

When Novelty is Not Enough

Giuseppe Cuccu and Faustino Gomez

IDSIA

Galleria 2, 6928 Manno-Lugano, CH

Email: {giuse,tino}@idsia.ch,

WWW: [http://www.idsia.ch/{~giuse,~tino}](http://www.idsia.ch/~giuse,~tino)

Abstract. The idea of evolving novel rather than fit solutions has recently been offered as a way to automatically discover the kind of complex solutions that exhibit truly intelligent behavior. So far, novelty search has only been studied in the context of problems where the number of possible “different” solutions has been limited. In this paper, we show, using a task with a much larger solution space, that selecting for novelty alone does not offer an advantage over fitness-based selection. In addition, we examine how the idea of novelty search can be used to sustain diversity and improve the performance of standard, fitness-based search.

1 Introduction

Almost all search algorithms inspired by natural evolution have been designed to optimize a user-defined objective function. Typically, a population of candidate solutions is evaluated on this function, and the resulting fitness values are used either directly or in modified form (e.g. fitness sharing [3]) to decide which solutions should be used to generate new, hopefully better solutions. This makes obvious sense as we usually have a particular goal in mind.

Recently, the idea of *goal-independent* search [6–9,11] has been gaining attention, where selection is focused on finding novel solutions rather than fit ones. Solutions are selected because of their novelty alone without measuring their fitness. The claim is that the continual discovery of novelty will lead to ever-increasing complexity, which will inevitably generate interesting and even useful behavior. So far, however, this idea of *novelty search* has only been applied to domains where novelty is defined over a relatively small solution space.

In this paper, we show that novelty search alone does not scale to large search spaces, but, when combined with fitness-based selection, it can be a useful diversity sustaining mechanism.

The next section describes in more detail the concept of novelty, and recent work in this area. In section 3, we present experiments on a deceptive version of the Tartarus (block packer) problem using a simple crowding based algorithm. The last section discusses the overall results, and suggests directions for further research.

2 Novelty Search

In [7], Lehman and Stanley present the idea of *Novelty Search* where individuals in an evolving population are selected based solely on how different they are compared to all of the other solutions evaluated so far. Each individual, x , in the population is assigned a novelty score that is computed as the average *novelty distance* from its k -nearest neighbors in both the population and an archive:

$$\text{nov}(x_i) = \frac{1}{k} \sum_{j=0}^k \text{dist}(\phi(x_i), \phi(x_j)), \quad (1)$$

where x_j is the j -th nearest neighbor with respect to novelty distance $\text{dist}(\cdot, \cdot)$ which compares features, $\phi(\cdot)$. The particular choice of features and distance measure are user-defined and problem specific. For example, novelty could be defined as simply as the Euclidean distance ($\text{dist}(x, y) = \|x - y\|$) between genotypes ($\phi(x) = x$), or more complex where $\phi(x) = \beta_x$ is the *behavior* of individual x in a sequential decision task:

$$\beta_x = (o_t, a_t, o_{t-1}, a_{t-1}, \dots, o_0, a_0), \quad (2)$$

where o_t and a_t are the observation of the environment and action taken at time t , respectively, and dist could be any similarity measure over sequences (e.g. edit distance, hamming distance).

Initially, the archive is empty. When a new individual is created, it replaces an individual the population according to some rule, e.g. replacing the least novel individual, and, if its novelty (as computed by equation 1) exceeds the *novelty threshold*, then it is also inserted into the archive.

Notice there is no notion of “fitness” in the normal sense—the probability of being selected is not determined by a fixed objective function, but instead depends entirely on the current state (population) and history (archive) of a given evolutionary run.

The archive provides a memory of previously seen novel behaviors that may no longer be present in the population, but does not preclude the evaluation of non-novel solutions as in e.g. tabu search [2]. Because solutions that are novel (with respect to the current knowledge of the system) are selected for, their offspring have a better chance of being novel themselves. This, in turn, diverts the search away from wastefully retracing the same ground.

While the idea of driving search towards solutions that provide the most novelty, surprise, information gain, etc. is not new [5,10,12,13], to our knowledge it has not previously used as the sole criteria for selection in artificial evolution. When originally introduced, novelty search was demonstrated on a maze navigation task where the novelty was defined by the Euclidean distance between the final x, y position of the individuals in the maze at the end of a trial. For even a very low novelty threshold, these mazes were small enough such that the archive quickly filled to cover the full feature space.

