Body symmetry in morphologically evolving modular robots

van de Velde, T.; Rossi, C.; Eiben, A. E.

published in
Applications of Evolutionary Computation
2019

DOI (link to publisher)
10.1007/978-3-030-16692-2_39

document version
Publisher's PDF, also known as Version of record

Body Symmetry in Morphologically Evolving Modular Robots

T. van de Velde[1(✉)], C. Rossi[2], and A. E. Eiben[3]

[1] University of Amsterdam, Amsterdam, The Netherlands
timon11444@hotmail.com
[2] Centre for Automation and Robotics UPM-CSIC, Madrid, Spain
[3] Vrije Universiteit Amsterdam, Amsterdam, The Netherlands

Abstract. Almost all animals natural evolution has produced on Earth have a symmetrical body. In this paper we investigate the evolution of body symmetry in an artificial system where robots evolve. To this end, we define several measures to quantify symmetry in modular robots and see how these relate to fitness that corresponds to a locomotion task. We find that, although there is only a weak correlation between symmetry and fitness over the course of a single evolutionary run, there is a positive correlation between the level of symmetry and maximum fitness when a set of runs is taken into account.

Keywords: Evolutionary robotics · Modular robots · Symmetry

1 Introduction

A particular field within Evolutionary Computing [1,2] is Evolutionary Robotics [3] that is "useful both for investigating the design space of robotic applications and for testing scientific hypotheses of biological mechanisms and processes" [4]. In general, a robot consists of a body with sensors and actuators that constitute the physical makeup of the robot (morphology, hardware), and a brain, which contains the operating logic of the robot (controller, software). The vast majority of Evolutionary Robotics studies is concerned with evolving robot controllers in fixed bodies. The evolution of robot morphologies has only been addressed in a handful of publications [5–9].

A recent development is fueled by the vision of the Evolution of Things [2,10] aiming at evolving robots in the real world, enabled by recent advances in technology (in particular, 3D-printing and rapid prototyping). However, as of today, a system where robots can reproduce and evolve has not been engineered yet. A generic system architecture for evolving robots in real-time and real-space has been established in [11], but, currently, mostly software implementations of this framework [12,13] have been explored. Exceptions are the proof-of-concept installation that demonstrated a physical 'robot baby' parented by two robots, relying on several simplifications and handwork in the reproduction step [14] and the setup proposed in [9], where robots are physically constructed and evaluated.

P. Kaufmann and P. A. Castillo (Eds.): EvoApplications 2019, LNCS 11454, pp. 583–598, 2019.
https://doi.org/10.1007/978-3-030-16692-2_39

In general, there are many parallels between evolutionary algorithms and natural evolution [2]. In nature, most living creatures exhibit forms of symmetry along the longitudinal axis, known as bilateral symmetry. Even though there do not appear to be any physical laws that state that this should be so, theories that explain bilateral symmetry have been proposed (see, e.g., [15]).

In this paper we examine the relationship between symmetry and fitness in a morphologically evolving robot population. The robots are based on the RoboGen system [8]. Their bodies are composed of predefined modules, their brains are artificial neural networks, and their fitness is determined by their ability to locomote. Our main hypothesis is that symmetrical robots perform better, and therefore evolution promotes symmetry.

To this end, we present three measures to quantify the level of symmetry in robots. Furthermore, we distinguish online evolution and offline evolution (see [13]) and run experiments with all six setups using the Revolve simulator [13]. These experiments are to provide an answer to the following research question: is there any relation between symmetry and fitness?

2 Robot Symmetry

To our best knowledge, not much work has been done on symmetry in evolutionary robotics. Certainly, it has been noted that modular robots may end up showing symmetry after having been evolved for locomotion in a straight line [7]. Other research specifically uses symmetry to evolve robot morphologies. For instance, by mirroring limbs along a spine [16] or by employing a mutation operator which can symmetrically replicate branches of robots [17]. Symmetry has also been used as a human evaluation criterion of evolved robots, where the ability of generative encoding to create symmetrical soft robots is seen as a positive quality [18]. Additionally, the importance of symmetry in locomotion has been shown to influence robot controller evolution through HyperNEAT [19]. Despite all this, not much work can be found that specifically measures symmetry in robots. There is, however, a lot of work that has been done in computer vision to detect symmetry [20,21]. Symmetry detection has even been used in robotics for tasks such as real-time object tracking [22]. Zabrodsky *et al.* [23,24] define a robust way to quantify symmetry based on how much an object would have to change to achieve true symmetry. The most relevant work to this study is that of Miras *et al.* [25] that defines several morphological descriptors for RoboGen-like robots in the Revolve system. This set of descriptors contains one that reflects a basic notion of mirror-symmetry. In this paper, we will use this measure as the *base symmetry*.

