Online Hyper-Evolution of Controllers in Multirobot Systems

Fernando Silva*†‡, Luís Correia† and Anders Lyhne Christensen*‡§
*BioMachines Lab, Lisbon, Portugal
†BioISI, Faculdade de Ciências da Universidade de Lisboa, Lisbon, Portugal
‡Instituto de Telecomunicações, Lisbon, Portugal
§Instituto Universitário de Lisboa (ISCTE-IUL), Lisbon, Portugal

Abstract—**In this paper, we introduce *online hyper-evolution* (OHE) to accelerate and increase the performance of online evolution of robotic controllers. Robots executing OHE use the different sources of feedback information traditionally associated with controller evaluation to find effective evolutionary algorithms and controllers online during task execution. We present two approaches: OHE-fitness, which uses the fitness score of controllers as the criterion to select promising algorithms over time, and OHE-diversity, which relies on the behavioural diversity of controllers for algorithm selection. Both OHE-fitness and OHE-diversity are distributed across groups of robots that evolve in parallel. We assess the performance of OHE-fitness and of OHE-diversity in two foraging tasks with differing complexity, and in five configurations of a dynamic phototaxis task with varying evolutionary pressures. Results show that our OHE approaches: (i) outperform multiple state-of-the-art algorithms as they facilitate controllers with superior performance and faster evolution of solutions, and (ii) can increase effectiveness at different stages of evolution by combining the benefits of multiple algorithms over time. Overall, our study shows that OHE is an effective new paradigm to the synthesis of controllers for robots.**

1. Introduction

Online evolution is an open-ended approach to the synthesis of behavioural control for robots that is part of the field of *evolutionary robotics*. In online evolution, the robot controllers are continuously optimised *during* task execution. An evolutionary algorithm is executed onboard the robots themselves. The evolutionary operators, namely selection and reproduction, are applied autonomously by the robots without any external supervision or human interaction. As a result, online evolution creates the potential for long-term adaptation and learning: robots can continuously self-adjust and learn new behaviours in response to, for example, changes in the task requirements or environmental conditions, and to faults in the sensors and actuators.

Research in online evolution started with the pioneering contributions of Floreano and Mondada, see [1] and references therein, which laid the foundation for a number of studies that followed. The authors evolved controllers for navigation and homing tasks on a real Khepera robot. The synthesis of successful controllers required up to ten days of continuous evolution on real robotic hardware, thus showing that the evaluation time is a critical aspect in real-robot experiments. Afterwards, Watson *et al.* [2] introduced *embodied evolution*, in which the evolutionary algorithm is distributed across a group of robots. Individual robots evolve controllers in parallel, and exchange partial solutions to the task when they meet. Such an approach enables a form of knowledge transfer that can speed up evolution and facilitate more effective collective problem solving [3]. However, the speed up of evolution is dependent on encounters between robots, which may be infrequent in large or open environments.

Over the past years, a number of online evolution algorithms have been introduced. Recent examples include mEDEA by Bredeche *et al.* [4], and odNEAT by Silva *et al.* [5]. However, at the current state of development, there are a number of key issues that must be addressed before online evolution becomes a feasible approach to adaptation and learning in real robotic systems. One main impediment to widespread adoption is the prohibitively long time that online evolution requires to synthesise solutions (typically several hours or days) [6]. Ideally, research in the field would enable the development of one or more prevalent algorithms able to effectively synthesise solutions to a large number of different tasks in a timely manner. However, as stated in the *No Free Lunch Theorem* [7], it is impossible to devise such general silver bullets given that all optimisation algorithms yield equivalent performance on average. Even if the tasks to which online evolution is applied share a common structure, an algorithm's performance is conditioned on its configuration (e.g. parameters and genetic encoding), which in turn constrains the ability for effective robot learning and adaptation. In this way, while the current quest for more efficient online evolution algorithms continues, the field may ultimately need more general *mechanisms* that can combine the benefits of different algorithms.

In this paper, we study a novel approach to accelerate and increase the performance of online evolution of robotic controllers. We introduce *online hyper-evolution* (OHE), in which the key principle is to use the different sources of feedback information traditionally associated with controller evaluation to find suitable evolutionary algorithms online.

Contrarily to current approaches, which rely on a single, predefined algorithm to find a high-performing controller, OHE searches across a space of candidate algorithms and configurations in order to find effective algorithms over time. In this way, OHE has the potential to increase the level of generality at which online evolution can operate. Similarly to embodied evolution, OHE is distributed across a group of robots, meaning that it can leverage the inherent parallelism in multirobot systems to speed-up the search. As robots can evaluate distinct algorithms in parallel, and a given robot can make use of several algorithms throughout task execution, OHE creates the potential for a multi-trajectory search for high-performing controllers.

To assess the potential of OHE, we introduce two approaches called OHE-fitness and OHE-diversity that respectively use the fitness score and behavioural diversity of controllers produced by candidate algorithms as the criterion to select a promising algorithm at a given point in evolution. We then define two variants of OHE-fitness, called **OHE-fitness+fit** and **OHE-fitness+div**, and two variants of OHE-diversity, called **OHE-diversity+fit** and **OHE-diversity+div**, in which the evolutionary algorithms underlying OHE-fitness and OHE-diversity are themselves driven by fitness-based evolution and by diversity-based evolution. We implement our OHE approaches over different variants of odNEAT [5], [8], a representative efficient algorithm that evolves the topology and weights of artificial neural network (ANN) controllers in a distributed manner. We assess the performance of our approaches in two foraging tasks with differing complexity, and in five configurations of a dynamic phototaxis task with varying evolutionary pressures. Results show that our proposed OHE approaches: (i) outperform multiple state-of-the-art algorithms, as they facilitate the synthesis of controllers with higher performance and significantly faster evolution of controllers, and (ii) increase effectiveness at different stages of evolution by combining the benefits of different algorithms over time. The main conclusion of our study is that OHE is an effective new paradigm because of its ability to leverage and combine the properties of different algorithms to solve a given task.

2. Related Work

In evolutionary computation, there is a long history of tuning operators and parameters [9]. For example, in order to tune parameters during a run, several alternatives exist [10], such as: (i) *meta-evolution*, that is, using an evolutionary algorithm to optimise the configuration of the parameters of a task-solver evolutionary algorithm, (ii) *self-adaptation*, that is, using a single evolutionary algorithm that configures itself while it tries to solve the task: the parameters to be self-adapted are encoded in the genome and co-evolve with the candidate solutions, and (iii) adaptation of the parameter values according to predefined heuristic rules (e.g. adaptive Gaussian mutation step-size optimised according to the *one-fifth rule* [10]). In effect, such approaches are related with the idea of searching over a space of candidate configurations, and can thus be considered as antecedents of *hyper-heuristics* [11].

