



# A Multi-brain Approach for Multiple Tasks in Evolvable Robots

Ege de Bruin<sup>(✉)</sup>, Julian Hatzky<sup>(✉)</sup>, Babak Hosseinkhani Kargar<sup>(✉)</sup>,  
and A. E. Eiben

Department of Computer Science, Vrije Universiteit Amsterdam,  
Amsterdam, The Netherlands

egedebruin@gmail.com, julianhatzky@gmail.com  
{b.hosseinkhanikargar,a.e.eiben}@vu.nl

**Abstract.** We investigate the joint evolution of morphologies (bodies) and controllers (brains) of modular robots for multiple tasks. In particular, we want to validate an approach based on three premises. First, the controller is a combination of a user-defined decision tree and evolvable/learnable modules, one module for each given task. Second, morphologies and controllers are evolved jointly for each task simultaneously by a multi-objective evolutionary algorithm. Third, after terminating the evolutionary process, the brain of the users' favorite morphology is optimized by a learning algorithm applied to the task-specific controller modules independently.

**Keywords:** Evolutionary Robotics · Morphological evolution · Controller evolution · Robot learning · Multi-Objective Optimization · Locomotion

## 1 Introduction

Evolutionary robotics is a field where evolutionary algorithms are used to evolve robots. In the early years of the field, the focus was mainly on evolving the controllers of robots with a fixed morphology [11,21], but more recently the evolution of robot morphologies has been addressed as well [2,6,10,26].

Much of the existing work is based on evolving bodies and brains for one task, where acquiring an adequate gait is a popular problem. However, simply moving around without a goal is not really practical. Functional robots need to move with purpose, for example, move towards a target in sight. There has been work done for this targeted locomotion, on robots with fixed morphologies [19] and robots with evolvable morphologies [15,16]. These works however assume that the robot knows where the target is, and if the target is not in sight it assumes the target is at the last known position. When it is completely unaware of where the target is, it will not search for it.

In this work we, consider a practically more relevant case, where the robot has to 1) find the target and 2) move to the target, and be able to repeat this for additional targets.

To achieve this, we want to end up with a robot body, that is fully capable to perform those tasks, together with a controller architecture to move the robot and that is a combination of a user-defined decision tree and evolvable brains, one brain for each given task.

The decision tree is to capture the users' domain knowledge where the solution needs not to be evolved, only coded. For instance, `IF target-not-in-sight THEN search-for-it` or `IF target-in-sight THEN move-towards-it`. The evolvable modules represent (morphology-dependent) sub-controllers specifying how that given task can be executed by the given body. Thus, the first research question we address is:

*Can a multi-objective evolutionary algorithm deliver good bodies for modular robots for handling two tasks?*

Our approach is, in essence, a multi-brain system: one brain for each task, combined through a decision tree. However, the first research question is focused on delivering good bodies for robots to handle two tasks regardless of doing the tasks separately or simultaneously. Hence, the second question we address is:

*How much can a secondary learning stage that separately optimizes each task enhance robot performance in a given morphology?*

To this end, we use a two-phase approach, where first the creatures are evolved to do the tasks simultaneously, and then the creatures learn to do the two tasks separately. This results in one morphology and two different brains, one for each task, which can be used in the robot's controller. Naturally, this procedure can be extended to an arbitrary number of tasks by creating an additional number of brains.

## 2 Background

Locomotion for modular robots is a difficult task, but there has been promising work with robot locomotion based on Central Pattern Generators (CPGs) [14]. Moreover, a neural network-based approach, for example using HyperNEAT [12], can be used to evolve good controllers for robot locomotion. For targeted locomotion, there are studies focusing on robots with fixed shapes [19]. Lan et al. propose a method for targeted locomotion of generic shapes by morphologically evolving the robots [16]. A CPG-based approach was used with a HyperNEAT generative encoding technique. This study was later extended to follow a moving target [15]. In this work, when the target went out of sight of the robot, the robot assumed the last-known angle towards the target to be the angle towards the target. These studies showed that a robot can evolve and learn to walk towards and follow a target.

Considering multiple objectives, the task of the robot influences its morphology [4]. This is especially the case when the tasks are conflicting, and when the multiple objectives are put into a single function. It is not certain that a single solution exists for the problems [7], and in a single fitness function, it is difficult

to find the correct trade-off between multiple behavioral terms [25]. It is therefore preferred to use a multi-objective function, and the NSGA-II algorithm has been shown to be a good and fast algorithm for multi-objective functions [20] [8].

A paper by Lipson et al. shows that when the morphology and brain of the robot are evolved together, we run into premature convergence, and this is especially seen in the morphology of the creature [5]. Moreover, when morphology and brain are evolved together to optimize multiple tasks, this will lead to worse results than when the creatures are evolved for the tasks separately [3]. Nygaard et al. try to overcome these convergences with a two-phase approach [22]. First, the morphology and brain are evolved together. Then, after convergence, the morphology is fixed and the brain evolves further. This paper shows that after convergence of evolving the morphology and brain together, the creature can still perform better when only the brain is evolved further. This is acknowledged by Eiben and Hart [9], who state that after the evolution of brain and body together, the brain needs to learn the optimal way to use the body. A paper by Lessin et al. showed results for an approach to evolve creatures for multiple objectives [18]. The robot was initially evolved for a locomotion task, after which most of the morphology was fixed and other tasks were learned after this. This was later extended by a method to make the morphology more flexible after the initial evolution phase [17]. This differs from our approach by evolving robots to more specific actions, like turning left and turning right, and by evolving the robot’s morphology per action instead of by all actions together.