The bodies of the robots we use here consist of a number of basic components arranged to form larger structures. They are defined by trees consisting of nodes and edges. Because trees are acyclical, so are the robots defined by them. The robots furthermore have some other requirements; they must be planar (i.e. flat), and need to possess at least one motor unit. In addition, the maximum number of parts a robot may have is limited. The nodes of the tree represent components,

their parameters (if applicable) and their orientation, while edges specify which components connect to each other. Table 1 shows an overview of the components used; for more details we refer to [25].

Table 1. Overview of used components.

Component	Connectivity	Parameters	Actuators	Sensors	Colour
Core	4	-	-	6	Yellow
Fixed brick	4	-	-	-	Blue
Parametric bar	2	3	-	-	Brown
Passive hinge	2	-	-	-	Purple
Active hinge	2	-	1	-	Orange

The controllers of the robots are fully connected recurrent neural networks with a single hidden layer. The input nodes are sensor inputs, and the output nodes are signals for the actuators. Hidden nodes and output nodes can have one of three different activation functions; linear, sigmoid, or sinusoidal. The sinusoidal activation function will ignore any input to the node and produce a sinusoid defined by three parameters for period, phase, and gain.

Our definitions of symmetry are based on a two-dimensional representation introduced in [25]. In this representation, each robot part appears as a coloured square, its colour indicating the part type (see Table 1). As each component can have at most four connection slots, we can draw squares adjacent to each other where components connect. Figure 1 shows an example of a robot and its 2D representation.

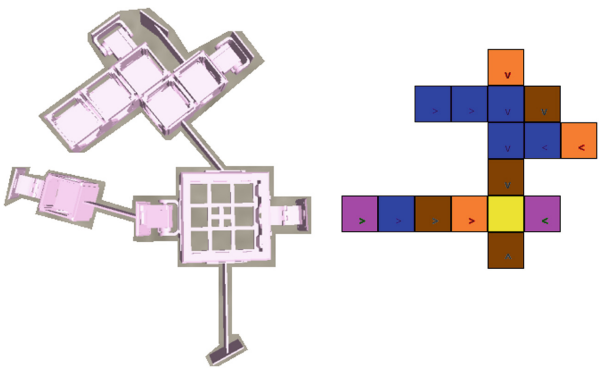

Fig. 1. Two representations of a robot. Left: the robot as rendered by the simulator. Right: its two dimensional approximation used to define symmetry measures.

2.1 Symmetry Measure of Miras *et al.*

Miras *et al.* [25] define a symmetry measure which we used as a base value to which novel symmetry measures are compared. Based on the 2D image of a robot, the core component is selected as the center of the robot. From there, a vertical line is drawn. This line will be the spine of the robot. Components on the spine do not factor into the final symmetry score. The rest of the algorithm is as follows:

```
total_components = 0
for each component:
    if component_a is on spine:
        continue
    total_components = total_components + 1
    coordinates_a = coordinates of component_a
    coordinates_b = coordinates_a mirrored in spine
    if another component exists on coordinates_b:
        matches = matches + 1
symmetry = matches / total_components
```

Because in the robots there is not a preferred longitudinal axis, this algorithm is performed twice. Once for a horizontal spine, and once for a vertical spine. The highest symmetry score of the two determines the final symmetry value for the robot. Note that the final calculation of symmetry is the ratio between matching components and total components. As such, the symmetry value will be 0 if the robot is not symmetrical at all, and 1 if it is completely symmetrical.

Figure 2 shows an example of the symmetry calculation in a robot. Computing its symmetry score, we would first draw a vertical line through its core component, the yellow square. Then, for all components that are not on the vertical axis, we would check if another part exists on the opposite side of the spine. With the vertical spine, that would only be the two blocks just adjacent the core. It's horizontal symmetry score would thus be $2/9 = 0.222$. If we now draw a horizontal line through the core component, we would end up with 8 matching components and a symmetry score of $8/14 = 0.571$. Choosing the larger value, this robot is now assigned a symmetry score of 0.571.

2.2 Novel Symmetry Measures

Several limitations exist to the previously described method. For one, snake-like robots that have most of their parts on their spine will not be accurately assessed, as all components on the spine are ignored. Secondly, it does not take into account if two components that are found on the opposite side of the symmetry line are the same component, or different ones. And finally, symmetry might develop along other lines than just the vertical and horizontal ones that pass through the core component. As such several possible solutions have been implemented.