Hyper-heuristics are a search methodology intended to solve a large variety of different tasks with little or no human input. Specifically, hyper-heuristics have been described as [12]: "heuristics to choose heuristics" or as "an automated methodology for selecting or generating heuristics to solve hard computational problems". Hyper-heuristics systems can be represented as a two-level model [12], as shown in Fig. 1. The base level encapsulates a set of predefined heuristics for a given task, a performance function, and a given search space. The hyper level is responsible for deciding which base-level heuristic should be used to solve a given task, and may additionally generate new heuristics with a meta-heuristic search mechanism. Importantly, the hyper level can operate based on *online learning* or on *offline learning*. Similarly to online evolution algorithms, systems that employ online learning hyper-heuristics can learn directly from what they experience during task execution.

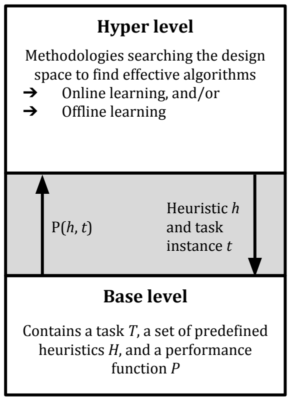

Hyper level
Methodologies searching the design space to find effective algorithms
➔ Online learning, and/or
➔ Offline learning
P(h, t)
Heuristic h and task instance t
Base level
Contains a task T, a set of predefined heuristics H, and a performance function P

Figure 1: Illustration of a hyper-heuristic system.

At the current state of development, hyper-heuristics have been successfully applied to a variety of domains, including [11]: (i) educational timetabling, (ii) production scheduling, (iii) workforce scheduling, (iv) constraint satisfaction, and (v) vehicle routing. That is, hyper-heuristics are a growing field of research with a number of practical applications. However, to the best of our knowledge, hyper-heuristics have not yet been assessed in evolutionary robotics domains, meaning that our study also includes the first hyper-heuristic driven by behavioural diversity. In the following section, we introduce hyper-heuristics in online evolution, which we term *online hyper-evolution*.

3. Online Hyper-Evolution

Online hyper-evolution (OHE) is a novel mechanism to accelerate and increase the performance of online evolution of controllers in multirobot systems. The key principle behind OHE is that, given a set of candidate algorithms, the different sources of feedback used to assess the quality of controllers during task execution can additionally be used *at the hyper level* to select suitable algorithms over time.

We propose OHE-fitness and OHE-diversity, which respectively use the fitness score and the behavioural diversity of controllers to select promising algorithms over time. In the following sections, we detail our proposed approaches.

3.1. OHE-fitness

Given a set of N candidate algorithms, OHE-fitness operates at the hyper level by monitoring the fitness score of controllers in the population of the robot, and by keeping track of which algorithm generated which controller. To that end, each genome is augmented with an integer identifier of the algorithm i that has originated it, with $i \in \{1..N\}$. When genomes are exchanged between robots, the fitness score of the corresponding controller and the identifier of the algorithm are also sent to the receiving robot. Thus, OHE-fitness can operate in a distributed and decentralised manner.

OHE-fitness intervenes in the evolutionary process when it is necessary to generate a new controller for a robot. Upon initialisation, that is, when the robot first starts executing, a randomly chosen algorithm is used. In subsequent interventions, OHE-fitness analyses the internal population of the robot and estimates the performance level P_i of every candidate algorithm i. The performance P_i of a candidate algorithm i is given by the mean fitness of the controllers it produced. An algorithm i is then selected proportionally to its performance level or randomly with a small probability p_{rs}. After the algorithm selection step, evolution proceeds as usual, with the configuration of the chosen algorithm during controller synthesis and evaluation.

3.2. OHE-diversity

Ultimately, online evolution of controllers synthesises *behavioural* control for robots. Based on the insight that controllers can be scored based on a characterisation of their behaviour during evaluation (see Fig. 2) and not only based on a traditional fitness function, Lehman and Stanley [13] introduced *novelty search*. In novelty search, the idea is to maximise the novelty of behaviours instead of their fitness, that is, to explicitly search for novel behaviours as a means to bootstrap evolution more effectively and to circumvent convergence to local optima. Because behaviours from more sparse regions of the behavioural search space receive higher novelty scores, the gradient of search is towards what is novel, with no explicit objective. Novelty search has attained considerable success in the evolution of controllers for a number of tasks, see [6], [13], [14], [15] and examples therein.

Inspired by novelty search, different studies have introduced behavioural diversity-based methods that explicitly reward the diversity of behaviours in a population [14]. In OHE-diversity, the behaviour is iteratively characterised at every control cycle of the robot (see next section for details). When a new controller for the robot is needed, a candidate algorithm is selected proportionally to its *behavioural diversity* score p_i or randomly with a small probability p_{rs}. The behaviour diversity score of a candidate algorithm

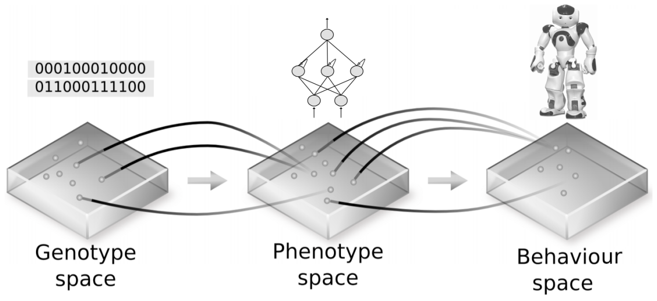


Figure 2: In traditional evolutionary algorithms, genomes of candidate solutions are translated into phenotypes whose fitness is assessed. In evolutionary robotics, the behaviour of the robot can also be characterised during task execution, which enables evolution to be guided by the search for novel or diverse behaviours instead of fitness. Adapted from [14].

is computed as the mean degree of behavioural diversity of the controllers it produced. In turn, the behavioural diversity score of a controller x is computed as the mean distance to its k-nearest neighbours in the population of the robot [14]. In the following section, we describe the two components required for behaviour diversity computations: (i) a characterisation of individual behaviours, and (ii) a distance metric $dist$ to quantify differences between pairs of behaviours.

3.2.1. Distributed State Count Characterisation. An effective task-dependent characterisation is usually the product of extensive trial-and-error experimentation [6]. We thus rely on a generic, task-independent characterisation. We use a modified version of the Combined State Count characterisation [15], henceforth called *Distributed State Count*.

