

Evolving Small Neurocontrollers With Self-Organized Compact Encoding

Shlomy Boshy Eytan Ruppin

January 6, 2003

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

Abstract

This paper presents a novel method for the evolution of artificial autonomous agents with small neurocontrollers. It is based on adaptive, self-organized compact genotypic encoding (SOCE) generating the phenotypic synaptic weights of the agent's neurocontroller. SOCE implements a parallel evolutionary search for neurocontroller solutions in a dynamically varying and reduced subspace of the original synaptic space. It leads to the emergence of compact successful neurocontrollers starting from large networks. The method can serve 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 Agent (EAA) models use neural networks to control their behavior. Such EAA neurocontrollers are evolved via genetic algorithms from a population of genomes undergoing natural selection and variation (see [21, 32, 9, 8] for a review). The majority of these studies employ direct genotype-to-phenotype encodings, where the magnitude of each synapse is encoded by a specifically designated gene. These direct encodings scale quadratically with network size and hence are hypothesized to be inadequate for solving complex tasks. Furthermore, the network size needed to solve a given task is unknown beforehand. These problems have led to considerable efforts aimed at developing “indirect” encodings, where the genome includes a developmental program for specifying the controller neural network and its weights. This paper presents a novel indirect encoding method. It, is based on an adaptively varying compact representation, where individual neurons may utilize different encoding levels. It enables the creation of near minimal phenotypic neurocontrollers.

The search for finding short, compact representations of solutions to problems has its origins in Occam’s Razor and the Minimum Description Length (MDL) principle. These principles suggest that among the viable solutions to a problem, the simpler, shorter solutions should be preferred. The rationale is that there are less simple solutions than elaborate ones. Hence, if one finds a simple solution that solves the problem, it is more likely to be a good one, i.e., with good generalization properties. The minimum description length principle has given rise to the idea that a program is a form of compact code, and that the job of a brain’s self-organization processes is to produce

compact codes ([3]). The Minimum Description Length principle has been typically implemented in EAA studies by introducing a complexity term that reflects network size into the fitness function which determines the selection of successful agents, e.g. [33]. In contradiction, the Self Organized Compact Encoding (SOCE) method introduced here does not introduce any explicit complexity term in the fitness function. The original fitness function is left intact, incorporating selective “pressure” towards minimal description length only *implicitly*.

Another implementation of the MDL principle in evolutionary computation have been *Variable length encoding* methods. The latter create compact genome representations by representing only essential neurons or synapses from an initially large network or using variable-size data structures ([16, 20, 25, 24]). Some methods contain a “switching” mechanism which turns synapses “on” and “off”. Only a certain subset of all possible neurons and synapses are represented at any given moment, and the fitness function includes an explicit complexity term which leads to smaller networks. In the SOCE method, in contrast, there is no such abrupt switching of representations. SOCE represents a wide range of approximation levels of neuronal representations, enabling a smooth evolutionary search through the space of variable length encodings. As such, it belongs to the more general class of indirect encodings ([4, 23, 19, 18, 31, 10, 30]), including *Grammar Encoding methods* ([15, 7, 17, 22]).

In general, indirect encoding methods search in smaller search spaces, aiming to increase the chance for finding good low-dimensional solutions for problems originally residing in high dimensional spaces. However, while de-

creasing the search space, indirect encodings usually restrict the set of possible solutions and thereby may fail to find good solutions. Indeed, some studies have favored large genome lengths over small ones in developing complex behavior of agents ([11, 12]). Devising indirect methods that selectively decrease the search space while maintaining a set of good solutions is hence an important challenge. The SOCE method addresses this challenge by adaptively varying the search space in a self organizing manner during the evolutionary process itself, seeking its best description. The advantages of *neutral genes* ([29, 28]) suggest that the basis of self-adaptation is the use of neutrality, allowing a variation of the search distribution without the risk of fitness loss. The SOCE method uses a random walk on neutral genes to create small solutions without a loss in fitness.

The rest of this paper is organized as follows: Section 2 describes the EAA model, the agents and the environment. Section 3 presents the SOCE method and section 4 describes the experimental results that testify to the benefits of SOCE and to its limitations. Section 5 discusses the implications of our research and suggests future directions.

