# Solving Deceptive Tasks in Robot Body-Brain Co-Evolution By Searching For Behavioral Novelty

Peter Krčah

Computer Center
Charles University
Prague, Czech Republic
peter.krcah@ruk.cuni.cz

**Summary.** Evolutionary algorithms are a frequently used technique for designing morphology and controller of a robot. However, a significant challenge for evolutionary algorithms is premature convergence to local optima. Recently proposed Novelty Search algorithm introduces a radical idea that premature convergence to local optima can be avoided by ignoring the original objective and searching for any novel behaviors instead. In this work, we apply novelty search to the problem of body-brain co-evolution. We demonstrate that novelty search significantly outperforms fitness-based search in a deceiving barrier avoidance task but does not provide an advantage in the swimming task where a large unconstrained behavior space inhibits its efficiency. Thus, we show that the advantages of novelty search previously demonstrated in other domains can also be utilized in the more complex domain of body-brain co-evolution, provided that the task is deceiving and behavior space is constrained.

## 1 Introduction

Evolutionary algorithms are a frequently used technique for designing morphology and controller of a robot [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12]. The advantage of using evolutionary algorithms compared to other optimization methods is that only a high level fitness function and a set of genetic operators (mutation, crossover) need to be specified. While such fitness-driven search can be very successful, a common problem in evolutionary algorithms is premature converge to local optima. This is often caused by too much focus on exploitation of already discovered areas of the search space as opposed to exploration of yet unknown solutions. The trade-off between exploration and exploitation is well understood and several methods for addressing this issue have been proposed [13]. Most methods attempt to increase the rate of exploration by maintaining the diversity of individuals in a population through techniques such as fitness sharing [14, 15], age-layered population structure [16], multi-objectivization [17] or hierarchical fair competition [18]. Several of the diversity maintaining methods have also been applied to evolutionary robotics [19, 20, 11].

However, recent works suggest that the underlying problem lies not just in the balance of exploration and exploitation, but in the deceitfulness of the fitness function itself [21, 22, 23, 24, 25]. Novelty search, a recently proposed approach [21], introduces a radical idea that for some problems convergence to local optima can be avoided by simply ignoring the original objective altogether and instead only searching for *any* novel behaviors regardless of their quality with respect to the fitness function. While seemingly counter-intuitive, this approach has already been successfully applied to problems in multiple domains: evolving neuro-controllers for robot navigation in a maze [22, 21], evolving genetic programs [25, 23] or evolving learning artificial neural networks for maze navigation [24].

In this work, we use search for behavioral novelty as a method for avoiding premature convergence to local optima in the evolution of body and controller of a simulated robot. Subjects of our experiments are robots placed in a virtual 3D environment with physics simulated using rigid-body dynamics. Both body structure and controller of the robot are subject to optimization by evolution, forming a larger and more complex search space than in domains where novelty search was previously tested. We demonstrate the advantages and disadvantages of novelty search in this domain in two experiments: one with a deceitful fitness function with constrained space of possible behaviors and one with a large behavioral space and a less deceitful fitness function.

We start the chapter by describing the representation of a robot (sections 2.1, 2.2), fitness evaluation (section 2.3) and search algorithms (sections 2.4 and 2.5) used in our experiments. Section 3 then describes the setup of our experiments. Results of experiments along with examples of evolved behavior are provided in section 4. Chapter concludes with the discussion of results (section 5) and conclusions (section 6).

## 2 Methods

### 2.1 The Robot

The robot is composed of boxes of varying sizes connected by joints (see figure 1 for examples of evolved robots). Genetic representation of a robot is a directed graph (cycles in the graph are permitted) of nodes representing the body parts. Robot is created from this genome by first adding the body part represented by the root node of the graph and then recursively traversing connections in depth-first order, adding encountered nodes and connections to the robot (see figure 2 for examples of manually constructed robots). To prevent infinite recursion, each node has a *recursive limit* which limits the number of passes through the given node during transcription. Each node can thus be copied multiple (but finite) times to the robot, forming repeated structures. Each genotype connection also has a *terminal flag*. Connection with the terminal flag enabled is applied only when the recursive limit of its source node is reached. Terminal flag can be used to represent structures appearing at the end of chains or repeating units. Transition from a genotype graph with a terminal connection to corresponding phenotype graph is illustrated in figure 3.

Each node also specifies the *size* of the resulting body part (i.e. dimensions of a box) and a type of joint connecting body part to its parent in the resulting tree. Each joint type is defined by a set of rotational constraints imposed on two connected body parts. Different joint types thus have a different number of degrees of freedom (DOF). The following joint types are used: fixed (0 DOF), hinge (1 DOF), twist (1 DOF), hinge-twist (2 DOF), twist-hinge (2 DOF), universal (2 DOF) and spherical (3 DOF). Joint type is subject to mutation and recombination.

Both genotype node and genotype connection have various other properties used for building their phenotype counterparts. Each genotype connection contains information about the position of the child node relative to its parent node. The position is represented by child and parent *anchor points*, relative *rotation*, *scaling factor* and a set of three *reflection flags*, one flag for each major axis. Each enabled reflection flag causes a mirrored copy of the child node to be added to the phenotype graph (along with the original non-mirrored child node). All enabled reflection flags are always applied (if one, two or three reflection flags are enabled, two, four or eight mirrored copies of a child node are created in the phenotype graph, respectively). All geometric transformations (such as scaling, rotation and reflection) are cumulative, i.e. they are applied to the entire subtree of the phenotype graph during its construction. Step-by-step construction of a sample robot is shown in figure 4.

This representation permits very compact encoding of complex body structures allowing for features such as symmetry (using reflection flags) and repetitive segmentation (using recursive transcription).

Although other encodings have been proposed for evolving body and brain of robots [26, 27, 1, 8, 10], encoding used in this work has been successfully used for the evolution of robots in several previous works [7, 28, 29, 4, 30, 6, 2]. Examples of evolved robots are shown in figure 1.

ODE physics engine was used to provide realistic environment for the robot [31]. Water environment was simulated by applying a drag force opposing the movement of each body part and disabling gravity.

### 2.2 The Controller

Robot's controller is distributed in the body of the robot. Each body part contains a local neuro-controller (an artificial neural network), as well as a local sensor and effector. Artificial neural network was used for controllers.

