Small is Beautiful : Near Minimal Evolutionary Neurocontrollers Obtained With Self-Organizing Compressed Encoding

Shlomy Boshy
School of Computer Science
Tel Aviv University
Tel-Aviv 69978,Israel
shlomy@post.tau.ac.il

Eytan Ruppin
School of Computer Science and
Sackler School of Medicine
Tel Aviv University
Tel-Aviv 69978,Israel
ruppin@post.tau.ac.il

Abstract

This paper presents a novel method for evolution of artificial autonomous agents. It is based on adaptive, self-organizing compressed genotypic encoding (SOCE) of the phenotypic synaptic efficacies of the agent's neurocontroller. The SOCE encoding implements a parallel evolutionary search for neurocontroller solutions in an dynamically varying and reduced subspace of the original one. The SOCE successfully maintains the quality of solutions obtained by performing search directly in the original space. The SOCE method leads to the robust emergence of compact, near minimal neurocontrollers even when starting from initially large networks. This is very important since in practical situations the network size needed to solve the problem is unknown beforehand. It hence may serve both to estimate the network size needed to solve a given task, and to delineate the relative importance of the neurons composing the agent's controller network.

1. Introduction

Many Evolutionary Autonomous Agents (EAAs) use neural networks to control their behavior. Such neurocontrollers of EAAs performing complex tasks can be evolved via genetic algorithms from a population of genomes undergoing natural selection and variation [see (J.A.Meyer and A.Guilliot, 1994) (X.Yao, 1999) (A.Gulliot and J.A.Meyer, 2001) (A.Gulliot and J.A.Meyer, 2000) for a review]. Much of these EAA studies employ direct genotype-to-phenotype encodings, where the magnitude of each synapse is encoded by a specifically designated gene. These direct encodings are problematic since they scale quadratically with network size and are inadequate for solving complex tasks. This problem has led to considerable efforts for developing "indirect" encodings, where the genome includes a developmental program for specifying the controller neural network and its weights. Addressing the inherent scaling limitations of direct encodings, we adopt a different, new, approach. Instead of utilizing a developmental program to compactly encode *all* the network information, we present an indirect encoding which adaptively maintains only an *approximate* description of the network. The new indirect encoding method introduced and studied here generates compressed synaptic encodings. The method can be used to obtain neurocontrollers containing a near minimal number of neurons for any given task, which are more amenable for an analysis of the structure and activity.

The search for finding short, compact representations of solutions to problems has its origins in *Occam's Razor* and the *Minimum Description Length (MDL)* principles. These principles suggest that among the viable solutions to a problem, the simpler, shorter solutions should be preferred. The rational is that there is a much smaller number of simple solutions (i.e. hypotheses) than elaborate ones, so if one finds a simple solution it is more likely to be a correct one (e.g. with good generalization properties) than a more complex one. In the same spirit, indirect encoding methods that narrow the search to a simpler low dimensional space increase the chance of finding a good solution for problems originally embedded in a high dimensional search space. However, while decreasing the search space, indirect encodings my restrict the set of possible solutions and thereby may fail to find good solutions. Devising indirect methods that selectively decrease the search space while maintaining a maximal set of good solutions is an important open research problem.

The basic contribution of this paper is to present a novel evolutionary process that maintains *a "good" subspace in which the search is performed.* Our indirect en-

coding method is based on *adaptive compression*. The synaptic efficacies of each neuron are encoded in a lossy, compressed manner, and an inverse decompression transform is used to transcribe the genome into a functioning phenotype. Individual neurons may utilize different encoding (compression) levels. The level of compression may vary from containing no information to direct specification of all the synaptic connection values. These levels of compression are themselves encoded in the genome and undergo evolution, adaptively varying in a self-organized manner. We term this method *Self-Organizing Compressed Encoding* (SOCE). Note that we do *not* use a fitness function that *explicitly* encourages shorter solutions, as done in the standard MDL approach. Compact solutions are obtained only because they enable a more efficient evolutionary search, which in turn leads to finding new compact solutions, in a reciprocal manner. With this scheme, the length of the genomic encodings of solutions to the problem in hand is optimized in an *implicit* manner.

Having adaptive, compressed encodings yields agents with compact genotypes but their phenotypes may remain unnecessary large. Exploiting a specific form of genetic variation in our method leads to the emergence of *compact near-minimal phenotype neurocontrollers*. Thus, our method leads to two important consequences: simpler controller networks that are more amenable for functional analysis, and an estimation of the complexity of the problem in hand. That is, if starting from different initial network sizes a good solution can be achieved by SOCE with X neurons, this gives us an upper bound on the number of neurons needed to solve the task. The SOCE algorithm automatically finds compact networks that solve the task, *irrespective of the initial network size*. This is very important since in practical situations the network size needed to solve the problem is unknown beforehand.

Lossy compression methods usually rely on exploiting regularities in the data encoded. Image compression, for example, relies on the fact that neighboring points in the image tend to have a similar level of brightness. This mutual information between neighboring points can be utilized to generate compact encodings. The task of encoding controller networks of EAAs in general does not offer such regularities, that may lend themselves for lossy compression. The synaptic values W_{ij} and W_{ik} projecting from neurons j and k on neuron i are uncorrelated to a first approximation. However, a simple observation shows that a series of numbers designating the weights of the input (or output) synapses which a neuron receives may still be subject to lossy compression, as illustrated in Figure 1 and explained below. For an uncorrelated series of synaptic random values the expected encoding level is about $\frac{P}{2}$, where P is the number of synaptic inputs a neuron receives. The compression levels obtained for EAA networks depend on a few factors, among them the sensitivity of neural processing to the fidelity of synaptic values and, more importantly, on the actual number of neurons that is really needed to solve the task. We show that the SOCE algorithm can be successfully used in EAA environments to attain high compression levels and, more importantly, near minimal neurocontrollers.