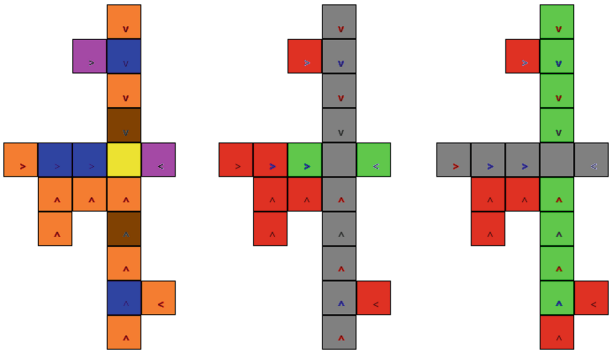

Fig. 2. Example of symmetry calculation. The first image is the robot in question, the middle image is for calculating horizontal symmetry, and the right image is for calculating the vertical symmetry. Grey components are not counted, red components have no symmetrical match, and green components do have a match. This robot has a symmetry value of 0.571 on a scale from 0 to 1. (Color figure online)

Symmetry Lines. No particular reason exists that should cause the core component to be at the center of the robot, other than that it is used as the root of the tree defining the robot. Should the core component be located in one of the robot's extremities, only drawing mirror lines from this point appears to limit the accuracy of the symmetry approximation. An example of this effect can be seen in Fig. 3. As before, the grey components represent the symmetry mirror line and core component in yellow. The third image shows that the robot is more symmetrical along some other mirror line that does not cross the core component. To account for this effect, the novel symmetry measures iterates over every possible symmetry line, to find the line that provides the highest symmetry value.

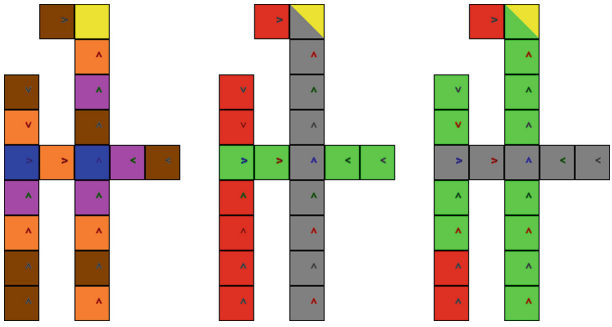

Fig. 3. Three images of the same robot. The first is just the robot. The second is the robot with highest symmetry as found by drawing a mirror line through the core component; symmetry value 0.36. The third image is the highest possible symmetry that can be found with a freely movable mirror line; symmetry value 0.8.

Component Symmetry. Another limitation of the base symmetry measure is that all components are considered equal, whereas in reality, different components fulfill different tasks. A robot arm consisting of solely hinges should perhaps not be symmetrical with a robot arm consisting of static components.

Two ways to take component differences into account have been developed. One will be called *strong* component symmetry, and the other *weak* component symmetry. In strong component symmetry, two components that are locationally symmetrical will only be counted as symmetrical if they are of the same type. For instance, a fixed brick component can only be symmetrical with another fixed brick, but not with a parametric bar.

Another way of measuring component symmetry is by component similarity. There are two types of hinge components; one is passive and the other is active. An argument can be made that they serve a similar purpose; to make movement possible in the robot. On the other hand, the fixed brick and the parametric bar serve only to connect other components, although they do so in very different ways. Nevertheless, it is possible to create a weak component symmetry by using these similarities. On locational matches, Table 2 is used to determine the symmetry value of the match. An example of this calculation can be seen in Fig. 4.

Table 2. Look-up table of similarity scores used for weak component symmetry.

	Parametric bar	Brick	Active hinge	Passive hinge
Parametric bar	1	0.5	0	0
Brick	0.5	1	0	0
Active hinge	0	0	1	0.5
Passive hinge	0	0	0.5	1

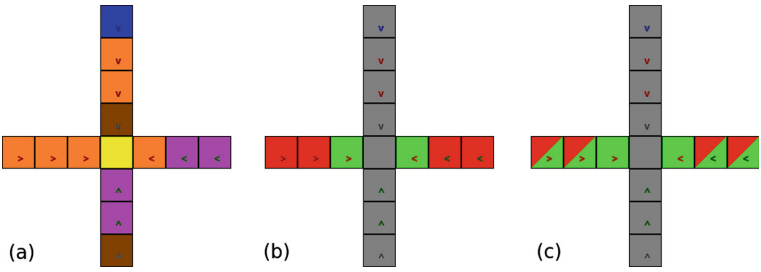

Fig. 4. (a) Image of the robot. (b) Image of symmetry calculation with strong component symmetry. Matching components are green, non-matching components are red. Symmetry value 0.33. (c) Image of symmetry calculation with weak component symmetry. Matching components are green, similar components are red and green. Symmetry value 0.66. (Color figure online)

Spinal Symmetry. Up until now, we have ignored the components that are located directly on the mirror line. In many cases this line could be considered the spine of the robot. There is one relatively common case in which ignoring the symmetry of the spine itself is not desirable. In particular in snake-like robots, where most of the components are aligned along a single axis, and represent a large percentage of the robot's total number of components. Intuitively, we would say that the robot of Fig. 5 is reasonably symmetrical, but ignoring the spine, we end up with a symmetry value of 0.

