Honda Research Institute EU
Honda Research Institute Europe GmbH
https://www.honda-ri.de/

An examination of different fitness and novelty based selection methods for the evolution of neural networks

Benjamin Inden, Yaochu Jin, Robert Haschke, Helge Ritter, Bernhard Sendhoff

2013

Preprint:

This is an accepted article published in Softcomputing. The final authenticated version is available online at: https://doi.org/[DOI not available]

Noname manuscript No.
(will be inserted by the editor)

An examination of different fitness and novelty based selection methods for the evolution of neural networks

Benjamin Inden · Yaochu Jin · Robert Haschke · Helge Ritter · Bernhard Sendhoff

the date of receipt and acceptance should be inserted later

Abstract It has been suggested recently that it is a reasonable abstraction of evolutionary processes to use evolutionary algorithms that select individuals based on the novelty of their behavior instead of their fitness. Here we study the performance of fitness- and novelty based search on several neuroevolution tasks. We also propose several new algorithms that select both for fit and for novel individuals, but without weighting these two criteria directly against each other. We find that behavioral speciation, behavioral near neutral speciation, and behavioral novelty speciation perform best on most tasks. Pure novelty search, as well as a number of hybrid methods without speciation mechanism, do not perform well on most tasks. Using behavioral criteria for speciation often yields better results than using genetic criteria.

Keywords Neuroevolution; selection; novelty search; evolutionary robotics; NEAT.

1 Introduction

1.1 The role of natural selection in evolutionary biology

The idea that species were not immutable but somehow connected by a process of evolution was already present before Charles Darwin presented his theories in 1859. Perhaps the most important distinguishing feature of his theory of evolution was the provision of a mechanism for evolution: natural selection. Although Darwin was careful not to try to explain all features of organisms as adaptations formed by natural

B. Inden
Research Institute for Cognition and Robotics, Bielefeld University, Universitätsstr. 25, 33615 Bielefeld, Germany
E-mail: binden@techfak.uni-bielefeld.de

Y. Jin
Department of Computing, University of Surrey, United Kingdom

R. Haschke · H. Ritter
Neuroinformatics Group, Bielefeld University, Germany

B. Sendhoff
Honda Research Institute Europe, Offenbach/Main, Germany

selection, there was a tendency in later work to consider only adaptation as possible explanation. Against this "adaptionist programme", various authors have pointed out that, among others, the following explanations must also be considered (Gould and Lewontin, 1979; Kauffman, 1993; Barton and Partridge, 2000; Lynch, 2007):

- A trait could be a byproduct of another trait that has selective advantages. As a special case, the two traits could be linked only at certain phases during the lifetime of an individual, in particular during development (ontogeny). So the inadequateness of adaptionist explanations can be rooted in the failure of selection to overcome internal constraints of the organism.
- A trait could have been fixed in a population by random genetic drift. In that case, it could be selectively neutral or even slightly deleterious. So the inadequateness of adaptionist explanations can be rooted in the failure of selection to compensate for mutations and genetic drift. A factor contributing to this is the isolation of subpopulations from each other. If speciation often occurs in small founder populations, these effects become particularly strong.
- A trait could be a result of physical forces or properties of biological materials. So the inadequateness of adaptionist explanations can be rooted in the failure of selection to overcome external constraints of the environment.

These insights are a part of mainstream evolutionary biology today, although it is still debated how much each of the respective factors can contribute to evolutionary explanations (Futuyma, 2005). Taken together, they imply that selection is not the only force that accounts for the diversity of life within evolutionary biology. In particular, it has been pointed out that selection is not necessarily a good explanation for the increase in maximal complexity that has taken place during the history of life on earth. On the contrary, strong selection can optimize existing function at the expense of any unnecessary features, leading to a decrease of complexity. The reduction of eyes in cave dwelling organisms and the absence of some organs in parasitic species are commonly cited as examples for such a decrease in complexity.

1.2 The role of selection in evolutionary algorithms

Necessarily, evolutionary algorithms have to strongly simplify the mechanisms of evolution to be implementable and runnable on today's serial or weakly parallel computers. Besides, computer scientists are not always familiar with the intricacies of evolutionary theories. As a result, the early evolutionary algorithms have the following characteristics:

- They work on simple representations that do not provide for the evolution of hierarchical structure in the genome (and corresponding mutation operators).
- The fitness of individuals is determined by computing a predefined, unchanging function that takes the phenotype (or behavior, if an environment is defined separately) as input.
- That fitness then determines how much offspring an individual will have, so selection is the dominant force in evolution (some selection methods like tournament selection include some amount of drift, but this is often viewed as an accidental or less desirable feature).

The first item in the list is less true for a number of later evolutionary algorithms. For example, many genetic programming methods allow the evolution of modular building

blocks in the form of subfunctions (Poli et al, 2008). Analog genetic encoding is an example of a method that imitates the way genes and regulatory elements are embedded into DNA to define a structure in an implicit way (Mattiussi and Floreano, 2007). Several methods include artificial ontogeny or abstractions thereof to allow for the evolution of complex genetic architecture (Stanley and Miikkulainen, 2003; Stanley, 2007; Harding and Banzhaf, 2008).

The second item in the list is still true in most cases and remains a serious challenge. Incremental evolution and coevolution have been explored as alternative approaches (Gomez and Miikkulainen, 1997; Nolfi and Floreano, 2000; Buason et al, 2005; Mouret and Doncieux, 2008; Miconi, 2009), but have their own intricacies and are not often used in practice, let alone in a systematic way.

Here we focus on the third issue — complementing selection with other methods in evolutionary algorithms to arrive at the next generation. Pure fitness based selection often leads to the problem of premature convergence. This means that diversity in the population gets lost, and with it the potential for future innovation. So a progression towards ever more complex solutions is rarely observed in artificial evolution.

Various measures have been taken to complement selection with other evolutionary forces. The possible exploitation of external constraints by artificial evolution has been suggested (Banzhaf et al, 2006), but has not been extensively explored in evolutionary algorithms. Isolation, on the other hand, has been frequently employed. In parallel evolutionary algorithms (Alba and Troya, 2002; Tomassini, 2005), evolution proceeds independently in several subpopulations, between which there is a regular exchange of individuals according to some rules and the spatial arrangement of the subpopulations. Of course, this exchange reduces diversity again and often leads to the eventual convergence of the subpopulations. So there is a trade-off between protection of diversity and the speed of evolution, which relies on the spread of beneficial mutations, in parallel evolutionary algorithms. Niching methods (Goldberg and Richardson, 1987; Darwen and Yao, 1995; Sareni and Krähenbühl, 1998) reduce the fitness of individuals that are in crowded areas of the genotype or phenotype space. How much to reduce the fitness is not always obvious because if there is a high fitness to gain in that local area, this may still be enough to attract the whole population. Another issue with niching techniques is that the reduction in fitness may prevent individuals that are close to the optimum from converging to it because they would mutually reduce their fitness. Speciation as has been used in neuroevolution previously (Stanley and Miikkulainen, 2002) increases diversity in the population by assigning similar individuals to one species. The selection pressure within a species is strong, while selection between species is much weaker (typically fitness-proportional selection that is based on the mean fitness of the species is used). This method is superior to standard selection methods on a number of neuroevolution problems, but it is still dependent on the absolute fitness differences: If they are too large, fitter individuals can still drive other lineages to extinction, whereas if they are too small, progress towards fitter individuals will be very slow.

More recently, novelty search (Lehman and Stanley, 2008) was introduced as a method that can avoid the problem of convergence altogether. The basic idea is to select not the fittest individuals, but those with the most novel behavior. It is assumed that ultimately there will be an individual with the desired behavior among them. Novelty is measured as the average distance to the k-nearest neighbors. A domain specific measure of distance in behavior space is therefore required.