The rest of the paper is organized as follows: Section 2 discusses previous work on indirect genetic encodings. Section 3 describes the model, including the agent and the EAA environment and the compressed encoding method. Section 4 describes our experimental results, and section 5 discussed their implications.

2. Indirect Encoding Methods

The work on developing indirect encodings has attracted ample efforts, focusing on a few distinct avenues:

- *Grammar Rewriting encodings* – which employ a set of rewriting rules that are encoded in the genome. For example, in (H.Kitano, 1990), an iterative decoding process uses rewriting rules to create a matrix specifying the network architecture. Simple grammar rewriting encodings typically generate restricted tree-like network architectures, but utilizing graph grammars one may develop more general recurrent networks (F.Gruau, 1994),(J.Kodjabachian and J-A.Meyer, 1998). Such encodings generate fairly compact genomes and hence reduce the search space of possible solutions. A variant, (B-T.Zhang and H.Muhlenbein, 1993) enables the evolution of compact target networks by including a complexity term in the fitness function.

- *Developmental, Ontogenetic encodings* – where the genome expresses a program for cell division and axonal migration that determines the phenotypic neural architecture [(R.K.Belew, 1993) (A.Cangelosi et al., 1994) (S.Nolfi and D.Parisi, 1995)]. In these encodings the objects undergoing development are localized on a 2-dimensional space, allowing for context-dependent effects from neighboring neurons, and the developmental program has a more biological flavor. Yet, the genomes generated are less compact than those generated by encoding graph grammars (scaling linearly with network size) and do not strongly bias toward the evolution of modular networks.

  Other developmental encodings incorporate more biologically motivated "regulatory" encodings, [(A.Cangelosi and J.L.Elman, 1995) (P.Eggenberger, 1996)], where the identity of the subset of genes active in each moment is determined by a complex interaction with the "transcription

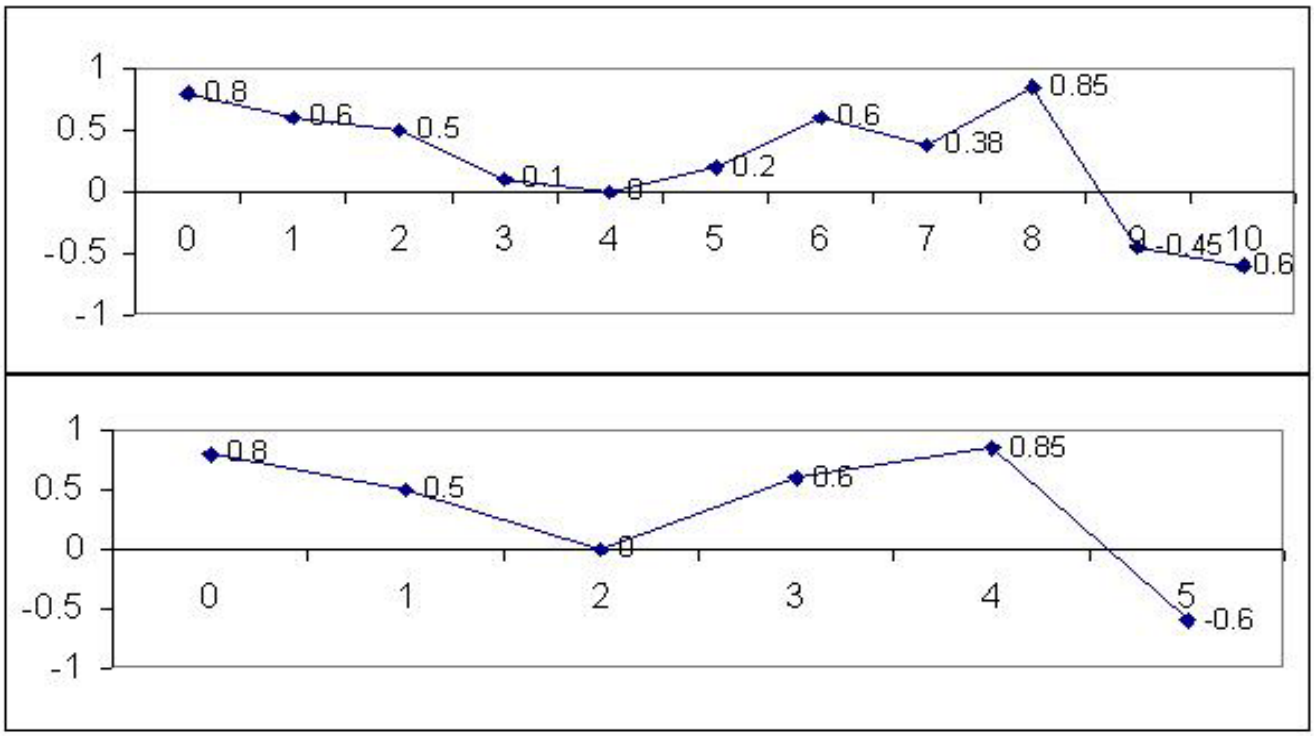


Figure 1: Encoding a series of 11 original numerical values (top) using 6 encoding parameters(below). To obtain a (possibly gross) approximation of the original series of numbers using a point interpolation transform, we need to represent a point in the compressed description only when the gradient of the line connecting the weight points switches direction. As long as the line keeps going up or down, the interpolation can use two end points to determine intermediate values along the line (the monotonic polygon between them can be approximated by a line). So, if the agent's performance is not very sensitive to perturbations of the synaptic values of some of its less important neurons, then a compression of this kind could approximate phenotypes with an encoding level equal to the average number of "switching" points in the line connecting the original weight values.

factors" it receives from the environment. These models require extensive computational resources and are still unable to evolve agents solving complex tasks.

