Towards Efficient Evolutionary Design of Autonomous Robots

Peter Krčah

Computer Center, Charles University
Ovocný Trh 5, 116 36 Prague 1, Czech Republic
peter.krcah@ruk.cuni.cz

Abstract. Recent works explored the possibility of designing physical robots using evolutionary algorithms. We propose a novel algorithm for the evolution of morphology and control of autonomous robots controlled by artificial neural networks. The proposed algorithm is inspired by NeuroEvolution of Augmenting Topologies (NEAT) which efficiently evolves artificial neural networks. All three main components of NEAT algorithm (protecting evolutionary innovation through speciation, effective crossover of neural networks with different topologies and incremental growth from minimal structure) are applied to the evolution of both morphology and control system of a robot. Large-scale experiments with simulated robots have shown that the proposed algorithm uses significantly less fitness evaluations than a standard genetic algorithm on all four tested fitness functions. Positive contribution of each component of the proposed algorithm has been confirmed with a series of supplementary ablation experiments.

1 Introduction

The current technology is capable of automatically designing hardware for autonomous robots using evolutionary methods [3,4,7]. However, the current methods are limited in the complexity of both control and physical design of evolved robots. Most of the evolutionary design techniques used today to automatically design both morphology and control of an autonomous robot do not evaluate the fitness directly on the physical robot. The primary reason is the technical difficulty of automatically evaluating different robot morphologies. Instead, the common approach is to evolve the robot in a simulated environment and to construct only the evolved champion in reality. This approach is, however, still very time-consuming. The high demand on the computer resources is one of the major factors prohibiting the evolution of more sophisticated robots. To allow evolution of more complex robots, it is necessary to propose methods which are more effective in evolving robot morphology and control.

Research in the area of evolution of autonomous robots was pioneered by Karl Sims in 1994 [9]. His virtual creatures inhabit a three-dimensional world with simulated physical laws and are controlled by a series of neural networks distributed along the body of each creature. Since its publication in 1994, virtual

G.S. Hornby et al. (Eds.): ICES 2008, LNCS 5216, pp. 153–164, 2008.

creatures have inspired much research in this area. Several works attempted to build evolved robots in reality [3], [7]. Others concentrated on the genetic representation of the robots and associated genetic operators [4], [8]. Bongard et al. [1] has recently proposed a method of automatic recovery from unexpected damage for physical robots, through adaptive self-modelling. Physical robot uses its sensor and actuator values to infer its (possibly damaged) morphology and then uses this self-model to move forward.

In summary, several works have successfully evolved autonomous robots. However, the major drawback in evolving autonomous robots is the high demand on computer resources. Karl Sims performed his experiments on a parallel computer with 32 processors – Connection Machine CM-5. According to Sims, single evolution has taken three hours to finish [9]. Today, single evolution takes typically multiple hours (or even days) to complete using a single standard PC computer. While computer speed is increasing fast, large-scale experiments with simulated robots are still difficult to carry out without the use of distributed computing. The algorithm proposed in this paper uses significantly less fitness evaluations to evolve robots with at least the given fitness value than a standard genetic algorithm. While not a definitive solution to the problem (distribution is still required), it is a step towards evolving robots faster than possible today.

The proposed algorithm – *Hierarchical NEAT* – is inspired by NeuroEvolution of Augmenting Topologies (NEAT) – a recent successful algorithm for the evolution of neural networks [11]. The proposed algorithm extends NEAT to evolve both morphology and control of robots. The first part of this paper (Sections 2 and 3) provides background about NEAT and robot representation. The second part (Sections 4 and 5) describes the proposed algorithm and experiments conducted to test its properties along with discussion and directions for future research (Sections 6 and 7).

2 NeuroEvolution of Augmenting Topologies

NEAT is an algorithm proposed by Stanley and Miikkulainen in 2002 [11]. NEAT is unique in its ability to evolve topology of a neural network along with weights of neural connections by introducing three concepts: a method of crossover of networks with different topologies using a concept of *historical markings*, a method of protecting innovation through *speciation* and starting the evolution of neural networks from a *minimal topology*. This section provides brief overview of key components of NEAT algorithm. For a more comprehensive description see [11].