2 The EAA Model

Agents directed by neurocontrollers are evolved and tested using SOCE and direct encoding schemes in the same environment. The agent structure and the environment are similar to those used by [2]. Agents are generated via an evolutionary process and perform a navigation and foraging task in a discrete grid 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 (see Figure 1). The food is concentrated in a limited zone in the southwest corner of the arena. Each agent has to eat as much food as possible and avoid poison items to maximize its fitness. It has to learn how 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 controller connected to its sensors and motors (Figure 1). It is the neural network controller that is genetically encoded and evolved, while the sensors and motors are given and constant. This neural network contains N internal neurons, L_{out} of them are also motor neurons, and L_{in} sensory input neurons. The neurons are simple binary McCulloch-Pitts neurons that use a zero-threshold activation function. That is, the neuron fires if its incoming field is positive, and is silent otherwise. Note that the outputs of motor neurons are recurrently fed back into the network. In the simulation used here the network has 5 input sensors and $10 - 50$ internal recurrently connected neurons, of which 4 are the motor neurons.

The agent receives very limited and local sensory information: Four sensors sense the grid cell which the agent is located in and the three grid cells immediately 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 food and poison. 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 neurons allow 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 about the environment, making navigation a challenging task. It has no information of its place coordinates or orientation in the arena. It must combine sensory inputs 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.

The findings of [2] show the emergence of one or a few “command” neurons that control two types of behavior patterns observed in the agents: *exploration*, that is seeking the food zone, and *foraging*, i.e. maximizing food consumption while inside the food zone. The “command” neurons serve as a switch between these two different behavior patterns.

3 Self Organized Compact Encoding

3.1 Overview

The majority of EAA studies apply a direct genotype-to-phenotype encoding to create the next generation of evolutionary evolved agents. In the task described in Section 2, the phenotype neurocontroller contains N internal neurons, L_{out} of them are also motor neurons, and L_{in} sensory input neurons. The direct encoding scheme employed by [2] in this task encodes the $L_{in} * N$ input synaptic values and $N * N$ internal synaptic values using a specific gene for each synaptic value, i.e. $L_{in} * N + N * N$ genes.

Instead of explicitly encoding *all* synaptic values, SOCE adaptively main-

tains only a restricted representation of the synaptic matrix. The genome is a list of N neuronal segments. Each segment encodes the incoming synapses of its corresponding neuron. The order of the neuronal segments in the genome list is arbitrary but fixed in all agents. Each neuronal segment contains a gene specifying the *encoding level* (EL) K of its neuron, and a list of K encoding parameter genes. These parameters are mapped using a *piece-wise interpolation transcription transform* to generate the $N + L_{in}$ incoming synaptic weights of each neuron. The SOCE method hence searches for the best solution in the space of synaptic matrices whose weights can be specified by a local, piece-wise interpolation performed separately for each neuron. Individual neurons in an agent’s neurocontroller may utilize different encoding levels of their synaptic connections. Each neuronal encoding level may vary from specifying uniform connection values (weights) to a one-to-one specification of all synaptic connection values. It undergoes evolution, adaptively varying in a self-organized manner. The neuronal encoding levels obtained in a given EAA network 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 are really required to solve the task. Since SOCE does not use a fitness function that explicitly encourages shorter solutions, compact solutions are obtained by a statistical (one-way bounded) random walk on neutral genes.

Having adaptive encodings yields agents with compact genotypes but their phenotypes may remain unnecessarily large. However, a variant of SOCE can lead to the emergence of small near-minimal phenotype neurocontrollers. This phenotypic reduction leads to two important consequences:

simpler controller networks that are more amenable for functional analysis, and an estimation of the complexity of the task in hand. That is, starting from arbitrary sized initial networks, SOCE can find a good neurocontroller solution with a small number of neurons, providing an upper bound on the number of neurons needed to solve the task.

3.2 The SOCE Algorithm : Genotypic Reduction

The genome of each agent is an interpolation-based encoding of the synaptic weights of the neurocontroller. 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 current population of genomes using a transcription transform.

