral diversity in evolutionary robotics: An empirical study," *Evolutionary Computation*, vol. 20, no. 1, pp. 91–133, 2012.

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

[33] K. Chatzilygeroudis, A. Cully, and J.-B. Mouret, "Towards semi-episodic learning for robot damage recovery," *arXiv preprint arXiv:1610.01407*, 2016.

[34] K. Chatzilygeroudis, V. Vassiliades, and J.-B. Mouret, "Reset-free trial-and-error learning for data-efficient robot damage recovery," *arXiv preprint arXiv:1610.04213*, 2016.

[35] M. Duarte, S. M. Oliveira, and A. L. Christensen, "Evolution of hybrid robotic controllers for complex tasks," *Journal of Intelligent and Robotic Systems*, vol. 78, no. 3, pp. 463–484, 2015.

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

[37] M. Duarte, S. M. Oliveira, and A. L. Christensen, "Automatic synthesis of controllers for real robots based on preprogrammed behaviors," in *Proceedings of the International Conference on Simulation of Adaptive Behaviour (SAB)*. Springer, 2012, pp. 249–258.

[38] M. Duarte, S. M. Oliveira, and A. L. Christensen, "Evolution of hierarchical controllers for multirobot systems," in *Proceedings of the International Conference on the Synthesis and Simulation of Living Systems (ALIFE)*. MIT Press, 2014, pp. 657–664.

[39] K. Spentzas, I. Alkhazali, and M. Demic, "Dynamics of four-wheel-steering vehicles," *Forschung im Ingenieurwesen*, vol. 66, no. 6, pp. 260–266, 2001.

[40] A. Haddoun, M. E. H. Benbouzid, D. Diallo, R. Abdessemed, J. Ghouili, and K. Srairi, "Modeling, analysis, and neural network control of an EV electrical differential," *IEEE Transactions on Industrial Electronics*, vol. 55, no. 6, pp. 2286–2294, 2008.

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

[42] K. O. Stanley and R. Miikkulainen, "Evolving neural networks through augmenting topologies," *Evolutionary Computation*, vol. 10, no. 2, pp. 99–127, 2002.

[43] M. Duarte, F. Silva, T. Rodrigues, S. M. Oliveira, and A. L. Christensen, "JBotEvolver: A versatile simulation platform for evolutionary robotics," in *Proceedings of the International Conference on the Synthesis and Simulation of Living Systems (ALIFE)*. MIT Press, 2014, pp. 210–211.

[44] E. Rohmer, S. P. Singh, and M. Freese, "V-REP: A versatile and scalable robot simulation framework," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE Press, 2013, pp. 1321–1326.

[45] J. Morimoto, G. Endo, J. Nakanishi, and G. Cheng, "A biologically inspired biped locomotion strategy for humanoid robots: Modulation of sinusoidal patterns by a coupled oscillator model," *IEEE Transactions on Robotics*, vol. 24, no. 1, pp. 185–191, 2008.

[46] K. Glette, G. Klaus, J. C. Zagal, and J. Torresen, "Evolution of locomotion in a simulated quadruped robot and transferral to reality," in *Proceedings of the Seventeenth International Symposium on Artificial Life and Robotics*, 2012, pp. 1–4.

[47] D. Belter and P. Skrzypczyński, "A biologically inspired approach to feasible gait learning for a hexapod robot," *International Journal of Applied Mathematics and Computer Science*, vol. 20, no. 1, pp. 69–84, 2010.

[48] N. Jakobi, "Running across the reality gap: Octopod locomotion evolved in a minimal simulation," in *European Workshop on Evolutionary Robotics*. Springer, 1998, pp. 39–58.

[49] G. Klaus, K. Glette, and J. Tørresen, "A comparison of sampling strategies for parameter estimation of a robot simulator," in *International Conference on Simulation, Modeling, and Programming for Autonomous Robots*. Springer, 2012, pp. 173–184.

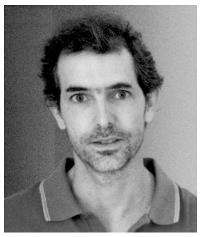
[50] J. Gomes, P. Urbano, and A. Christensen, "Evolution of swarm robotics systems with novelty search," *Swarm Intelligence*, vol. 7, no. 2–3, pp. 115–144, 2013.

[51] V. Vassiliades, K. Chatzilygeroudis, and J.-B. Mouret, "Scaling up MAP-Elites using centroidal voronoi tessellations," *arXiv preprint arXiv:1610.05729*, 2016.

[52] A. Gaier, A. Asteroth, and J.-B. Mouret, "Data-efficient exploration, optimization, and modeling of diverse designs through surrogate illumination," in *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO)*. ACM Press, 2017, in press.

Miguel Duarte Received a Ph.D. degree from the University Institute of Lisbon (ISCTE-IUL), Portugal in 2016. His research focuses on swarm robotics systems, evolutionary robotics, and automatic robotic control synthesis.

Jorge Gomes Received a Ph.D. degree in Informatics Engineering from University of Lisbon, Portugal, in 2017. He is currently a postdoctoral researcher collaborating with the BioMachines Lab (ISCTE-IUL) and BioISI-MAS (FCUL) laboratories. His current research interests include coevolutionary algorithms, diversity-driven evolution, multirobot and multiagent systems, and artificial life in general.



Sancho Moura Oliveira Received the licentiate degree in computer science from Instituto Superior Técnico and a Ph.D. degree in physics from the University of Lisbon, Portugal in 1996 and 2008, respectively. He is currently an Assistant Professor with the Instituto Universitário de Lisboa and a researcher at Instituto de Telecomunica cões. His research is focused on swarm intelligence, evolutionary computation and communication in large-scale systems.

Anders Lyhne Christensen Received a Ph.D. degree from IRIDIA-CoDE, Université Libre de Bruxelles, Belgium in 2008, and he is currently an assistant professor at University Institute of Lisbon (ISCTE-IUL), Portugal. Dr. Christensen is the founder and head of the Bio-inpsired Computation and Intelligent Machines Lab at ISCTE-IUL. His research interests include bio-inspired machine intelligence, large-scale multirobot systems, evolutionary computation, and fault tolerance.

1089-778X (c) 2017 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.