### 3 Experimental Work

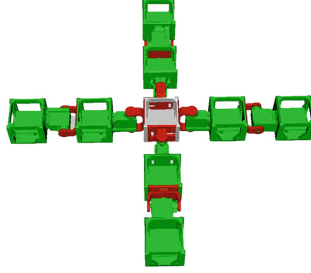
#### 3.1 Phenotypes and Simulation

The robots are simulated on a Gazebo-based simulator *Revolve* [13]<sup>1</sup> The robot design is based on RoboGen [1]. There are three different robot modules which can be used as building blocks for the robots. At first, every robot has one *core* component which, when the robot is built in real life, contains the controller board. This block has four possible connections to other components. Another component is the *fixed brick*, which has four possible slots to attach other components. Lastly, there is an *active hinge* component and this is the component responsible for the robot’s movement. It is a joint that can be attached on two lateral sides. In the simulator the orientation of the robot is a virtual sensor, if the robot would be built in real life the camera would be located in the core component of the robot. In Fig. 1 an example of robot with its possible components is shown.

#### 3.2 Central Pattern Generators

As a controller for the robots, Central Pattern Generators (CPGs) have been shown to perform well for a task like targeted locomotion. CPGs are neural

<sup>1</sup> (see Revolve <https://github.com/ci-group/revolve>).



**Fig. 1. A possible robot from RoboGen.** The middle white component is the core component, the green square components are the fixed brick components, and the red parts are the active hinge components. (Color figure online)

networks that are responsible for the rhythmic movement of animals, without any sensory information or rhythmic inputs [14]. Because they are independent of higher control centers, this reduces time in the motor control loop and reduces the dimensionality to control movements. This concept is applied to robots, where CPG models are being used to control the locomotion of robots. In this work, CPGs are used in the hinges of the robots, with differential oscillators, which are responsible for rhythmic movement, as its main components. The oscillators generate patterns by calculating activation levels of neurons  $x$  and  $y$  shown at the top of Fig. 2. Moreover, there is an output neuron that outputs the value for the CPG model. For these neurons, the equations for calculating the difference per time step can be seen in Eqs. 1 and 2.

$$\Delta x = w_{yx}y + bias_x \quad (1)$$

$$\Delta y = w_{xy}x + bias_y \quad (2)$$

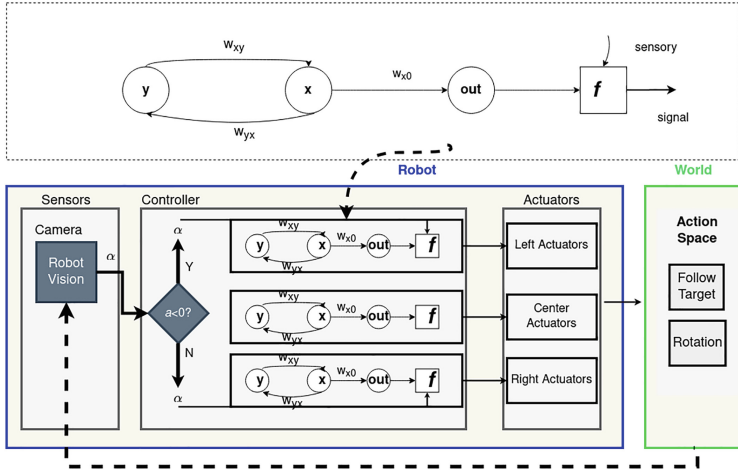
The activation functions of the output neurons are *tanh* functions. The hinges also affect each other, therefore the CPG components are extended by taking into account the neurons of neighboring hinges. To be more precise, for each pair of neighboring hinges the  $x$  neurons in the CPG models also have an influence on each other. Therefore, the activation values for the  $x$  and  $y$  nodes can be calculated by the equations in Eqs. 3 and 4.

$$x_i(t) = x_i(t-1) + \Delta x_i(t) + \sum_{j \in N_j} x_j(t-1) * w_{ji} \quad (3)$$

$$y_i(t) = y_i(t-1) + \Delta y_i(t) \quad (4)$$

In these equations,  $N_j$  are all neighboring hinges of hinge  $i$ , and  $\Delta x_i$  and  $\Delta y_i$  are calculated from Eqs. 1 and 2. To also take into account the angle from

the robot’s vision direction towards the target, the CPG model is extended with sensory input as shown in Fig. 2. When the angle towards the target is lower than  $\alpha$ , the target is assumed to be on the left and otherwise on the right. This has an influence on the actuators, where the left actuators are influenced when the target is on the left and visa versa. Whether the actuator is on the left or on the right depends on its lateral position and the direction of the target. An overview of this approach can be seen at the bottom of Fig. 2.



**Fig. 2. Overview of a CPG component with angle towards target taken into account.** When the target is on the left,  $\alpha$  is assumed to be lower than 0 and only the left actuators are influenced. Otherwise the right actuators are influenced.