2.1 Recombination Using Historical Markings

The concept of historical markings is central to NEAT algorithm. Markings bring the possibility of tracing individual structure elements (e.g. neurons and neural connections) throughout the evolution. Each element is assigned a unique identifier (i.e. a historical marking) upon its creation (e.g. during mutation). Markings are inherited, so each element can be traced back to its oldest ancestor.

Recombination of two arbitrary neural networks with different topologies is a challenging task. Without any additional information, expensive topology matching algorithms must be employed for finding correspondences between neurons in parents. Historical markings, however, offer a simple solution to this problem. Parental structures are first scanned for the presence of structure elements with matching historical markings. An offspring is constructed by first copying the parent with a higher fitness value, followed by a random exchange of internal parameters of all matching nodes and connections with another parent.

2.2 Speciation

Speciation in NEAT serves two purposes: it maintains the diversity of the population and it protects structural innovation. Neural networks in the population are divided into species according to their similarity. Each network is compared to a representative of each species one at a time (representative is chosen randomly, e.g. as the first member of a species). If the value of compatibility distance is smaller than the specified compatibility threshold, the network is placed in that species. If none of the species satisfies the condition, a new species is created.

The compatibility distance measure in NEAT is defined as a linear combination of the number of non-matching genes (i.e. those without a counterpart in other parent) and average weight difference in matching neural connections.

Neural networks compete only against networks in the same species (explicit fitness sharing mechanism is used for reproduction [2]). This allows new structural innovation (which might initially be disadvantageous) to be optimized in a separate species, instead of being immediately dominated by currently better networks in the entire population.

2.3 Minimizing Dimensionality

Many approaches to the evolution of structure start with a population of networks with randomly generated structure. However, it has been shown that starting from a complex randomly-generated structure might decrease the performance of the evolution, because the random generation introduces a lot of unjustified structure, not tested by a single fitness evaluation [11].

NEAT introduces a concept of starting with a minimal structure (small random networks) and increasing complexity of the network as the evolution proceeds. Starting the search this way minimizes the dimensionality of the search space during the early stages of the search.

3 Robot Representation

This section presents an overview of the robot representation used in this paper. Robot simulator has been implemented using ODE physics engine [10]. For a more comprehensive description, see [5,6]. The genetic representation and mating operators (grafting and crossover) are inspired by Karl Sims' virtual creatures [9].

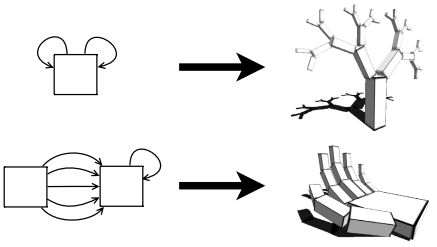


Fig. 1. Manually designed examples of a transcription of genotype to a physical robot

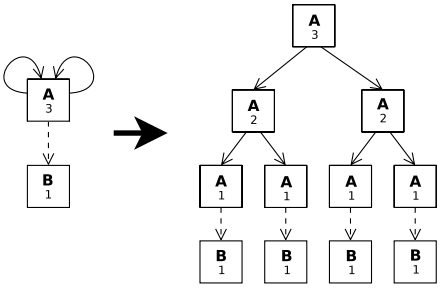


Fig. 2. Transcription example with terminal connections (dashed lines). Recursive limits are shown inside each node.

3.1 Morphology

A physical robot is represented by a rooted tree of morphological nodes. Each node corresponds to a single body part (e.g. a box) and each connection corresponds to a physical joint between two body parts (see Figure 1 for examples).

The robot phenotype is created from a corresponding genetic template (i.e. a *genotype*). Genotype is a directed graph (not necessarily a tree; cycles are permitted). Phenotype is created from the genotype by first copying the *root node* and then recursively traversing connections in depth-first order, adding encountered nodes and connections to the phenotype graph. To prevent infinite recursion, each genotype node has a *recursive limit* which limits the number of passes through the given genotype node. Each genotype node can thus be copied multiple (but finite) times to the phenotype graph. 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 (see Figure 2).