The key principles behind the Distributed State Count characterisation are: (i) to define states based on the sensor readings and actuation values at every control cycle during controller evaluation, (ii) to compute the number of times a controller visited a given state, and (iii) when robots meet and exchange genomes, to propagate states counts along with genetic information. For algorithmic efficiency, each characterisation is represented as a map from states to counts [15]. In Algorithm 1, we describe the computation of the characterisation as independently executed by each robot during controller evaluation.

The function $read\_state(r)$ retrieves the vector of sensor readings and actuation values ϑ_r for robot r:

$$\vartheta_r = \{\mathbf{s}(r), \mathbf{a}(r)\} \tag{1}$$

where $\mathbf{s}(r)$ and $\mathbf{a}(r)$ are the sensor readings and actuation values. The function $discretise(\vartheta_r)$ computes the vector ϑ'_r by independently normalising each value in ϑ_r, and rounding the resultant value to the nearest integer:

$$\vartheta'_{i,r} = \left\| \frac{\vartheta_{i,r} - \vartheta_{i,min}}{\vartheta_{i,max} - \vartheta_{i,min}} \cdot (B-1) \right\|, i \in [1, N] \tag{2}$$

where $\vartheta_{i,max}$ and $\vartheta_{i,min}$ are respectively the maximum and minimum value of the i-th sensor/actuator, B is the

Algorithm 1 Pseudo-code of the Distributed State Count characterisation for a robot r.

```
$controller_r \leftarrow assign\_as\_controller(genome, r)$
$map \leftarrow Map < Integer, Integer >$
$alg\_id \leftarrow id\_of\_current\_algorithm$
while $controller_r$ is under evaluation do
    $execute\_control\_cycle()$
    $\vartheta_r \leftarrow read\_state(r)$
    $\vartheta'_r \leftarrow discretise(\vartheta_r)$
    $h \leftarrow hash(\vartheta'_r)$
    $map[h] \leftarrow map[h] + 1$
    if $broadcast?$ then
        $send(genome, alg\_id, map, robots\_in\_range)$
    end if
end while
```

number of discrete bins, and N is the length of ϑ_r and ϑ'_r. In this way, the parameter B is used to define the level of detail of the behaviour characterisation. Finally, the function $hash(\vartheta'_r)$ employs Jenkins's one-at-a-time function to hash the vector ϑ'_r into a single integer in order to improve the space- and time- complexity of the algorithm [15]. The behavioural distance $dist$ between two characterisations is given by a modified version of the Bray-Curtis dissimilarity, a well-known statistic used to quantify the difference between samples of abundance data, see [15] for details.

3.3. Driving the Search Process

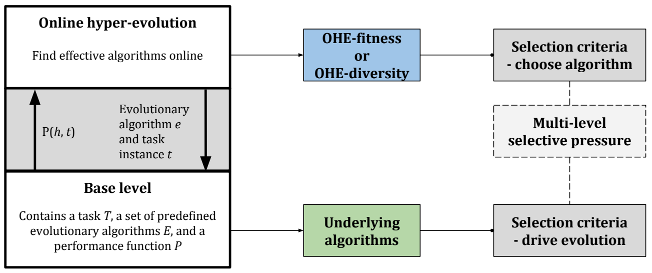
OHE-fitness and OHE-diversity can aid evolution by changing the algorithm used to search for promising solutions. However, the performance of OHE is fundamentally tied to how the search is guided at the *base level*. That is, the selective pressure and effectiveness of OHE influences and is influenced by the selective pressure of the individual algorithms at base level, giving rise to a *multi-level* selective pressure, see Fig. 3.

To investigate the multi-level selective pressure within OHE, we study: (i, ii) **OHE-diversity+fit** and **OHE-diversity+div**, which respectively represent OHE-diversity with fitness-based selection and diversity-based selection driving the base-level algorithms, and (iii, iv) **OHE-fitness+fit** and **OHE-fitness+div**, that is, OHE-fitness operating with fitness-based selection and diversity-based selection at the base level. Because behavioural diversity methods are an exploration procedure in the sense that they encourage a more expansive search, and fitness-based algorithms are an exploitation procedure as they typically focus on increasingly narrow regions of the search space, these approaches allow us to assess the relative merits of encouraging exploration and/or exploitation at different levels.

4. Methods

In this section, we define our simulation platform, the robot model, and the tasks used in the study.[1]

1. The source code of the experiments can be found at http://fgsilva.com/?page_id=319.



Online hyper-evolution
Find effective algorithms online
P(h, t)
Evolutionary algorithm e and task instance t
Base level
Contains a task T, a set of predefined evolutionary algorithms E, and a performance function P
OHE-fitness or OHE-diversity
Selection criteria - choose algorithm
Multi-level selective pressure
Underlying algorithms
Selection criteria - drive evolution

Figure 3: Scheme illustrating the multi-level selective pressure in online hyper-evolution.

4.1. Simulation Platform and Robot Model

Our simulation-based experiments are conducted using JBotEvolver [16], an open-source, multirobot simulation platform and evolutionary framework. We model the robots after the e-puck [17], a 7.5 cm in diameter differential drive robot that can move at speeds of up to 13 cm/s. Robots are equipped with infrared sensors that multiplex obstacle sensing and communication at a range of up to 25 cm. The controller details are listed in Table 1. Each sensor and each actuator are subject to noise, which is simulated by adding a random Gaussian component within $\pm$ 5% of the sensor saturation value or of the current actuation value.

The control system of each robot is a based on a discrete-time neural network with connection weights in the range [-10,10]. The inputs of the neural network are the readings from the sensors, normalised to the interval [0,1]. The output layer is composed of two neurons. The values of the output neurons are linearly scaled from [0,1] to [-1,1] to set the signed speed of each wheel. In the two foraging tasks, each robot is additionally equipped with a gripper, which allows the robot to collect the closest resource within a range of 2 cm, if there is any. A third output neuron is used to control the gripper (activated if the output value of the neuron is higher than 0.5, otherwise it is deactivated).

TABLE 1: Controller details. Light sensors have a range of 50 cm. Other sensors have a range of 25 cm.

Foraging tasks – controller details	
Input neurons: 25	
	4 for IR robot detection
	4 for IR wall detection
	1 for energy level reading
	8 for resource A detection
	8 for resource B detection
Output neurons: 3	
	2 for left and right motor speeds
	1 for controlling the gripper
Phototaxis task – controller details	
Input neurons: 25	
	8 for IR robot detection
	8 for IR wall detection
	8 for light source detection
	1 for energy level reading
Output neurons: 2	Left and right motor speeds

4.2. Foraging Tasks

In the foraging tasks, robots have to search for and collect objects spread across the environment. Foraging is a classic task in cooperative robotics, and is evocative of tasks such as search and rescue, harvesting, and toxic waste clean-up. In our experimental configuration, robots spend virtual energy at a constant rate, and gain energy by first finding and then collecting resources. When a resource is collected by a robot, a new resource of the same type is placed randomly in the environment to keep the number of resources constant. Following [3], [8], we setup two foraging tasks with different types of resources that robots have to collect: (i) the *standard foraging* task, in which there are only type A resources, and (ii) the *concurrent foraging* task, in which there are both type A and type B resources that have to be consumed alternately (a resource of type A followed by a resource of type B).