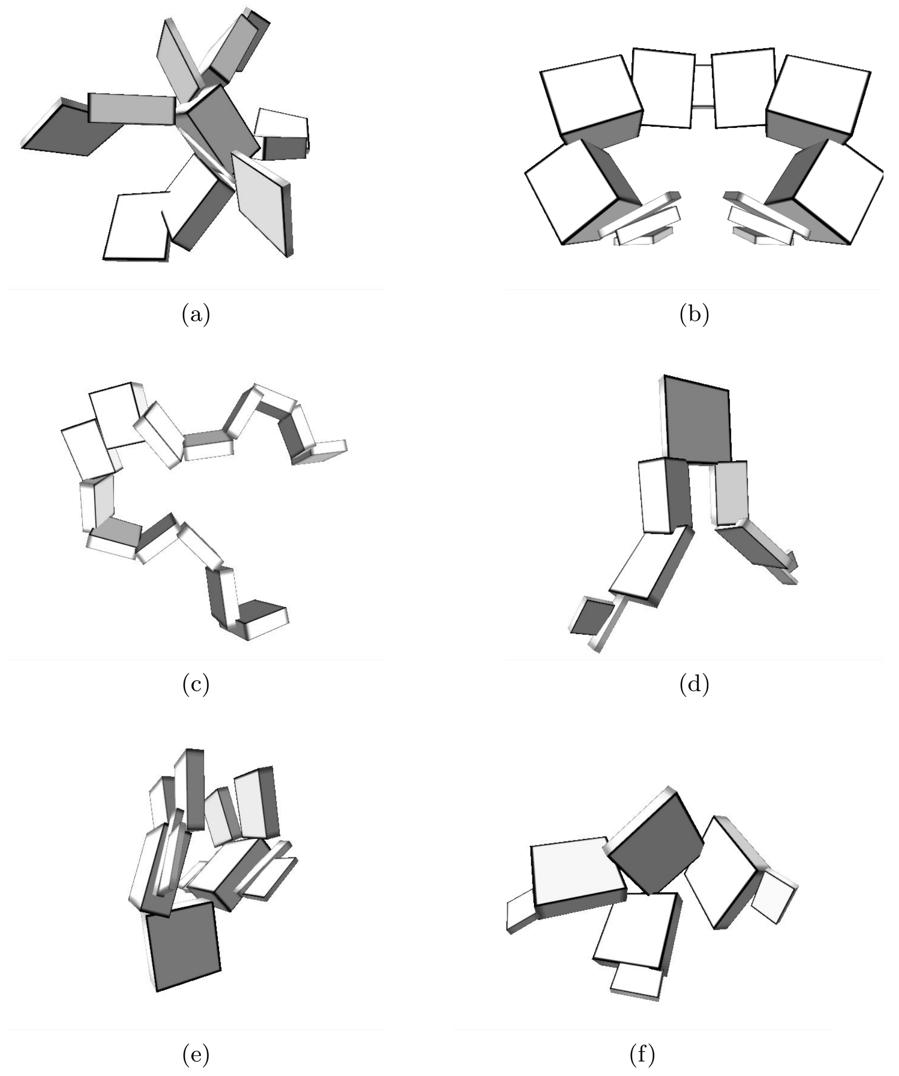


**Fig. 1. Examples of Evolved Robot Morphology.** Several evolved robots exhibit symmetry (1a, 1b, 1d) and segmentation (1c).

Apart from standard sigmoidal transfer function, oscillatory transfer function was used to enable faster discovery of efficient swimming strategies. A sensor in each body part is measuring current angle of each degree of freedom of a joint.

Effectors allow the robot to move in the simulated world by applying torque to the joints between body parts of the robot. Each degree of freedom

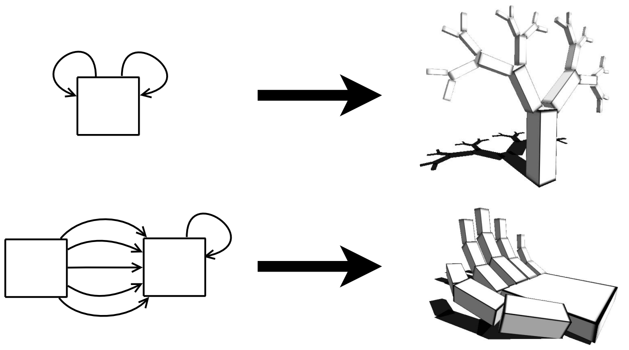


**Fig. 2.** Manually designed examples of a transcription of genotype to a robot phenotype.

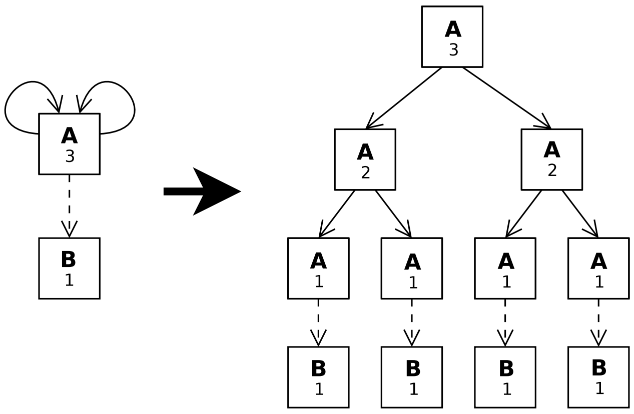
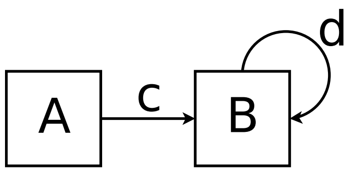
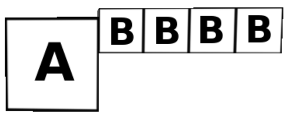


**Fig. 3.** Robot genotype (left) and phenotype (right). Dashed line represents a terminal connection. Recursive limit of each node is shown under the node mark (A, B).