The transcription transform works by serially processing every neuronal gene segment in the genome list. Denote by P the number of synaptic weights each neuron receives from the network in a full-recurrent state. For each neuronal segment denote by K the number of its synaptic encoding parameters, i.e. its encoding level. Let $G[i](i = 0, \dots, K - 1)$ be the parameter value written in the i 'th place in the genome from the beginning of this neuronal gene (segment). Let T_k denote the transcription transform with an encoding level K . Let $S[j](j = 0, \dots, P - 1)$ be the weight 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 of the neuron and uses them to generate its P synaptic values by a piece-wise linear interpolation. The synaptic values $S(j)$ are obtained via an interpolation between the values of the 2 “nearest” encoding parameters, with weights

proportional to the distance from each parameter (note that some of these interpolation weights may be zero). Figure 2 presents an illustration of this process. More formally, for a neuron with encoding level K , $1 < K < P$ the j 'th synaptic value is determined by

$$S(j) = \frac{G[P_{left}] * W_{left} + G[P_{right}] * W_{right}}{M} \quad (1)$$

where $M = \lfloor \frac{P}{K} \rfloor$ is the number of synaptic values created from each two “neighboring” encoding parameters, $P_{left} = \lfloor \frac{J}{M} \rfloor$ and $P_{right} = P_{left} + 1$. The interpolation weights $W_{left} = (M - |J - P_{left} * M|)$ and $W_{right} = (M - |J - P_{right} * M|)$ are given to each encoding parameter according to the distance along the interpolation line. In the limit cases, for encoding level $K = 0$ all incoming synapses of a neuron receive a zero value. For $K = 1$ they all receive a fixed value of the single genome location $G[0]$. For the encoding level $K = P$ we obtain a direct encoding - each location $G[i]$ in the genome encodes a unique synaptic weight value $S[j]$ in the controller neural network.

Since the encoding level K of each neuron is encoded in the genome, the agent adaptively distributes the description length between the neurons: some neurons receive a more detailed and longer encoding, while other neurons receive coarser (possibly zero-length) encodings. Note that when all neurons use an 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 fitness function governing 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 ($K = P$). Yet even in simulation experiments where the initial population contained full direct encodings for all the agents, the successful agents employed compact encodings. Good solutions to the task examined do not occupy a larger volume of the compact low-dimensional solution space than the volume they occupy in the initial, arbitrary high-dimensional problem space (as indicated by Figure 6, in which the mean fitness does not increase when searching smaller fixed size subspaces). The SOCE method hence is likely to adaptively vary the search space for each neuron, reaching small solutions while maintaining the fraction of “good” solutions over the search space and the overall fitness.

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

1. *Similarity Conservation* - phenotypes created from similar genomes should 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 the phenotypic level, resulting in an erratic, almost random, evolutionary process.
2. *Completeness* - the set of phenotypes that can be created from a population of genomes should span extensive parts of the phenotypic space. This requirement assures that good solutions are reachable.

SOCE maintains the second property by its definition, through the self-organization and variation of encoding levels. Most phenotypes can be rep-

resented to a certain approximation; the transform dynamically changes the encoding level, reaching direct encoding in the limit if required. The first condition, similarity conservation, requires a closer look at the impact of various genetic variations when using SOCE. It is examined in depth in section 4.5.

3.3 Phenotypic Reduction with SOCE

In the basic SOCE, the encoding level K of each neuron is initialized with a random genotypic value (from a uniform distribution) in the interval $[-1, 1]$, which is mapped to a phenotypic encoding level in the interval $[0, 1, 2, \dots, P]$. During evolution, values larger than 1 are interpreted as encoding level P and values lower than -1 as encoding level 0. Obviously, a neuron with encoding level $K = 0$ will never transmit any information to other neurons because it receives zero, sub-threshold inputs. This enables the usage of SOCE for *phenotypic reduction*: finding near minimal phenotypic neurocontroller solutions. Starting from a possibly large initial network, SOCE will retain only a small number of *viable neurons* (i.e. with $K > 0$) to construct the agents' controller network. During the evolutionary search the neuron's encoding level may also increase and thus a neuron can regain its viability in a single mutation. When an encoding level is increased uniformly distributed random values are used to instantiate the new encoding parameters that are added. The phenotypic encoding level is made to vary in the interval $[-L, P]$, where $-L$ is a negative lower bound (negative encoding levels are transcribed to an encoding level of $K = 0$, that is, all incoming synapses are zero). Interestingly, if the lower bound L is negative enough, unimportant redundant neurons will generally obtain an encoding level of $K \leq 0$ (see Section 4).