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).

Each genotype node contains information about the *shape* and the *size* of the resulting morphological node (in the case of a box, its dimensions are specified) and a *joint-type*. The following joint types are used: *fixed*, *hinge*, *twist*, *hinge-twist*, *twist-hinge*, *universal* and *spherical*. Each joint type is defined by a set of rotational constraints imposed on two connected body parts.

3.2 Control

Robot's control system is distributed over its entire body. Each morphological node has its local sensors, effectors and a local controller. Besides local

controllers, a single global controller (i.e. the "brain") is also present to allow global coordination. Local controller in a child node can also communicate with its parent node controller (neural connections in both directions are allowed). This way, neural signal can spread through the robot body in a fashion similar to natural organisms.

In this project, artificial neural network (ANN) controllers are used exclusively. Connection weights are limited to the range $[-2, 2]$. Feed-forward topology of the network is used.

Large set of transfer functions is used, as proposed by Sims [9]: *abs, atan, cos, differentiate, exp, integrate, log, memory, osc-saw, osc-wave, sigmoid, sign-of, sin, smooth-linear, divide, max, min, product, sum, greater-than, if, interpolate, sum-threshold*. Each neuron has one, two or three inputs, depending on the transfer function. Multiple connections can be connected to a neuron input. Neuron computes its value by summing weighted values of all enabled connections for each input and computing transfer function value on these inputs.

Each sensor is contained in a specific morphological node. Following sensor types were used in experiments with the robots: *a joint angle sensor*, *a gyroscopic sensor*, *a touch sensor* and *a photo sensor*.

Only one type of effector, *a joint effector*, is currently used. Each degree of freedom of a joint is controlled separately, by a single effector output.

3.3 Validity Testing

During the evolution, robots often exploit properties of the physical simulation to their advantage. However, several pre-simulation tests can be carried out to discover abusive robots early (e.g. test for self-penetration, excessively small body parts, excessively large number of body parts). Operators such as mutation, grafting and crossover are performed repeatedly until a genotype passing the test is introduced or a specified number of unsatisfactory genotypes is created (in that case, the last created genotype is returned). Without the testing process, most of the newly created population contains invalid robots, effectively disabling the evolution process.

4 The Proposed Algorithm

This section introduces *Hierarchical NEAT* (hNEAT) – a novel algorithm for evolving autonomous robots (for a more comprehensive description of the proposed algorithm see [5,6] or visit http://ero.matfyz.cz). The algorithm is an application of the three main concepts introduced by NEAT to the morphology and control of evolving robots.

NEAT uses the concept of historical markings to trace individual neurons and neural connections during the evolution of neural networks. Our algorithm extends this concept by introducing *hierarchical* historical markings to trace structural elements of both morphology and control. Markings on the level of morphology are managed as in NEAT (see Section 2): each morphological node and connection is assigned a unique and inheritable historical marking upon

its creation. In addition to that, historical markings are also assigned to all newly created neurons. Markings on the morphological level are independent of markings on the control system level (i.e. they are never compared to each other). Moreover, markings of neurons in different morphological nodes are also independent.

4.1 Recombination

Phenotypes are mated hierarchically, based on the correspondence given by historical markings. Morphology graphs of both parents are first searched for the presence of nodes and connections with the same historical markings. The parent with higher fitness value is then copied to become the first draft of an offspring. Values of internal properties of all matching nodes and connections in the offspring are then randomly chosen either from the first or from the second parent.

The only exception from this rule is the local neural network contained in each morphological node. When a pair of corresponding morphological nodes is found, the neural network of the offspring is *not* created by copying one of the parental neural networks. Instead, neural networks themselves are combined internally, based on the historical markings of individual neurons. Recombination of neural networks is, again, based on the correspondence given by historical markings. All internal properties of matching neurons (transfer function) and neural connections (weight) are chosen from a random parent.