- *Compound encodings* – recently, newly developed genetic encodings that flexibly encode both the synaptic weights and their learning rules have been shown to be superior to direct synaptic encodings, but these results await further corroboration in a wider set of EAA models [(D.Floreano and J.Urzelai, 2000) (D.Floreano and J.Urzelai, 2001)].

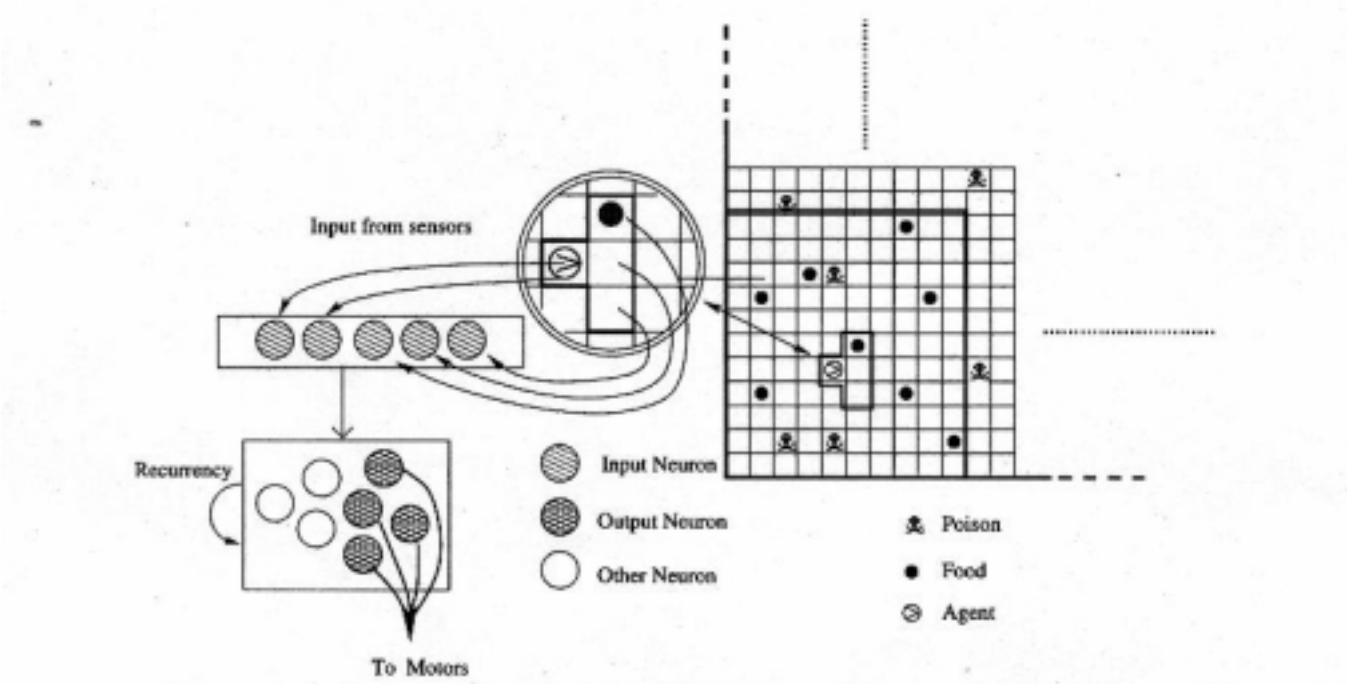
In summary, the superiority of existing indirect encodings over direct ones has not yet been shown in a convincing manner, due to the absence of convincing examples of agents solving complex tasks with indirect encodings that were otherwise unsolvable with direct encodings(D.Floreano and J.Urzelai, 2001).

3. The Model

3.1 The EAA Environment

The agent and environment used to study our compression encoding method are as in (R.Aharonov-Barki et al., 2001): Agents generated via an evolutionary algorithm perform a navigating and foraging task in a discrete arena. The arena is surrounded by walls, and contains two kinds of resources: "food", which increases the agent's fitness, and "poison", which decreases it. The food is concentrated in a limited zone in the southwest corner of the arena (see Figure 2). Each agent has to eat as much food as possible and avoid poison to maximize its fitness. It has to learn to distinguish between food and poison, and how to speedily locate the food zone. 30 food items are randomly placed in a 10*10 food zone inside the 30*30 arena. 250 poison items are placed in random locations in the arena.

The agent contains a fully recurrent neural network



An outline of the grid arena (southwest corner) and the agent's controlling network. The agent is marked by a small arrow on the grid, whose direction indicates its orientation. The curved lines indicate where in the arena each of the sensory inputs comes from. Output neurons and inter-neurons are all fully connected to each other.

Figure 2: The Agent and the Environment. The agent has only local information and sensory input combination is needed to perform the task (adopted from Aharonov-Barki,2001).

controller connected to its sensors and motors (Figure 2). It is the neural network controller which is genetically encoded and evolved, while the sensors and motors are given and constant. In the simulation used here the network had 5 input sensors and 10 internal recurrently connected neurons, of which 4 are motor neurons.