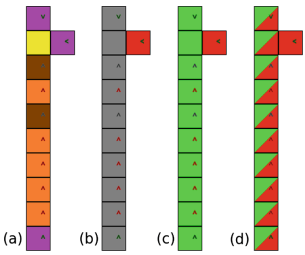

Fig. 5. (a) Snake-like robot. (b) Symmetry calculation without counting the spine. There is only one red component, thus a symmetry of 0. (c) Symmetry calculation while also counting the spine results in a score of 0.91. (d) Symmetry calculation with discounted spine. The red and green components all count as half components for a symmetry score of 0.83. (Color figure online)

To correct this, we want to make sure the spine also influences the symmetry. However, assigning full symmetry points would skew the results as well. Other components in the robot have to satisfy both locational symmetry and component symmetry. The parts on the mirror line would always be symmetrical with themselves. To further adjust this bias, components on the spine will only count as half a component. Keeping mind that our formula for symmetry was:

```
symmetry = matches / total_components
```

In Fig. 5, doing full spinal symmetry would then result in 10 parts being symmetrical, with a final symmetry score of $10/11 = 0.91$. Discounted spinal symmetry would mean that those 10 symmetrical components now only count as half components; both as `matches` and `total_components`. The resulting symmetry score then follows from $5/6 = 0.83$. Reducing the value of `total_components` is necessary to assure symmetry can still reach a value of 1 if the robot is completely symmetrical. Also note that the spine is nothing more than the location of the mirror line. This line is still shifted throughout the robot to find the highest fitness value.

Final Measures. To summarise, we now have two symmetry measure variations which we can compare to our base symmetry. The two novel measures implement all techniques described above, but differ in *weak* and *strong* component symmetry. Consequently they shall be referred to as such.

3 Experimental Setup

There are two distinct evolutionary approaches available in Revolve; offline and online. The following subsection describes the evolutionary process as specified in Hupkes *et al.* [13]. In offline evolution, individuals are evaluated separately and only once, whereas with online evolution, individuals will be continuously evaluated until they die by some criteria. In either case, the fitness function is based on how well the robots are able to move. The formula is as follows:

$$f(\rho) = v + 5s. \tag{1}$$

Here, $f(\rho)$ is the fitness value of robot ρ. The velocity of the robot, or the total distance travelled since the beginning of the evaluation is denoted as v. Thus, if a robot moves back and forth repeatedly, the value v increases. The variable s is the speed of the robot, and is obtained by measuring the distance between the robot's location at the start of evaluation and the robot's location at the end of evaluation.

This fitness value is then used for selecting parents through tournament selection. This means that out of the entire population, several random individuals are selected, and out of those, the individual with the highest fitness is selected to be a parent. This process is repeated to select the other parent, with the only requirement that both are distinct robots. Through crossover, a new robot is generated and born into the simulator. The simulation ends after a set amount of births. The difference between offline and online solution has some consequences to population management.

During offline evolution, each individual is inserted into the simulator separately. They are then evaluated according to the fitness function for some period of time. Afterwards, their fitness is recorded, the robot removed and a new one added. After an entire generation has been evaluated, an equal amount of offspring is produced. Out of these two generations, the fittest individuals are kept, and the rest removed. This way, the size of the living generation remains constant. Specific settings are listed in Table 3.

Table 3. Settings for offline evolution.

Offline evolution	
Generation size	15 individuals
Evaluation time	12 s
Parent selection	4-tournament
Survivor selection	Best 15
Stopping criteria	3000 births

With online evolution, the entire initial population is inserted into the world at once. They are randomly placed within a circular area called the *birth clinic*. The individuals are then continually evaluated within a sliding time-window. They will have matured after having been evaluated through an entire time-window, but the fitness value will keep updating throughout the robot's lifespan. The simulator is set up so that new robots are born at regular intervals. New robots are simply inserted into the world, along with the already existing population. Robots can only be selected as a parent if they are mature.