In later work [6], the features were expanded to include 200 intermediate points along the path of an individual through the maze, instead of just the final

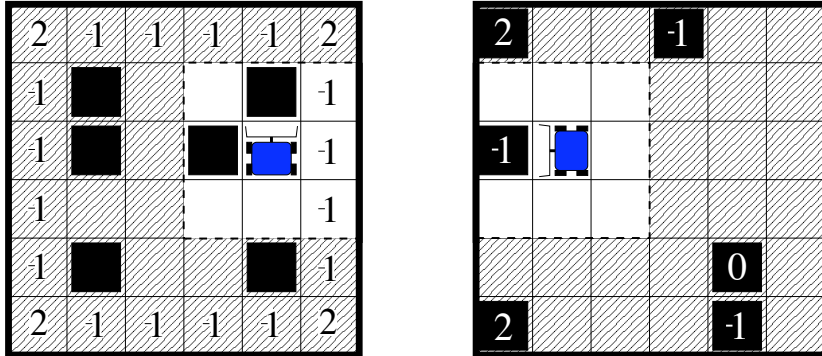


Fig. 1. The Deceptive Tartarus Problem. The Tartarus board on the left shows a possible initial state with the six blocks and the bulldozer placed at random squares away from the walls; the orientation of the bulldozer is also random. The bulldozer must select an action (either turn left, turn right, or go forward) at each time-step based on the situation within its visual field (shown in white), and its internal state (memory). The bulldozer can only move forward if its path is unobstructed or if the block in its way has no block behind it, otherwise it will remain in its current position. On the right is a possible final state after the allotted 80 moves. The score for this configuration is 1: two blocks receive a score of two for being in the corner, minus one point each for the three other blocks that are against a wall, and zero points for the block that is away from the walls.

position, thereby embedding the behaviors in a 400-dimensional vector space. Although this increases the number of possible behaviors, only a small portion of this space is reachable since adjacent points in a given path are highly correlated (i.e. the agent can only move so far in the interval between samples). Subsequent work [8] covertly reintroduced fitness into novelty search by requiring that individuals satisfy a goal-dependent objective or *minimal criterion* in order to be selected.

In the next section, fitness and novelty based search are combined explicitly in order to study the relationship between these two foci of selective pressure, and evaluate both approaches in a task where novelty is measured in a much larger space.

3 Experiments

3.1 The Deceptive Tartarus Problem

The Tartarus environment [14] consists of a 6×6 grid-world surrounded by walls within which six blocks and an agent, or *bulldozer*, are placed away from the walls. The bulldozer is allotted 80 moves, during each of which it takes one of three actions: turn left, turn right, or go forward. If the agent goes forward it can push a block that is in its path, as long as the block is not against a wall or another block. The score is defined in function of the final position of the

Algorithm 1: GENERICCROWDING(p, k, n, m, ρ)

```
1 Initialize the population  $P$  with  $p$  individuals
2 and evaluate them
3 for  $i=1$  to  $k$  do
4   parentA  $\leftarrow$  TOURNAMENTSELECT( $n, \rho$ )
5   parentB  $\leftarrow$  TOURNAMENTSELECT( $n, \rho$ )
6   (childA, childB)  $\leftarrow$  CROSSOVER(parentA, parentB)
7   MUTATE(childA) //mutate and evaluate
8   EVALUATE(childA) //the two offspring
9   MUTATE(childB)
10  EVALUATE(childB)
11   $l_A \leftarrow$  CROWDINGSELECT( $P, m, \text{childA}$ )
12   $l_B \leftarrow$  CROWDINGSELECT( $P, m, \text{childB}$ )
13   $P[l_A] \leftarrow \text{childA}$  //replace losers with
14   $P[l_B] \leftarrow \text{childB}$  //offspring
15 end
```

Function CROWDINGSELECT(P, n, x)

```
1 for  $i \leftarrow 1$  to  $n$  do
2    $j \leftarrow \text{RAND}(|P|)$  //choose random genotype
3   distance  $\leftarrow d(x, P[j])$  //compute distance
4   if distance  $<$  min then
5     min  $\leftarrow$  distance
6     loser  $\leftarrow j$ 
7   end
8 end
9 return loser //return the most similar
```

blocks. In the standard version of the task the objective is for the bulldozer to push the blocks against the walls, receiving one point for each block so placed, and corners are worth two points, for a maximum score of 10.