### 3.3 HyperNEAT and Compositional Pattern Producing Networks

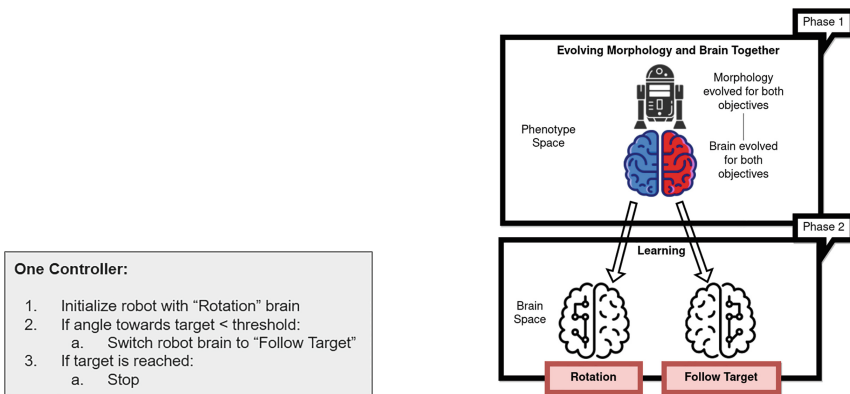
We use HyperNEAT, which is a hypercube-based encoding to evolve large-scale neural networks [23]. HyperNEAT is an effective choice when learning the weights for the CPG controllers for a given task. The idea behind HyperNEAT is that there is a substrate network, represented by for example a hyper-dimensional cube, of which the coordinates of a node are used as input for a CPPN. The CPPN is then evolved using NEAT [24] to find the optimal structure and activation functions of the CPPN for the problem at hand. For our work, to get the correct connection weight values, the CPPN will have as input the coordinates of the source and target CPG component, including a 1 if the target node is an x-CPG-node, a -1 if it is a y-CPG-node, and 0 if it is an output-CPG-node. The output of the CPPN is then the weight of the connection.

To evolve and generate the morphology of a robot a second CPPN is used. This network takes as input the x, y and z coordinates of the possible location for a module of the robot, and the length towards the core module as well as input. The output is the module that is used, hinge, block or no module, and the

rotation of the module. The generation of the robots starts at the core module with coordinates  $(0,0,0)$ , and from this module the robot is extended with new modules dependent on the outputs of the CPPN. This CPPN is also evolved using NEAT.

### 3.4 Two-Brain Approach

Other work related to targeted locomotion has mostly focused on only the locomotion part. For example, recent work by Lan et al. [15] showed progress in making evolvable robots move towards a moving target, but it was the robot's only task. When the target was out of sight of the robot, it assumed the latest known angle towards the target to be the current angle, so there was no specific task for searching for the target. In this work we aim to add an additional task to the robot, that enables it to explore its surrounding by rotating in search for the target. Once the target's position is known, the robot will switch back to the task of moving towards the target. To achieve this multi-task behaviour, we need to evolve and learn robots to do two tasks, *rotation* and *follow target*. We want to end up with a morphology that is able to perform both behaviours well, alongside two sets of CPG-weights, or two different *brains*. The two modules can then be used in the robots controller. The controller starts with the *rotation brain* module, and once the target is in sight it will switch to the *follow target brain* module. An overview of the controller that will be used in this work can be seen in Fig. 3.



**Fig. 3. The controller of the robot to end up with.** There will be two sets of CPG-weights, or two brains: one for *rotation* and one for *follow target*. The robot will switch brains dependent on the angle towards the target.

**Fig. 4. The two-phase approach to evolve and learn the robot to do two tasks.** First a robot is evolved for both objectives, then it learns the two tasks separately.

### 3.5 Two-Phase Approach

The goal is to end up with one morphology alongside two different sets of CPG-weights for the two tasks, and we will do that using a two-phase approach [22]. First, the morphology of the robot is evolved to do both tasks at the same time. After that, the morphology of the robot is fixed and we will learn the robot twice. Once for the *rotation* task and once for the *follow target* task. An overview is given in Fig. 4.

**Phase 1: Morphology Evolution.** We want to have a morphology that can perform the two distinct tasks: *rotation* and *follow target*. To achieve this we use a multi-objective algorithm to simultaneously evolve the morphology and brain. Previous work has shown that the NSGA-II algorithm is a good and fast algorithm for multiple objectives [20]. The NSGA-II algorithm is an elitist genetic algorithm, which uses non-dominated sorting for selection. A non-dominated solution, in the context of multi-objective optimization, refers to a solution that cannot be surpassed by any other solution in terms of its performance on any objective, while maintaining equal or better performance on all other objectives. A front of such non-dominated solutions is called a Pareto-front. In the NSGA-II algorithm, there are multiple fronts, where the first front contains all non-dominated solutions, the second front contains all non-dominated solutions without the first front, and so on. A selection is made by choosing the solutions which are on the highest fronts, and ties are decided on crowding distance sorting to ensure diversity. This evolutionary process aims to arrive at a morphology that is good in doing both tasks, however not yet separately. This morphology can then be used to make the robot learn the tasks separately.