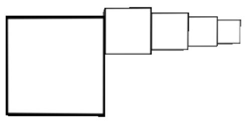
of each joint can be controlled separately. The straightforward application of the effector values to physical joints causes undesirable effects and instability in the simulation. Therefore, several transformations are applied to effector values before their application. Effector values are first limited to range $[-1, 1]$ and then scaled by a factor proportional to the mass of the smaller of two connected body parts. This transformation limits the maximum size of a force to some reasonable value and consequently improves simulation stability. Effector values are then smoothed by averaging previous ten values. The resulting value is used as the physical torque applied to a joint of the robot. This modification eliminates sudden large forces and also improves stability of the simulation.



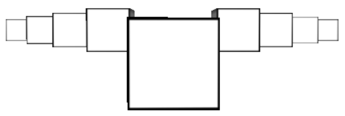
(a) Robot genotype. Recursive limits of nodes **A** and **B** are 1 and 4, respectively. Node **A** is the root node of the graph



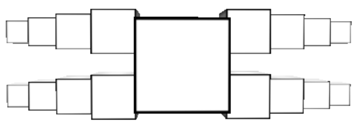
(b) Robot phenotype created from the genotype shown on the left. Four copies of node **B** are created during the recursive traversal of the genotype. Copies of node **B** are connected using connection **d**, while nodes **A** and **B** are connected using connection **c**.



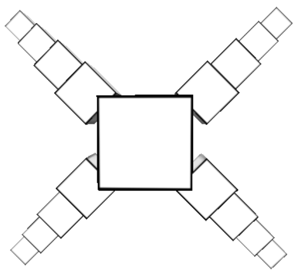
(c) Scaling factor of connection **d** has been changed from 1 to 0.8.



(d) Reflection flag for x axis of connection **c** has been enabled.



(e) Reflection flag for z axis of connection **c** has been enabled.



(f) One of three rotation angles of connection **c** has been changed from 0 to 45 degrees.

**Fig. 4.** Step-by-step construction of a robot. The structure of the robot genotype (a) is fixed during all steps. Robot is constructed by successive application of scaling (c), reflection for x axis (d), reflection for z axis (e) and rotation (f).

### 2.3 Fitness Evaluation

Fitness of each robot is evaluated in a simulated 3D physical world. Simulation proceeds in discrete simulation steps at the rate of 30 steps per simulated second. During each time step, following phases occur for each simulated robot:

1. Sensor values are updated based on the robot environment.
2. Neural signal is propagated through the distributed control system of the robot.
3. Torque is applied to each joint between body parts, according to the current value of the corresponding output neuron.
4. Simulation step is taken and the positions of all objects in the world are updated.

The simulation runs for a specified amount of time (60 seconds of simulated time) after which the fitness value of the robot is evaluated (see section 3).

Open Dynamics Engine (ODE [31]) is used to simulate the rigid body dynamics (including collision detection and the friction approximation). ODE has proved to be sufficient for the purposes of the simulation of evolving robots. However, search algorithms often exploit errors and instabilities in the physics engine to increase the fitness value of robots. Several mechanisms are needed to prevent such exploits. During the fitness evaluation, relative displacement of each two connected parts of each robot is watched and when it rises above a specified threshold (which signals that the robot is behaving unrealistically), the robot is immediately assigned zero fitness and its simulation is stopped. Individual body parts are also prohibited to reach angular or linear velocity above the specified threshold. Robot with such body parts is assigned zero fitness. Finally, an oscillation detector is used to prevent robots from moving using small oscillations (which is another way of exploiting inaccuracies in ODE). If an oscillating robot is detected, it is also discarded.

Several pre-simulation tests also need to be carried out to discover abusive robots before the simulation is started. E.g. the volume of each body part must be larger than the specified threshold as extremely small body parts cause instability in the physical engine. Such tests help detect invalid robots early, so that they do not consume computational resources during full-scale physical simulation.

### 2.4 Novelty Search

Instead of following the gradient of the fitness function, novelty search directs the search towards any yet unexplored parts of the behavior space. This is achieved by modifying the search algorithm to use the measure of individual's behavioral novelty instead of the fitness function. No other modifications to the underlying search algorithm are required.

Measure of individual's novelty is computed using an archive of previously discovered behaviors [23] as an average distance of the individual's behavior from $k$ closest behaviors recorded in the archive:

$$\rho(x) = \frac{1}{k} \sum_{i=0}^{k} dist(x, \mu_i) \tag{1}$$

where $\mu_i$ is the $i$th-nearest neighbor of $x$ and with respect to distance measure *dist*. This measure provides an estimate of local sparseness in the vicinity of the behavior being measured. Novelty search thus promotes individuals which are further away from already discovered behaviors.

If novelty of an individual exceeds some threshold value, individual is added to the archive. In this work the size of an archive was unlimited (previous works have successfully used limited archive [23] as well).

## 2.5 NeuroEvolution of Augmenting Topologies (NEAT)

NEAT is an efficient algorithm proposed originally for evolution of artificial neural networks (ANNs)[15] and recently extended for body-brain coevolution of robots [20]. The extended NEAT is used as an underlying search method in all experiments in this work. NEAT is unique in being able to meaningfully compare and crossover individuals represented by graphs with different topologies (e.g. robots with different morphology or ANNs with different structure).

This is achieved by using *historical markings*–unique inheritable identifiers assigned to each newly created structural element (a neuron or a body part). Historical markings bring the possibility of tracing individual structure elements throughout the evolution. Each structural element is assigned a unique identifier upon its creation (either during the construction of the initial population or during mutation). Historical markings are inherited, so each node and connection can be traced back to its oldest ancestor. Historical markings are also computationally inexpensive. The genetic algorithm only needs to keep track of the value of the most recent marking assigned to a neuron or a body part. Upon each request for a new marking (e.g. when the mutation creates a new node or a new connection), the value of the most recent marking is incremented and returned.

Historical markings offer an elegant method of sensibly mating structures with different topology. Parental structures are first scanned for the presence of structure elements with matching historical markings (these are the ones, which share the common ancestor and are thus considered compatible). Since many organisms during the evolution are very close relatives, the number of matching structure elements is expected to be fairly high. An offspring is constructed by first copying the parent with a higher fitness value, followed by random exchange of internal parameters of all matching nodes and connections with another parent. This way, new offspring inherits all non-matching nodes and connections from its better parent, while all matching elements are formed by a random recombination of properties of both parents.