Although the grid-world is small, the problem is challenging because the bulldozer can only see the adjacent grid cells, so that many observations that require different actions look the same, (i.e. perceptual aliasing: there are approximately 20,000 times more states than there are observations) so that optimal behavior requires that the agent remember the relative locations of the blocks it has seen.

Due to the difficulty of the task, evolution can easily converge prematurely to strategies which employ simple, mechanical behaviors to produce better than random performance, but do not exhibit the sophisticated use of memory needed to place all of the blocks correctly. For example, because the arena is small in relation to the number of blocks, simply pushing a block forward as soon as it is encountered can earn a mediocre score (i.e. fitness between 4 and 6 points). Surpassing a score of about 6 requires instead a different, more complex behavior where the agent now has to first push a block against a wall, move around to the other side of the block, and then push it along the wall into a corner.

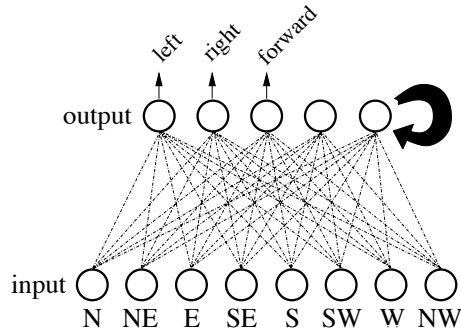


Fig. 2. Bulldozer controller. The bulldozer is controlled by a fully recurrent neural network with five sigmoidal units. At each time step the network receives the state of the eight surrounding grid cells through its input layer. The input is propagated to the output layer, along with the activation of the output layer from the previous time step (i.e. the recurrent connections denoted by the large black arrow), and the action corresponding to action unit (left, right, forward) with the highest activation is chosen for next time step.

For the standard Tartarus task, both fitness and novelty based selection encourage similar types of behavior. The deceptive version of the task introduced here decouples novelty and fitness. Now blocks lying against a wall (but not in a corner) receive a score of -1 , and the corners are still worth 2 points (figure 1), so that the optimal final configuration is: four blocks in the corners, and the other two away from the walls in a central position, for a maximum score of 8. In order to get points, the agent cannot simply push blocks around aimlessly, because fortuitously hitting a wall is no longer rewarded. This new scoring scheme is deceptive for fitness-based selection because the subgoal of getting a block against the wall is penalized. Pure novelty-based selection will behave as before, oblivious to the change in fitness measure.

3.2 Setup

Algorithm 1 presents pseudocode for the simple steady state GA used in the experiments. The algorithm takes five parameters: the size of the population p , the number of iterations k , the tournament size for selection n , the *crowding factor* [1] (used for replacement) m , and ρ described below. After the population of p individuals is initialized, each individual, $x_i, i = 1..p$, is evaluated to measure both fitness, $fit(i)$, and novelty, $nov(i)$ (according to equation 1).

Fitness and novelty measure two independent types of information about an individual on which selection can operate. A more general measure combines both. The simplest way to achieve this is to combine them linearly:

$$score(i) = (1 - \rho) \cdot \overline{fit}(i) + \rho \cdot \overline{nov}(i), \quad (3)$$

where $\rho \in [0, 1]$ controls the relative importance of fitness and novelty, which are normalized according to:

$$\overline{fit}(i) = \frac{fit(i) - fit_{\min}}{fit_{\max} - fit_{\min}}, \quad \overline{nov}(i) = \frac{nov(i) - nov_{\min}}{nov_{\max} - nov_{\min}},$$

and nov_{\min} , fit_{\min} , are the lowest novelty and fitness in the current population, respectively, and nov_{\max} and fit_{\max} are the corresponding highest values. A high ρ means that selection pressure is biased towards novelty and away from fitness; low values favor fitness. At $\rho = 0$, the algorithm uses pure fitness-based selection (goal-dependent); at the other extreme, $\rho = 1$, selection is entirely based on novelty (goal-independent).

Each iteration begins by selecting two parents based on the score computed according to equation (3) using tournament selection, and recombining them using 1-point crossover to produce two children. The latter undergo mutation and evaluation themselves, then the CROWDINGSELECT function chooses for each child, by tournament selection (tournament size m), the individual (i.e. the *loser*) that is most similar to the child according to the similarity measure $d(\cdot, \cdot)$. Each child replaces its corresponding loser, and the cycle repeats.