4 Results

SOCE was tested by running evolutionary experiments in the model of [2] described in Section 2. A population of 100 agents was evaluated in each generation, performing a navigation and foraging task. Selection, mutation and crossover are performed to create the next generation. Simulations were typically run for 10,000 or 20,000 generations. The various parameters of the evolutionary process in SOCE were kept similar to those used by the direct encoding benchmark, to focus on the effects of using different encoding schemes on the simulation results. The mutation rate was 0.02 for each gene in the genome. A mutation changes the gene by adding a random number between -0.6 and 0.6 to its current value. Uniform crossover rate was 0.04. Elitist selection was performed: The top 10% of the most successful agents did not undergo mutations and crossover, and were transferred as is to the next generation. The rest of the agents in the next generation are selected from the current population with a selection probability proportional to their fitness and undergo mutations and crossover. The neurocontrollers contained 10-50 neurons in the initial generation, created from the genotypic representations using the SOCE transcription transform. Fitness was measured by the number of food items consumed by the agent minus the number of poison items it consumed. The fitness scores are normalized, dividing by the number of food items in the arena (30). Due to the limited lifespan of the agent and limited sensory information it receives, a normalized fitness value above 0.3 is considered good performance [2].

Our aim has been to study the ability of SOCE to obtain the following goals:

- Solving navigation and foraging tasks with compact genomes. Small genome sizes are reached implicitly, maintaining the original fitness function.
- Obtaining small phenotype neurocontrollers containing a near minimal number of neurons for a given task.
- Simplifying neuronal analysis by generating small networks.

4.1 Genotypic Reduction

Evolutionary experiments of the task described above were conducted with SOCE agents and direct-encoded agents. Figure 3 shows that SOCE agents can obtain performance levels similar to those obtained by direct encoded agents, both starting with neurocontrollers containing 10 neurons. The best SOCE agents are able to solve the navigating problem with a genome length of about 22% of the direct encoding genome length of the neurocontrollers studied in [2], which is 150 for 10 neurons with 15 synapses each, as shown in Figure 4. The distribution of encoding levels across different neurons in evolved SOCE agents is shown in Figure 5: Output neurons receive high encoding levels, and the agents maintain a small number of viable internal neurons with a non-zero encoding level. Each agent may maintain different internal neurons because their order is arbitrary, and therefore internal neurons tend to have small encoding levels with a large standard deviation. Overall, the distribution of encoding levels in a given agent is typically sharply bimodal. Most of the neurons obtain a vanishing encoding level, and most of the remaining neurons take a high encoding level value.

The importance of dynamic modification of *individual* encoding levels was studied using a baseline benchmark in which all the population had a *fixed, uniform* encoding level K . In this setting the agents were able to solve the problem employing large K values, but with smaller K values many runs did not yield successful solutions and the mean fitness dropped, as shown in Figure 6. The performance of the best agent is less affected by lowering the fixed encoding level. In this figure the maximum value may be smaller than one standard deviation above the mean because of the skewed nature of the fitness distribution. Lowering the uniform encoding length under 60% of the direct encoding size results in a fairly significant decrease in fitness. In comparison, SOCE agents are able to reduce the genome size up to 22% of its original size in the same task while maintaining high levels of performance. These results testify to the importance of the dynamic nature of SOCE, where each neuron is free to select its own encoding level.

4.2 Phenotypic Compactness : Near Minimal Neuro-controllers

Using SOCE with encoding levels $[-L, N]$ (see section 3.3) leads to the self-organized elimination of unimportant neurons, creating near minimal size networks. This occurs because, after the incorporation of negative bounds, the encoding levels of unimportant neurons tend to vanish and their corresponding neurons can be eliminated.

Starting from initially large networks composing the initial population, containing 30 or 50 neurons, the SOCE algorithm still reaches near minimal networks. In both cases the evolutionary process obtains solution with good

performance using only 4 – 5 viable neurons, as shown in Figure 7. The relatively narrow standard deviations testify that the SOCE algorithm is robust in its ability to converge to near minimal networks starting from fairly large neurocontrollers.

The small neurocontrollers evolved are much more amenable for analysis of their operation than larger neurocontrollers. Typically, a successfully evolved SOCE agent had a network of only 3 – 4 neurons, with the following key 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, typically only one has a non-zero encoding level and remains viable in the final controller network. This resembles the findings of very few important “command neurons” in [2]. The latter switch between two modes of behavior: navigation (searching for the food zone) and foraging (while inside the food zone). In some SOCE runs successful agents with only 3 motor neurons evolved, using no internal neurons (as, for example, the agent whose encoding levels are depicted in Figure 5(a)).