The energy level of a controller is initially set to 100 units, and is limited to the range [0,1000]. At each control cycle, E is updated according to the resources collected by the robot, as follows:

$$\frac{\Delta E}{\Delta t} = \begin{cases} r_{item} & \text{if right type of resource is collected} \\ w_{item} & \text{if wrong type of resource is collected} \\ \text{-0.02} & \text{if no resource is collected} \end{cases} \quad (3)$$

where r_{item} = 10 and w_{item} = -10. The number of resources of each type is set to the number of robots multiplied by 10. Note that the w_{item} component applies only to the concurrent foraging task.

4.3. Phototaxis Tasks

In traditional versions of the phototaxis task, a widely-used task in evolutionary robotics, robots have to find and move towards a fixed-position light source. Following previous studies [3], [8], we use a variant of the phototaxis task in which the light source is dynamic and is periodically moved to a new random location. In this way, the robots have to continuously search for and reach the light source, which eliminates controllers that find the light source by chance.

The virtual energy level is limited to the range [0,1000] units. Each controller is assigned an initial value of 100 units. At each control cycle, E is updated as follows:

$$\frac{\Delta E}{\Delta t} = \begin{cases} S_r & \text{if } S_r > 0.5 \\ 0 & \text{if } 0 < S_r \leq 0.5 \\ penalty & \text{if } S_r = 0 \end{cases} \quad (4)$$

S_r is the maximum value of the readings from light sensors, between 0 (no light) and 1 (brightest light). To study the relation between evolutionary pressure and evolutionary dynamics, we setup five task variants. In each variant, we changed the value of the $penalty$ component when the light source is not within a robot's sensory range. The $penalty$ component is set to a value of -0.01, -0.02, -0.05, -0.08, and -0.10 per control cycle, and a controller unable to find the light source can only survive for a period of 1000, 500, 200, 125, and 100 seconds, respectively. These experimental setups are henceforth referred to as **p1**, **p2**, **p5**, **p8**, and **p10** setups. In all variants, light sensors have a range of 50 cm, that is, robots are only rewarded if they are close to the light source. The remaining sensors have a range of 25 cm.

4.4. Base-level Evolutionary Algorithms

Given its generality, OHE can be implemented over a number of different algorithms. We use three variants of the odNEAT algorithm [5], a distributed online evolution algorithm that optimises both the weights and the topology of ANNs. odNEAT has been successfully used in a number of simulation-based studies, in which it was shown to enable: (i) adaptivity, that is, the ability to effectively evolve controllers for robots that operate in dynamic environments [18], (ii) robustness, as the controllers evolved can often adapt to changes in environmental conditions without further evolution [5], and (iii) fault tolerance, that is, robots executing odNEAT are able to adapt their behaviour in the presence of sensor faults [5]. odNEAT is thus used here as a representative online evolutionary algorithm.

odNEAT starts with minimal artificial neural networks with no hidden neurons, that is, with each input neuron connected to every output neuron. Throughout evolution, topologies are gradually complexified by adding new neurons and new connections through mutation. In addition, the internal population of each robot implements a niching scheme comprising speciation and fitness sharing, which allows each robot to maintain a healthy diversity of candidate solutions with distinct topologies [5].

During task execution, each robot is controlled by an ANN-based controller that represents a candidate solution to the task. Each controller maintains a virtual energy level reflecting its individual task performance. The fitness score is defined as the mean energy level, sampled at regular time intervals. When the virtual energy level reaches a minimum threshold, the current controller is considered unfit for the task. A new controller is then synthesised via selection of a parent species and two genomes from that species (the parents), crossover of the parents' genomes, and mutation of the offspring. Mutation is both structural and parametric, that is, it adds new neurons and new connections, and optimises connection weights and neuron bias values. Once the new genome is decoded into a new controller, it is guaranteed a maturation period during which no controller replacement takes place. The new controller can continue to execute after the maturation period if its energy level is above the threshold. That is, a controller remains active as long as it is able to solve the task.

4.4.1. odNEAT variants. In online evolution algorithms, two key aspects are: (i) the controller evaluation policy, and (ii) the controller exchange policy. Because the effectiveness of both aspects is task-dependent, we use two variants of odNEAT that differ from the standard version of odNEAT in the controller evaluation policy and in the controller exchange policy. That is, we use algorithm variations with relative advantages and disadvantages. In this section, we review the two variants of standard odNEAT.

Controller evaluation policy: Traditional online evolution algorithms employ a policy in which robots substitute controllers at regular time intervals, see [4] for an example. While the approach is suitable for individual tasks, it has been shown to lead to incongruous group behaviour and to poor performance in collective tasks that explicitly require continuous collective coordination and cooperation [5]. Algorithms such as odNEAT, on the other hand, allow a controller to remain active as long as it is able to solve the task. However, such approach can also be too conservative and delay the synthesis of more effective solutions [3] by evolving intermediate controllers that can operate for a long period of time before they fail.

To cut short the evaluation of inferior intermediate controllers, odNEAT was recently extended [8] with a racing approach for multirobot systems, a variant henceforth called **racing**. The approach relies on the task performance of the controllers assessed by the different robots, and on a non-parametric statistical approach. The evaluation of a controller x is aborted, and a new controller is generated, if the performance of x is below $M_c(t)$. $M_c(t)$ is a progressively stricter minimal criterion of performance based on the P-th percentile of the fitness scores in the population, see [8].

Controller exchange policy: The exchange of genetic information between robots is a crucial feature in distributed, online evolutionary algorithms. In traditional approaches, individual robots transmit to neighbouring robots either part of a genome [2] or a complete genome [4], [5]. The genome is the unit in the selection process, and the population of robots is a distributed substrate across which genetic information can spread. In order to give robots the potential to leverage the genetic information they have accumulated, and to enable a more effective knowledge transfer, **racing** was additionally extended with a population cloning technique [8]. We henceforth refer to the population cloning technique as **ppc**. The approach puts the selection and reproductive processes at a higher level by considering the elements involved in the selection process to be the internal population of each robot. As a result, a robot can transmit to neighbouring robots *a copy* of any part of its population (e.g. a single genome or a set of genomes representing high-performing controllers) or of the complete population, which can push evolution towards higher-quality solutions [8].