Later work has extended novelty search in several ways. One extension was to prune the search space by setting minimal criteria that must be fulfilled by all individuals

for selection (Lehman and Stanley, 2010). Another was to use a multiobjective evolutionary algorithm that uses both fitness and novelty (and possibly also diversity of the population) as criteria (Mouret, 2009). A weighted sum of fitness and novelty was also used in combination with the crowding method for solving a deceptive problem (Cuccu and Gomez, 2011). Novelty has been used as a criterion for restarting after convergence in a special kind of evolution strategy (Cuccu et al, 2011). Recently, a multiobjective algorithm has been used to reward novelty, genotypic diversity, and fitness relative to nearest neighbors (Lehman and Stanley, 2011). In this article, we complement previous research by exploring how a number of straightforward extensions and hybridizations of fitness and novelty based search perform in comparison to the basic techniques.

1.3 Neuroevolution

There are a number of methods to evolve both the topology and weights of artificial neural networks (Yao, 1999; Floreano et al, 2008). These artificial neural networks are then often used to control robots or simulated agents. While neural networks have a number of desirable properties, the mapping from weights to behavior is highly nonlinear, which means that there will be many local optima onto which evolution is likely to converge. In addition, some methods encode connection weights indirectly into the genome, which makes generating large neural networks from small genomes possible. That way, the search spaces becomes smaller, but also more rugged. For these reasons, it can be expected that neuroevolution benefits from going beyond mere fitness-based evolutionary algorithms as outlined in the previous section.

Here we will use the NEATfields neuroevolution method, which is an extension of the well known NEAT (Stanley and Miikkulainen, 2002; Stanley, 2004) neuroevolution method. It uses a particular kind of indirect encoding to solve problems with large input and output spaces. NEATfields networks have three levels of architecture. The highest level is a NEAT-like network of neural fields. The intermediate level is a field of identical subnetworks, called field elements, with a two-dimensional topology. The lowest level is a NEAT-like subnetwork of neurons. The topology and connection weights of these networks are evolved with methods derived from the NEAT method. In addition, evolution is provided with further building blocks to enable information flow between field elements, and to dehomogenize neural fields. We have previously shown that the NEATfields method can solve a number of high dimensional pattern recognition and control problems (Inden et al, 2012, 2011b,a). Here we compare the performance of NEATfields on a number of tasks using hybrid selection methods with the performance achieved using standard fitness based selection methods.

2 Methods

We will first briefly describe the NEAT method in section 2.1, and then our NEATfields method which is based on NEAT. In section 2.3, we will describe the different selection methods compared by us.

2.1 The NEAT neuroevolution method

The NEAT method (Stanley and Miikkulainen, 2002; Stanley, 2004) is a well known method for simultaneously evolving the topology and the connection weights of neural networks. It starts evolution with the simplest possible network topology and proceeds by complexification: There is a mutation operator that adds neurons (always between two connected neurons), and adjusts the connection weights such that the properties of these connections changes as little as possible. Another operator adds a connection between two previously unconnected neurons.

Once a new gene arises by mutation, it receives a globally unique reference number. These numbers are used to align two genomes during the process of recombination. They are also used to define a distance measure between networks, as will be explained in section 2.3.2.

Like NEAT, the NEATfields method used here also starts with the smallest possible topology for a given task. If all higher-level mutation operators are switched off, the higher-level structure will remain trivial, and NEATfields will reduce to a NEAT implementation. We use this reduced version in two simple experiments. The details of its mutation operators, as well as the parameter settings, are the same as those given for NEATfields in the following.

2.2 The NEATfields method

2.2.1 General approach

The NEATfields method (Inden et al, 2010, 2012) has been designed to extend the principles and abilities of NEAT to the evolution of much larger neural networks. The assumption that the input and output spaces of a task can largely be decomposed into a number of equal or similar subspaces is built into NEATfields. Many real-world tasks indeed require one or two dimensional fields of networks that do the same or similar calculations. For example, an eye or a camera provides large amounts of sensory data with a natural two-dimensional topology. Also, robots with actuated limbs often require a number of similar controllers in addition to a coordinating mechanism.

NEATfields networks have three levels of architecture (see figure 1). At the highest level, a network consists of a number of fields that are connected just like individual neurons are in a NEAT network. At the intermediate level, fields are collections of identical (or similar) subnetworks with a two dimensional topology. At the lowest level, these subnetworks, or field elements, are NEAT networks of individual neurons. In accordance with the idea of searching in smaller topologies first, NEATfields usually starts evolution with a single field of size 1×1.

2.2.2 Neural networks

The activation of the neurons in NEATfields is a weighted sum of the outputs of the neurons $j \in J$ to which they are connected, and a sigmoid function is applied on the activation: $o_i(t) = \tanh(\sum_{j \in J} w_{ij} o_j(t-1))$. Like in some other NEAT implementations, connection weights are constrained to the range $[-3, 3]$. There is no explicit threshold value for the neurons. Instead, a constant bias input is available in all networks.

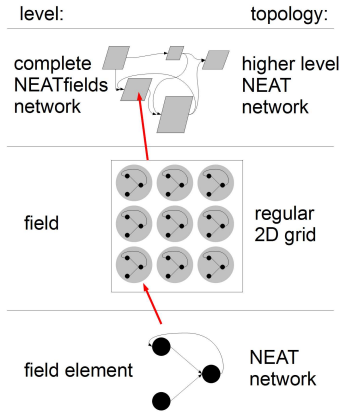


Fig. 1 The three levels of architecture in NEATfields.

```
3       number of inputs / outputs
0       '0' for input / '1' for output
16      field size in x dimension
16      field size in y dimension
1001  }   globally unique input /
1002  }   output identification
1003  }   numbers
```

Fig. 2 For construction or mutation of a network, NEATfields gets information about input and output geometry from the task. Here the specification of a single input field is shown as an example. Specifications for several fields can simply be concatenated.

A field element in NEATfields is a recurrent neural network with almost arbitrary topology, where almost arbitrary means the used operators will ensure that no disconnected neurons exist and that there exists only one connection at most between all pairs of neurons. A field is a two-dimensional array of field elements. In special cases, the field size along one or both dimensions is 1. A complete NEATfields network is a NEAT-like network where the nodes are fields. It consists of at least one internal field (internal fields are specified by the genome, see the next section), and fields for network input and output (these are specified by the given task, see figure 2). There can be several input and output fields with different dimensions. For example, a bias input can be provided as an input field of size 1×1. Within the NEATfields network, connections can be local (within a field element), lateral (between field elements of the same field), or global (between two fields). It should be noted that connections between field elements or fields are in fact connections between individual neurons in these field elements or fields. How they are established will be described below.

2.2.3 Encoding connections and other network elements in the genome

The parameters for an individual field are encoded in the genome on a corresponding chromosome (see figure 3). The first gene in a chromosome specifies the field size in

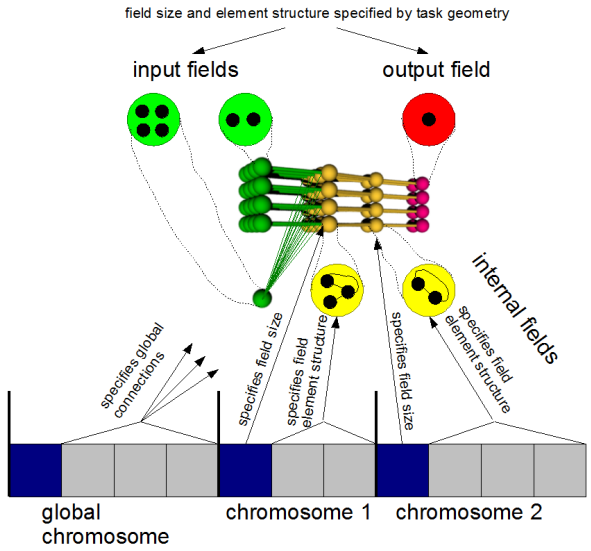


Fig. 3 Construction of a NEATfields network from its genome (bottom). The balls in the central network represent field elements. Their contents are shown in the circles above and below. In these circles, black dots represent the individual (input, output or hidden) neurons.

x and y dimensions. After that node and connection genes for one field element are specified. In the current implementation, all genes have a length of 160 bit and may contain several numerical parameters and flags as well as unused space for extensions. However, each flag or numerical value within the gene basically has its own specialized mutation operator, so the genes could equally well be implemented as any kind of data structure as long as there are enough bits available for sufficient numerical accuracy. All genes contain a unique reference number that is assigned once the gene is generated through a mutation. In addition, connection genes contain a connection weight, a flag indicating whether the connection is active, and the reference numbers of the source and target neurons (as well as additional data that is explained below). Node genes contain additional values that are not used for the experiments described in this article.