These observations testify to the minimality of the neurocontroller solutions found by SOCE. It is hard to conceive a successful solution to the task

in hand without including at least three motor neurons - one for moving forward, one for turning and one for eating. In addition, at least one command neuron that switches the agent’s behavior modes from exploration to foraging is helpful in many of the evolved solutions. The natural drift process of the encoding levels of unimportant neurons in SOCE almost always ends in the lower bound. Evidently, the statistical one-way bounded random walk itself pushes towards small encoding level values.

An important advantage of the SOCE method for obtaining small neurocontrollers is the implicit manner in which the self organization occurs, without any need to add explicit minimization terms to the fitness function. Modifying the fitness function by adding a cost term representing the size of the network may turn to be a nontrivial task in certain problems. Its success may delicately depend on careful selection of the exact parameters and functions used in the minimization term.

Figure 12 shows the results of such an explicit minimization experiment: Agents were evolved to solve our basic foraging task, adding an explicit cost term to the direct encoding. This term adds a “reward” of fixed value to the fitness of the agent for each inactive synapse (A synapse is considered inactive if its absolute value drops below 0.05 (recall that synaptic values range between [-1..1])). For small reward terms the evolutionary process “ignores” the cost of large size and the networks remain large. For large reward terms the agents fail to achieve high foraging performance levels, and even the network size fails to reach small values because the algorithm converges after a small number of generations. In the transition region between these two regimes, the runs end stochastically in one of these two distinct outcomes,

but no run was able to decrease network size significantly while maintaining reasonable fitness in our foraging task.

4.3 Analysis of SOCE Dynamics

4.3.1 The Importance of Synaptic Compression

Neurocontroller reduction with the SOCE algorithm is composed of two basic elements: elimination of neurons which reach a zero encoding level, and the interpolation encoding of synaptic values. A simple “neuronal switching” encoding was used as a benchmark to dissociate which of these two basic elements contributes to the emergence of successful small networks. This benchmark uses direct encoding of synaptic values and a “switch” that can deactivate or activate the genotypic representation of each neuron, regulating its phenotypic transcription. This encoding hence dissociates the effect of elimination and revival of neurons from the use of interpolation encoding. As shown in Figure 8, the neuronal switching encoding was incapable of achieving near minimal networks when solving our foraging task. Neurocontroller sizes start to decrease but reach a plateau above half of the initial network size. This result implies that the use of the interpolation encoding of synaptic values is crucial for reaching the near minimal neurocontrollers obtained by SOCE. It is likely that the gradual decrease in encoding levels in SOCE agents serves to “smooth” the fitness landscape and allows the agents to reach much smaller networks.

This observation is reminiscent of the findings of [13], in which random learning was used to smooth the fitness landscape searched by a genetic algo-

rithm. Hinton and Nowlan discussed a problem in which the fitness landscape is extremely flat, with only one sharp peak, making it a problem difficult to solve with standard genetic algorithms. They show that an addition of a primitive form of “learning” in which random changes are made to some of the genes during the search “smoothes” the fitness landscape, giving a better chance for reaching a good solution. The SOCE algorithm accomplishes the “smoothing” of the fitness landscape using a different mechanism, *on an evolutionary level*. The gradual decrease of encoding levels of each neuron enables the evolutionary process to eliminate only those neurons which are really unimportant. Neural elimination is dependent on the consequences of gradually decreasing the neuron’s synaptic encoding level over the generations. Thus, only after reaching a low encoding level, the final neuronal elimination (i.e., setting the encoding level to zero) is safely done. In contrast, in the “neuronal switching” encoding, the elimination of neurons is abrupt and occurs indiscriminantly with equal probability in all neurons. This leads to a much smaller chance of finding good compact encodings.

4.3.2 Neuronal Elimination in SOCE

To better understand the nature of neuronal elimination in SOCE, we ran a contribution analysis [1, 26] on the best neurocontrollers obtained along the evolutionary process. Such a contribution analysis of a given neurocontroller computes a contribution value, i.e., a measure of “importance” to each neuron in the neurocontroller, denoting the precise neuron’s value to the overall performance (and hence survival) of the agent. Our goal in running this analysis was to examine the correlation between the contribution values of the

neurons (as computed by the contribution analysis) and their corresponding encoding levels (determined by SOCE), as an indication of how well does SOCE succeed in eliminating the unimportant neurons and retaining the important ones.