This simple Generic Crowding algorithm (also used in [4]) allows us to control the *selective* pressure that drives the population to convergence and at the same time an opposing, *replacement* pressure via the crowding factor that seeks to delay convergence.

Eleven sets of 20 experiments were run, each with a different value of ρ ranging from 0 to 1 in increments of 0.1, $\rho = \{0.0, 0.1, 0.2, \dots, 1.0\}$. As a baseline Random Weight Guessing (RWG) was used, where the network weights are chosen at random (i.i.d.) from a uniform distribution. This approach is used to give an idea of how difficult each task is to solve by simply guessing a good set of weights. All simulations were run for $k = 250,000$ iterations (two individuals are evaluated each iterations, for a total of 500,000 evaluations) using a population of $p = 100$ bulldozer controllers represented by fully recurrent neural networks with five sigmoidal units (figure 2). Three of the five units served as the outputs, one for each of the actions. The network genotypes were represented by real-valued vectors encoding the inputs and recurrent weights of each of the units using initial values chosen at random from $[-10, 10]$. The mutation operator changed the value of each weight with probability α to a new value chosen at random from the same range as the initial weights. Best results under all setups have been achieved through mutation probability $\alpha = 0.3$, tournament size of $n = 10$ and a crowding factor of $m = 10$, with all parameters being robust to small variations.

Each controller was evaluated on 100 random board configurations. To reduce evaluation noise, the set of 100 initial boards was chosen at random for each simulation, but remained fixed for the duration of the simulation, so that all of the networks in a given run were evaluated on the same initial boards.

In all experiments, the novelty of an individual was the average novelty distance (equation 1) of from $k = 15$ closest neighbors¹, where the novelty distance

¹ This is the value for k found in the source code used in the original novelty search paper [7]

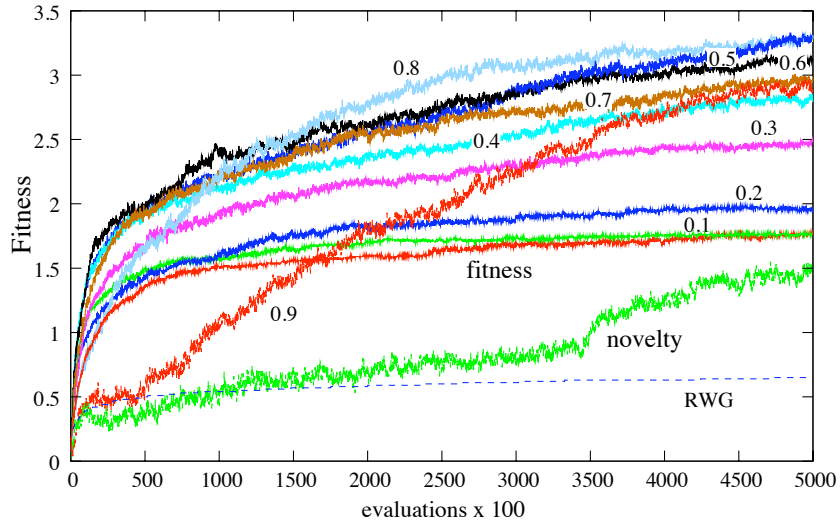


Fig. 3. Performance on the Deceptive Tartarus Task. Each curve shows the max fitness in the population over the course of run (average of 20 runs) for a particular value of ρ .

between two individuals, x and y , was computed as the average Manhattan distance between the corresponding blocks in the two sets of final (i.e. at the end of 80 moves) boards:

$$d(x, y) = \frac{1}{100} \sum_{i=1}^{100} \sum_{j=1}^6 \text{Manhattan}(b_{ij}^x, b_{ij}^y),$$

where b_{ij}^x is the final position of block j on board i for individual x .

The Hamming distance between the action sequences was used as the distance measure for CROWDINGSELECT. Each action sequence consisted of a string of 8,000 {Left=1 | Right=2 | Forward=3} actions; 80 actions for each of the 100 trials concatenated together.

3.3 Results

Both pure fitness and pure novelty runs performed poorly (figure 3). Intermediate values of ρ , ($0 < \rho < 1$) perform better, and as ρ approaches 0.8 (i.e. the score is 80% novelty and 20% fitness) the performance improves steadily. The size of the novelty archive (figure 4) grows dramatically with ρ , doubling twice in the last two increments (i.e. ρ from 0.8 to 1.0).