The agent has very limited sensory information: Four sensors sense the grid cell in which the agent is located and the three grid cells ahead of it. These sensors sense the difference between an empty cell, a cell with a resource and a wall, but do not distinguish between a food and a poison item. The fifth sensor is a "smell" probe which discriminates between food and poison just under the agent, but gives a random reading if the agent does not stand on a resource. The motor system allows the agent to go forward, turn 90 degrees in each direction and open its "mouth" in an attempt to eat. Eating is performed only when stopping and having the agent's mouth open, thus consuming a whole time step.

The agent has only local information on the environment, making navigation a challenging task. It has no information of its place coordinates or orientation on the arena. It must combine sensory outputs to see a resource, move towards it, discriminate it as "food" and stop to eat it. An agent has a lifetime "epoch" of 150 steps, which it begins from a random location and orientation.

A population of 100 agents is evaluated in each generation. Selection, mutation and crossover are performed to create the next generation. Simulations were typically run for 10,000 generations. Mutation rate is 0.02 for each place in the genome. A mutation changes the encoding parameter value by a random number between -0.6 and 0.6. Crossover rate is 0.04. The 10% most successful agents do not undergo mutations and crossover.

3.2 Methods:The SOCE Algorithm

The genome of each agent is a compressed indirect encoding of the features of the developed phenotype, i.e. the synaptic weights of the controller network. The evolutionary process is governed by a standard genetic algorithm, with a developmental stage occurring at the beginning of every new generation, creating the controller networks from the genomes by a decoding transform. The controller recurrent neural network contains L_{in} in-

put sensors (neurons) and N internal neurons (of which L_{out} provide output to the motors), i.e., $L_{in} * N + N * N$ synapses.

The decoding transform works on the neuronal level. Each neuron has a distinct compressed representation of its synapses and utilizes a distinct compression level. Let us look at the encoding for a given neuron: Denote by P the number of weights the neuron receives, and by K the number of its encoding parameters (its encoding level) . Let $G[I]$ be the value of the Ith place in the genome from the beginning of this neuron's encoding string. Let T_k be the indirect decoding transform with an encoding level K. Let $S[J]$ be the weight value of the input synapse J of the neuron (for simplicity we omit the subscripts that denote that these are neuronal parameters).

The transform T_K takes the K encoding parameters and uses them to generate P values by piece-wise linear interpolation. The synaptic values $S(j)(0 \leq j \leq P-1)$ are obtained via an interpolation between the values of the 2 "nearest" encoding parameters, with weights proportional to the distance from each parameter (see Figure 3). Note that some of these weights may be zero.

More formally, for $2 \leq K \leq P-1$ the jth synapse value is $S(j) = \frac{G[P_{left}]*W_{left}+G[P_{right}]*W_{right}}{M}$ where $M = \frac{P}{K}$ is the number of synapse values created from each two "neighbor" generative encoding parameters $P_{left} = \frac{J}{M}$ and $P_{right} = P_{left} + 1$. $W_{left} = (M - |J - P_{left} * M|)$, $W_{right} = (M - |J - P_{right} * M|)$ are the weights given to each encoding parameter according to the distance along the interpolation line. The division sign denotes integer division.

For $K = 0$ all incoming synapses of a neuron receive a zero value. For $K = 1$ they all receive a fixed value of the single genome place $G[0]$. For $K = P$ we obtain a direct encoding - each place $G[i]$ in the genome encodes a single, unique weight value $S[j]$ in the controller neural network.

The distinct encoding (compression) level K of each neuron is also encoded in the genome. This enables the agent to "choose" during the evolutionary process to which neurons it should allocate a more detailed (but hence longer) encoding and which should receive coarser (possibly zero length) encodings. E.g., if all neurons use encoding level K, the genetic algorithm performs its search in a K-dimensional subspace instead of the original P-dimensional search space. Allowing each neuron to individually vary its encoding level allows the agent to self organize its search space according to the environment in which it evolves, in an ongoing adaptive manner.

The decoding transform $T_K : [-1,1]^K \rightarrow [-1,1]^P$ must obey certain properties to serve as an efficient genotype-to-phenotype encoding:

1. *Topology Conservation* - Phenotypes created from similar genomes must be similar to each other in the phenotypic space. If this requirement is not met, an agent created from a slightly varied genome could be very different in each generation resulting in an erratic, almost random, evolutionary process. The transform we present fulfills this required property because of its *local* piecewise interpolation.

2. *Completeness* - The set of phenotypes that can be created from a population of genomes should span, at least in principle, extensive parts of the phenotype space. This requirement assures that good solutions are reachable. In the SOCE method this is achieved via the self organization and individual encoding levels that each neuron employs, compressing and spanning the search space in a dynamic manner. Most phenotypes can be represented to a certain approximation, and our transform dynamically changes the encoding level, reaching direct encoding in the limit, when required.

Obviously, if $K = 0$ the corresponding neuron will never transmit any information to other neurons because it receives zero, sub-threshold inputs. This is a prime feature of the SOCE algorithm enabling its usage to find near minimal solutions to a task (minimal by the number of neurons) starting from *any* sufficiently large initial network sizes, and letting the self organization process select a small number of important neurons (i.e. with $K > 0$) to construct the agents' controller network. This property has practical importance since in almost all tasks we have poor intuitive sense of the magnitude of the network required to solve them. In fact, the SOCE method can achieve very large compression/minimization levels if one chooses to start from arbitrary large networks (see next section).

