**1. Methods**

To provide some independent support of the proposed hypothesis that a genotype to phenotype mapping that includes epigenetic operators may be more evolvable than an equivalent mapping without such operators, we evolved simulated robots to solve the same task as the physical robot. Fig. 1 shows the behavior of the simulated robot when equipped with the best controller evolved using EO.

|  |  |
| --- | --- |
| a | b |
| c | d |
| **Fig 1:** Behavior produced by an evolved controller. Each controller was evaluated on the robot four times at four different starting locations. Dark gray objects are obstacles; the light gray object is the light source. | |

* 1. **The robot**

The robot behaved in a square, walled enclosure. The enclosure contained three obstacles between itself and the light source, and the light source itself. The robot was equipped with four sensors: two light sensors (placed at front left and front right) and two proximity sensors (placed at front left and front right). Light sensors were set to *1/d2*, where *d* is the distance from the sensor to the light source. Occlusion by the obstacles was not simulated here. Proximity sensors were set to *d*, where *d* represents the length of a ray emitted by the sensor, facing the direction the robot is facing, and with length equal to that determined by the first collision of the ray with an obstacle or the inside boundary of the arena. The robot had two wheels that it could rotate forward or backward at different speeds, and a third caster wheel.

* 1. **The controller**

The robot was controlled by an artificial neural network containing four sensor neurons, four hidden neurons, and two motor neurons. Synapses could connect the sensor neurons to the motor neurons; the sensor neurons to the hidden neurons; the hidden neurons to each other; and the hidden neurons to the motor neurons. At each time step, the sensor neurons were set to the raw values of the sensors (without normalization or thresholding). The hidden- and motor neurons were updated according to

*yi(t) =* tanh*( yi(t-1) + τi(****Σ****j wjiyj(t-1) ) )*

where *yi(t)* denotes the value of the *i*th neuron at the *t-*th time step, *τi* denotes the time constant controlling the rate of change of the *i*th neuron (here all *τi*=0.3 following previous work), and *wji* denotes the weight of the synapse connecting neuron *j* to neuron *i*.

* 1. **The genotype to phenotype mapping**

Genomes were encoded as bit strings of length 560. Genomes dictated where synapses should be added to the neural network controller. This was accomplished by transforming the bit string into automatons that `walked’ across a set of simulated pins. As it does, it may drop several threads – each of which is in turn made up of one or more wires -- on to the pins. When finished, the threads are converted into synapses.

Each bit string is divided into seven substrings of 80 bits. These seven substrings are transformed into seven automata in the following manner. Each set of 80 bits are further divided into 20, four-bit groupings. Each grouping is converted into an integer in [0,15]. If the integer is above nine it is discarded. The remaining 20 (or fewer integers) guide the automaton as follows. The first two numbers indicate the starting position of the automaton. The automaton moves across a 10x20 set of pins: the 10 columns correspond to the 10 neurons; the 20 rows denote that there are 20 pins available for each neuron.

The next two numbers indicate the direction and distance it will move. This gives a new position. If the new position is valid, the automaton moves there, laying a wire from the starting position to the ending position. The fifth number indicates the strength of this wire: this number is converted from [0,9] to [-0.5,-0.4,-0.3,-0.2,-0.1,0.1,0.2,0.3,0.4,0.5].

The next two numbers indicate the new direction and distance the automaton wishes to move, and so on. The automaton terminates when there are no more numbers in the string to guide it. The next automaton is created and moved across the pins. This continues until all seven automata have been executed.

The automaton can fail in the following cases: (1) its target location is outside the boundary of the pins; (2) the target pin is already occupied; or (3) the target location equals the starting position.

In both the GO and EO evolutionary trials, the automaton terminates if it experiences condition (1). In the GO trials, it also terminates when it experiences conditions (2) and (3). In the EO trials, the automata will attempt to find an empty pin on that row. If it can, it attaches there. If it cannot, it terminates.

After all seven automata terminate, the wires are converted into synapses as follows. The weight of a synapse connecting neuron *j* to neuron *i* is set to

*wji =* **Σ***k pik*

if there are one or more wires traveling from pin row *j* to pin row *i*, and *wji=0* otherwise. *pik* denotes the weight of a wire that originates on pin row *j* and pin column *k*, and *pik* =0 if there is no wire emanating from pin row *j* and pin column *k*. Only valid pin row pairs are considered, where a valid pin row pair is one that connects a sensor pin row to a hidden pin row, a sensor pin row to a motor pin row, a hidden pin row to another hidden pin row (including its own pin row), or a hidden pin row to a motor pin row.

* 1. **The evolutionary algorithm**

Given the greater computational budget available for evolving these simulated robots, we employed the AFPO algorithm [[[1]](#footnote-1)] to evolve the controllers. Each evolutionary trial began with a population of 50 random bitstrings. Each was converted into a controller and embedded in the simulated robot. The robot was then evaluated four times as shown in Fig. 1. Each evaluation lasted 300 time steps. After evaluation, fitness was calculated as

*f =* **Σ***e=1..4* **Σ***t=1..300 (LPet + RPet)*

where *LPet* and*RPet* denote the values of the left- and right photo sensors in the *e*-th environment at the *t*-th time step, respectively.

After all 50 controllers were evaluated, the dominated individuals were deleted, using fitness and age as the two objectives (fitness is maximized while age is minimized). The population was filled back up to 49 individuals by randomly choosing a non-dominated individual, copying it, mutating it, and placing it in the population. The 50th slot was filled with a random bitstring and assigned an age of zero. The next generation was then conducted, and continued until 50 generations had elapsed.

**2. Results**

adsfas

|  |
| --- |
|  |
| **Fig. 2.** Relative performance of the GO (red lines) and EO (blue lines) genotype to phenotype mappings. Twelve evolutionary trials of GO and 13 evolutionary trials of EO were performed, each lasting for 500 generations. Thick lines report the mean fitness of the best individual in the population, averaged across the trials. Thin lines report ± one unit of standard error of the mean. By the end of the trials, the EO treatment produced statistically significantly more fit individuals than the GO treatment. (*p*<0.001 according to a Student’s *t*-test, assuming unequal variances.) |

**3. Discussion**

adsfas

1. Schmidt, Michael, and Hod Lipson. "Age-fitness pareto optimization." *Genetic Programming Theory and Practice VIII*. Springer New York, 2011. 129-146. [↑](#footnote-ref-1)