There is a special chromosome that contains genes coding for global connections. Global connections contain reference numbers of a source and a target. These can be reference numbers either of neurons or of inputs and outputs as specified by the task description. They must be in different fields — global connections between two neurons in the same field are never created by the NEATfields method. Due to the higher level architecture of NEATfields, a neuron with a given reference number will be present n times in a field with n field elements. A single global connection gene implicitly specifies connections for all these neurons. If a global connection is between neurons in fields with the same sizes, every field element in the target field will get a connection from the field element in the source field that has the same relative position (in x and y dimension) in its field. Their connection weights are all the same because they are all derived from one gene. If field sizes are different in a dimension, then the fields will still be connected using a deterministic and topology preserving method (see figure 4): if the source field is smaller than the target field, each source field neuron projects to several adjacent target field neurons, whereas if the source field is larger than the

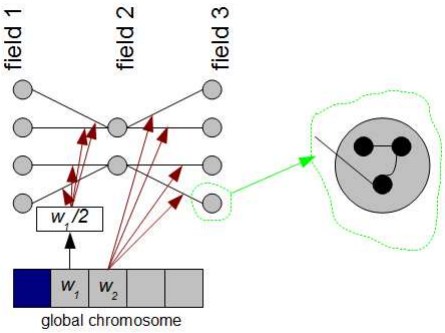


Fig. 4 An example of how the global connections between neurons in fields of different sizes (shown here as one dimensional) are created using a deterministic and topology preserving method. The genetically specified weights are automatically scaled if necessary. As shown in detail on the right side, connections go in fact to individual neurons within the field elements as specified by the connection gene.

target field, the target field neurons get input from a number of adjacent source field neurons, while the genetically specified connection weight is divided by that number. That way, field sizes can mutate without changes in the expected input signal strength.

2.2.4 Mutation operators for the field element networks

The NEATfields method uses mutation operators that are very similar to those of the original NEAT implementation for evolving the contents of the field elements. The probabilities of these operators are also chosen based on experience with NEAT and the derivative NEON method (Inden, 2008; Stanley, 2004). The most common operation is to choose a fraction of connection weights and either perturb them using a normal distribution with standard deviation 0.18, or (with a probability of 0.15) set them to a new value. The application probability of this weight changing operator is set to 1.0 minus the probabilities of all structural mutation operators, which amounts to between 0.8815 and 0.949 in the experiments reported here. In general, structural mutations are applied rarely because they will cause the evolutionary process to operate on larger neural networks and search spaces. A structural mutation operator to connect neurons is used with probability 0.02, while an operator to insert neurons is used with probability 0.001. The latter inserts a new neuron between two connected neurons. The weight of the incoming connection to the new neuron is set to 1.0, while the weight of the outgoing connection keeps the original value. The existing connection is deactivated but retained in the genome where it might be reactivated by further mutations. There are two operators that can achieve this: one toggles the active flag of a connection and the other sets the flag to 1. Both are used with probability 0.01.

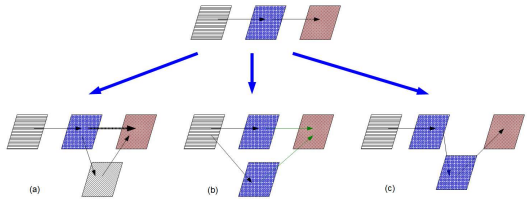


Fig. 5 Three methods to create a new field in NEATfields. (a) A completely new field (with one neuron per field element) is inserted into an existing connection. The existing connection is deactivated. This is the equivalent of the split/insert neuron mutation of NEAT on the level of global network topology. (b) A field is duplicated, and the copy is made parallel to he original, and the outgoing connection weights are halved. (c) A field is duplicated, and the copy is inserted after the original. Respective neurons in respective field elements are connected between the two copies. Note that in all cases, the connections displayed here are in fact sets of connections specified by one global connection gene.

2.2.5 Evolving network topology on higher levels

For evolving higher level topology, NEATfields introduces some new operators. At the level of a single field, one operator doubles the field size along one dimension (with a probability of 0.001) and another changes the size (for both dimensions independently) to a random value between its current size and the size of the largest field it is connected to (with a probability of 0.005).

At the level of the complete NEATfields network, there is an operator that inserts global connections (with a probability of 0.01) and an operator that inserts a new field into an existing global connection (with a probability of 0.0005). The size of the new field is set randomly to some value between 1 and the larger of the sizes of the two fields between which it is inserted. This is done independently for both dimensions. These two operators correspond to operators already used to evolve the topology of the individual field elements. In addition, an existing field can also be duplicated, where all elements of the new field receive new reference numbers. The new field can either be inserted parallel to the old one (in this case, the outgoing connection weights will be halved to prevent disruption of any existing function) or in series with the old one (in this case, every neuron in every field element in the new field gets input from the corresponding neuron in the corresponding field element in the old field, while the output from the new field goes to where the output from the old field went previously). These two operators allow for reuse of previously evolved structure. The serial duplication operator is also applied on input fields, in which case an internal field is created that contains one neuron for every input in the input field. Both mutations occur with probability 0.0005. The three possible methods of creating a new field in NEATfields are depicted in figure 5.

2.2.6 Flow of information within neural fields

NEATfields networks can also have lateral connections between field elements of the same field. These enable flow of information within the neural field. Lateral connections are like local connections: they are between two neurons in the NEAT network that describes the field elements. However, the connection from the source neuron does not

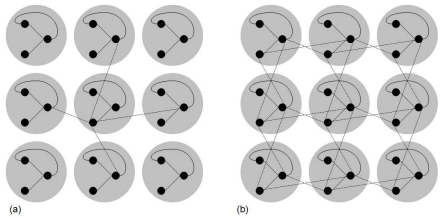


Fig. 6 Lateral connections are established between a neuron in a field element and another neuron in each of the up to four neighbor field elements. There are less neighbors if it is at the border of the field. (a) Lateral connections are only shown for the central element as dotted lines here for clarity. (b) All lateral connections constructed from a single gene.

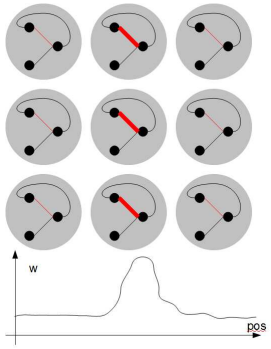


Fig. 7 Dehomogenization of neural fields by the focal areas technique. The thickness of connections here symbolizes their weights.

go to a neuron in the same field element, but to the corresponding neurons in the up to four neighbor field elements instead (see figure 6). The gene coding for a lateral connection specifies source and target neuron reference numbers just as genes coding for local connections do; it is also located in the same chromosome. However, it has a lateral flag set to 1, and is created by a lateral connect operator (with a probability of 0.02).