The minimization of the phenotypic networks from the compact genomes obtained by SOCE is performed by letting each neuron's compression level K obtain values on the interval $[-\inf, U]$ where U is some positive upper bound (naturally one may choose $U = N$). With this left-open interval bound non-important redundant neurons will obtain $K = 0$ values with probability 1 due to the random walk performed on these values during the evolutionary process.

Note that the fitness function that governs the evolutionary process is based on the agent's performance solely and does *not* contain any *explicit* preference for shorter genomes. Nothing explicitly prevents the agent from always using a full direct encoding ($K = P$). Even in simulation experiments where we have started from an initial population of full direct encodings for all the agents, the emerging successful agents employed compact encodings. It is likely that decreasing the search space enables the shorter genomes to find good solutions faster, thus driving the population towards compact solutions, both on the genotypic and phenotypic levels (see below).

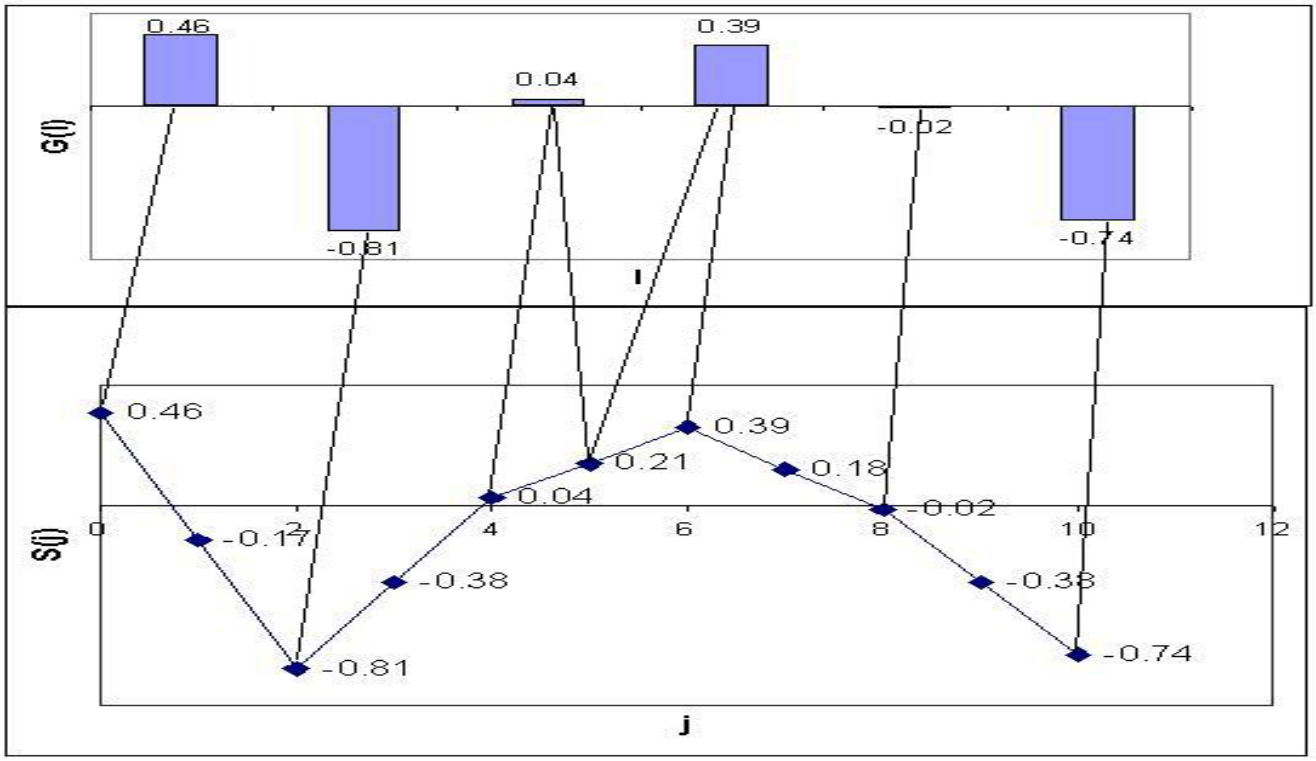


Figure 3: An example of the SOCE genotype-to-phenotype T_K transform: The top figure shows the encoding parameters in the genome, which are decompressed to form the synaptic weights shown in the bottom figure. For example, synapses S(4) and S(6) receive their values directly from the corresponding encoding parameters G(3) and G(4), while S(5) is determined by interpolating G(3) and G(4).

4. Results

The compressed encoding algorithm was tested by running evolutionary experiments in the environment of (R.Aharonov-Barki et al., 2001) described in the Model section. Figure 4a shows that the average performance of direct and SOCE agents is about similar throughout the simulation run but the SOCE encoding leads to much smaller networks. The compressed encoded agents were able to solve the navigating problem with a genome length of about 33% of the direct encoding genome. The average encoding level decreases throughout the evolution of a SOCE agent from initial levels of about 50% of the direct encoding size to about 33% (initial encoding levels of about 50% are obtained by the uniform selection of encoding levels from distribution U[0,N] at generation 1). The SOCE agents also showed robustness: more than 90% of the runs produced good solutions to the problem, about the same percentage as direct encodings achieve in this task.

In order to see the importance of the *dynamic* modification of individual compression levels, we studied the baseline, "null hypothesis", case in which all the population has a *fixed* compression level K. The agents were able to solve the problem, but as K got smaller the results became less and less robust: many runs did not achieve working solutions and the average fitness dropped . These results testify to the importance of the dynamic, self-organizing nature of the SOCE encoding.