When two **ppc**-executing robots meet, their internal populations compete, and the losing robot receives a portion of the population of the winning robot. Firstly, winner and loser are determined by comparing the performance of each population according to their $M_c(t)$ value (as defined in **racing**). The robot with the highest $M_c(t)$ value is considered the winner. Secondly, the internal population of the losing robot is subject to an extinction event. The genomes of the losing robot that yield a fitness score below the winner robot's $M_c(t)$ are removed before the injection of new genomes. Finally, the genomes from the population of the winning robot that have a fitness score above $M_c(t)$ are injected into the losing robot.

4.5. Experimental Parameters and Treatments

For each task variant and each algorithm considered, we conduct 30 independent runs. Each run lasts 100 hours of simulated time. Robots operate in a square arena surrounded by walls. The size of the arena is chosen to be 3 x 3 meters. The parameters of odNEAT and variants are set as in previous studies [5], including a population size of 40 genomes per robot. Regarding the minimal criterion for **racing**, we follow [8] and set P to the 50th percentile of the fitness scores found in the population, meaning that $M_c(t)$ amounts to the median fitness score. For OHE approaches, p_{rs} is set to 0.10. OHE approaches with a behaviour diversity component use a k value of 5 nearest neighbours. The number of bins is set to $B = 10$ bins. k and B were tuned empirically. These parameter settings are robust to moderate variation, and were found to perform effectively in preliminary experiments.

In addition to the quality of the controllers evolved, another relevant aspect is how the different algorithms explore the behaviour space, individually and with respect to each other. To visualise how the different approaches traverse the behaviour space, we use *Sammon's nonlinear mapping* [19]. Sammon's mapping is a multidimensional scaling algorithm that performs a point mapping of high-dimensional data to two- or three-dimensional spaces, such that the structure of the data is approximately preserved. The algorithm minimises the error measure E_m, which represents the disparity between the high-dimensional distances δ_{ij} and the resulting distance d_{ij} in the lower dimension for all pairs of points i and j. E_m is computed as follows:

$$E_m = \frac{1}{\sum_{i=1}^{n-1}\sum_{j=i+1}^{n}\delta_{ij}} \sum_{i=1}^{n-1}\sum_{j=i+1}^{n} \frac{(\delta_{ij} - d_{ij})^2}{\delta_{ij}} \qquad (5)$$

We use the two-tailed Mann-Whitney U test to determine the statistical significance of differences between results because it is a non-parametric test, and therefore no strong assumptions need to be made about the underlying distributions. When multiple comparisons are performed using the results obtained in a given set of runs, we adjust the ρ-value using the two-stage Hommel method [20].

5. Experimental Results

In this section, we assess our proposed OHE approaches. Firstly, we compare the performance of the most straightforward OHE approach, namely **OHE-fitness+fit**, to the performance of each individual evolutionary algorithm. We then compare the different OHE approaches in terms of task performance and exploration of behaviour space.

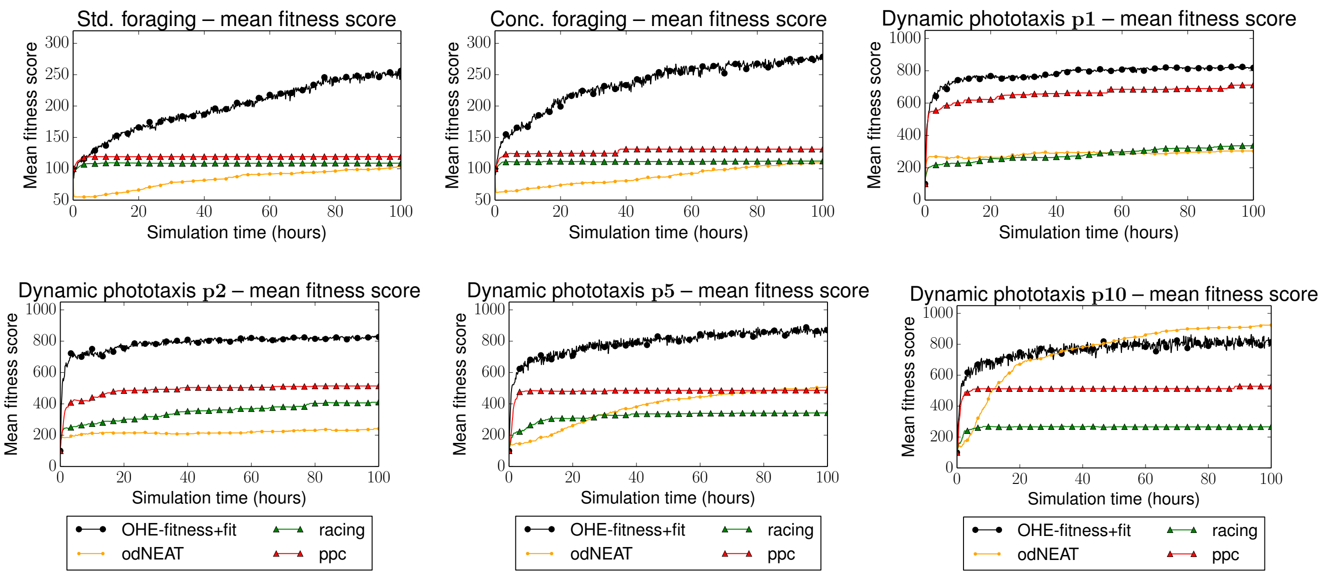


Figure 4: Mean fitness score of controllers evolved by base-level algorithms and by **OHE-fitness+fit** throughout the simulation trials of the standard foraging and concurrent foraging tasks (top, left and center), and four variants of the dynamic phototaxis task (top right, and bottom).

5.1. OHE-fitness+fit vs. Base-level Algorithms

The mean fitness score of controllers throughout the simulation trials is shown in Fig. 4 for the two foraging tasks, and for four variants of the phototaxis task (**p1**, **p2**, **p5**, and **p10** setups). Overall, results show that **OHE-fitness+fit** typically yields superior performance and that it is able to synthesise effective solutions in the early stages of evolution.

In the two foraging tasks, **OHE-fitness+fit** typically yields performance levels superior to those of the individual algorithms, and the highest-performing individual algorithm is **ppc**. Differences between the mean group fitness of the final controllers evolved by **OHE-fitness+fit** and that of those evolved by other algorithms are statistically significant across every comparison ($\rho < 0.0001$, Mann-Whitney).