Thanks to the historical markings, mating is capable of recombination of genes even on the level of individual properties of neurons and neural connections inside morphological nodes of a robot. This is not possible with other methods such as crossover, grafting [9] and picking [8].

4.2 Speciation and Compatibility Measure

Mechanism of speciation used in the proposed algorithm is the same as in NEAT (see subsection 2.2). Method of explicit fitness sharing uses compatibility distance measure to assign robots to species. Therefore, for speciation to work with robots, compatibility distance of two robots needs to be defined.

Robots are represented by a complex data structure composed of a directed graph representing the morphology, with each node containing a local controller which is, again, a directed graph. Measuring similarity of two robots represented by graphs with different topologies without any additional information is a challenging task. However, hierarchical historical markings offer an easy and efficient solution. Since the hierarchical markings provide a correspondence between structural elements of two morphological graphs, the compatibility distance can also be computed hierarchically.

The similarity of two robots is computed using a hierarchy of compatibility distance measures (CDMs): δ for robots; δ_n, δ_c for morphological nodes and connections, δ_a for neural networks, δ_{an} for neurons and δ_{ac} for neural connections. The range of all CDMs is $[0, 1]$. This constraint is not necessary, but simplifies visualization and hierarchical composition of CDMs. CDMs are constructed

hierarchically, down to the level of primitive properties (represented by a real number, boolean value, etc.). CDM for these primitive objects is defined so that they are zero if their arguments are the same and one if their arguments are entirely different (valid ranges of all values are known in advance).

The CDM δ between two robots is computed as follows:

$$A = \sum_{(a_1,a_2)\in N_m} \delta_n(a_1, a_2), \qquad B = \sum_{(b_1,b_2)\in C_m} \delta_c(b_1, b_2),$$

$$\delta = \frac{|N_n| + |C_n| + A + B}{|N_n| + |N_m| + |C_n| + |C_m|},$$

where N_m, C_m are sets of all pairs of matching nodes and connections, respectively and N_n, C_n are sets of all non-matching nodes and connections from both parents, respectively. In the case of small morphological graphs, the denominator can be replaced by a constant.

CDM of two morphological nodes δ_n is computed as a weighted average of CDMs of their recursive limits, geometric shapes, physical joint types and local neural controllers (δ_a). Similarly, CDM of two morphological connections is computed as a weighted average of CDMs of their anchoring coordinates, three angles of rotation, the scale factor, the terminal flag, the recursive limit and three reflection flags. The influence of individual properties on the CDM is controlled by a set of weights: w_s for properties affecting the morphological structure (recursive limits, terminal and reflection flags), w_m for other morphological properties (such as scale, rotation, shape, joint-type) and w_c for the control system. Values of 2.0, 1.0 and 1.0 have been used for w_s, w_m and w_c respectively in this work (giving higher importance to the morphology structure).

The CDM δ_a between two neural networks is computed using the formula for morphology δ, substituting δ_{an} for δ_n and δ_{ac} for δ_c. CDM for neurons (δ_{an}) is one if they have the same transfer function and zero otherwise. For neural connections (δ_{ac}) it is an average of compatibility distances of their weights.

Compatibility distance defined in this manner is capable of measuring differences on a very fine level of detail. This allows species to form not only according to the differences in their morphology, but also according to the differences in their neural networks.

4.3 Minimizing Dimensionality

Starting from minimal structure has been shown to increase the performance of the evolutionary search in NEAT algorithm (see subsection 2.3 for details). The same approach has been taken here. The initial population is filled with randomly generated robots with small number of nodes in the genotype (size of the generated robots is further discussed in subsection 5.2). Neural network inside each morphological node starts without neurons, with the inputs randomly connected to the outputs of the network.

5 Experiments and Results

This section presents two sets of experiments conducted with hNEAT. First set compares the performance of hNEAT and the performance of a standard genetic algorithm with grafting and crossover [9]. The second set shows how each component of hNEAT contributes to the performance of the search.