Using the compressed encoding algorithm with encoding bounds $[-\inf, P]$ leads to the self-organized elimination of many unimportant neurons with a bimodal distribution of encoding levels (Figure 4b), and to near minimal size networks. Bounding the synapses only from the upper side leads to many zero information neurons while the remaining neurons are free to select small encoding levels.

A concrete example of such a successful evolved agent has a genome with only a third of the length of the direct encoding, with the following main properties:

- Out of 4 motor (output) neurons, only 3 have non vanishing encoding levels. The apparent elimination of a motor neuron was initially surprising. However, in this problem it turns out that a very good solution can be based only on one-sided turns, so one turning motor neuron can be eliminated.
- Out of the 6 internal neurons that are not connected directly to the motors, only one had a non-zero encoding level and remained viable in the fi-

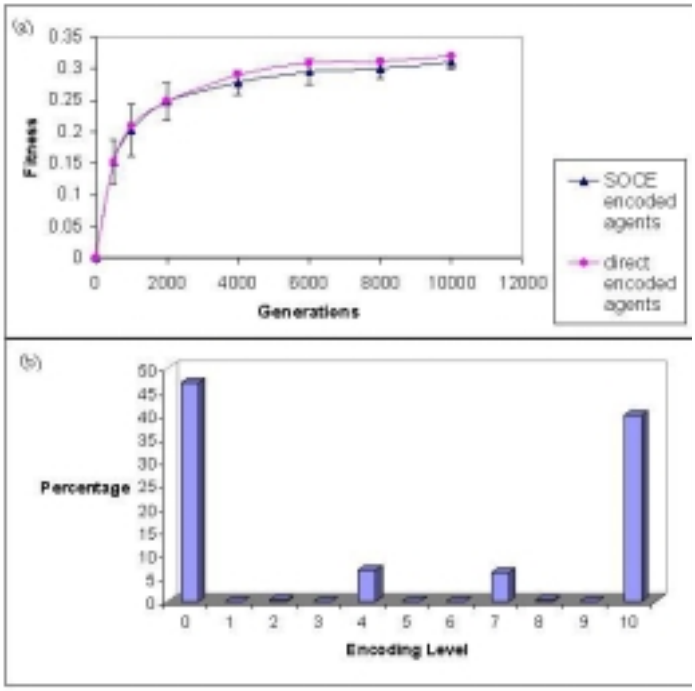


Figure 4: (a) Fitness of direct versus compressed encoding, plotted across generations of evolution. The means and SDs of 20 experiments are shown. 0.3 denotes a normalized high performance level due to the limited lifespan of the agent.(b) Distribution of encoding levels in an evolved SOCE agent. A bimodal distribution is created by the random walk dynamics. This enables to create near minimal networks because zero encoding level neurons do not effect the network and can be removed

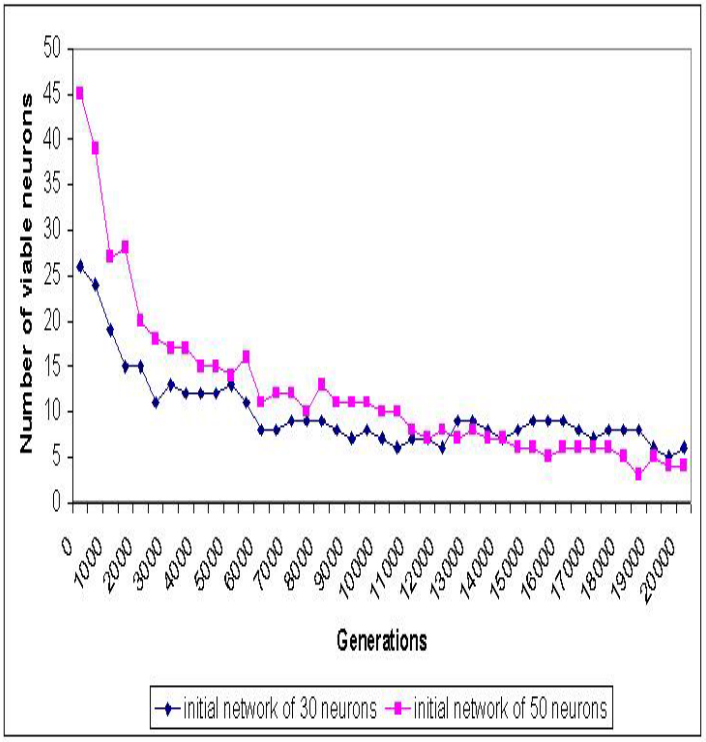


Figure 5: Number of viable neurons (with non zero encoding levels) in the neurocontroller of SOCE agents, starting from initial large networks of 30 and 50 neurons. The SOCE algorithm leads to a near minimal network with only 4-5 effective neurons. In each generation, the agent with minimal network size is shown. At the end of the runs, all agents achieve normalized high performance levels, around 0.31.

nal controller network. This resembles the findings of a single, important "Command neuron" in (R.Aharonov-Barki et al., 2001), which switches between two modes of behavior of navigation and foraging: searching for the food zone, an foraging while inside the food zone. It is an example of how this method can provide important clues to the analysis of evolved agents.

Compressed encodings emerge even with bounded encoding levels dynamics (i.e. without phenotypic compactness). Even when we used bound weight values in the range $[0, P]$ and started with direct encoding levels ($K = P$) for all the population, the selection process and the genetic operators generate a population in which most of the neurons do not have a full encoding level.