Historical markings are used to efficiently measure similarity of two individuals which is used in NEAT to assign individuals to species. An explicit fitness sharing is used to maintain a set of species in the population, encouraging innovation by protecting it in a separate species. Complete description of NEAT method is presented in [15, 20]. Novelty search is integrated with NEAT by simply replacing the fitness value of an individual with a measure of its behavioral novelty as described in the previous section.

## 3 Experiments

By searching *only* for novelty in behavior, novelty search is less prone to falling into traps set up by the objective-based fitness function in the form of local optima. However if the behavior space is too large and unconstrained, novelty search may spend most of the time exploring behaviors that are uninteresting with respect to the objective and never find any useful solutions. To test both ends of this scale in the domain of body-brain co-evolution, two experiments were performed: one non-deceptive with a large unconstrained space of behaviors (the swimming experiment) and one highly deceptive with a constrained behavioral space (the barrier avoidance experiment).

In each experiment, the performance of novelty search was compared to the performance of standard fitness-based search. Random search was used as a baseline when comparing performance in each experiment (random search was performed by assigning each individual a random fitness value).

### 3.1 Swimming Experiment

The objective in the swimming experiment is to evolve robots capable of moving in a water environment the largest distance from the starting position in the allocated time. The difficulty in this task is that the robot must evolve useful morphology *and* control to successfully move through water. At the start of each test, robot is placed at the origin of the coordinate system. Robot is then free to move in any direction for 60 seconds after which the position of its center of mass is recorded. Fitness function is defined as the distance between the final and starting positions of the robot. Behavior of a robot is defined as the position of the robot at the end of the test and thus consists of a vector of 3 real values. Behavior metrics is defined as Euclidean distance between the final positions of two robots. There were no barriers for the robot to avoid and no constraints on the behavior of the robot apart from the limits of the physics simulation.

### 3.2 Barrier Avoidance Experiment

In the barrier avoidance experiment, the robot is placed at one end of a 20m x 20m x 25m container while the target is placed at the other end of the

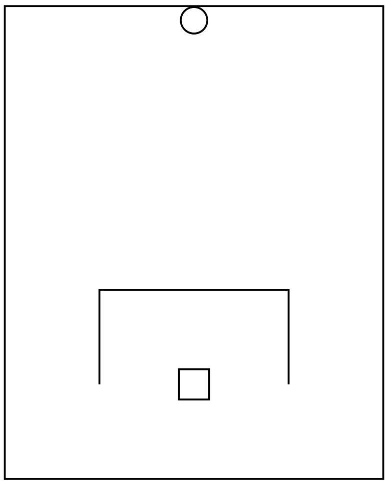


**Fig. 5. Barrier Avoidance Experiment.** Dimensions of the environment are 20m x 20m x 25m. The barrier is located between the robot and the target and is formed by a hollow 10m x 10m x 5m box with the bottom face left out. The target (indicated by the circle) is located at the top of the container at the initial distance of 20m from robot's initial position (indicated by the square).

container (see figure 5). Robot's goal is to reach the target within the time limit of 60 seconds. Task is made deceptive by putting the barrier directly in front of the robot, so it obstructs the direct path to the target. Moreover, the shape of the barrier only makes it possible to reach the target if the robot first moves *away* from it, which makes this task highly deceptive. Fitness function for a robot with final position $x$ is defined as

$$f(x) = \max(0, d(p_{start}, p_{target}) - d(x, p_{target})) \tag{2}$$

where $d(p_{start}, p_{target})$ is the constant distance from the starting position of the robot to the target (in this case 20m) and $d(x, p_{target})$ is the distance from the final position of the robot to the target. Behavior of the robot is defined as robot's final position (same as in the swimming experiment) and behavior metrics is defined as the Euclidean distance between final positions of two robots. Target position is fixed for all experiments and robot has no information about the position of the target. As in the swimming experiment, barrier avoidance experiment is performed in a simulated water environment.

### 3.3 Parameter Settings

The number of individuals in a population is set to 300, with 150 generations per run. In all experiments, initial generation was initialized with uniform robots consisting of a single box with no neurons except for inputs/outputs. All parameters of the underlying search algorithm (hierarchical NEAT[20]) were set to the same values in both the standard fitness-based configuration and in the novelty-based configuration. The configuration of hierarchical NEAT was the same as in our previous works [20]. Parameter $k$ for computing the novelty of an individual was set to 15 and the novelty threshold for adding an individual to the archive was set to 0.1. Archive size was not limited. Each configuration was tested independently 25 times in the barrier avoidance experiment and 30 times in the swimming experiment. All significance levels were computed using Student's t-test.

## 4 Results

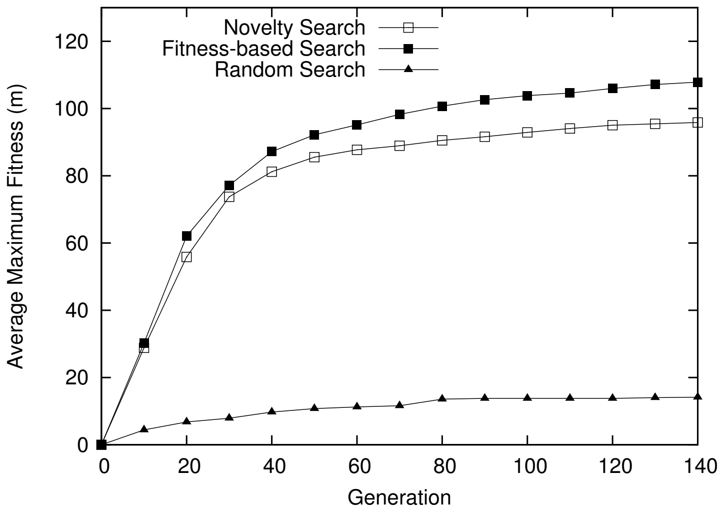
In the swimming experiment, both novelty search and fitness-based search were able to consistently find effective solutions (see figure 6a). Average maximum distance of the robot from the starting position after 150 generations was 108.32m for fitness-based search, 96.23m for novelty search and 14.46m for random search. Fitness-based search thus outperformed novelty search by 12.47% ($p < 0.05$), confirming that unconstrained behavior space without deceptive fitness function is favourable to the standard fitness-based approach.