In order to compare performance of different configurations of evolution, *the winning fitness* is first defined for each fitness function. Algorithms are compared based on the number of fitness evaluations necessary to find an robot with at least the winning fitness value. Failed validity tests (as described in subsection 3.3) are also included in the number of fitness evaluations. Fitness values for all four fitness functions used for experiments in this section have been chosen so that the large majority of the evolutionary runs are able to find the winner within the first 30000 evaluations. In case when the search fails to find the winner, the number of evaluations of 30000 is used. Fitness functions for light following, swimming, walking and jumping have been defined as proposed by Sims [9].

Population size of 300 was used. New population in standard GA was composed of one champion from the previous population, 75% robots created by mating (half by crossover and half by grafting) and the rest created by mutation. Each offspring had an 80% probability of being mutated. Surviving robots have been selected uniformly from the elite 20% robots of the previous population. The same setup has been used for populating species in hNEAT algorithm except for the mating method – recombination based on historical markings was used instead of grafting and crossover.

Each configuration of the evolutionary algorithm was tested 30 times for 100 generations. A single evolution took on average 70 minutes to finish using 7 PC computers. Computation of all experiments has taken more than 8 months of CPU time. Using distributed computation on 70 computers (divided into groups of 7), all experiments have been computed in under a week of real time.

All p-values have been computed using Welch two sample t-test for unequal variances with significance level of 5%.

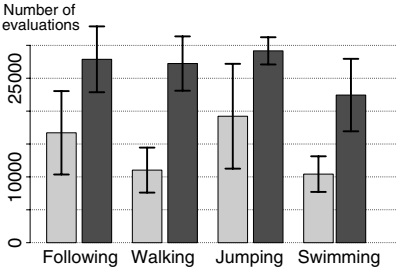
5.1 Measuring Performance

The goal of experiments presented in this section is to show whether hNEAT improves the performance of the evolutionary search compared to the standard GA. Four sets of experiments have been performed, each set with one of the following fitness functions: light following, walking, jumping and swimming. For each fitness function, two algorithms have been tested:

1. **hNEAT algorithm.** The initial population has been generated randomly; each robot having three nodes and three connections.
2. **Standard GA with grafting and crossover.** Speciation and historical markings have been disabled. Grafting and crossover have been used as mating methods. Otherwise, all parameters (including the initial population) have been set to the same values as in hNEAT.

Table 1. Comparison of hNEAT and the standard GA. Columns show mean number of fitness evaluations along with the standard deviation and the percentage of failed evolutions. Standard GA is represented by dark color and hNEAT by light color.

Behavior	Config.	Mean	Std	Failed
Following	hNEAT	16704.53	6339.75	8%
	GA	27873.94	5002.57	72%
Walking	hNEAT	11033.09	3419.77	0%
	GA	27230.64	4128.62	63%
Jumping	hNEAT	19221.51	7966.50	22%
	GA	29159.23	2068.91	80%
Swimming	hNEAT	10425.67	2707.13	0%
	GA	22438.95	5507.82	16%



Results of the performed experiments are presented in Table 1. Standard GA with grafting and crossover is outperformed by hNEAT in all tested scenarios ($p < 10^{-7}$). hNEAT is faster than standard GA by the factors of 1.67, 2.47, 1.52 and 2.15 respectively and also fails much less frequently. A large diversity of successful robots has been discovered for each fitness function. Examples of robots evolved using hNEAT are shown in Figure 3.

5.2 Ablation Experiments

Ablation experiments are designed to show the contribution of individual hNEAT components to the overall performance of the algorithm:

1. **Non-mating hNEAT.** Mating is disabled. Otherwise as hNEAT.
2. **Non-speciated hNEAT.** Speciation is disabled. Otherwise as hNEAT.

3.-5. **hNEAT with randomly created initial population.** Three different configurations have been tested, each with different size of robots in the initial population (robots were generated with the same number of nodes and connections – 2, 3 and 4).