In the dynamic phototaxis task, **OHE-fitness+fit** is still the highest-performing approach, but the differences in performance between **OHE-fitness+fit** and the individual algorithms is dependent on the evolutionary pressure. In the least demanding configuration (**p1** setup), **OHE-fitness+fit** significantly outperforms **odNEAT** and **racing** ($\rho < 0.0001$, Mann-Whitney), and **ppc** is the algorithm that gets closest to the performance levels of **OHE-fitness+fit**. In the **p2** and **p5** setups, **OHE-fitness+fit** continues to outperform every other algorithm ($\rho < 0.001$, Mann-Whitney). However, as the task becomes even stricter, the performance of **odNEAT** furthermore increases, and the algorithm is able to outperform **OHE-fitness+fit** in the final part of the **p10** setup ($\rho < 0.001$, Mann-Whitney). The increasingly higher performance performance of **odNEAT** as the dynamic phototaxis task becomes stricter is due to the controller replacement frequency of the algorithm. Specifically, as the task difficulty is increased, **odNEAT** replaces the controller of each robot more frequently: on average, each robot executing **odNEAT** produces on average 9 controllers in the **p1** setup, 24 controllers in the **p2** setup, 120 controllers in the **p5** setup, 420 controllers in the **p8** setup, and 920 controllers in the **p10** setup. This result is consistent with previous studies [8], which have shown that if the evolutionary pressure is set above a certain limit, **odNEAT** can, under certain conditions, display increased performance in the long term.

5.1.1. Algorithm Selection Analysis.

To investigate the dynamics of **OHE-fitness+fit**, we analysed the base-level algorithms used by the approach. Throughout this section, we refer to the selection and execution of a given algorithm by OHE as an *algorithm instance*.

Figure 5 shows the mean number of algorithm instances used by **OHE-fitness+fit** in the **p1** setup. Remaining tasks yield similar trends. Throughout evolution, **OHE-fitness+fit** selects **ppc** significantly more often than the other two algorithms to synthesise effective controllers ($\rho < 0.0001$, Mann-Whitney). Specifically, **ppc** is used to generate, on average, 90% or more of the controllers evaluated during the search process. For example, in the **p1** setup, **ppc** is chosen approximately 158 times per robot, while **odNEAT** and **racing** are each chosen approximately six times per robot. Importantly, **ppc** is typically used more intensively in the early stages of evolution, thus indicating that it helps to boost the evolutionary process and push towards higher-quality solutions. It is also noteworthy that the base-level algorithms used in this study cause different controllers to have potentially different evaluation times, as discussed in Section 4.4.1. In this way, it is also relevant to consider

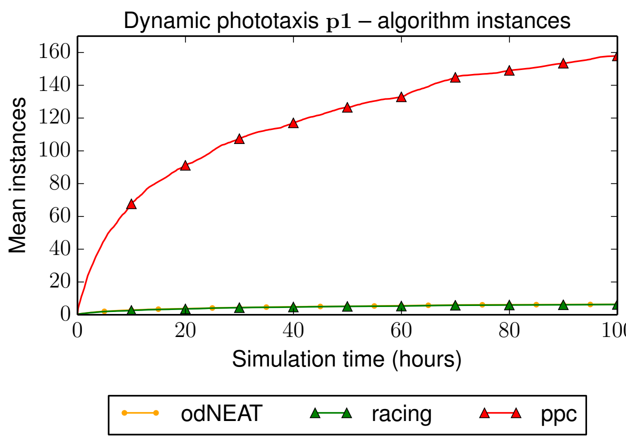


Figure 5: Mean number of algorithm instances used by **OHE-fitness+fit** throughout evolution in the **p1** setup.

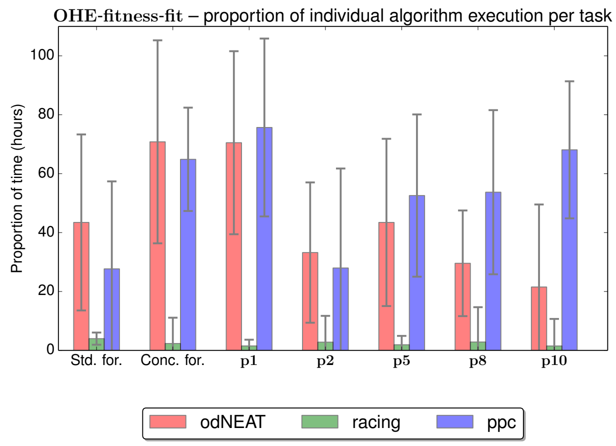


Figure 6: Mean absolute proportion of time per robot that each algorithm was used by **OHE-fitness+fit** in each task.

the *amount of time* that a controller produced by a given algorithm executed.

Figure 6 shows the mean proportion of time per robot that each algorithm was used by **OHE-fitness+fit** in the different tasks, which is given by the execution time of the *controllers* it produced. Despite the more consistent selection of **ppc**, **odNEAT**-produced controllers execute for a comparable amount of time in the majority of the tasks. In effect, when **odNEAT** and **ppc** are selected to synthesise controllers, the fitness score of controllers increases on average from approximately 22% to 160%; **racing**, on the other hand, causes the fitness scores to decrease by approximately 19%. The performance of **OHE-fitness+fit** is thus, not solely due to the **ppc**, but instead is caused by an effective combination of **odNEAT** and **ppc** throughout evolution. The key result is thus that different algorithms are important at different phases of evolution and used for different periods of time. This indicates that OHE stands for more than an effective algorithm selection approach: OHE can increase effectiveness and boost performance at different stages of evolution.

Overall, our results illustrate the potential of **OHE-fitness+fit**, namely that it can effectively combine the benefits of different algorithms to speed up and increase the performance of the evolutionary process. In the following section, we study if and how the performance of OHE changes with respect to how the search is guided at the hyper level and at the base level.

5.2. Multi-level Selective Pressure

In this section, we analyse the performance and behaviour of different OHE variants, namely **OHE-fitness+fit**, **OHE-fitness+div**, **OHE-diversity+fit**, and **OHE-diversity+div**, see Sect. 3.3 for a description.

Figure 7 shows the mean fitness score throughout evolution for the two foraging tasks and for the dynamic phototaxis **p5** setup. Remaining variations of the dynamic phototaxis task yield comparable results. Firstly, results show that driving evolution towards behavioural diversity at the base level is detrimental to the performance of OHE. In this respect, differences between the fitness scores of final controllers synthesised by fitness-based evolution and those synthesised by diversity-based evolution are significant in the two foraging tasks ($\rho < 0.0001$), in the **p1** setup ($\rho < 0.05$, scores not shown in Fig. 7), and in the **p5** setup ($\rho < 0.005$). Secondly, with respect to **OHE-fitness+fit** and **OHE-diversity+fit**, the two approaches yield comparable performance levels in all tasks, although there is a slight advantage in favour of **OHE-diversity+fit** in the standard foraging task. However, as shown by the proportion of individual algorithm executions, see Fig. 6 and Fig. 8, the two approaches use the base-level algorithms differently, both in terms of the number of times each algorithm instance is used and the amount of time that controllers produced by a given instance execute. These combined results show that **OHE-fitness+fit** and **OHE-diversity+fit** follow different approaches to the synthesis of high-performing solutions to the tasks, which in turn highlights that the search mechanisms underlying the different levels in OHE can exert significant influence on the evolutionary path over time.