**Phase 2: Task Learning.** The multi-objective evolution of the morphology alongside the brain (phase 1) results in a brain that is optimized to perform both objectives simultaneously. However, since our goal is a clear separation between the desired behaviors, the idea of this phase 2 is to learn a brain for each task separately. We achieve this by fixing a chosen morphology after phase 1 and optimizing the weights of two randomly initialized copies of its CPG-brain (Fig. 4). These two sets of CPG-weights are then used by the controller (Fig. 3) to accomplish the execution of multiple separate tasks. Note that we randomly re-initialize the CPG-weights before the learning step, instead of keeping the already evolved weights intact. The reason is that by keeping the evolved CPG-weights, we would need to unlearn the opposing task instead of learning the desired task, and this would likely need an adjustment of the fitness function by penalizing the undesired behavior. Instead, the robot learns the task from scratch by randomly re-initializing the CPG-weights, so there is no unlearning involved and we can use the same fitness function for both phases.

### 3.6 Fitness Functions

There are two tasks for the robots, *rotation* and *follow target*, and for both tasks a separate fitness function is used. The robot should be able to rotate as quickly as possible to get a full overview of the environment, so for the objective value of rotation the forward orientation, which is the direction of the robot's vision, returned from the Revolve Simulator is used. At every time step the forward orientation of the robot is compared to the previous orientations, and this is summed up. This will then result in the total orientation of the robot, and the goal is to maximise this. The function for this can be seen in Eq. 5, where  $T$  are all time points during evaluation, and  $o(t)$  is the forward orientation at time point  $t$ .

$$rotation = \sum_{t \in T} |o(t) - o(t - 1)| \quad (5)$$

For the fitness function of the *follow target* task several factors are taken into account, and a visualisation can be seen in Fig. 5. There,  $T0$  is the start position of the robot,  $T1$  the end position and the red dotted line is the line towards the target. It is important that the robot is moving into the correct direction, so the distance travelled on the ideal trajectory line is taken into account, which is the distance between  $p_0(x_0, y_0)$  and  $p(x_p, y_p)$ . Secondly, the distance between the end point of the robot and the ideal trajectory line is taken into account, which is the distance between  $p_1(x_1, y_1)$  and  $p(x_p, y_p)$  in Fig. 5. It is preferred that the robot moves in a straight line towards the target, so distance travelled is minimized simultaneously. Hence, the travelled path is also taken into account, shown in the figure as two different solid red trajectory lines. Finally, the angle between the optimal direction and travelled direction is also taken into account, as shown by the difference between  $\beta_0$  and  $\beta_1$ . The combination of all this results in Eq. 6, where  $\epsilon$  is an infinitesimal constant,  $e1$  is the distance on ideal trajectory,  $e2$  is the distance between the end point of the robot and the ideal trajectory,  $e3$  is total distance travelled, and  $\delta$  is the angle between the optimal direction and travelled direction. Finally,  $p1$  is a penalty weight set to 0.01.

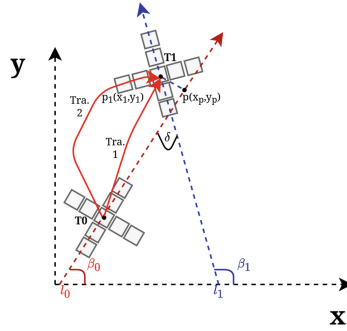
$$followTarget = \frac{e1}{e3 + \epsilon} \cdot \left( \frac{e1}{\delta + 1} - p1 \cdot e2 \right) \quad (6)$$

### 3.7 Experiment Parameters

For phase 1 we use a  $(\mu + \lambda)$  selection mechanism with  $\mu = 100$  and  $\lambda = 50$  to update the population which is initially generated randomly. In each generation 50 offspring are produced by selecting 50 pairs of parents through tournament selection with replacement, creating one child per pair by crossover and mutation according to the MultiNeat implementation<sup>2</sup>. Out of the  $(\mu + \lambda)$  solutions  $\mu$  solutions are selected through NSGA-II selection for the next generation. The evolutionary process is terminated after 300 generations and we do a total of 30 runs.

<sup>2</sup> (see MultiNeat <https://github.com/MultiNEAT/>).





**Fig. 5. Visualisation of calculating the *follow target* fitness value.**  $T_0$  and  $T_1$  are the start and end point of the robot respectively. The red dotted line is the line towards the target, and the red solid lines are two possible trajectory lines of the robot. (Color figure online)

The morphologies for phase 2 are selected among the pareto front of generation 300 and then the specified objective is learned for additional 200 generations. For phase 2 the same set of parameters is used, but this time the morphology of the phenotype is fixed and only the brain weights are evolved for a single objective, again using MultiNeat (Table 1).