In barrier avoidance experiment, fitness-based search never successfully found a way out of the barrier, reaching an average maximum fitness of 4.56m (see figure 6b). Performance of the fitness-based search was in this case only marginally better than random search which achieved maximum average fitness of 3.68m (although the difference was statistically significant; $p < 10^{-8}$). Novelty search was the only method that consistently escaped the trap and reached an average maximum fitness of 13.16m, significantly outperforming both fitness-based and random search ($p < 10^{-10}$).

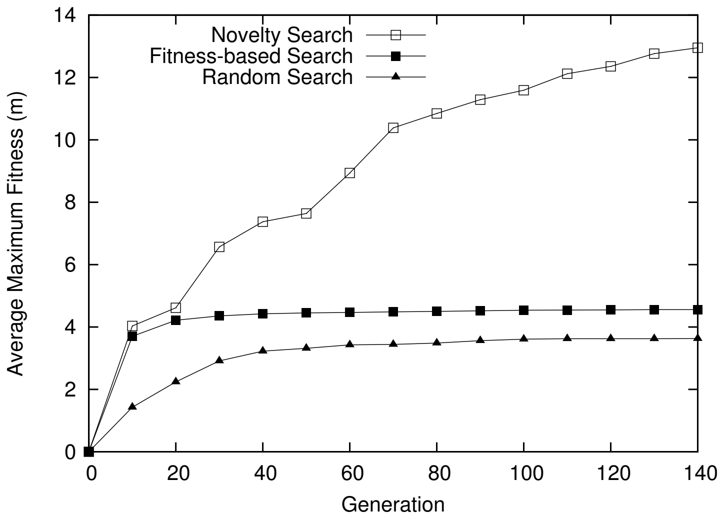
### 4.1 Analysis of Behavioral Diversity

To provide better insight into how individual search methods explore the space of behaviors, the coverage of behavior space was computed for each experiment. The coverage was computed as a total number of cells from a 1m x 1m x 1m grid that contained the final position of at least one robot. This statistic was computed after each generation using all behaviors seen since the start of the run. Resulting values were averaged over 25 runs (see figure 7).

Comparison of the amount of behavior space covered by individual methods shows that novelty search was exploring the behavior space faster than the fitness based search in both experiments ($p < 10^{-10}$). In swimming experiment the results demonstrate that, although novelty search achieved on average lower fitness values than fitness-based search, it was still able to efficiently explore the behavior space. The difference in how the two methods explore the behavior space can best be seen in the set of typical runs shown in figures 8a and 8b. While novelty search thoroughly and evenly explores the search space



(a) Swimming



(b) Barrier Avoidance

**Fig. 6. Comparison of Maximum Fitness Achieved by Novelty Search and Fitness-Based Search.** The average maximum fitness is shown for the swimming experiment (6a) and for the barrier avoidance experiment (6b) averaged over 25 runs. Results show that novelty search significantly outperforms fitness-based search in constrained deceptive task (barrier avoidance), while reaching similar performance in unconstrained non-deceptive task (swimming).

progressing in an ever-expanding sphere (figure 8a), fitness-based search tends to exhaustively search a small number of promising directions (8b).

In the barrier avoidance experiment, coverage of the behavior space for fitness based search never exceeded $500m^3$, which corresponds to the volume of the interior space of the barrier. A typical distribution of behaviors for such run is shown in figure 8d. While standard fitness-based approach always failed to escape the barrier, novelty search consistently explored the entire container, eventually reaching the target as well. Example of a typical run is shown in figure 8c.

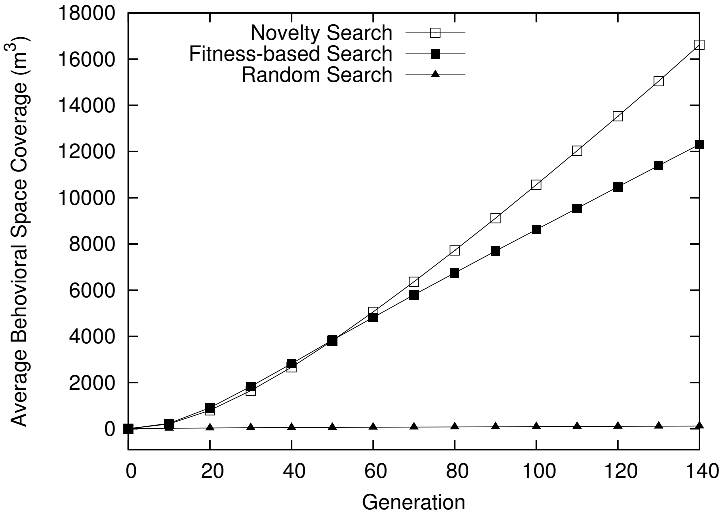
### 4.2 Combining Novelty Search with Fitness-based Search

To further analyze both search methods, additional experiments were performed using a combination of the novelty search and the fitness-based search. The combined search starts with novelty search and switches to fitness-based search after 20, 40, 60 or 80 generations. The motivation for switching the methods in the middle of the search is to more efficiently exploit strengths of each method in the barrier-avoidance experiment. Once novelty search overcomes the barrier (by focusing on the exploration part of the search), fitness based search may be able to quickly converge to a solution (by focusing on the exploitation).

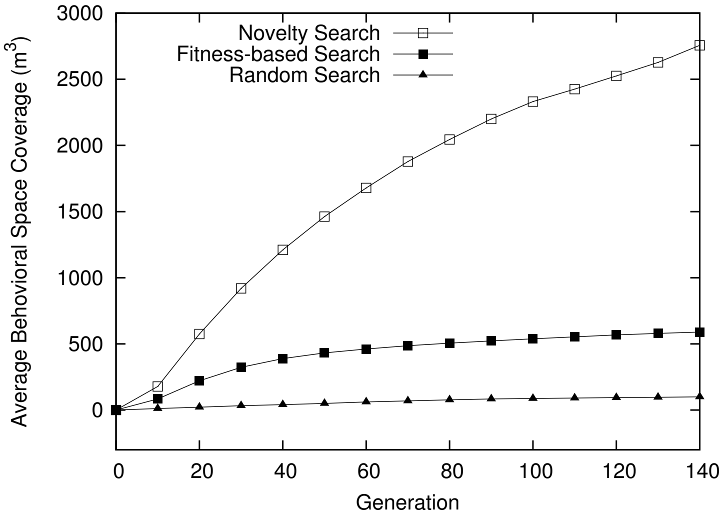
Results of experiments with combination of Novelty Search and Fitness-based search (shown in figure 9) show, that the later the switch was made, the higher the final fitness value was achieved. The best average fitness values were reached by the novelty search (13.16m), and the combined method switching at generation 60 (12.26m) and generation 80 (13.65m). The differences between these best methods were not statistically significant ($p > 0.25$). Two other two combined search methods achieved average fitness value of 10.39m (switch after 40 generations) and 6.81m (switch after 20 generations).