2.2.7 Dehomogenizing neural fields

By default, corresponding connections in different field elements all have the same strength so they can be represented by one gene. The same is true for the global connections between field elements of two fields. For some tasks, it is useful to have field elements that react slightly differently to input, which can create what has been called "repetition with variation" (Stanley, 2007). One way to realize this is to have larger connection weights in a neighborhood of some center coordinates on the field. Here, connection weights are scaled according to $\exp(-\epsilon(\frac{distance}{field\ size})^2)$ (in our implementation, this is done separately for the x and y dimensions), where $\epsilon = 5.0$ is chosen such that connections close to the center have a large weight and the rest have weak weights (see figure 7). The center coordinates are specified in the following way: There are

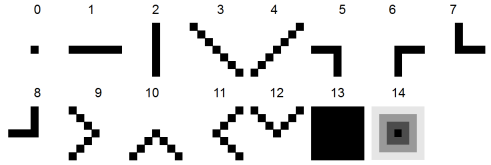


Fig. 8 Feature patterns for use with the NEATfields method. Black denotes a connection weight of 1.0, white a connection weight of -1.0, and the gray scales denote connection weights between 0.0 and 1.0.

two eight bit values in the gene encoding a connection, one for each dimension. These are converted to two numbers between -1 and 1. If a number is between -0.15 and 0.15, the connection weights are homogeneous in the corresponding dimension. This is the default for all connections because they are usually created with these values set to 0.0. If a value is outside this range, on the other hand, then it is mapped linearly to a position between the two field borders. There is a mutation operator that (at a probability of 0.03) sets the values for a single connection gene.

2.2.8 Building local feature detectors

NEATfields also has another kind of global connections that serve to detect local patterns by connecting to a number of adjacent field elements in another field. For example, if the input field contains camera data, they can detect lines or edges in particular orientations. These global connections have an additional number on the connection gene that refers to one of at most 16 prespecified connection weight patterns. These patterns have been designed by hand with the aim of providing useful building blocks for local feature detection (see figure 8). A value from the pattern is multiplied by the original connection weight specified in the gene to get the weight of the connection from the field element at the respective position. The patterns have a size of 7×7. The central element of the pattern corresponds to the weight modification of the connection between exactly corresponding field elements. The feature detectors do not have to use an entire pattern. Instead, the maximal offsets between the field element positions in the source and target fields can be between 0 and 3 separately for the x and y dimensions, and are also specified on the genome. There is a mutation operator (used with probability 0.02 if local feature detectors are used) that mutates the pattern reference and maximal offsets of an already existing connection. Both in that case and when a new connection is created, choices of pattern use and sizes are made based on probabilities that are externally specified for a given experiment. In the experiments reported below, the default for method one is to use patterns 1 to 4 from figure 8 with equal probabilities. The maximal offsets in x and y direction are 0 or 1 with equal probabilities by default. Comparisons to other parameter settings and a number of alternative approaches can be found in earlier work (Inden et al, 2012).

2.3 Selection methods

2.3.1 Tournament selection

Tournament selection is used here as a baseline for comparisons, and is implemented as described in the literature (De Jong, 2006). A tournament of size 2 is used, and the tournament winner is selected with probability 1.0. For a population of 1000 individuals, an elite consisting of the 10 best individuals is copied into the next generation unchanged before the rest of the space is filled by tournament selection and mutation.

To limit the demand on computational resources, the fitness of networks that exceed absolute limits of the numbers of nodes or connections is set to 0.0. The limits are obviously implementation and hardware specific. Here we set them to 10000 nodes and 100000 connections. This is also done in combination with all other selection methods described below.

2.3.2 Genetic speciation selection

NEAT and NEATfields use speciation selection by default. This method is fitness based, but uses some techniques to protect innovation that may arise during evolution against competition from fitter individuals that are already in the population. The globally unique reference numbers assigned to each gene are used to calculate a distance measure between two genomes, and partition the population into species. The number of offspring assigned to a species is proportional to its mean fitness. This rather weak selection pressure prevents a slightly superior species from taking over the whole population, and enables innovative yet currently inferior solutions to survive. In contrast, the selection pressure between members of the same species is much stronger in NEAT.

Within NEATfields, the dissimilarity between two networks is calculated as follows:

$$d = c_n \#ref_n + c_r \#ref_c + c_w \sum \Delta w \\ + c_f \#ref_f + c_s \sum \log(1 + \Delta s_x + \Delta s_y)$$

where $\#ref_n$ is the number of nodes present in just one of these networks, $\#ref_c$ is the number of connections present in just one of these networks ($\#ref_n$ and $\#ref_c$ are only counted in fields that are present in both networks, otherwise the excess nodes and connections are just ignored), $\#ref_f$ is the number of fields present in just one of these networks, Δw are the connection weight differences (summed over pairs of connections that are present in both networks), the Δs are the field size differences in the x and y dimension (summed over pairs of fields present in both networks), and the c variables are weighting constants. For basic NEAT, some of the variables will be zero as the corresponding features are not present in NEAT.

Using the dissimilarity, the population is partitioned into species by working through the list of individuals. An individual is compared to representative individuals of all species until the dissimilarity between it and a representative is below a certain threshold. It is then assigned to this species. If no compatible species is found, a new species is created and the individual becomes its representative. The number of offspring a species has is proportional to its mean fitness. Inside the species, the worst 60% of its members are deleted, after which uniform selection is used for the rest. Species with an offspring size greater than five also keep their best performing individual. If the

maximum fitness of a species has not increased for more than 200 generations and it is not the species containing the best network, its mean fitness is multiplied by 0.01, which usually results in its extinction. Also, in order to keep the number of species in a specified range, the dissimilarity threshold is adjusted in every generation if necessary.

For the experiments reported here, we use an initial speciation threshold of 4.0, and set $c_n = 0.0$, $c_r = 1.0$, $c_w = 1.0$, $c_f = 1.0$, $c_s = 1.0$. A population of 1000 individuals has a target number of species of between 35 and 45.

2.3.3 Behavioral speciation selection

Behavioral speciation selection works exactly like genetic speciation selection with one exception: The dissimilarity is calculated based on the behavior, not based on the genome. The calculation methods, as well as the speciation thresholds, are task specific and will be explained when the respective tasks are described.

2.3.4 Novelty search

Novelty search has been implemented based on the description in the literature. However, here we use a generation-based variant of the method. The first step in the process of selection is to calculate the pairwise behavioral dissimilarities of all individuals. An approximation of novelty is then calculated for each organism by taking the mean of its 15 lowest dissimilarity values. If an individual is above a task specific novelty threshold, it is copied into an archive. Every 10 generations, it is checked how many individuals have been copied into the archive. If this number is above 3, the threshold is multiplied by 1.05, whereas if it is below 3, the threshold is multiplied by 0.95 (these parameters have been taken from Lehman and Stanley (2008)). When dissimilarities are calculated, individuals from both the current population and the archive are taken into account. For a population of 1000 individuals, an elite of the 10 most novel organisms is copied unchanged, after which the rest of the new generation is chosen by tournament selection using novelty as criterion, and subjected to mutation as described above.

2.3.5 Selection for fitness and novelty

This hybrid method (called "F-N selection" in the following) combines fitness based tournament selection and novelty search. Half of the population is chosen based on fitness, and the other half is chosen based on novelty. Each half has its own elite of size 10.

There is also a variant of this method ("F/N selection") where the novelty of the individuals is calculated in a slightly different way: Only the dissimilarities to the 50% less fit individuals are taken into account. The idea behind that approach is that an individual that is similar in behavior to the currently fittest individuals (perhaps due to common descent) might be disadvantaged because the fittest individuals have so many close relatives due to selection for fitness. F/N selection might remove this disadvantage by not taking the distance to the fittest individuals into account.

2.3.6 Novelty speciation

Novelty speciation is a new hybrid method between novelty search and speciation selection. It proceeds just like behavioral speciation selection. However, the number of offspring a species has is not determined by its mean fitness, but by its novelty. The novelty of a species is defined as the novelty of its representative individual with respect to both the representatives of the other species and the individuals in the archive. It should be noted that this method has two thresholds as important parameters: a dissimilarity threshold for speciation, and a novelty threshold for inclusion in the archive. Both are task specific. All other parameters are like in speciation selection and novelty search.