The simulator will, at certain fixed time intervals, remove robots that do not have a fitness value over a certain threshold. To ensure the success of the simulation, robots will not be killed if that would mean that the population size drops below a set minimum. On the other end of the spectrum, if a maximum number of living individuals is exceeded, a percentage of the population with the lowest fitness will be culled. Specific settings are listed in Table 4.

Table 4. Settings for online evolution.

Online evolution	
Initial population	15 individuals
Evaluation window	12 s
Parent selection	4-tournament
Birth interval	15 s
Kill interval	30 s
Survivor selection	>70% average fitness
Minimum population size	8 individuals
Maximum population size	30 individuals
Population culling	Weakest 70%
Stopping criteria	3000 births

The process of robot generation works through several sequential steps. First of all, the total number of components is decided by randomly drawing from a normal distribution defined by $\mu_{\text{initial parts}}$ and $\sigma_{\text{initial parts}}$. This number is required to be between R_{minimum} and R_{maximum}. Starting from the core component, a random attachment slot is chosen to attach another part to. This part, in turn, is also randomly selected, as is its attachment slot that will be attached to the robot. If the selected part has parameters to be set, as would be the case with the parametric bar joint, these are also initialised at random. This process continues until the total number of components has been reached, or until there are no further attachment slots available. As previously stated, the neural network of the robot will have its input and output neurons determined by the presence of sensors and actuators in the robot's body. The number of hidden neurons will be a randomly chosen value between 0 and a variable h_{max}. For every hidden and output node, a random activation with randomised parameters is chosen.

Finally, all weights in the fully connected recurrent neural network are initialised to 0. They are then with a random chance $p_{\text{connect neurons}}$ set to a value between $[0, 1]$. Parameters for robot generation are listed in Table 5.

Full offspring generation consists of a specific sequence. Every step happens only with a probability. For instance, it is possible to set different possibilities for subtree removal and subtree duplication. If the result of a step produces a robot that violates set restrictions as described in Sect. 2 (that is, planarity, number of motor units, total number of parts), crossover is aborted and no offspring is created. The following crossover steps exist, each step continuing with the result of the previous step:

- **One point crossover**: A node is selected on parent a, and replaced with a random node from parent b.
- **Subtree removal**: A random subtree is selected and removed.
- **Subtree duplication**: A random subtree is duplicated.
- **Subtree swap**: Two random subtrees are swapped.
- **Hidden neuron and neural connection removal**: Each hidden neuron or neural connection has a probability to be removed.
- **Parameter mutation**: Each parameter, for either body parts or neurons is updated with a randomly drawn value. They are changed according to $(1 - \epsilon)\pi + \epsilon\pi*$, where π is the current parameter value, $\pi*$ a new randomly drawn value, and ϵ the mutation rate.
- **Hidden neuron and neural connection addition**: With a probability, new hidden neurons and neural connections are added.
- **Part addition**: A new part is added to the robot body.

Table 5. Parameters and restriction values for robot generation.

Parameter	Value
$\mu_{\text{initial parts}}$	12
$\sigma_{\text{initial parts}}$	5
R_{minimum}	3
R_{maximum}	30
h_{max}	10
$p_{\text{connect neurons}}$	0.1

Table 6. All probabilities and parameters used during crossover.

Parameter	Value
$p_{\text{delete subtree}}$	0.05
$p_{\text{duplicate subtree}}$	0.1
$p_{\text{swap subtree}}$	0.05
$p_{\text{remove brain connection}}$	0.05
$p_{\text{delete hidden neuron}}$	0.05
$\epsilon_{\text{body mutation}}$	0.05
$\epsilon_{\text{brain mutation}}$	0.1

Table 6 lists the probabilities of the crossover actions. The probabilities of neuron addition neural connection addition and part addition are tuned in such a way that the number of parts, number of neurons, and number of neuron connections stay the same on average and do not, for instance, tend to zero [13].

The setup of this experiment is to determine whether there is any relationship between a robot's fitness and its symmetry. By running many simulations and collecting data on the born individuals' fitnesses and morphologies (out of which we extract their symmetry values), we can compute linear correlations between the two. Data was collected both by running online evolution and offline evolution.

Combining these data, we have several possible measurements we can do. For each separate simulation run, we can try to see if the symmetry of robots increases as their fitness increases. This can be done simply by verifying whether a linear correlation between the two exists. The same linear correlation can be calculated not only on separate runs, but also on the combined datasets of all runs.