Comparison of the amount of behavior space covered by individual methods confirms that novelty search explores the behavior space faster than the fitness based search. In combined experiments, later switch to fitness-based search consistently resulted in higher coverage of behavior space. Combined search methods switching at generation 20, 40, 60 and 80 reached 10.57%, 17.33%, 23.48% and 26.49% of the container, respectively. Standalone novelty search reached the highest behavior space coverage of all methods: 28.87%. All differences in behavior coverage are statistically significant ($p < 0.05$) except the difference between novelty search and combined search switching after 80 generations ($p < 0.06$).

The hypothesis that switching to fitness-based search after the barrier is overcome will improve the performance of the search was not confirmed by the experiments. Experiments with combined search switching at generations 60 and 80 indicate that switching to fitness search may provide a speedup. However, the speedup was only temporary and the difference from novelty-search was not sufficiently significant ($p > 0.1$). Moreover, behavior space

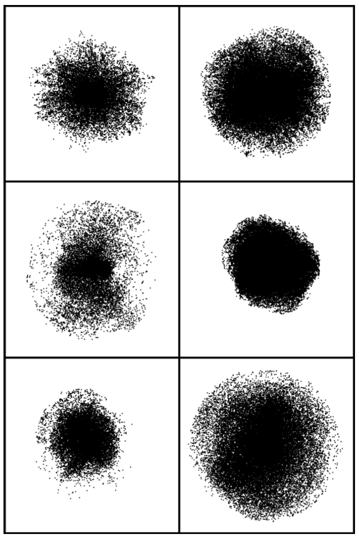


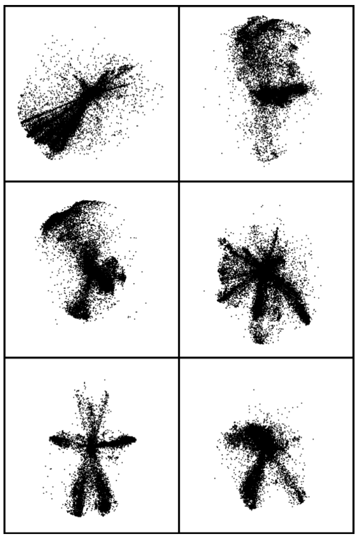
(a) Swimming

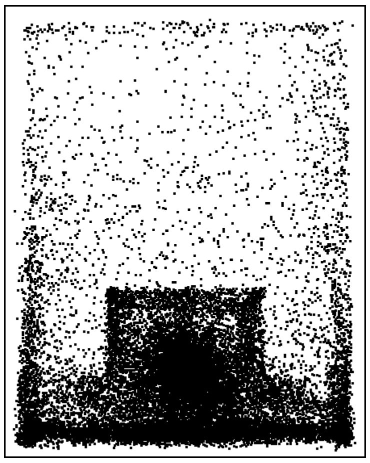


(b) Barrier Avoidance

**Fig. 7. Comparison of Behavioral Diversity In Novelty Search and Fitness-Based Search.** Behavioral diversity of a single evolutionary run is measured as a number of cells in 1m x 1m x 1m grid that contain at least one robot behavior discovered during the run. The cumulative behavioral diversity per generation averaged over 25 runs is shown for the swimming task (7a) and for the barrier avoidance task (7b). The main conclusion is that in both tasks, novelty search discovers significantly more behaviors than fitness-based search.


(a) Swimming (Novelty)


(b) Swimming (Fitness)


(c) Barrier Avoidance (Novelty)

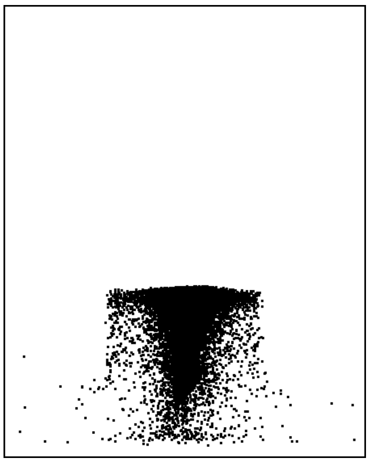
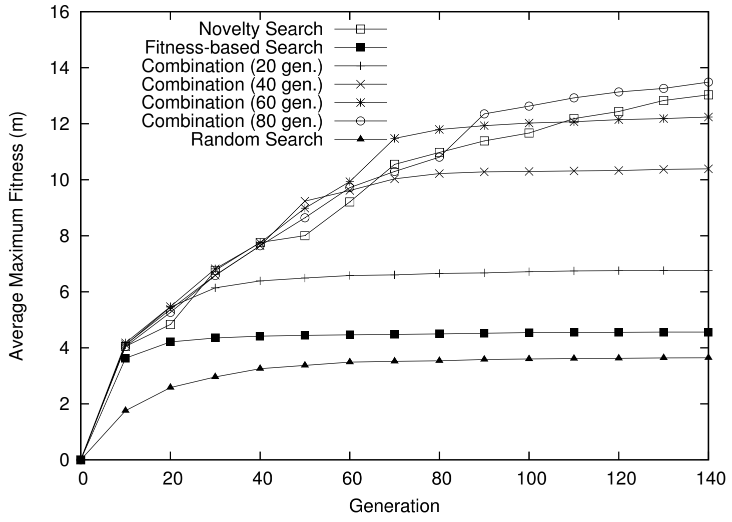
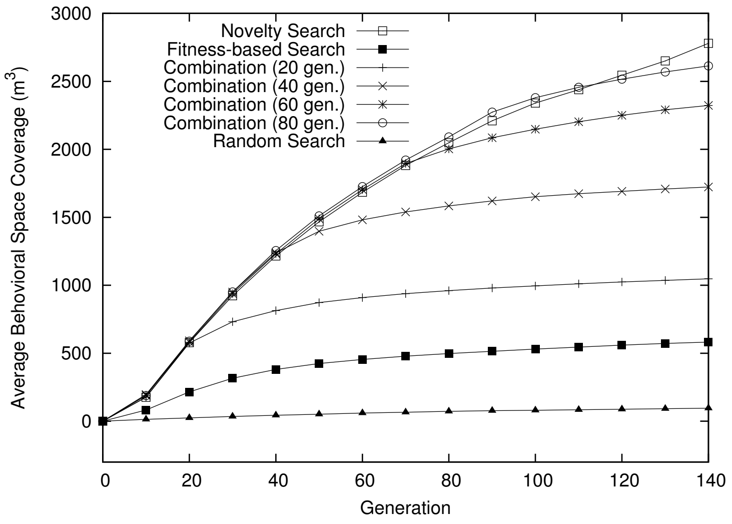