Figure 5 shows the average diversity in (a) genotype space, and (b) behavior space, every 100 iterations. The genotype diversity was measured as the average Euclidean distance between the chromosomes in the population, whereas behavioral diversity is the average Hamming distance between all the action sequences

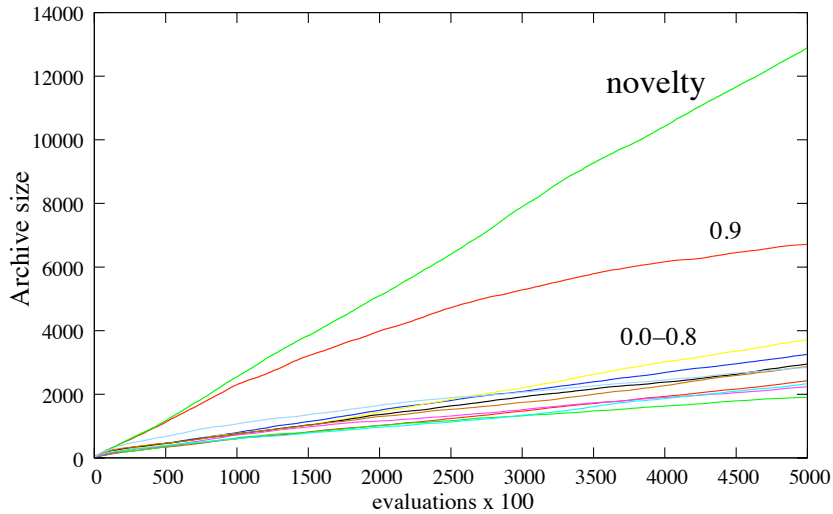


Fig. 4. Archive size. Each curve shows the size of the archive over the course of a run (average of 20 runs) for a particular value of ρ .

generated by the population. Both pure novelty and pure fitness are the least genotypically diverse. All intermediate- ρ runs sustained higher diversity in genotype space. The higher the value of ρ , the more behavioral diversity, but this does not translate into high genotypic diversity. This relationship has already been noted in [4] on the standard Tartarus problem. In that study, however, the most diverse populations with respect to behavior were also those with higher average fitness. Here, this does not hold. While the high- ρ populations are the most behaviorally diverse, the behaviors themselves are not necessarily fit. The number possible final board configurations, roughly $\binom{36}{6} = 1,947,792$, is simply too large to be searched efficiently by novelty alone. Though the archive size grows the most rapidly for pure novelty, the average number individuals stored there after 5000 iterations represents only a tiny fraction of the possible solutions.

4 Discussion

The results show that using novelty, as defined in [7], as the sole criterion for selection encourages behavioral diversity, but does not necessarily lead to high average fitness. In small search spaces, such as the one induced by the maze navigation task in [7], novelty search works because the novelty of an individual is correlated with an intuitive measure of utility: its final position in the maze. With proper tuning of the novelty threshold, the archive fills to cover the space of possible behaviors, some of which are later interpreted as “useful” behavior because they represent individuals near the goal.

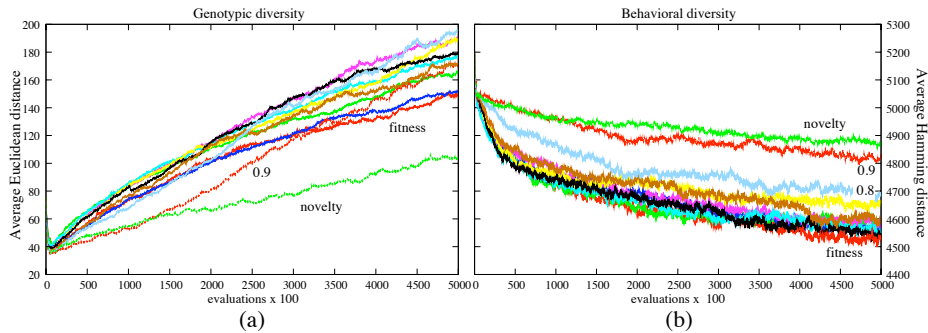


Fig. 5. Genotypic and behavioral diversity. (a) shows the average Euclidean distance between the chromosomes in the population every 100 evaluations, for each value of ρ . (b) shows the average Hamming distance between the action sequences (behaviors), in the population. All curves are the average of 20 runs.