In our contribution analysis, we used a novel method [14] that is more precise and efficient than the one presented previously in [1, 26]. This method, the Multi-lesion Shapley value Analysis (MSA), is based on a sound axiomatic foundation using the concept of Shapley value from Game theory [27] to construct a fair and canonical solution to the classic problem of attributing contribution values to the individual system elements. The solution found is unique and in essence, optimally fair. The MSA method processes a dataset of multi-lesion experiments and the corresponding performance level (the fitness) of the agent measured after each such multi-lesion is induced. In each such multi-lesion experiment a few neurons are “lesioned”, that is their effect on the other network elements is abolished, and the other “un-lesioned” neurons are left intact (except for the loss of the inputs from the lesioned neurons). Given this data the MSA yields the precise contribution of each neuron to the agent’s survival. While in traditional Game theory the Shapley value is more a theoretical tool which requires full knowledge of the behavior of the system under all possible lesioning configurations, [14] have developed a novel method to calculate it from relatively small samples of the lesioning space, making this method of contribution analysis a practical tool for the analysis of evolved large neurocontrollers (in the current analysis, however, the neurocontroller is rather small and the full set of all multi-lesion configurations was used).

Figure 13 plots the correlation coefficient between the contribution values (CVs) and encoding levels (ELs) of the neurocontroller’s neurons at every 1000 generations as evolution proceeds. The evolution was performed in our basic foraging environment, and the best agent in each step of a 1000 generations was examined. As evident, there is a fairly high level of correlation between the CV and EL values of each neuron. Interestingly, this correlation tends to increase as evolution proceeds. These results support the notion that SOCE successfully operates to preserve the important neurons during evolution, and eliminates the unimportant ones. However, obviously, one should keep in mind that it is a correlational measure and not a causal test. That is, it is likely that in reality a complex feedback loop takes place during a SOCE-guided evolution; as the EL of less important neurons is decreased, they become even less functional and important, making them more “vulnerable” to further EL-decrease, and so on. The increase of the correlation during evolution testifies to the powerful effect of this feedback “coupling”.

4.4 Extended Foraging Tasks

SOCE was tested on two more complicated tasks, which are extensions of the original task we have studied. The first task contained two food zones located at two opposite corners of the arena instead of just one corner in the original task. In the second task the agent’s food consuming procedure became more complex: the agent had to stand on the food item without moving for precisely two steps in order to gain its fitness increase. The SOCE agents were able to achieve small networks both in mean encoding level and in the number of viable neurons in the network, while maintaining

the same fitness levels obtained with direct encoding, as shown in Figures 10 and 11. The slight increase (about one neuron) in the final network size can be referred to the more complex memory mechanism needed for counting the waiting steps during the eating procedure in the second task, and the additional behavioral switch between food zone exploration that is required in the first task.

4.5 Similarity Conservation

An important characteristic of an efficient genotype-to-phenotype encoding is its ability to maintain the similarity conservation property: Phenotypes created from similar genomes should be similar to each other in the phenotypic space. This property maintains a consistent evolutionary process and increases the robustness of the results. After studying the SOCE method in depth, let us now revisit and examine its similarity conservation properties. To this end, we calculate its Similarity Distortion Factor (SDF): This is the ratio between the mean change per synapse in the phenotype (C_p) and the mean change in the genome (C_g), when applying the genetic variation operators used in our simulation to obtain the next generation agents, where

$$C_p = \frac{\sum_i \sum_j |W'_{ij} - W_{ij}|}{(N + L_{in}) * N} \quad (2)$$

$$C_g = \frac{\sum_i \sum_g |G'_{ig} - G_{ig}|}{N_g} \quad (3)$$

and

$$SDF = \frac{C_p}{C_g} \quad (4)$$

where W_{ij} and W'_{ij} are the synaptic values before and after applying genetic variations, G_{ig} and G'_{ig} denote the encoding parameters before and after applying genetic variations, N_g is the total number of genes representing the network in the encoding and the summations run over all synapses and genes, correspondingly, i.e., i running over all neurons, and $j(g)$ over all the synapses (genes) per neuron.