(d) Barrier Avoidance (Fitness)

**Fig. 8. Final Positions Of The Robot Visited In Typical Runs.** Final positions of all robots discovered in selected typical runs is shown for the swimming experiment (8a, 8b) and for the barrier avoidance experiment (8c, 8d). Novelty search in both experiments explores the space of behaviors more evenly, while fitness-based search focuses on optimization of few chosen areas. For swimming experiment (8a, 8b) a 260m x 260m x 260m part of the behavior space is shown. Final positions are three-dimensional and are shown using a two-dimensional orthogonal projection.



(a) Fitness Value (see equation (2))



(b) Behavior Space Coverage

**Fig. 9. Comparison of Behavioral Diversity and Maximum Fitness for Different Combinations of Novelty Search and Fitness-based Search.** Combined algorithm performs fitness-based search for the first fixed number of generations and then switches to using Novelty Search for the rest of the search. Figure 9a shows the average maximum fitness averaged over 25 runs and figure 9b shows the cumulative behavioral diversity per generation averaged over 25 runs.

analysis shows that after switching to fitness-based search, the exploration rate slowed significantly.

## 5 Discussion

In the swimming experiment the behavior space was only constrained by the limits of the physical simulation (the speed at which a robot can move is limited by the maximum amount of torque it can exert by its joints). Such large space of possible behaviors together with absence of any obvious deceptiveness makes this problem unfavorable to the novelty search. This intuition was confirmed by the results of the swimming experiment, where novelty search was outperformed by the fitness-based search, although it was able to explore more of the behavior space (figures 6a and 7a).

However, the results of the swimming experiment do not confirm the expectation that in such unconstrained environment the novelty search will perform just as poorly as a random search. On the contrary, novelty search performed almost as well as the fitness-based search and it significantly outperformed random search. The reason is that even though novelty search does not explicitly optimize solutions towards the main objective, it still explores behavior space in a structured way (even when the behavior space is practically unlimited) progressing from simple to more complex behaviors. This is unlike the random search which often revisits the same solutions repeatedly.

In the barrier avoidance experiment, the deceptiveness of the task lies in the fact that fitness-based search leads the robot directly to the target until it reaches the barrier. At that point, in order to find better solutions, the search needs to explore behaviors that are temporarily further away from target. In the standard objective-based approach, getting further away from the target decreases the fitness value; fitness function thus has a local optimum inside the barrier where fitness-based search is likely to be trapped. This was confirmed by the results of the barrier experiment with fitness-based configuration, where individuals never fully escaped the barrier. In the same experiment, novelty search was able to explore the entire container and was capable of consistently finding the target. The effectiveness of the novelty search in this experiment can be explained by the fact that the space of all possible behaviors was constrained to the size of the container; the boundaries of the container in this case served in a sense as a guide directing the search towards the target.

## 6 Conclusions

In this work, we applied recently proposed novelty search algorithm to the evolution of body and brain of a simulated robot. We demonstrated advantages and disadvantages of novelty search on two tasks: swimming and barrier avoidance. Results from the swimming experiment have confirmed that novelty search does not provide an advantage for tasks where behavior space is unconstrained and fitness function is not deceptive. For such tasks, novelty search may still explore the space of possible behaviors more effectively than fitness-based search, but it takes longer to optimize solutions towards the objective.

However, the barrier avoidance experiment has shown that if the range of possible behaviors is limited (in this case by putting the robot inside a container) and fitness function is deceptive then the novelty search algorithm significantly outperforms fitness-based search.

In summary, this work confirms that the advantages of novelty search demonstrated previously in other domains can be successfully leveraged in the co-evolution of body and brain of robots. Results presented in this work demonstrated that directing the search towards behavioral novelty can help in solving deceptive problems in body-brain evolution that standard fitness-based search methods often struggle with.

## Acknowledgment

The research is supported by the Charles University Grant Agency under contract no. 9710/2011.

## References

1. G. S. Hornby, H. Lipson, and J. B. Pollack, "Generative representations for the automated design of modular physical robots," *IEEE Transactions on Robotics and Automation*, vol. 19, no. 4, pp. 703–719, 2003.
2. N. Lassabe, H. Luga, and Y. Duthen, "A new step for evolving creatures," in *IEEE-ALife'07, Honolulu, Hawaii, 01/04/2007-05/04/2007*. http://www.ieee.org/: IEEE, avril 2007, pp. 243–251.
3. H. Lipson and J. B. Pollack, "Automatic design and manufacture of robotic lifeforms." *Nature*, vol. 406, no. 6799, pp. 974–978, August 2000.
4. T. Miconi, "In silicon no one can hear you scream: Evolving fighting creatures." in *EuroGP*, ser. Lecture Notes in Computer Science, M. O'Neill, L. Vanneschi, S. Gustafson, A. Esparcia-Alczar, I. D. Falco, A. D. Cioppa, and E. Tarantino, Eds., vol. 4971. Springer, 2008, pp. 25–36.
5. S. Nolfi and D. Floreano, *Evolutionary Robotics. The Biology, Intelligence, and Technology of Self-organizing Machines.* Cambridge, MA: MIT Press, 2001, 2001 (2nd print), 2000 (1st print).
6. K. Sims, "Evolving 3d morphology and behavior by competition," *Artificial Life*, vol. 1, no. 4, pp. 353–372, 1994.
7. ——, "Evolving virtual creatures," in *SIGGRAPH '94: Proceedings of the 21st annual conference on Computer graphics and interactive techniques.* New York, NY, USA: ACM Press, 1994, pp. 15–22.