2.3.7 Speciation with near-neutral interspecies selection

Four variants of this new method have been explored. All work in principle like speciation selection. However, the number of offspring a species has is not dependent on its mean fitness here. It is just the population size divided by the number of species. To prevent the population from being flooded by new species, the maximal number of species that can exist is set such that each species can have the minimum size to use elitism. As this minimal size is 5 (like in standard speciation) and the population size is 1000, it follows that the species maximum is 200. The species are sorted first for higher mean fitness and then for higher maximal fitness. If there are too many, the worst are just truncated. In theory, this should happen rarely because, as described in section 2.3.2, other measures are taken to keep the number of species much lower.

Speciation can either be based on genes or on behavior. In addition, we can either use speciation with variable threshold as described in section 2.3.2, or use a fixed threshold in a way that is closer to the original NEAT implementation. If we use this original method, two measures are taken in order too ensure that the number of species does not grow too much: Species without increase in maximal fitness are practically removed after 20 generations by multiplying their mean fitness with 0.01. Besides, if the maximal fitness of the whole population does not increase for 50 generations, all but the two best species are removed. Compared to speciation with variable threshold, this method removes variation from early generations in favor of variation from later generations (if the threshold is variable, it might have increased so much that innovative individuals from later generation never get the benefit of being assigned into their own species).

So, altogether we have genetic (G-) NN-(near neutral) speciation, behavioral (B-) NN-speciation, genetic NN-FT-(fixed threshold) speciation, and behavioral NN-FT-speciation.

2.3.8 Mixing genetic and behavioral criteria

So far, we have used all selection methods either together with measuring genotype similarity or together with measuring behavioral similarity. In principle, we can also use a weighted sum of these similarities for every selection method. Here, we do this only for speciation selection and F/N selection (this approach is denoted by the prefix G/B- in the following).

3 Experiments

We study the performance of different selection methods on five different tasks. Pole balancing (section 3.1) and maze navigation (section 3.2) do not require large input and output spaces and are therefore tackled with NEATfields reduced to NEAT. For two pattern recognition tasks (section 3.3 and 3.4) and a quadruped walking task (section 3.5), we use the full NEATfields method.

3.1 Pole balancing

Pole balancing tasks are a family of benchmark tasks, in which a nonlinear system has to be controlled and fast reactions are required. We include pole balancing here because it is extensively used as benchmark problem, and some members of this task family are difficult to solve for standard methods like tournament selection.

In the version of pole balancing tackled here, a cart can drive back and forth on a track, and it has to balance two poles mounted on top of it by a hinge joint (the second pole being one tenth the length of the first). Performance is measured by the number of time steps (a maximum is set at 100000) that the cart stays within some distance from its point of origin, and both poles do not deviate from the upright position by more than a certain angle. Neural networks get only the current cart and pole positions as inputs. In addition, a bias input is provided. We use a particular "anti-wiggling" fitness function that punishes wiggling of the poles and tests for generalization to at least 200 out of 625 starting angles. This task has been described in more detail previously (Wieland, 1991; Stanley and Miikkulainen, 2002). A population of 150 individuals is evolved for 1000 generations.

A behavioral distance can be defined as follows: The position of the cart on its track (which can be in the range $[-2.4, 2.4]$) is recorded at time steps 10, 50, 100, 200, 500, 1000, 5000, and 10000 if the poles are still upright then. (The exact timing of the recordings is not expected to be important. No comparisons were done to other choices.) For the two lists of positions belonging to two individuals, a distance is calculated by summing the squares of the position differences at corresponding times. For every recording that is just present in one individual (because the other had already failed before the time of recording), a value of 1.0 is added. The novelty threshold is set to 4.8 based on initial tests on how fast the archive was growing for different settings.

3.2 Maze navigation

This task is similar in principle to the task used for introducing novelty search (Lehman and Stanley, 2008). Like in the original study, it is a deceptive problem. Because the deceptiveness of the problem is immediately visible, it is a good problem to think about going beyond fitness-based search. In this problem, a robot has to reach a goal position in a simple maze. The maze is a discrete grid here of size 256×256 with several walls that obstruct the path to the obstacle (see fig. 9). The robot has four range finders in the four principle directions that yield a graded response to obstacles within a distance of up to 12 positions in their exact direction. It has also four goal finder sensors that yield a graded response if the goal is within a distance of up to 5 points within their respective quadrant. In addition, there is a bias input to the neural

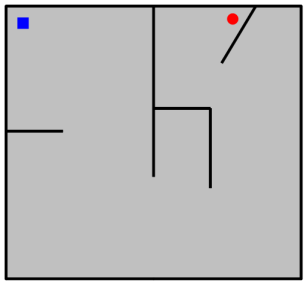


Fig. 9 Maze navigation task. The blue square represents the start, and the red circle the goal.

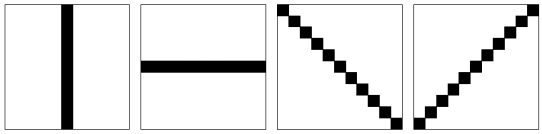


Fig. 10 Input patterns for the "four patterns" task.

network. The network has four outputs. One pair controls movement in the East/West axis: if one output is active (i.e. larger than 0.0) and the other is not, the robot can move one step in the corresponding direction. If both are active, or both are inactive, there is no movement. Another pair controls movement in the North/South axis in the same way. The fitness is $360 - d(r, g)$, where d is Euclidean distance, r is the end position of the robot, and g is the goal position. The robot starts at position $(2, 2)$, and the goal is at position$(190, 5)$. The simulation ends after 1200 time steps. Behavioral distance is calculated as the distance between the end positions of two individuals. The novelty threshold is set to 10.0. With the given fitness function, evolution can easily be trapped in local optima, i.e. in the corners of the maze (Lehman and Stanley, 2008).

3.3 Distinction of four visual patterns

This task is rather simple to solve as a previous study has shown (Inden et al, 2012), but it requires a large neural network to be solved. We use it mainly for comparing the results with those for the more difficult task in the next section. Here, an 11×11 input field is used (in addition, there is a bias input). There are four outputs for classification. The four patterns shown in figure 10 (a) are presented to the network in separate episodes. The output with the highest activation after 20 time steps is considered to be the classification result. The mapping that has to be learned is prespecified, so, for example, the first pattern has to be mapped to the first output. For every episode where classification is correct, the network scores 200 fitness points. For the other episodes, the network scores between 0 and 100 points depending on the difference between the maximally activated output and the output that should have been maximally activated.

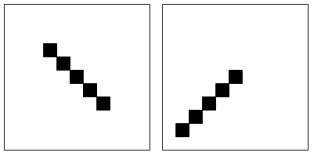


Fig. 11 Example input patterns for the visual discrimination task.

That value v, which can be between 0.0 and 2.0, determines the fitness score according to $f = 100 - 50v$. The behavioral distance between two indviduals is defined as the number of episodes in which a pattern was assigned to different classes. The novelty threshold is set to 0.5, so there are only 16 different classes of behavior.

3.4 Distinction of two visual patterns with variable positions

The task described in this section was found to be much more difficult than the task from the previous section in a previous study (Inden et al, 2012). This study also showed that to achieve good performance on the task, NEATfields features like de-homogenization of neural fields and local feature detectors must be used. However, it seems unlikely that using such features will initially increase the fitness of a neural network, especially if the neural fields that use these features are still smaller than the input field. Therefore, we suspect this task to be quite deceptive. The task provides a 9×9 visual input field. In addition, there is a bias input. There is a single output for classifying the input pattern according to the two possible orientations orientation of a diagonal bar (see fig. 11). The bar can be in any position provided that it fits onto the matrix completely. Therefore, the center of the bar must be at least 2 pixels away from the border of the visual field. This leaves 25 possible positions, so there are 50 episodes altogether. For every episode where classification is correct (either between 0.5 and 1.0 for one orientation, or between -0.5 and -1.0 for the other orientation), the network scores 200 fitness points. For the other episodes, the network scores between 0 and 100 points depending on the distance between the correct and actual output. The distance measure for this task is just the sum of the absolute differences of the output values over all episodes. The novelty threshold is set to 12.5.