The population size used during experiments is quite small; around 15 individuals for online evolution, and a fixed 15 individuals for offline evolution. As such, there exists a possibility that the global morphology of the entire population during a simulation run converges, meaning that all individuals end up with a similar morphology. In this case it is interesting to make comparisons between the runs. One way to do this, is to create another dataset by taking the average symmetry and highest found fitness of each run, and computing the linear correlation between those two values.

Finally, all of these experiments have 3 different measures for symmetry available. There are 2 novel symmetry measures and the base symmetry. For further use of symmetry measures, it would be useful to confirm whether there are any differences. One measure might correlate better with fitness than others, and would therefore be more likely to be useful when symmetry is used to modify the fitness function during the simulation.

4 Experimental Results

4.1 Fitness and Symmetry Correlations

Thirty runs were performed in both offline and online evolution. By computing fitness and the three symmetry measures for each individual we can examine the correlation between them. Table 7 shows the averaged correlations of all the separate runs. From this we can see that the correlation between symmetry and fitness is low. Most of the values are around 0, indicating a random relationship between fitness and symmetry. It appears that symmetry does not generally increase as a run progresses, even though fitness does.

Table 8 shows some slightly more positive numbers. These values were computed by first concatenating the fitness and symmetry values of each run and then finding the correlations, rather than correlating each run separately and then averaging. The difference between Tables 7 and 8 can be explained by the fact that different runs achieve different maximum fitness values. By averaging correlations of the separate runs, the difference in fitness between runs is removed. By not doing so, runs with higher fitness values essentially have a higher impact while calculating correlations.

Finally, the difference between online and offline evolution can be explained by the way new robots are added to the simulation. Because in offline evolution only fitter individuals are added to the database, there is more statistical weight for individuals with higher fitness. Or, in other words, the average fitness of the recorded individuals in offline evolution is higher than that of online evolution.

Table 7. Correlation values averaged over all separate runs.

Symmetry	Correlations	
	Offline	Online
Base	−0.184	−0.162
Weak	−0.059	−0.052
Strong	−0.037	−0.040

Table 8. Correlation values as calculated over the entire dataset.

Symmetry	Correlations	
	Offline	Online
Base	0.035	−0.071
Weak	0.152	0.018
Strong	0.191	0.030

4.2 Average Symmetry and Maximum Fitness Correlations

In the previous analyses, we have not found a strong correlation between fitness and symmetry. However, visual inspection of the generated robots suggests that a general morphology is settled on during each run. It is therefore possible that some morphologies have a higher potential for fitness. We can verify this by correlating the maximum obtained fitness values and average symmetries of each run. The resulting graphs can be seen in Fig. 6. The corresponding correlation numbers can be found in Table 9.

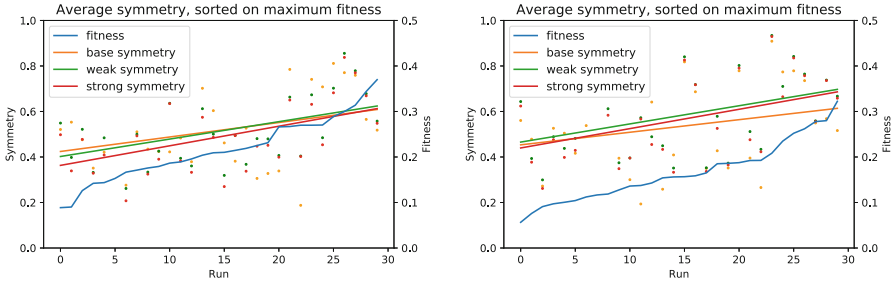

Fig. 6. Relationship between the maximum fitness and the average symmetry in 30 independent runs ordered by the maximum fitness at termination. Symmetry values are represented by scatter points and a linear fit. Left: offline evolution. Right: online evolution.

Table 9. Correlation values as calculated between the maximum fitness values of runs and their average symmetries.

Symmetry	Correlations			
	Offline		Online	
	r	p-value	r	p-value
Base	0.300	0.110	0.235	0.210
Weak	0.447	0.013	0.380	0.038
Strong	0.493	0.006	0.393	0.032

We can now see stronger correlations between the average symmetry of a run and its maximum obtained fitness. Strong symmetry correlates a little better than weak symmetry, and both of them better than base symmetry. It should be noted that strong symmetry is always more conservative than weak symmetry. That is, for any robot the strong symmetry score is generally lower than the weak symmetry score. This difference is greatest when symmetry is low. At higher symmetry values, the measures get closer together.

As in Table 8, there is some difference between online and offline evolution, with lower values for online evolution. Aside from the bookkeeping differences between the two approaches as mentioned before, we can now also see that the maximum achieved fitness of each run in online evolution is lower than in offline evolution. A difference likely caused by the different population management approaches, as noted in [13].