**Table 1. Experiment parameters**

Parameters	1, 2*	Description
Population	100, 100	Individuals per generation
Offspring	50, 50	Offspring per generation
Generations	300, 200	Termination condition
Mutation	NEAT, NEAT	Mutation operator
Crossover	NEAT, NEAT	Crossover operator
Parent selection	Tournament, Tournament	Parent selection operator
Survivor selection	NSGA-II, Tournament	Survivor selection operator
Tournament size	2, 2	Number of individuals used in the tournament
Evaluation time	50, 50	Duration of the test period per fitness evaluation in seconds
Runs	30, 30	Repetitions per experiment

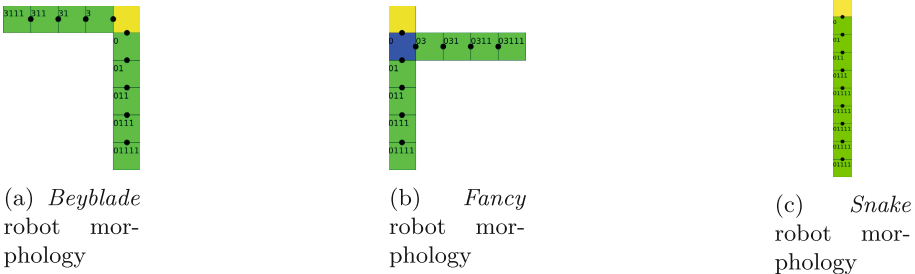
\*Values for Phase 1, Phase 2

## 4 Results

### 4.1 Phase 1: Morphology Evolution

With the multi objective evolutionary process using NSGA-II, many individuals perform well on the objectives at hand. Figure 7 shows that the dominating morphology of generation 300 is of type *snake* (Fig. 6c). However, after further investigation of the behaviour we decide to exclude this phenotype, since it not only rotates horizontally but also vertically, and a vertical rotation is not practical with the hardware we are using.

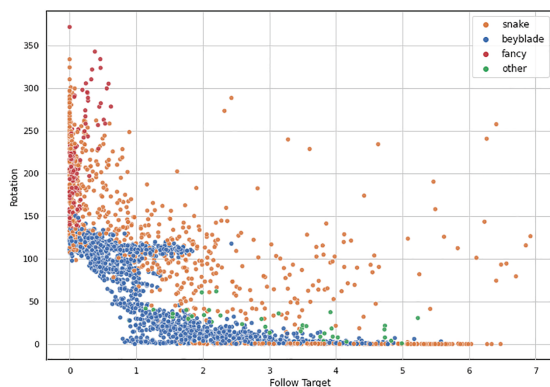
Hence, we decided to proceed by filtering out the snake morphologies. The other two morphologies we call *beyblade* (Fig. 6a) and *fancy* (Fig. 6b). Lastly, there is a category *other* that contains morphologies that are present in only a very small amount. The distribution shows that the *fancy* morphology is mostly good in rotating but not in following the target, whereas the *beyblade* scores well for both objectives, though not as high in *rotation* as *fancy*. The rotation score of the *beyblade* seems good enough for the task, and because following a target is the more difficult task it makes sense to continue to phase 2 with a *beyblade* morphology.



**Fig. 6.** The three most occurring morphologies after 300 generations of evolution.

### 4.2 Phase 2: Task Learning

By following the argument of the foregoing Sect. 4.1, we choose the *beyblade* morphology that is given by the NSGA-II pareto-front of generation 300, for the second phase of learning. As explained in Sect. 3.5, the brain weights are reset to random values, and all morphologies of the *beyblade* robots are the same. Therefore, we can pick any *beyblade* morphology. To this end the morphology stays fixed and the brain weights are re-initialized and then the two objectives are learned separately, resulting in two separate brains. In Table 2 the fitness after 200 generations of learning - using the morphology of generation 300 - is compared to the fitness of the evolution-only approach, which is extended to 500 generations for a fair comparison. It can be observed that for the *rotation*



**Fig. 7. Individuals in Generation 300 of 30 runs.** It is observable that the snake morphology (orange, Fig. 6c) is dominating the population. The morphology is naturally good for the *follow target* objective, because of its straight line shape, and the exploitation of the rotation along its horizontal axis leads to high scores for rotation as well. However, as we state in Sect. 4.1 it comes with undesired behavioral trades. Besides the snake morphology, *fancy* (red, Fig. 6b) is very good in *rotation* although not so good in *follow target* and *other* (green) are potentially better in *follow target* than *rotation*. The *beyblade* (blue, Fig. 6a) morphology seems to lead to a more diverse distribution, getting decent scores for both objectives. *Rotation* is given in radians and *follow target* in meters. (Color figure online)

objective the learning results in a higher fitness, whereas for the learning of the *follow target* objective, the single objective learning performs slightly worse than NSGA-II.

**Table 2. Phase 1 results versus phase 2 results.** The results of the learning only approach with NSGA-II (phase 1) after 500 generations versus the separate learning of each behaviour for 200 generations (phase 2, after 300 generations of NSGA-II) averaged over 30 runs. It can be seen, that for rotation the learning leads to a higher overall fitness function while for the follow line objective, the NSGA-II approach performs better. The rotation performance is given in radians and the follow line performance in meters.

	evolution only	evolution + learning
rotation	66.38	<b>82</b>
follow line	<b>2.02</b>	0.72