3.5 Quadruped walking task

Control of legged robots is another potentially deceptive problem because initially robots that just fall onto the floor might get further from their point of origin than robots that are in the process of evolving regular gaits, and therefore gain higher fitness. We use a task similar to the one reported in Clune et al (2009), where both the motivation for the problem and previous results are described in more detail. Robots with four legs are simulated using the ODE physics engine for 60 seconds using a time step of 0.01 s. The fitness is calculated based on the total distance traveled by a robot

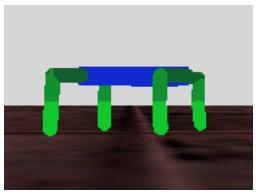


Fig. 12 Robot morphology for the legged locomotion task.

in x and y direction as $(x^2 + y^2)^2$. A trial is aborted if any body segment of the robot other than the four lowest leg segments touches the ground, or if the directions of movement of the joints have been changed more than 960 times altogether. Besides, a trial is aborted if a robot jumps higher than some threshold value because that can only be achieved by exploiting the inaccuracies of the physics simulation. In the latter case, the fitness is set to 0.0. The squared distance between the end positions of two individuals serves as a behavioral distance measure. The novelty threshold is set to 0.001 because in the first few generations differences between the end positions of individuals are very small.

The robot (see Fig. 12) has a rectangular torso of size $0.15 \times 0.3 \times 0.05$ units. Each of its four legs consists of three cylindrical segments of length 0.075 and radius 0.02. They are connected to the body or their preceding segments by hinge joints. The first joint allows forward and backward movements and is constrained within a maximum of 180° rotation. The second joint allows the two lower leg segments to swing towards or away from the body. The third joint, which, like the second, is unconstrained, allows the lowest segment to move forward or backward. The controller gets a 2×2 input field where the field element of each leg contains the three joint angles and a touch sensor for the lowest segment. It gets another input field with the yaw, pitch, and roll values of its torso, as well as a bias input. The output field, which is of size 2×2, contains three outputs for the desired joint angles p_i^s for each field element. The simulator takes these values to move the joints with a desired speed $v_i = 2.0(p_i - p_i^s)$ and a maximal force of 100 units for each joint i (where p_i is the current position of the joint). Unlike in Clune et al (2009), no periodic signal was provided as input because it was found in preliminary experiments that it did not facilitate the evolution of periodic gaits.

A more thorough discussion on how NEAT and NEATfields compare on this task and some related leg control tasks can be found elsewhere (Inden et al, 2011b).

4 Results

20 runs were performed for each experiment. Results are shown in table 1. A performance index was also calculated for individual runs by dividing the highest fitness by the number of evaluations (for the pole balancing task, the highest number of sucessfully tackled cart-pole configurations was used instead of the highest fitness). This make possible a comparison of the different methods as shown in figure 13 for pole balancing, figure 14 for maze navigation, figure 17 for walking, figure 15 for the four patterns task, and figure 16 for the two patterns task.

selection method	pole balancing	maze navigation	walking	four patterns task	two patterns task
tournament selection	0.50 65896	0.20 674943	3.3	1.00 84457	0.35 547340
G-speciation	1.00 33961	0.15 857141	7.2	1.00 73947	0.65 517301
B-speciation	1.0 25721	0.95 233748	6.6	1.00 32205	0.90 469584
G/B-speciation	1.00 26958	0.30 427200	x	1.00 85531	0.40 472152
G-novelty search	0.00 -	0.00 -	x	1.00 292508	0.00* -
B-novelty search	0.00 -	0.40 558371	8.4	1.00 288696	0.00 -
G-F-N-selection	0.15 25986	0.30 584106	x	1.00 132223	0.25 452588
B-F-N-selection	0.30 49813	0.70 466781	x	1.00 79058	0.40 379649
G-F/N-selection	0.25 60743	0.15 479248	x	1.00 135311	0.60 516240
B-F/N-selection	0.30 37476	0.90 557315	8.4	1.00 121885	0.45 531947
G/B-F/N-selection	0.2 65614	0.35 415122	8.9	1.00 72688	0.53* 521581
B-novelty speciation	1.0 19898	0.90 214708	8.4	1.00 42536	0.20 501720
G-NN-speciation static	0.10 25545	0.80 499252	x	1.00 87754	0.15 522684
G-NN-speciation dynamic	0.8 54807	0.10 507482	8.0	1.00 77767	0.50 599767
B-NN-speciation static	0.15 33236	1.00 190096	x	1.00 40423	0.30 327029
B-NN-speciation dynamic	1.0 28393	1.00 106843	8.0	1.0 40423	0.20 472959

Table 1 The fraction of successful runs, and the average number of evaluations in successful runs, for different selection methods. For the walking task, just the mean distances covered by the winners of the runs are shown. Besides, only selected configurations were tested for the walking task. For configurations marked with "*", not all runs could finish within the allocated time budget, so those runs were excluded from the statistics.

5 Discussion

No method outcompetes the other methods on all tasks, so a recommendation to always prefer one method cannot be given. This is not too surprising since different tasks have different demands. However, a few general trends can be observed in the results:

- Tournament selection (chosen as a baseline) is among the methods with lowest performance except for the four patterns task, which is very simple to solve.
- Pure novelty search, whether genetic or behavioral, is also among the methods with lowest performance except for the quadruped walking task.
- It is not good idea to use novelty search with genetic distance measures because this will lead to the evolution of ever larger genotypes, which in turn will slow down evolution. G-novelty search performs worst of all selection methods.
- The same method in most cases performs better if behavioral distance measures are used (alone or in combination with genetic distance measures).

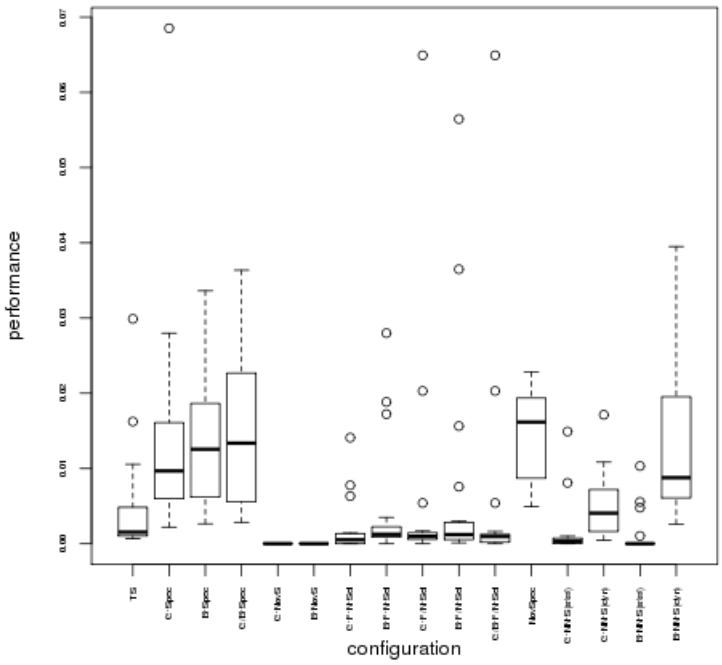


Fig. 13 Performance of different selection methods on double pole balancing. Abbreviations: TS, tournament selection; G-Spec, genetic speciation; B-Spec, behavioral speciation; G/B-Spec, genetic/behavioral speciation; G-NovS, genetic novelty search; B-NovS, behavioral novelty search; G-F-N-Sel, genetic F-N selection; B-F-N-Sel, behavioral F-N selection; G-F/N-Sel, genetic F/N selection; B-F/N-Sel behavioral F/N selection; G/B-F/N-Sel, genetic/behavioral F/N selection; NovSpec, novelty speciation; G-NN-S(stat), genetic near neutral-speciation with static speciation threshold; G-NN-S(dyn), genetic near neutral-speciation with dynamic speciation threshold; B-NN-S(stat), behavioral near neutral-speciation with static speciation threshold; B-NN-S(dyn), behavioral near neutral-speciation with dynamic speciation threshold.