Starting from initially much larger networks composing the initial population containing 30 and 50 neurons we were able to show the SOCE algorithm is still able to reach a near minimal network. The evolutionary process reached high performance solutions with only $4-5$ effective neurons (see Figure 5). The baseline network discussed so far started with 10 neurons and the reached similar minimal solutions of 4 effective neurons. Thus the SOCE algorithm provides near minimal controller networks, yielding an estimate of the number of neurons needed to solve a given task by starting with large networks and sorting out the unimportant neurons.

The harder the problem the larger is its search space, and the benefit gained from SOCE is expected to be greater. To test this hypothesis we ran the SOCE on an extended, more difficult task than our original one. The agent now has a more complex behavior to complete before gaining a food item reward: It has to remain for X consecutive steps on the food item, afterwhich it should close its mouth to ingest the food. The number of steps X was incremented during the evolutionary run from 1 to 3 every 2500 generations of a 10000 generations run (incremental evolution). In this problem the percentage of successful runs reached by SOCE agents was much greater that direct-encoded agents (30% as opposed to 10% out of 10 simulation runs). The time to recover from each increase in the difficulty level of the problem, adopt to the new task and regain high food intake and

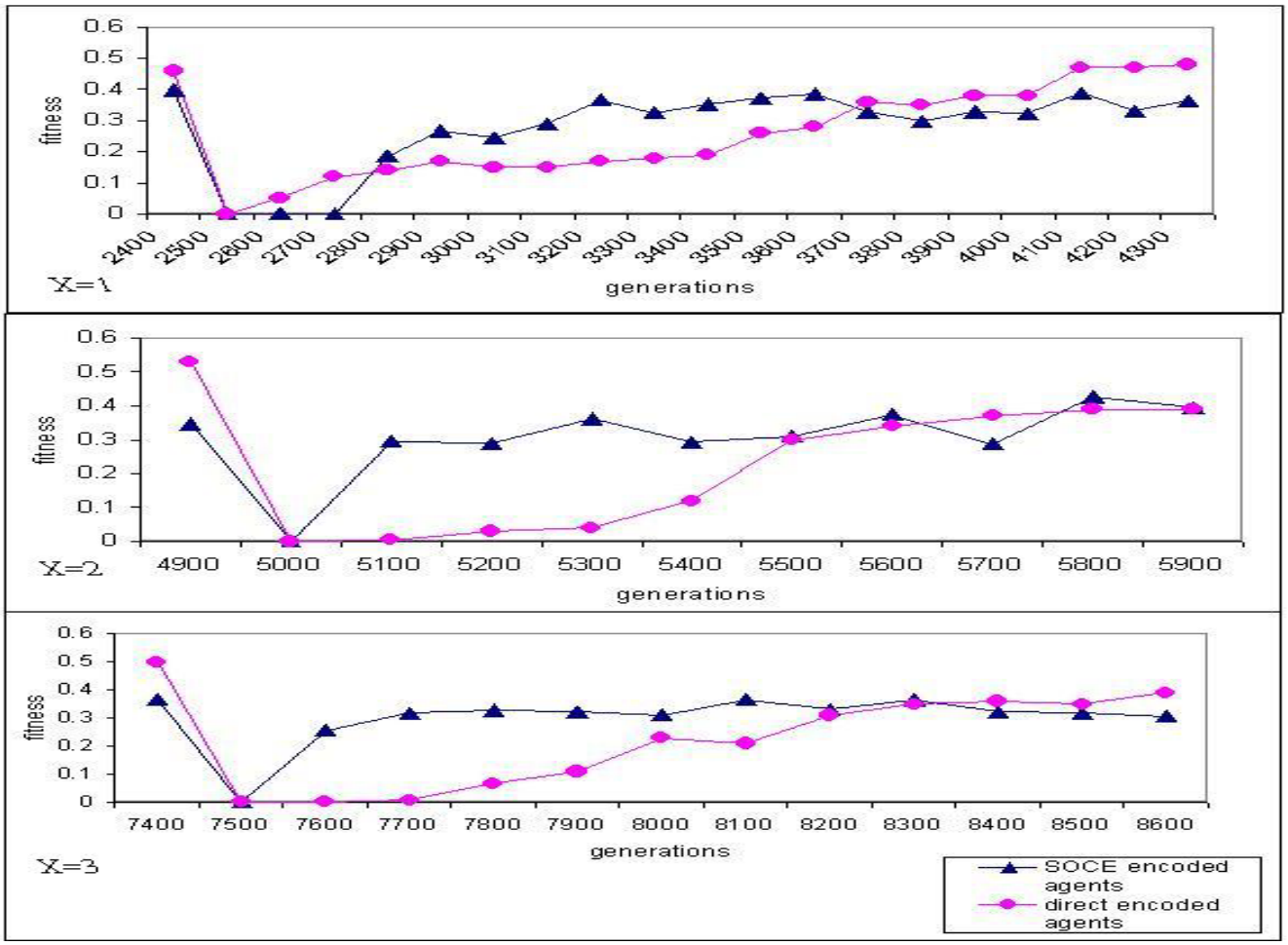


Figure 6: Fitness of SOCE encoded and direct encoded agent when problem difficulty increases every 2500 generations.The SOCE agents recover much faster than direct encoded agents in most cases. Awaiting X steps before food ingestion makes the task more difficult for the agents because of the increased memory span they impose on solving the task successfully.

fitness levels, was much shorter for SOCE agents in most cases, as can be seen in figure 6.