To further assess the dynamics of the OHE approaches, we analysed how they traverse the behaviour space using Sammon's mapping [19], see Sect. 4.5. The distance in the high-dimensional space δ_{i_j} between the behaviour of two controllers i and j is given by the count of states (number of states and respective cardinality) in which the behaviours differ. The distance between two points in the two-dimensional space is their Euclidean distance. Note that **OHE-fitness+fit** does not make use of behavioural diversity during evolution.

The three mappings in Fig. 9 show how the different approaches compare with one another in terms of behaviour space exploration (**p5** setup, remaining tasks yield similar results). To obtain a clear visualisation and a representative selection of behaviours, we map the 250 most behaviourally different controllers produced by each of the four approaches. The area of each point in the two-dimensional

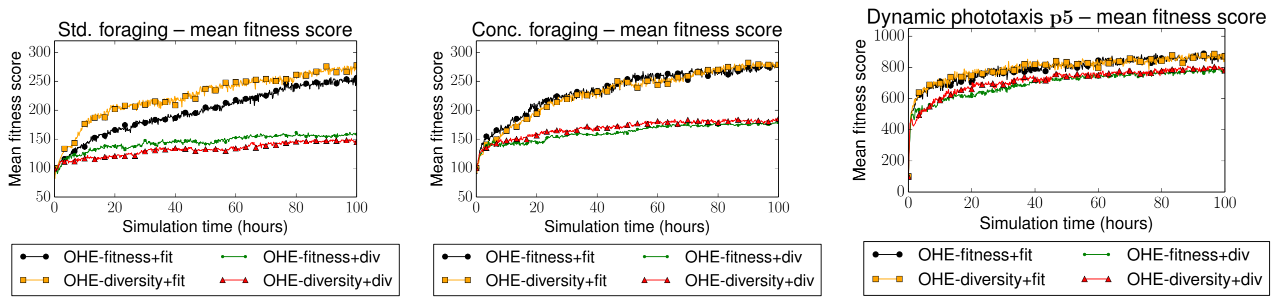


Figure 7: Mean fitness score of controllers evolved by the different OHE techniques throughout the simulation trials of: (i) the standard foraging and concurrent foraging tasks (left and center), and (ii) in the **p5** setup.

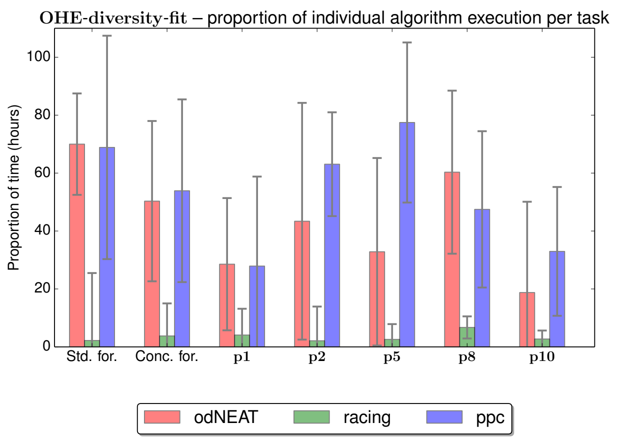


Figure 8: Mean absolute proportion of time per robot that each algorithm was used by **OHE-diversity+fit** in each task.

space is set proportionally to the fitness score of the corresponding controller. Behaviours belonging to high-fitness regions (fitness $> 75\%$ of maximum fitness) are marked with a grey square in the centre of the corresponding point. The error value is respectively $E_m = 0.025$ for the first two mappings, and 0.027 for the third mapping, which indicates that the distances between behaviours are well-preserved.

The first mapping compares the two OHE-fitness variants, **OHE-fitness+fit** and **OHE-fitness+div**. Given its ability to guide evolution towards diversity, **OHE-fitness+div** naturally performs a more uniform exploration of the behavioural search space than **OHE-fitness+fit**. The second mapping compares the two approaches that exploit fitness-based evolution and diversity-based evolution in opposite ways. Comparing with **OHE-fitness+div**, the **OHE-diversity+fit** approach is able to both cover more regions of the behaviour search space, *and* to find higher-performing solutions in those regions, see behaviours located in the right half of Fig. 9 (middle). Complementarily to the results in Fig. 7, which show that **OHE-diversity+fit** is typically the highest-performing approach, the behaviour space exploration indicates that the multi-level selective pressure of OHE benefits if the hyper-level search is driven towards

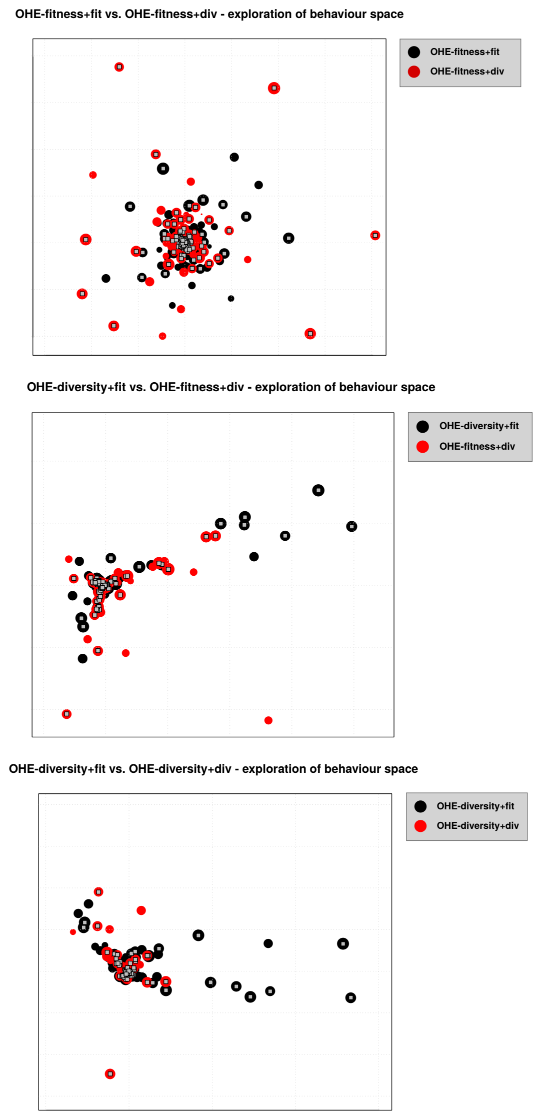


Figure 9: Sammon's mapping for the **p5** setup: (top) **OHE-fitness+fit** vs. **OHE-fitness+div**, (middle) **OHE-diversity+fit** vs. **OHE-fitness+div**, and (bottom) **OHE-diversity+fit** vs. **OHE-diversity+div**. Behaviours in high-fitness regions are marked with a grey square.

behavioural diversity and the base-level search is driven towards higher fitness regions. In effect, as shown in the third mapping, **OHE-diversity+fit** also explores more regions of *high-performing* solutions than **OHE-diversity+div**, which is driven towards diversity in both levels.