- There is no strong difference between F-N-selection and F/N-selection. Both methods do not perform well on tasks other than the quadruped walking task.
- Behavioral speciation yields good performance except for the quadruped walking task.
- B-NN-speciation with dynamic threshold and B-novelty speciation, which are identical to B-speciation except for the way the future species size is calculated, also perform quite well in general. They do slightly better than B-speciation on the quadruped walking task, but fail almost completely on the two patterns task. Why that happens is not clear.

Of course, it is not possible to distinguish between the effects of the difficulty of a task and the effects of a particular behavioral distance measure based on the shown results. Here we consider these two factors as given. Designing a good behavioral measure is a separate topic of research just like designing good fitness functions. Alternatively, general distance measures (Doncieux and Mouret, 2010; Gomez, 2009) can be used.

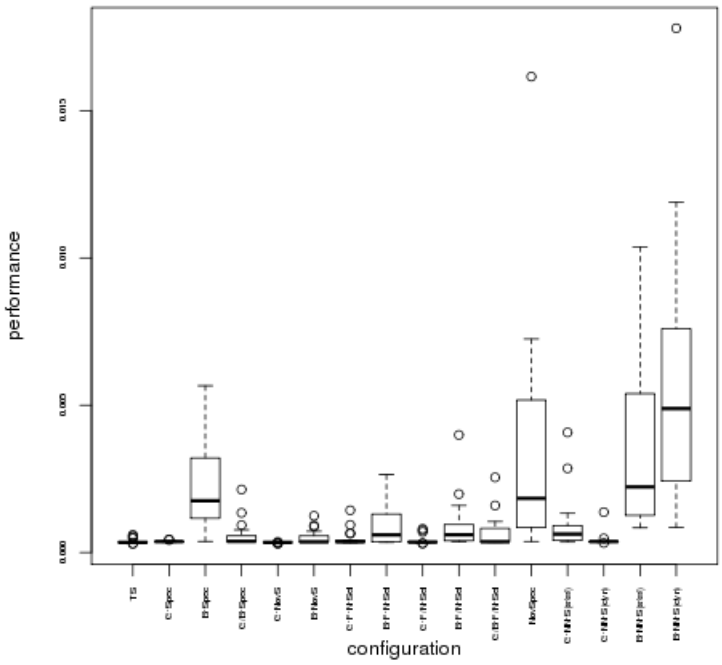


Fig. 14 Performance of different selection methods on maze navigation. Abbreviations like in figure 13.

The following rules of thumb can be derived from the results of our experiments: Methods like behavioral speciation, B-novelty speciation, and B-NN-speciation should be preferred over simple methods like tournament selection or novelty search if the goal is to obtain a reasonably good solution in an unknown fitness landscape because this fitness landscape can usually be assumed to be deceptive to some degree, but also require some measure of optimization effort. Setting parameters like speciation threshold and archive threshold is not too critical because these are automatically adapted. Genetic distance measures can be combined with behavioral distance measures if it has to be suspected that those networks with the smallest topologies will not solve the task, but will be superior to the first networks with larger topologies that emerge. (This was the rationale behind genetic speciation as originally introduced in Stanley and Miikkulainen (2002)). But the weight of genetic criteria should not be too large to prevent runaway complexification of neural networks.

It is also interesting to note that both novelty speciation and NN-speciation implement what is called *local competition* in Lehman and Stanley (2011), although they do so in a different and much simpler way than the multiobjective algorithm used in that article. Future work can compare the different methods to create local competition, and study new algorithms that use local competition, perhaps even in a nested hierarchy. As also suggested in Lehman and Stanley (2011), one should not only look at the fittest individual when comparing different selection methods. It will also be

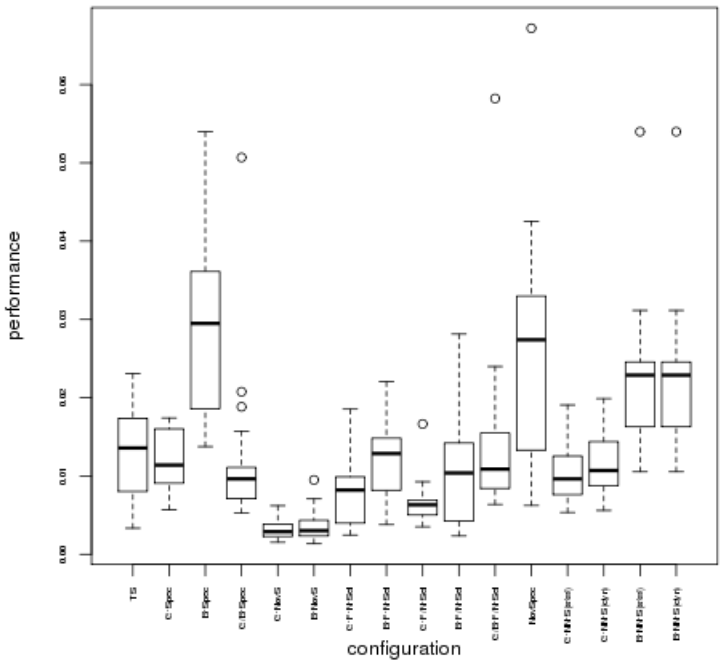


Fig. 15 Performance of different selection methods on the four patterns task. Abbreviations like in figure 13.

interesting to find algorithms that produce a diversity of different useful solutions for a given task.

Acknowledgments

Benjamin Inden gratefully acknowledges the financial support from Honda Research Institute Europe for the project "Co-Evolution of Neural and Morphological Development for Grasping in Changing Environments". Pole balancing code was adapted from the NEAT implementation by Kenneth Stanley.

References

Alba E, Troya JM (2002) Improving flexibility and efficiency by adding parallelism to genetic algorithms. Statistics and Computing 12:91–114

Banzhaf W, Beslon G, Christensen S, Foster JA, Képès F, Lefort V, Miller JF, Radman M, Ramsden JJ (2006) From artificial evolution to computational evolution: a research agenda. Nature Reviews Genetics 7:729–735

Barton N, Partridge L (2000) Limits to natural selection. BioEssays 22:1075–1084

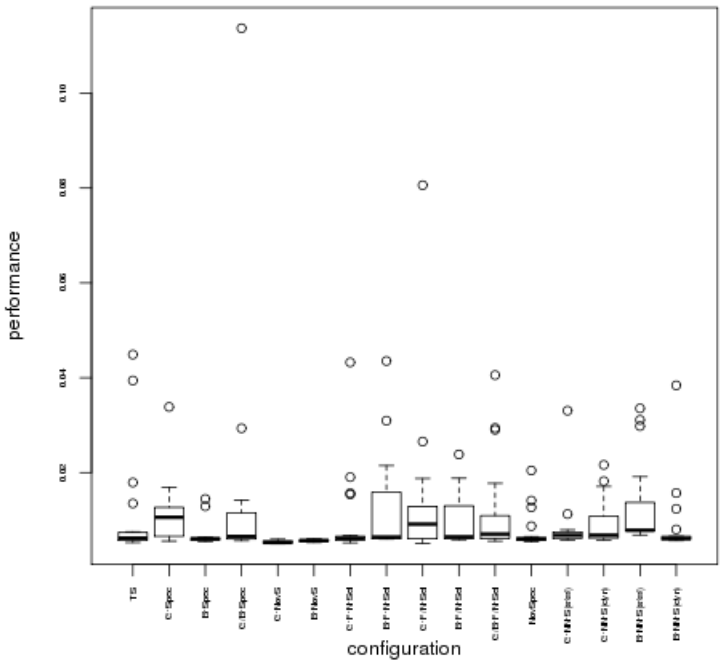


Fig. 16 Performance of different selection methods on the two patterns task. Abbreviations like in figure 13.