8. M. Komosinski and S. Ulatowski, "Framsticks: Towards a simulation of a nature-like world, creatures and evolution," in *Proceedings of 5th European Conference on Artificial Life (ECAL99)*, J.-D. N. D. Floreano and F. Mondada, Eds. Springer-Verlag, 1999, pp. 261–265.
9. G. S. Hornby, M. Fujita, and S. Takamura, "Autonomous evolution of gaits with the sony quadruped robot," in *Proceedings of the Genetic and Evolutionary Computation Conference.* Morgan Kaufmann, 1999, pp. 1297–1304.
10. F. Gruau, "Cellular encoding for interactive evolutionary robotics," in *Fourth European Conference on Artificial Life.* Cambridge, MA, USA: MIT Press, 1997, pp. 368–377.
11. J. C. Bongard and G. S. Hornby, "Guarding against premature convergence while accelerating evolutionary search," in *GECCO '10: Proceedings of the 12th annual conference on Genetic and evolutionary computation.* New York, NY, USA: ACM, 2010, pp. 111–118.
12. M. L. Pilat and C. Jacob, "Evolution of vision capabilities in embodied virtual creatures," in *GECCO '10: Proceedings of the 12th annual conference on Genetic and evolutionary computation.* New York, NY, USA: ACM, 2010, pp. 95–102.
13. D. E. Goldberg, *Genetic Algorithms in Search, Optimization, and Machine Learning.* Addison-Wesley Professional, January 1989.
14. D. E. Goldberg and J. Richardson, "Genetic algorithms with sharing for multimodal function optimization," in *Proceedings of the Second International Conference on Genetic Algorithms on Genetic algorithms and their application.* Mahwah, NJ, USA: Lawrence Erlbaum Associates, Inc., 1987, pp. 41–49.
15. K. O. Stanley and R. Miikkulainen, "Efficient reinforcement learning through evolving neural network topologies," in *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2002).* San Francisco, CA: Morgan Kaufmann, 2002.
16. G. S. Hornby, "Alps: the age-layered population structure for reducing the problem of premature convergence," in *GECCO '06: Proceedings of the 8th annual conference on Genetic and evolutionary computation.* New York, NY, USA: ACM, 2006, pp. 815–822.
17. J. D. Knowles, R. A. Watson, and D. Corne, "Reducing local optima in single-objective problems by multi-objectivization," in *EMO '01: Proceedings of the First International Conference on Evolutionary Multi-Criterion Optimization.* London, UK: Springer-Verlag, 2001, pp. 269–283.
18. J. Hu, E. Goodman, K. Seo, Z. Fan, and R. Rosenberg, "The hierarchical fair competition (hfc) framework for sustainable evolutionary algorithms," vol. 13, no. 2. Cambridge, MA, USA: MIT Press, 2005, pp. 241–277.
19. J.-B. Mouret and S. Doncieux, "Overcoming the bootstrap problem in evolutionary robotics using behavioral diversity," in *CEC'09: Proceedings of the Eleventh conference on Congress on Evolutionary Computation.* Piscataway, NJ, USA: IEEE Press, 2009, pp. 1161–1168.
20. P. Krcah, "Towards efficient evolutionary design of autonomous robots," in *ICES '08: Proceedings of the 8th international conference on Evolvable Systems: From Biology to Hardware.* Berlin, Heidelberg: Springer-Verlag, 2008, pp. 153–164.
21. J. Lehman and K. O. Stanley, "Exploiting open endedness to solve problems through the search for novelty," in *In Proceedings of the Eleventh International Conference on Artificial Life.* Cambridge, MA: MIT Press, 2008.

22. ——, “Revising the evolutionary computation abstraction: minimal criteria novelty search,” in *GECCO ’10: Proceedings of the 12th annual conference on Genetic and evolutionary computation.* New York, NY, USA: ACM, 2010, pp. 103–110.
23. ——, “Efficiently evolving programs through the search for novelty,” in *GECCO ’10: Proceedings of the 12th annual conference on Genetic and evolutionary computation.* New York, NY, USA: ACM, 2010, pp. 837–844.
24. S. Risi, S. D. Vanderbleek, C. E. Hughes, and K. O. Stanley, “How novelty search escapes the deceptive trap of learning to learn,” in *GECCO ’09: Proceedings of the 11th Annual conference on Genetic and evolutionary computation.* New York, NY, USA: ACM, 2009, pp. 153–160.
25. J. Doucette and M. I. Heywood, “Novelty-based fitness: An evaluation under the santa fe trail,” in *Genetic Programming, 13th European Conference, EuroGP 2010*, ser. Lecture Notes in Computer Science. Springer, 2010, pp. 50–61.
26. J. Bongard, “Evolving modular genetic regulatory networks,” in *Proceedings of the 2002 Congress on Evolutionary Computation, CEC ’02*, vol. 2, 2002, pp. 1872–1877.
27. P. Eggenberger, “Evolving morphologies of simulated 3D organisms based on differential gene expression,” in *Proceedings of the Fourth European Conference on Artificial Life.* Cambridge, MA: MIT Press, 1997, pp. 205–213.
28. Y.-S. Shim and C.-H. Kim, “Generating flying creatures using body-brain coevolution,” in *SCA ’03: Proceedings of the 2003 ACM SIGGRAPH/Eurographics symposium on Computer animation.* Aire-la-Ville, Switzerland, Switzerland: Eurographics Association, 2003, pp. 276–285.
29. N. Chaumont, R. Egli, and C. Adami, “Evolving virtual creatures and catapults,” *Artificial Life*, vol. 13, pp. 139–157, 2007.
30. T. Miconi and A. Channon, “An improved system for artificial creatures evolution,” in *Artificial Life X: Proceedings of the Tenth International Conference on the Simulation and Synthesis of Living Systems*, L. M. Rocha, L. S. Yaeger, M. A. Bedau, D. Floreano, R. L. Goldstone, and A. Vespignani, Eds. Bloomington, IN, USA: MIT Press, 3-7 June 2006, pp. 255–261.
31. R. Smith, “Open dynamics engine manual,” available at http://www.ode.org.