Direct encoding agents maintain an SDF equal to 1 by definition, since each gene directly encodes a single synaptic value. Figure 9 shows that SOCE agents have SDF levels larger than one, increasing during the evolutionary process, but maintaining values bounded under 3.

To understand these results let us examine the effect of different genetic variations under SOCE. In direct encoding a mutation in one gene changes a single synaptic value with a uniform distribution $U(-0.6, 0.6)$. In SOCE one can distinguish between two types of mutations by their effect on the phenotype: a mutation in a synaptic encoding parameter gene, and a mutation in an encoding level gene. When a mutation occurs in a synaptic encoding gene of a neuron with a reduced encoding level, it effects more than one synaptic value, but the change to each individual value is smaller than that caused by a similar mutation in a direct encoding, due to the piece-wise linear interpolation. The number of synaptic changes remains the same as in direct encoding (more synapses are affected by a change in a single gene, but there are less genes since the representation is shorter), but the mean magnitude of each change is smaller. This kind of mutation thus tend to decrease the

SDF. However, a mutation in the encoding level of a neuron causes changes in all the synapses of this neuron. This kind of mutation increases the SDF of the agent above 1. Moreover, larger phenotypic changes occur in advanced agents, when the encoding levels become smaller. These changes are small in magnitude, but are spread over many phenotypic synapses. SOCE agents with dynamic encoding levels employ both kinds of mutations, overall resulting in SDF values larger than 1. These values increase during the evolutionary process due to the continuing decrease in the mean encoding level of the agent’s neurons. Still, SDF values remain quite small and do not diverge, maintaining fair levels of similarity conservation.

5 Discussion

The SOCE algorithm was shown to successfully solve a simple navigation and foraging task, while providing compact solutions. Via the SOCE method the agents select compact encoding levels without any explicit fitness pressure. The self-organizing dynamics of SOCE lead to an adaptive focus of the evolutionary search on relevant subspaces, using a statistical one-way bounded random walk on neutral genes. This reduction of genome size can be utilized for the minimization of network size. 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 then reduced during the evolutionary process to a near-minimal size.

SOCE can also be used as an analysis method: The encoding level of each neuron is a good clue for its importance and “input sensitivity”. More-

over, since the objects of the analysis are now near-minimal networks, the insights gained probably better reflect the true, inherent nature of successful solutions to the task in hand than that of a 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 the neurons’ “importance” in the network. This is so because the encoding level 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 to the evolved agents, as in [5, 6]. Such synaptic learning rules can be encoded using the SOCE method. The agent then decides what encoding levels should be allocated to the data (the initial synaptic weights) vs. the theory (the synaptic learning rules). Combining data and theory using SOCE could yield an implicit comprehensive framework for finding a minimal description length solution to a task, with an efficient tradeoff between data and theory. When learning is applied, changes in synaptic weights during the agent’s lifetime may be inserted back into the genome of descendants in a Lamarckian way. In this framework SOCE could then be viewed as a *compression method*, maintaining a compact representation of learned synaptic weights.

The SOCE method utilizes a local encoding method (piecewise interpolation) to obtain a similarity conserving genotype-to-phenotype mapping. This however may be modified in future studies. Global encodings that conserve similarity less than local ones may be incorporated into the SOCE scheme. Interestingly, SOCE lends itself quite naturally to incorporate a global en-

coding (e.g., a Fourier series representation of the synaptic values in the network). This may lead to more compact encodings and perhaps to more robust neurocontrollers.

In summary, we presented a new genotype-to-phenotype encoding method based on adaptive, self-organizing interpolation. This method provides efficient and compact solutions to EAA tasks, starting from a population of inefficient and arbitrarily large network solutions. It provides an estimation of the complexity of a task, and grades the importance of neurons composing the emerging neurocontrollers.