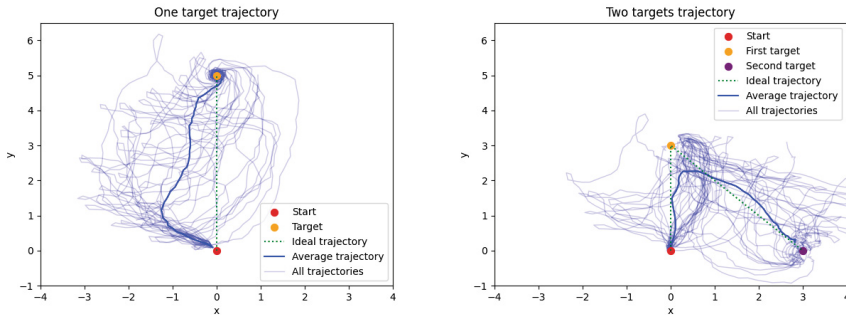
### 4.3 Combining the Tasks

To achieve a more complex *search-and-chase behavior* we now use the controller to switch back and forth between the two learned sets of CPG-weights. We showcase this by using the beyblade morphology of phase 1, together with the

two best-learned brains of phase 2. These two brains are used in the controller of the robot (Fig. 3). Figure 8 shows the trajectory of 30 runs for the robot, as well as the average trajectory. There is one target and its position is at coordinates (0,5). We estimate the performance of the robot by using the Mean Absolute Error (MAE) of its trajectory versus the optimal (shortest) trajectory. The MAE is calculated as  $MAE = \frac{1}{N} \sum_{i=1}^N |Robot_i - Ideal_i|$ , with N being the number of data points that are sampled, and *Robot* and *Ideal* being the coordinates of the robot and the optimal trajectory at time-step i. Compared to the optimal trajectory, the green dotted line, the average trajectory has an MAE of 0.6, with a standard deviation of 1.3. While none of the individual runs is able to transition to the target with an optimal, or close to optimal, trajectory all of them do reach the target. It seems that the robot keeps correcting itself when it is not going in the right direction, and once the target is reached it stays there.

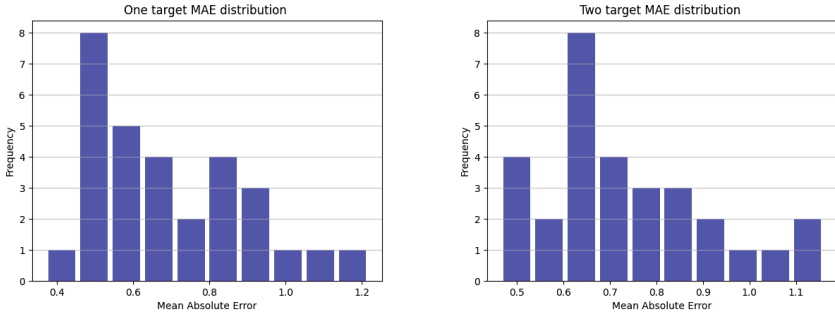
Figure 8b shows the trajectory of 30 runs for two targets, target 1 is at coordinates (0,3) and target 2 is at coordinates (3,0). The robot will first search for the first target and move towards it. Once the first target is reached, it will search for the second target and move towards it. Compared to the ideal trajectory the average trajectory has an MAE of 0.9, and the standard deviation of the trajectories is 1.2. The trajectories again do not seem optimal, but in 27 cases the robot reached both targets. Overshooting the target does not seem to be a big problem and since more distance is traveled between the starting point and the first target, the plain average over the trajectories does not reach the target.

Figure 9 shows the distribution of the MAE of the 30 trajectories for both the one target scenario and the two target scenario. Both distributions look



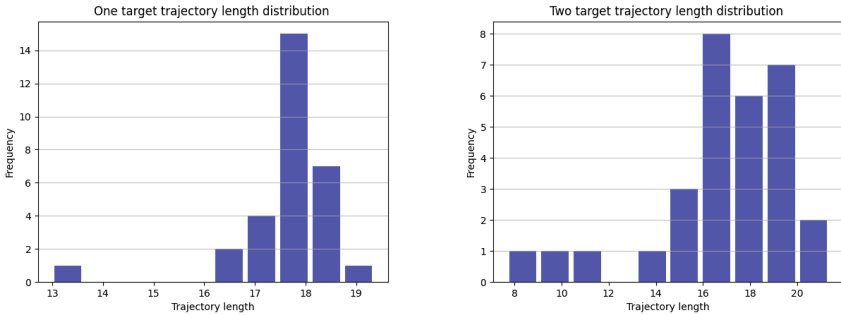
(a) The MAE of the average trajectory is 0.6 with a standard deviation of 1.3. (b) The MAE of the average trajectory is 0.9 with a standard deviation of 1.2.

**Fig. 8. Trajectories in meters of 30 runs of the resulting beyblade morphology with the learned controller.** The trajectories are of a single beyblade robot, with its best-learned *rotation* and *follow target* brains, starting at coordinates (0,0). The dark blue line is the average trajectory of the 30 runs and the other blue lines are the individual runs. The left Figure is in a two target environment and the right Figure is in a one target environment. (Color figure online)



(a) The distribution of the MAE of all 30 trajectories for the one target situation. The average MAE is 0.6  
 (b) The distribution of the MAE of all 30 trajectories for the two target situation. The average MAE is 0.7

**Fig. 9.** Mean Absolute Error distribution of the 30 trajectories



(a) The distribution of the length of all 30 trajectories for the one target situation. Most lengths are between 16 and 19, but there is one short outlier at 13. The ideal trajectory is 5  
 (b) The distribution of the length of all 30 trajectories for the two target situation. Most lengths are between 14 and 20, but there are some below 12. The ideal trajectory is 7.2

**Fig. 10.** Length of trajectory distribution of the 30 trajectories

similar, as most MAE are between 0.5 and 0.7. This is to be expected, as we work with the same robot in both scenarios. Figure 10 shows the distribution of the trajectory length of the 30 trajectories. The lengths are very similar to each other, except for some outliers. As mentioned before, for the two target scenario 27 were able to reach both targets, meaning 3 of them did not. These are the outliers between lengths 8 and 12.