Buason G, Bergfeldt N, Ziemke T (2005) Brains, bodies and beyond: Competitive co-evolution of robot controllers, morphologies and environments. Genetic Programming and Evolvable Machines 6:25–51

Clune J, Beckmann BE, Ofria C, Pennock RT (2009) Evolving coordinated quadruped gaits with the hyperneat generative encoding. In: Proceedings of the IEEE Congress on Evolutionary Computing

Cuccu G, Gomez F (2011) When novelty is not enough. In: Proceedings of evo*

Cuccu G, Gomez F, Glasmachers T (2011) Novelty restarts for evolution strategies. In: Proceedings of the IEEE Congres on Evolutionary Computation

Darwen P, Yao X (1995) A dilemma for fitness sharing with a scaling function. In: Proceedings of the 1995 Conference on Evolutionary Computation

De Jong KA (2006) Evolutionary Computation — A Unified Approach. MIT Press

Doncieux S, Mouret JB (2010) Behavioral diversity measures for evolutionary robotics. In: IEEE Congress on Evolutionary Computation

Floreano D, Dürr P, Mattiussi C (2008) Neuroevolution: from architectures to learning. Evolutionary Intelligence 1:47–62

Futuyma DJ (2005) Evolution. Sinauer Associates

Goldberg D, Richardson J (1987) Genetic algorithms with sharing for multimodal optimization. In: Proceedings of the Second International Conference on Genetic Algorithms

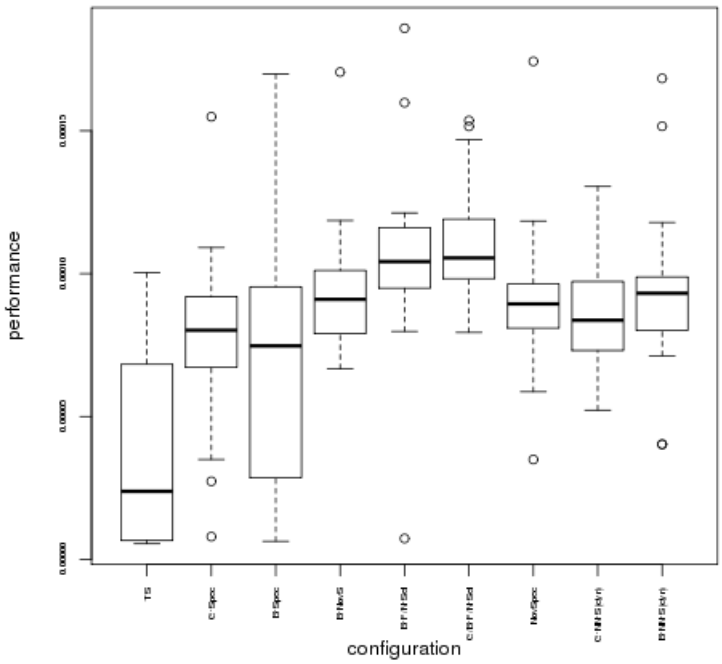


Fig. 17 Performance of different selection methods on the quadruped walking task. Abbreviations like in figure 13.

Gomez F, Miikkulainen R (1997) Incremental evolution of complex general behavior. Adaptive Behavior 5:317–342

Gomez FJ (2009) Sustaining diversity using behavioral information distance. In: Proceedings of the Genetic and Evolutionary Computation Conference

Gould SJ, Lewontin RC (1979) The spandrels of san marco and the panglossian paradigm: A critique of the adaptionist programme. Proceedings of the Royal Society of London B 205:581–598

Harding S, Banzhaf W (2008) Organic Computing, Springer-Verlag, chap Artificial Development

Inden B (2008) Neuroevolution and complexifying genetic architectures for memory and control tasks. Theory in Biosciences 127:187–194

Inden B, Jin Y, Haschke R, Ritter H (2010) Neatfields: Evolution of neural fields. In: Proceedings of the Conference on Genetic and Evolutionary Computation

Inden B, Jin Y, Haschke R, Ritter H (2011a) Evolution of multisensory integration in large neural fields. In: Tenth International Conference on Artificial Evolution

Inden B, Jin Y, Haschke R, Ritter H (2011b) How evolved neural fields can exploit inherent regularity in multilegged robot locomotion tasks. In: Third World Congres on Nature and Biologically Inspired Computation

Inden B, Jin Y, Haschke R, Ritter H (2012) Evolving neural fields for problems with large input and output spaces. Neural Networks 28:24–39

Kauffman SA (1993) The Origins of Order — Self-Organization and Selection in Evolution. Oxford University Press

Lehman J, Stanley KO (2008) Exploiting open-endedness to solve problems through the search for novelty. In: Proceedings of the Eleventh International Conference on Artificial Life

Lehman J, Stanley KO (2010) Revising the evolutionary computation abstraction: Minimal criteria novelty search. In: Proceedings of the Genetic and Evolutionary Computation Conference

Lehman J, Stanley KO (2011) Evolving a diversity of creatures through novelty search and local competition. In: Proceedings of the Genetic and Evolutionary Computation Conference

Lynch M (2007) The frailty of adaptive hypotheses for the origins of organismal complexity. Proceedings of the National Academy of the Sciences 104:8597–8604

Mattiussi C, Floreano D (2007) Analog genetic encoding for the evolution of circuits and networks. IEEE Transactions on Evolutionary Computation 11:596–607

Miconi T (2009) Why coevolution doesn't "work": superiority and progress in coevolution. Proceedings of the EuroGP Conference

Mouret JB (2009) Novelty-based multiobjectivization. In: Proceedings of the Workshop on Exploring New Horizons in Evolutionary Design of Robots, 2009 IEEE/RSJ International Conference on Intelligent Robots and Systems

Mouret JB, Doncieux S (2008) Incremental evolution of animat's behaviors as a multiobjective optimization. In: Simulation of Adaptive Behavior

Nolfi S, Floreano D (2000) Evolutionary Robotics — The Biology, Intelligence, and Technology of Self-Organizing Machines. MIT Press

Poli R, Langdon WB, McPhee NF (2008) A Field Guide to Genetic Programming. Published via http://lulu.com, URL http://www.gp-field-guide.org.uk

Sareni B, Krähenbühl L (1998) Fitness sharing and niching methods revisited. IEEE Transactions on Evolutionary Computation 2:97–106

Stanley K (2004) Efficient evolution of neural networks through complexification. PhD thesis, Report AI-TR-04-314, University of Texas at Austin

Stanley K (2007) Compositional pattern producing networks: A novel abstraction of development. Genetic Programming and Evolvable Machines pp 131–162

Stanley K, Miikkulainen R (2002) Evolving neural networks through augmenting topologies. Evolutionary Computation 10:99–127

Stanley K, Miikkulainen R (2003) A taxonomy for artificial embryogeny. Artificial Life 9:93–130

Tomassini M (2005) Spatially Structured Evolutionary Algorithms — Artificial Evolution in Space and Time. Springer-Verlag

Wieland AP (1991) Evolving controls for unstable systems. In: Touretzky D (ed) Connectionist Models: Proceedings of the 1990 Summer School

Yao X (1999) Evolving artificial neural networks. Proceedings of the IEEE 87:1423–1447