5. Discussion

The SOCE algorithm was shown to successfully solve navigation and foraging problems, providing robust and compact solutions. Via the SOCE, the agents select minimal encoding levels without any explicit fitness pressure. The self-organizing dynamics of the SOCE method allows them to adaptively focus the search in relevant subspaces. This results in a reduction of genome size, and consequently leads to minimization of network size. This utilization of genotypic compression to obtain phenotypic completeness is the prime advantage of SOCE from a practical standpoint. In the absence of prior knowledge of the size of the controller network required to solve the task, one can start from initially large networks which are reduced during the evolutionary process to near-minimal size.

The SOCE algorithm can also be used as an analysis method: The encoding level of each neuron is a good clue for its importance and "input sensitivity". Moreover, since the object of the analysis are near-minimal size networks, the insights gained probably reflect more the true, inherent nature of successful solutions to the task in hand, than the specific idoisynoracies of one specific solution out of many possible non-minimal ones obtained with standard, direct encodings. One cautionary note, however, is in place: the encoding levels of the remaining, viable, neurons correlate but do not directly correspond to these neurons "importance" in the network. This is so because the compression/encoding level permitted also reflects the fidelity of the neuron's input/output function, and not just the importance of the information it transmits to other neurons in the network.

Future research should naturally add learning capabilities into the genome. Learning rules can be encoded using the SOCE method and used in the development stage of the agent. The agent can decide what encod-

ing levels should be allocated to the data vs. the theory (the synaptic weights versus the synaptic learning rules). Combining data and theory using SOCE will give us a comprehensive framework for finding a Minimal Description Length solution to a task with an efficient tradeoff between data (saved synapse weights) and theory (learning or development rules).

The SOCE method utilizes a local encoding method (piecewise interpolation) to obtain a topology conserving genotype-to-phenotype mapping (see section 3). This, however, has its drawbacks since global encodings can compress the data more efficiently than local ones. Interestingly. the SOCE method lends itself quite naturally to incorporate a global encoding (e.g., a Fourier series representation of the synaptic values in the network). This may be accomplished by adding an additional global-encoding layer which encodes the current, local encoding parameters of the agent's genome. That is, evolution proceeds in its "local phase" for a few thousand years iterations with the present, local SOCE method. Thereafter, successful agents are selected and evolved further using a globally-encoded genomic representation. The feasibility and efficiency of this global phase is an interesting subject of future research.

In summary, we presented a new genotype-to-phenotype encoding method based on adaptive, self-organizing compression. This method provides efficient, robust and compact (near minimal) solutions to EAA tasks, starting from a population of inefficient and arbitrarily large network solutions. It thus provides a novel and powerful way to efficienctly evolve EAA solutions to non-trivial tasks, to estimate the complexity of these tasks, and to grade the importance of neurons composing the emerging neuroncontrollers.

References

A.Cangelosi, D.Parisi, and S.Nolfi (1994). Cell division and migration in a 'genotype' for neural networks. *Network(5),pages 497-515.*

A.Cangelosi and J.L.Elman (1995). Gene regulation and biological development in neural networks:an exploratory model. *Technical report,CRL-UCSD,University of California at San Diego,www.citeserr.nj.nec.com/context/15377/132530.*

A.Gulliot and J.A.Meyer (2000). From sab94 to sab2000:what's new,animat? *Proceedings of the Sixth International Conference on Simulation of Adaptive Behavior. MIT Press,Cambridge,MA.*

A.Gulliot and J.A.Meyer (2001). The animat contribution to cognitive systems research. *Journal of Cognitive Systems Research(2),pages 157-165.*

B-T.Zhang and H.Muhlenbein (1993). Evolving optimal neural networks using genetic algorithms with occam's razor. *Complex Systems, 7, pages 199-220.*

D.Floreano and J.Urzelai (2000). Evolutionary robots with online self-organization and behavioral fitness. *Neural Networks(13),pages 431-443.*

D.Floreano and J.Urzelai (2001). Neural morphogenesis, synaptic plasticity and evolution. *Theory in Biosciences.*

F.Gruau (1994). Automatic definition of modular neural networks. *Adaptive behavior, 3, pages 151-183.*

H.Kitano (1990). Designing neural networks using genetic algorithms with graph generation system. *Complex System, 4, pages 461-476.*

J.A.Meyer and A.Guilliot (1994). From sab90 to sab94:four years of animat research. *Proceedings of the third international Conference on Simulation of Adaptive Behavior.,ed D.Cliff and P.Husbands and J.A.Meyer and S.K.Wilson MIT Press,Cambridge,MA.*

J.Kodjabachian and J-A.Meyer (1998). Evolution and development of modular control architectures for 1-d locompotion in six-legged animats. *Connection Science, 10, pages 211-254.*

P.Eggenberger (1996). Cell interactions as a control tool of developmental processes for evolutionary robotics. *Proceedings of the Forth International Conference on Simulation of Adaptive Behavior.,ed. P.Maes and M.Mataric and J-A.Meyer and J.Pollack and H.Roitblat and S.Wilson MIT Press,Cambridge,MA.*

R.Aharonov-Barki, T.Beker, and E.Ruppin (2001). Emergence of memory-driven command neurons in evolved artificial agents. *Neural Computation,13,pages 691-716.*

R.K.Belew (1993). Interposing an ontogenetic model between genetic algorithms and neural networks. *Advances in Neural Information Processing(NIPS5).,ed. J.Cowan Morgan Kaufmann,San Mateo,CA.*

S.Nolfi and D.Parisi (1995). Evolving artificial neural networks that develop in time. *Advances in Artificial Life:Lecture Notes in Artificial Intelligence(929),pages 353-367.*

X.Yao (1999). Evolving artificial neural networks. *Proceedings of the IEEE,87(9).pages 1423-1447.*