## 5 Discussion

The NSGA-II evolution shows an expected and steady increase in the ability of the phenotypes to score well on the objectives. Another observation is that through this process, the diversity among the morphologies is decreasing over time, and converging to mostly three types of morphologies. A mechanism that ensures a higher diversity amongst the population could be a great improvement in this regard.

In phase 2, the learning of the rotation task for the *beyblade* morphology is clearly outperforming the NSGA-II performance. However, while the learning of the *rotation* objective is even able to outperform the NSGA-II we can not observe that for the learning of the *follow target* objective. We hypothesize that this is because the phenotype has a natural bias for rotation and the rotation behaviour does not drastically harm the robots abilities to follow the target. Moreover, another possibility is that the selection pressure is not high enough and thus the learning algorithm can not keep up with the elitist strategy of NSGA-II. However, our goal here is not only to outperform NSGA-II in first place, but also to *unlearn* the other task. The best morphologies out of the first phase are able to do both tasks well, but only simultaneously. The resulting trajectory plots of Fig. 8 show as well that on average the robots follow the target directly. Lastly, it is to point out that we use the same objective functions for phase 1 and phase 2 in order to have better comparable results. However, when the goal is a robot that is able to perform the desired tasks as accurately and separately as possible, a penalty term in phase 2, that penalizes the task that shall be unlearned has great potential to vastly improve on the resulting behaviour.

### 5.1 Future Work

As of now, only the learning for the *beyblade* morphology is analyzed. In the future it can be interesting to also apply the two phase approach to the *fancy* morphology as well as others. Furthermore, the state-machine controller could be exchanged by a reward-driven system like reinforcement learning. Another interesting idea would be to test the approach with different or more objectives and other types of environments. It is also of interest how the learning after generation 300 would behave without re-initialization of the brain weights, using more of a Lamarckian approach. Lastly, the forward orientation of the robot, which is also the potential view field, is not necessarily its direction of movement. The robot might be moving backwards, sideways or diagonally while facing forward. We would like to investigate how the enforcing of a forward orientation changes the phenotype space and behavioural space of the evolution.

## 6 Conclusion

The first question that we ask is: *Can a multi-objective evolutionary algorithm deliver good bodies for modular robots for handling two tasks?* We demonstrate

that this is possible with the two-phase approach. Initially, we evolve a set of candidate morphologies using the multi-objective NSGA-II algorithm, and then we further separate the desired behaviors with a second phase of learning. To this end, we compare the effects of evolving the phenotypes for the two objectives (*follow target* and *rotation*) with only evolving the brain for the specified objectives separately. We show that using a two-phase approach leads to a better separation of desired behaviors and an increase in overall fitness performance for at least one objective. The second question we ask is: *How much can a secondary learning stage that separately optimizes each task enhance robot performance in a given morphology?* By using the evolved morphology of phase 1 and the two separately learned brains of phase 2, together with a controller that switches between the brains, we demonstrate that the robot is able to navigate through its environment while performing the tasks of searching and chasing one or multiple targets. Videos of the robot can be found on a video playlist<sup>3</sup>.

## References

1. Auerbach, J., et al.: Robogen: robot generation through artificial evolution. In: ALIFE 14: The Fourteenth International Conference on the Synthesis and Simulation of Living Systems, pp. 136–138 (2014)
2. Beer, R.D.: The dynamics of brain–body–environment systems. In: Handbook of Cognitive Science, pp. 99–120. Elsevier (2008)
3. Carlo, M.D., Ferrante, E., Ellers, J., Meynen, G., Eiben, A.E.: The impact of different tasks on evolved robot morphologies. In: Proceedings of the Genetic and Evolutionary Computation Conference Companion. ACM, July 2021
4. Carlo, M.D., Zeeuwe, D., Ferrante, E., Meynen, G., Ellers, J., Eiben, A.: Robotic task affects the resulting morphology and behaviour in evolutionary robotics. In: 2020 IEEE Symposium Series on Computational Intelligence (SSCI). IEEE, December 2020
5. Cheney, N., Bongard, J., Sunspiral, V., Lipson, H.: On the difficulty of co-optimizing morphology and control in evolved virtual creatures. IN: Proceedings of the Artificial Life Conference 2016, July 2016
6. Cheney, N., Bongard, J., SunSpiral, V., Lipson, H.: Scalable co-optimization of morphology and control in embodied machines. *J. Roy. Soc. Interface* **15** (2018)
7. Coello, C.C.: Evolutionary multi-objective optimization: a historical view of the field. *IEEE Comput. Intell. Mag.* **1**(1), 28–36 (2006)
8. Deb, K., Pratap, A., Agarwal, S., Meyarivan, T.: A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Trans. Evol. Comput.* **6**(2), 182–197 (2002)
9. Eiben, A.E., Hart, E.: If it evolves it needs to learn. In: Proceedings of the 2020 Genetic and Evolutionary Computation Conference Companion. ACM, July 2020
10. Eiben, A., et al.: The triangle of life: evolving robots in real-time and real-space. In: Lio, P., Miglino, O., Nicosia, G., Nolfi, S., Pavone, M. (eds.) Proceedings of the 12th European Conference on the Synthesis and Simulation of Living Systems (ECAL 2013), pp. 1056–1063. MIT Press (2013)
11. Floreano, D., Husbands, P., Nolfi, S.: Evolutionary robotics. In: Siciliano, B. and Khatib, O. (ed.) Handbook of Robotics, 1st edn, pp. 1423–1451. Springer, Heidelberg (2008). [https://doi.org/10.1007/978-3-540-30301-5\\_62](https://doi.org/10.1007/978-3-540-30301-5_62)