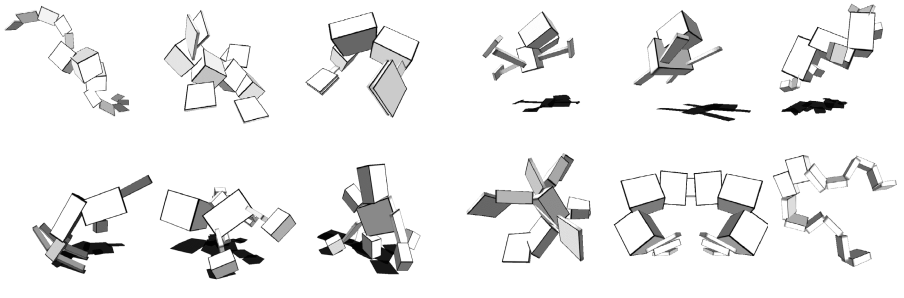


Fig. 3. Examples of robots evolved using the proposed algorithm for light following (top left), jumping (top right), walking (bottom left) and swimming (bottom right)

Table 2. Results of ablation experiments. Second and third column show mean and standard deviation for the number of fitness evaluations performed until the winner has been found. Fourth column shows the percentage of failed evolutions.

Configuration	Mean	Std	Failed
Full hNEAT	16704.53	6339.75	8%
Non-mating hNEAT	22441.38	6146.31	20%
Non-speciated hNEAT	20456.40	6313.65	20%
Random - 2 nodes	11503.77	5871.19	3%
Random - 3 nodes	16704.53	6339.75	8%
Random - 4 nodes	29920.69	387.05	93%
Uniform - 2 nodes	16811.39	8190.81	21%

6. **hNEAT with uniform initial population.** All robots in the initial population are copies of a single hand-designed robot with two nodes and one connection.

Because of the high demand on the computational resources, all of the ablation experiments were carried out using only the light following fitness function. Fitness function for the light following behavior has been chosen, because it is the only fitness function, which explicitly requires robots to use their sensory input and to evolve their neural network controllers.

Results of ablation experiments are presented in Table 2. Full hNEAT is 1.34 times faster in finding the winner and fails less frequently than non-mating hNEAT (rows 1 and 2). Difference is statistically significant ($p < 0.0001$). This confirms the positive effect of recombination based on historical markings.

Experiments with various methods of generating the initial population have shown, that starting with small randomly generated robots increases the performance of the search. When evolution starts with population of robots with two body parts, random initialization (row 4) is 1.46 times faster than uniform population (row 7; $p < 0.005$). Experiments with random initialization show that the performance of the search decreases with increasing size of generated robots (rows 4-6; $p < 0.001$ in all cases). This fact supports the hypothesis that starting minimally helps the search by minimizing the dimensionality of the search space[1].

In summary, it can be concluded that starting the evolution with small randomly generated robots is more beneficent than starting either with hand-designed uniform population or with a population of robots with larger genomes.

The positive effect of speciation has also been confirmed. Results of experiments show, that full NEAT is 1.22 times faster than non-speciated NEAT (rows 1 and 3; $p < 0.02$). The failure rate is also significantly higher in the non-speciated version of the algorithm.

[1] Experiments show that generating robots with three nodes and three connections in hNEAT was suboptimal. This, however, does not affect conclusions of the experiments.

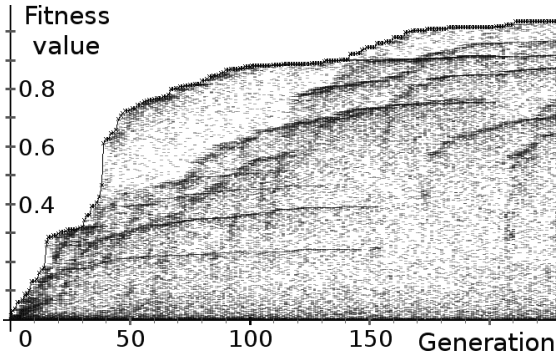


Fig. 4. Fitness values of all robots in an evolution. Robots were evolved for light-following using hNEAT. Threads of black color correspond to individual species.