5 Discussion

Throughout all the offline simulations, population size was maintained at the default setting of 15 individuals. This is not a particularly big number and may have had some effects on the results, such as the body morphology quickly converging. Effective population sizes were even smaller in online evolution. As opposed to offline evolution, where every generation spawns a new generation and the 15 best individuals of both generations are kept, online evolution just periodically kills weaker individuals. The practical consequence of this is that oftentimes there will only be a mature population of close to 8 individuals. Premature convergence could also be averted by reducing evolutionary pressure.

The different ways of population management and record keeping also have a profound effect on the average fitness scores. In offline evolution, only the best 15 robots out of 15 children + 15 parents have their fitness recorded in the database every generation. Whereas in online evolution, all new individuals are kept, even if their fitness scores are a lot lower than those of the previous generation. This is why average fitness scores are a lot lower in online evolution than in offline evolution. The effective population size and population management may also explain why the maximum fitness values of online evolution simulations are lower than those of offline evolution.

5.1 Symmetry Limitations

The first limiting factor for the symmetry measures is the conversion of the robots' tree representation to the 2D graph representation. As mentioned in Sect. 2, not all components have the same size, some are positioned at an angle, and some are not static. All of these factors are ignored in the 2D representation, leading to some fundamental differences between the two representations. It is difficult to estimate how much this affects the results without doing an overhaul on this conversion.

There are also parts of the symmetry evaluation that are based on personal design choices. For instance, symmetry detection works by shifting the mirror lines to find the highest symmetry value available in the robot. This means, that the core component, which contains all the sensors, may not be located at the center of the robot. If symmetry is to be evaluated and used during runtime as part of a fitness function, it may be more beneficial to actually keep the core component centered. One case in which this may be true is if visual sensors are situated in the core component.

Another design choice has been to only count components that are located on the mirror line as half components. This was done to value the robots' extremities more than their spine, and to compensate for the fact that the specific components always match with themselves. Whether this is has been the right choice may need to be examined further.

Finally, mirror line placement is currently constrained to only be along the centers of the 2D components. Allowing the lines to be placed on the borders between components would allow for certain robot configurations to be rated as more symmetrical than they would be now. For instance a robot consisting of two components might be more symmetrical if the mirror line was to be placed in between its components, rather than through their center.

6 Conclusions and Further Work

Over the course of single runs, there do not seem to be any strong correlations between symmetry and fitness. The average correlation values tend to zero, indicating a random relationship between the two. However, when we take into account that each run quickly settles on specific morphologies that only change slowly over time, we can find more interesting figures. Correlating the maximum attained fitness of a run with the entire population's average symmetry values, we observed a positive correlation with fitness. There was a positive correlation up to 0.49 for offline evolution, and 0.39 for online evolution. Out of the three measures strong symmetry seems to correlate best with fitness, though the difference with weak symmetry is minor.

One way to examine the cause of the disparity between single run correlations and maximum fitness correlations would be to seed the simulator with handcrafted symmetric robots. If these runs attain high fitness values, it may indicate that either the genome is unsuited for developing symmetry by itself or that the simulator parameters may need to be changed.

Another limitation of our current setup is that newborn robots do not learn after birth; their neural networks only change randomly during crossover. We chose for this setup to see an isolated, clean effect of evolution. However, in general, any neural connection that worked well in a parent robot, may not work after crossover has changed large portions of the body layout. This will cause the robot to perform poorly, even if the given body may be better suited for locomotion than the bodies of the parents. Future investigations shall therefore include lifetime learning capabilities to the robots at least during their infancy, e.g., attending a 'Robot School' [26]. In this way, the problem of having good but poorly controlled morphologies should be mitigated. In addition, this might also allow body morphology to change more throughout the simulation runs.

Finally, a promising line of investigation is to experiment with exploiting the symmetry and fitness relationship to produce better robots faster. This could be done through a multi-objective approach based on an objective that measures the locomotion abilities and one that measuring symmetry.