<sup>3</sup> (see [shorturl.me/JtySjtH](https://shorturl.me/JtySjtH)).

12. Haasdijk, E., Rusu, A.A., Eiben, A.E.: HyperNEAT for locomotion control in modular robots. In: Tempesti, G., Tyrrell, A.M., Miller, J.F. (eds.) ICES 2010. LNCS, vol. 6274, pp. 169–180. Springer, Heidelberg (2010). [https://doi.org/10.1007/978-3-642-15323-5\\_15](https://doi.org/10.1007/978-3-642-15323-5_15)
13. Hupkes, E., Jelisavcic, M., Eiben, A.E.: Revolve: a versatile simulator for online robot evolution. In: Sim, K., Kaufmann, P. (eds.) EvoApplications 2018. LNCS, vol. 10784, pp. 687–702. Springer, Cham (2018). [https://doi.org/10.1007/978-3-319-77538-8\\_46](https://doi.org/10.1007/978-3-319-77538-8_46)
14. Ijspeert, A.J.: Central pattern generators for locomotion control in animals and robots: a review. *Neural Netw.* **21**(4), 642–653 (2008)
15. Lan, G., van Hooft, M., Carlo, M.D., Tomczak, J.M., Eiben, A.: Learning locomotion skills in evolvable robots. *Neurocomputing* **452**, 294–306 (2021)
16. Lan, G., Jelisavcic, M., Roijers, D.M., Haasdijk, E., Eiben, A.E.: Directed locomotion for modular robots with evolvable morphologies. In: Auger, A., Fonseca, C.M., Lourenço, N., Machado, P., Paquete, L., Whitley, D. (eds.) PPSN 2018. LNCS, vol. 11101, pp. 476–487. Springer, Cham (2018). [https://doi.org/10.1007/978-3-319-99253-2\\_38](https://doi.org/10.1007/978-3-319-99253-2_38)
17. Lessin, D., Fussell, D., Miikkulainen, R.: Adopting morphology to multiple tasks in evolved virtual creatures. In: *Artificial Life 14: Proceedings of the Fourteenth International Conference on the Synthesis and Simulation of Living Systems*. The MIT Press, July 2014
18. Lessin, D., Fussell, D., Miikkulainen, R.: Open-ended behavioral complexity for evolved virtual creatures. In: *Proceedings of the 15th annual conference on Genetic and evolutionary computation*. ACM, July 2013
19. Matos, V., Santos, C.P.: Towards goal-directed biped locomotion: combining CPGs and motion primitives. *Robot. Auton. Syst.* **62**(12), 1669–1690 (2014)
20. Moshaiov, A., Abramovich, O.: Is MO-CMA-ES superior to NSGA-II for the evolution of multi-objective neuro-controllers? In: *2014 IEEE Congress on Evolutionary Computation (CEC)*. IEEE, July 2014
21. Nolfi, S., Floreano, D.: *Evolutionary Robotics: The Biology, Intelligence, and Technology of Self-organizing Machines*. MIT Press, Cambridge (2000)
22. Nygaard, T.F., Samuelson, E., Glette, K.: Overcoming initial convergence in multi-objective evolution of robot control and morphology using a two-phase approach. In: Squillero, G., Sim, K. (eds.) *EvoApplications 2017*. LNCS, vol. 10199, pp. 825–836. Springer, Cham (2017). [https://doi.org/10.1007/978-3-319-55849-3\\_53](https://doi.org/10.1007/978-3-319-55849-3_53)
23. Stanley, K.O., D’Ambrosio, D.B., Gauci, J.: A hypercube-based encoding for evolving large-scale neural networks. *Artif. Life* **15**(2), 185–212 (2009)
24. Stanley, K.O., Miikkulainen, R.: Evolving neural networks through augmenting topologies. *Evol. Comput.* **10**(2), 99–127 (2002)
25. Trianni, V., López-Ibáñez, M.: Advantages of task-specific multi-objective optimisation in evolutionary robotics. *PLoS ONE* **10**(8), e0136406 (2015)
26. Weel, B., Crosato, E., Heinerman, J., Haasdijk, E., Eiben, A.E.: A Robotic Ecosystem with Evolvable Minds and Bodies. In: *2014 IEEE International Conference on Evolvable Systems*, pp. 165–172. IEEE Press, Piscataway (2014)