6 Discussion

The proposed algorithm performs better than the standard GA consistently over all four fitness functions. Moreover, ablation experiments have shown, that each component improves its performance. There are two main reasons for this.

First, speciation in hNEAT increases the robustness of the search. With standard GA, maximum fitness value of a population often gets stuck in a local optimum. In hNEAT, however, if one species stops improving, another species can take its place. For example, evolution in Figure 4 is dominated by a single species from generation 30 to generation 140. This species starts to stagnate around generation 100. As a result of that, it is outperformed by another species in generation 140 (and by a different one in generation 170).

Second, recombination in hNEAT uses information offered by historical markings to find correspondences between parts of different robots. Therefore, advantageous features discovered by different robots can be better combined in their offspring. Since structural elements in different robots with the same value of historical marking are descendants of the common ancestral element, it is very likely that both elements serve the same purpose in both parents. Therefore, it is reasonable to expect that the recombination of their internal properties mixes genetic information without introducing destructive changes. This is a major difference from operators such as grafting, crossover and picking, which often generate an invalid offspring.

7 Future Works and Conclusions

The next direction of research is to construct robots evolved using hNEAT in reality. Morphology of evolving robots can be constrained to the parameters of available hardware components (e.g. fixed-size body parts, servo-joints). This way, all evolved robots would be constructible in reality.

The contribution presented in this paper is the proposal of a novel algorithm for the evolution of autonomous robots. The proposed algorithm is inspired by NEAT – an algorithm for the evolution of neural networks. Large-scale distributed experiments (taking over 8 months of CPU time) conducted to measure various properties of the algorithm have shown that the algorithm significantly increases the performance of the evolution on the tasks of swimming, walking, jumping and light-following. Ablation experiments have shown, that each individual component of hNEAT improves the search performance. For more information about the project and examples of evolved robots, visit http://ero.matfyz.cz.

Acknowledgments

This work is supported by the Grant Agency of Charles University in Prague under Grant-No.358/2006/A-INF/MFF.

References

1. Bongard, L.H., Zykov, J., Resilient, V.: machines through continuous self-modeling. Science 314(5802), 1118–1121 (2006)
2. Goldberg, D.E., Richardson, J.: Genetic algorithms with sharing for multimodal function optimization. In: Proceedings of the Second International Conference on Genetic Algorithms on Genetic algorithms and their application, pp. 41–49. Lawrence Erlbaum Associates, Inc, Mahwah (1987)
3. Hornby, G.S., Lipson, H., Pollack, J.B.: Evolution of generative design systems for modular physical robots. In: IEEE International Conference on Robotics and Automation (ICRA) (2001)
4. Hornby, G.S., Pollack, J.B.: Body-brain co-evolution using L-systems as a generative encoding. In: Proceedings of the Genetic and Evolutionary Computation Conference (GECCO 2001), pp. 868–875. Morgan Kaufmann, San Francisco (2001)
5. Krcah, P.: Evolutionary development of robotic organisms. Master's thesis, Charles University in Prague (September 2007)
6. Krcah, P.: Towards efficient evolutionary design of autonomous robots. Technical Report 2008/4, Charles University in Prague (2008)
7. Lipson, H., Pollack, J.B.: Automatic design and manufacture of robotic lifeforms. Nature 406, 974–978 (2000)
8. Miconi, T., Channon, A.: An improved system for artificial creatures evolution. In: Proceedings of the 10th conference on the simulation and synthesis of living systems (ALIFE X). MIT Press, Bloomington (2006)
9. Sims, K.: Evolving virtual creatures. In: SIGGRAPH 1994: Proceedings of the 21st annual conference on Computer graphics and interactive techniques, pp. 15–22. ACM Press, New York (1994)
10. Smith, R.: ODE manual, `http://www.ode.org`
11. Stanley, K.O., Miikkulainen, R.: Efficient reinforcement learning through evolving neural network topologies. In: Proceedings of the Genetic and Evolutionary Computation Conference (GECCO 2002). Morgan Kaufmann, San Francisco (2002)