References

1. Eiben, A., Smith, J.: Introduction to Evolutionary Computing, 2nd edn. Springer, Heidelberg (2015). https://doi.org/10.1007/978-3-662-05094-1
2. Eiben, A., Smith, J.: From evolutionary computation to the evolution of things. Nature **521**(7553), 476–482 (2015)
3. Bongard, J.C.: Evolutionary robotics. Commun. ACM **56**(8), 74–83 (2013). http://doi.acm.org/10.1145/2493883
4. Floreano, D., Husbands, P., Nolfi, S.: Evolutionary robotics. In: Siciliano, B., Khatib, O. (eds.) Springer Handbook of Robotics, vol. Part G. 61, pp. 1423–1451. Springer, Heidelberg (2008). https://doi.org/10.1007/978-3-540-30301-5_62
5. Sims, K.: Evolving 3D morphology and behavior by competition. Artif. Life **1**(4), 353–372 (1994). https://doi.org/10.1162/artl.1994.1.353
6. Sims, K.: Evolving virtual creatures. In: Proceedings of the 21st Annual Conference on Computer Graphics and Interactive Techniques SIGGRAPH 1994, pp. 15–22. ACM, New York (1994). https://doi.org/10.1145/192161.192167
7. Lipson, H., Pollack, J.B.: Automatic design and manufacture of robotic lifeforms. Nature **406**, 974–978 (2000)
8. Auerbach, J., et al.: Robogen: robot generation through artificial evolution. In: Artificial Life 14: Proceedings of the Fourteenth International Conference on the Synthesis and Simulation of Living Systems, pp. 136–137 (2014)
9. Brodbeck, L., Hauser, S., Iida, F.: Morphological evolution of physical robots through model-free phenotype development. PLoS One **10**(6), e0128444 (2015)
10. Eiben, A.E., Kernbach, S., Haasdijk, E.: Embodied artificial evolution - artificial evolutionary systems in the 21st century. Evol. Intell. **5**(4), 261–272 (2012). http://www.few.vu.nl/~ehaasdi/papers/EAE-manifesto.pdf
11. Eiben, A.E., et al.: The triangle of life: Evolving robots in real-time and real-space (2013)
12. Eiben, A.E.: EvoSphere: the world of robot evolution. In: Dediu, A.-H., Magdalena, L., Martín-Vide, C. (eds.) TPNC 2015. LNCS, vol. 9477, pp. 3–19. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-26841-5_1

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
14. Jelisavcic, M., et al.: Real-world evolution of robot morphologies: a proof of concept. In: Artificial Life (2017)
15. Werner, E.: The origin, evolution and development of bilateral symmetry in multicellular organisms. Quantitative Biology ArXiv:1207.3289 (2012)
16. Marbach, D., Ijspeert, A.J.: Online optimization of modular robot locomotion. In: IEEE International Conference Mechatronics and Automation 2005, vol. 1, pp. 248–253, July 2005
17. Faiña, A., Bellas, F., López Peña, F., Duro, R.: Edhmor: evolutionary designer of heterogeneous modular robots. Eng. Appl. Artif. Intell. **26**, 2408–2423 (2013)
18. Cheney, N., MacCurdy, R., Clune, J., Lipson, H.: Unshackling evolution: Evolving soft robots with multiple materials and a powerful generative encoding. In: Proceedings of the 15th Annual Conference on Genetic and Evolutionary Computation GECCO 2013, pp. 167–174. ACM, New York (2013). https://doi.org/10.1145/2463372.2463404
19. Clune, J., Beckmann, B.E., Ofria, C., Pennock, R.T.: Evolving coordinated quadruped gaits with the hyperneat generative encoding. In: 2009 IEEE Congress on Evolutionary Computation, pp. 2764–2771, May 2009
20. Sun, C., Sherrah, J.: 3D symmetry detection using the extended gaussian image. IEEE Trans. Pattern Anal. Mach. Intell. **19**(2), 164–168 (1997)
21. Mitra, N.J., Guibas, L.J., Pauly, M.: Partial and approximate symmetry detection for 3D geometry. ACM Trans. Graph. **25**(3), 560–568, July 2006. https://doi.org/10.1145/1141911.1141924
22. Li, W.H., Zhang, A.M., Kleeman, L.: Bilateral symmetry detection for real-time robotics applications. Int. J. Robot. Res. **27**(7), 785–814 (2008). https://doi.org/10.1177/0278364908092131
23. Zabrodsky, H., Peleg, S., Avnir, D.: A measure of symmetry based on shape similarity. In: Proceedings 1992 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, pp. 703–706, June 1992
24. Zabrodsky, H., Peleg, S., Avnir, D.: Symmetry as a continuous feature. IEEE Trans. Pattern Anal. Mach. Intell. **17**(12), 1154–1166 (1995)
25. Miras, K., Haasdijk, E., Glette, K., Eiben, A.E.: Search space analysis of evolvable robot morphologies. In: Sim, K., Kaufmann, P. (eds.) EvoApplications 2018. LNCS, vol. 10784, pp. 703–718. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-77538-8_47
26. Rossi, C., Eiben, A.: Simultaneous versus incremental learning of multiple skills by modular robots. Evol. Intell. **7**(2), 119–131 (2014)