Overall, the analyses in this section show that the OHE approaches have different dynamics both in terms of how they use the base-level algorithms, and of how they traverse the behaviour space. In this respect, although the four approaches consistently yielded high-performance across the tasks, **OHE-diversity+fit** was found to be the most effective one.

6. Conclusions and Discussion

In this paper, we introduced a novel approach to accelerate and increase the performance of online evolution of controllers in multirobot systems. We proposed *online hyper-evolution* (OHE), in which the evolutionary process can search across a space of candidate algorithms online. We studied four OHE approaches: **OHE-fitness+fit**, **OHE-fitness+div**, **OHE-diversity+fit**, and **OHE-diversity+div**, in which exploration and exploitation are conducted differently in the hyper level and in the base level. Experimental results showed that our OHE approaches: (i) facilitate the evolution of controllers with superior performance, and faster synthesis of solutions to the task, and (ii) effectively combine different algorithms to increase the effectiveness of the evolutionary process at distinct phases.

The main conclusion of our study is that OHE represents a simple and effective approach to online evolution of robotic controllers. In this respect, **OHE-diversity+fit** was shown to be the most effective approach as it can successfully push towards diversity and performance. In ongoing work, we are studying how **OHE-diversity+fit** fares against other approaches that combine diversity and fitness (e.g. multiobjective and linear scalarisation algorithms), assessing alternative algorithm selection strategies, and if and how low-performing algorithms can be removed from the selection process. Our final goal is to enable adaptation at different levels (e.g. parameter configurations such as mutation and/or crossover rates, and algorithmic components such as speciation or crossover), and the *construction* of complete algorithms during task execution.

Acknowledgments

This work was partly supported by FCT under grants SFRH/BD/89573/2012, UID/EEA/50008/2013, and UID/Multi/04046/2013.

References

[1] D. Floreano and F. Mondada, "Evolution of homing navigation in a real mobile robot," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 26, no. 3, pp. 396–407, 1996.

[2] R. Watson, S. Ficici, and J. Pollack, "Embodied evolution: Distributing an evolutionary algorithm in a population of robots," *Robotics and Autonomous Systems*, vol. 39, no. 1, pp. 1–18, 2002.

[3] F. Silva, L. Correia, and A. L. Christensen, "A case study on the scalability of online evolution of robotic controllers," in *Proceedings of the 17th Portuguese Conference on Artificial Intelligence*. Springer, Berlin, Germany, 2015, pp. 189–200.

[4] N. Bredeche, J. M. Montanier, W. Liu, and A. Winfield, "Environment-driven distributed evolutionary adaptation in a population of autonomous robotic agents," *Mathematical and Computer Modelling of Dynamical Systems*, vol. 18, no. 1, pp. 101–129, 2012.

[5] F. Silva, P. Urbano, L. Correia, and A. L. Christensen, "odNEAT: An algorithm for decentralised online evolution of robotic controllers," *Evolutionary Computation*, vol. 23, no. 3, pp. 421–449, 2015.

[6] 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.

[7] D. H. Wolpert and W. G. Macready, "Coevolutionary free lunches," *IEEE Transactions on Evolutionary Computation,*, vol. 9, no. 6, pp. 721–735, 2005.

[8] F. Silva, L. Correia, and A. L. Christensen, "Leveraging online racing and population cloning in evolutionary multirobot systems," in *Proceedings of the 19th European Conference on the Applications of Evolutionary Computation*. Springer International Publishing, Switzerland, 2016, pp. 165–180.

[9] W. Kantschik, P. Dittrich, M. Brameier, and W. Banzhaf, "Empirical analysis of different levels of meta-evolution," in *Proceedings of the IEEE Congress on Evolutionary Computation*. IEEE Press, Piscataway, NJ, 1999, pp. 2086–2093.

[10] A. E. Eiben, Z. Michalewicz, M. Schoenauer, and J. E. Smith, "Parameter control in evolutionary algorithms," in *Parameter Setting in Evolutionary Algorithms*, ser. Studies in Computational Intelligence. Springer, Berlin, Germany, 2007, vol. 54, pp. 19–46.

[11] E. K. Burke, M. Gendreau, M. Hyde, G. Kendall, G. Ochoa, E. Özcan, and R. Qu, "Hyper-heuristics: A survey of the state of the art," *Journal of the Operational Research Society*, vol. 64, no. 12, pp. 1695–1724, 2013.

[12] P. Ryser-Welch and J. F. Miller, "A review of hyper-heuristic frameworks," in *Proceedings of the 50th Annual Convention of the Society for the Study of Artificial Intelligence and the Simulation of Behaviour*, 2014, available at http://doc.gold.ac.uk/aisb50/.

[13] 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.

[14] 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.

[15] J. Gomes and A. L. Christensen, "Generic behaviour similarity measures for evolutionary swarm robotics," in *Proceedings of the 15th Genetic and Evolutionary Computation Conference*. ACM Press, New York, NY, 2013, pp. 199–206.

[16] 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 14th International Conference on the Synthesis and Simulation of Living Systems*. MIT Press, Cambridge, MA, 2014, pp. 210–211.

[17] F. Mondada, M. Bonani, X. Raemy, J. Pugh, C. Cianci, A. Klaptocz, S. Magnenat, J. Zufferey, D. Floreano, and A. Martinoli, "The e-puck, a robot designed for education in engineering," in *Proceedings of the 9th Conference on Autonomous Robot Systems and Competitions*. IPCB, Castelo Branco, Portugal, 2009, pp. 59–65.

[18] F. Silva, P. Urbano, and A. L. Christensen, "Online evolution of adaptive robot behaviour," *International Journal of Natural Computing Research*, vol. 4, no. 2, pp. 59–77, 2014.

[19] J. Sammon Jr., "A nonlinear mapping for data structure analysis," *IEEE Transactions on Computers*, vol. C-18, no. 5, pp. 401–409, 1969.

[20] G. Hommel, "A stagewise rejective multiple test procedure based on a modified Bonferroni test," *Biometrika*, vol. 75, no. 2, pp. 383–386, 1988.