The problem arises as soon as novelty and utility are decoupled. For example, in the deceptive Tartarus task, simply pushing blocks around is no longer rewarded. The configurations yielding good scores are much fewer compared to the standard task, and they are harder to achieve (comparable to the highest scores possible in the standard task). The point is that one can always design a fitness function such that the solutions discovered by novelty alone perform arbitrarily badly. For example, by changing the reward from -1 to -100 for each block against a wall, the average population score will drop drastically, since the progress of novelty search will be unaffected.

With such a large number of possible board configurations, the size of the archive grows steadily, dramatically slowing down computation because each new individual must be compared with both the population and the entire archive, resulting in a complexity per generation of $O(|P| * (|P| + |archive|)) = O(|P|^2)$. In our experiments, the size of the archive, $|archive|$, grew linearly (see figure 4) at a rate of 25 every 1000 evaluations, for pure novelty, $\rho = 1.0$. This means an increase in computational overhead of more than four times compared to $\rho < 0.8$, and around 100 times for pure fitness, since in practice it does not use an archive. Of course, the particular archiving system used does not form part of the general novelty search concept, and more sophisticated implementations requiring fewer comparisons per individual could make it more scalable.

Injecting some goal-direction into selection, or seen the other way, adding diversity to goal-dependent search, produced the best results in our experiments ($\rho = 0.4-0.9$). An obvious next step would be to investigate methods for automatically determining the ρ parameter. Novelty and fitness could be balanced for example by updating ρ based on the overall performance of the search. If the average or best fitness of the population does not improve over some number of generations, ρ could be turned up to provide more diversity and escape a local optimum. If the population fitness is instead steadily improving, a ρ could be turned down to better focus the search.

Acknowledgments

This research was supported by Swiss National Science Foundation grant #120061: “Advanced Cooperative NeuroEvolution for Unsupervised Learning and Autonomous Control”.

References

1. De Jong, K.A.: An Analysis of the Behavior of a Class of Genetic Adaptive Systems. Ph.D. thesis, The University of Michigan, Ann Arbor, MI (1975), university Microfilms No. 76-09381
2. Glover, F., Laguna, M.: Tabu Search. Kluwer Academic Publishers (1997)
3. Goldberg, D.E., Richardson, J.: Genetic algorithms with sharing for multimodal function optimization. In: Grefenstette, J.J. (ed.) Proceedings of the Second International Conference on Genetic Algorithms. pp. 148–154. San Francisco, CA: Morgan Kaufmann (1987)
4. Gomez, F.: Sustaining diversity using behavioral information distance. In: Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-09). pp. 113–120 (2009)
5. Krause, A., Guestrin, C.: Nonmyopic active learning of gaussian processes: An exploration-exploitation approach. In: Proceedings of the International Conference on Machine Learning (2007)
6. Lehman, J., Stanley, K.: Abandoning objectives: Evolution through the search for novelty alone. To appear in: Evolutionary Computation Journal (2010)
7. Lehman, J., Stanley, K.O.: Exploiting open-endedness to solve problems through the search for novelty. In: Proceedings of the Eleventh International Conference on Artificial Life (ALIFE XI). MIT Press (2008)
8. Lehman, J., Stanley, K.O.: Efficiently evolving programs through the search for novelty. In: Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-10) (2010)
9. Lehman, J., Stanley, K.O.: Revising the evolutionary computation abstraction: minimal criteria novelty search. In: Proceedings of the Genetic and evolutionary computation (GECCO-10). pp. 103–110. ACM, New York, NY, USA (2010)
10. MacKay, D.J.C.: Information-based objective functions for active data selection. neural computation. Neural Computation 4, 550–604 (1992)
11. Risi, S., Vanderbleek, S.D., Hughes, C.E., Stanley, K.O.: How novelty search escapes the deceptive trap of learning to learn. In: Proceedings of the 11th Annual Conference on Genetic and Evolutionary Computation (GECCO-09). pp. 153–160. ACM, New York, NY, USA (2009)
12. Schmidhuber, J.: Curious model-building control systems. In: Proceedings of the International Joint Conference on Neural Networks, Singapore. vol. 2, pp. 1458–1463. IEEE press (1991)
13. Schmidhuber, J.: Developmental robotics, optimal artificial curiosity, creativity, music, and the new arts. Connection Science 18, 173–187 (2006)
14. Teller, A.: Advances in Genetic Programming, chap. 9. MIT Press (1994)