References

- [1] R. Aharonov, I. Meilijson, and E. Ruppin. Localization of function via lesion analysis. *Neural Computation, To appear*, To appear.
- [2] R. Aharonov-Barki, T. Beker, and E. Ruppin. Emergence of memory-driven command neurons in evolved artificial agents. *Neural Computation, 13, pages 691-716*, 2001.
- [3] D.H. Ballard, editor. *An introduction to Natural Computation*. The MIT Press, Cambridge, Massachusetts, 1997.
- [4] D. Dasgupta and D.R. McGregor. Designing application-specific neural networks using the structured genetic algorithm. *Proceedings of COGANN-92 (Internat. Workshop on Comb. of Genetic Alg. and NN), 1992, Ed. Whitley and Schaffar, IEEE Computer Society Press*, 1992.
- [5] D. Floreano and J. Urzelai. Evolution of plastic control networks. *Autonomous Robots, 11, 311-317*, 2001.
- [6] D. Floreano and J. Urzelai. Neural morphogenesis, synaptic plasticity, and evolution. *Theory in Biosciences, 120, 225-240*, 2001.
- [7] F. Gruau. Automatic definition of modular neural networks. *Adaptive behavior, 3, pages 151-183*, 1994.
- [8] A. Gulliot and J.A. Meyer. From SAB94 to SAB2000: What's new, animat? *Proceedings of the Sixth International Conference on Simulation of Adaptive Behavior. MIT Press, Cambridge, MA*, 2000.

- [9] A. Gulliot and J.A. Meyer. The animat contribution to cognitive systems research. *Journal of Cognitive Systems Research (2)* , pages 157-165, 2001.
- [10] S.A. Harp, T. Samad, and A. Guha. Towards the genetic synthesis of neural networks. *Proceedings of the third International Conference on Genetic Algorithms and Their Applications*, pages 360-369, 1989.
- [11] I. Harvey. Evolutionary robotics and saga: the case for hill crawling and tournament selection. In C.Langton, editor, *Artificial Life III, Santa Fe Institute studies in the Sciences of Complexity, Proc. Vol. XVI*, pages 299-326, 1993.
- [12] I. Harvey, P. Husbands, D. Cliff, A. Thompson, and N. Jakobi. Evolutionary robotics: The sussex approach. *Robotics and Autonomous Systems, 1996*, 1996.
- [13] G.E. Hinton and S.J. Nowlan. How learning can guide evolution. *Complex Systems 1*, 495-502, 1987.
- [14] A. Keinan, I. Meilijson, and E. Ruppin. Fair attribution of contribution: The Shapley value analysis. *Neural Computation*, Submitted.
- [15] H. Kitano. Designing neural networks using genetic algorithms with a graph generation system. *Complex System, 4*, pages 461-476, 1990.
- [16] K. Kobayashi and M. Ohbayashi. A new indirect encoding method with variable length gene code to optimize neural network structures. *Proceedings of 1999 International Joint Conference On Neural Networks (IJCNN99)*, Vol. 6, pp. 4409-4412., 1999.

- [17] J. Kodjabachian and J-A. Meyer. Evolution and development of modular control architectures for 1-d locomotion in six-legged animats. *Connection Science*, 10, pages 211-254, 1998.
- [18] P. Koehn. Genetic encoding strategies for neural networks. *Proceedings of Information Processing and Management of Uncertainty in Knowledge-Based Systems 1996, Granada, Spain, Volume II*, pages 947-950. citeseer.nj.nec.com/koehn96genetic.html, 1996.
- [19] S. Luke and L. Spector. Evolving graphs and networks with edge encoding: Preliminary report. *Late Breaking Papers at the Genetic Programming 1996 Conference Stanford University*, pages 117-124, 1996.
- [20] J.J. Merelo, A. Prieto, and F. Moran. Design of neural classifiers using variable-length genetic algorithms. citeseer.nj.nec.com/merelo95design.html, 1995.
- [21] J.A. Meyer and A. Guilliot. 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*, 1994.
- [22] S. Nolfi and D. Parisi. Evolving artificial neural networks that develop in time. *European Conference on Artificial Life*, pages 353-367, 1995.
- [23] J. Pujol and R. Poli. Efficient evolution of asymmetric recurrent neural networks using a PDGP-inspired two-dimensional representation. *Proceedings of the First European Workshop on Genetic Programming (EU-*

- ROGP*), volume 1391 of *Lecture Notes in Computer Science*, pages 130-141, 1998.
- [24] J. Rissanen. Modeling by shortest data description. *Automatica*, 14:465-471, 1978.
- [25] N.N. Schraudolph and R.K. Belew. Dynamic parameter encoding for genetic algorithms. *Machine Learning*, 9(1): 9-21, 1992.
- [26] L. Segev, R. Aharonov, I. Meilijson, and E. Ruppin. High-dimensional analysis of autonomous agents. *Artificial Life*, To appear.
- [27] L.S. Shapley. A value for n-person games. In H. W. Kuhn and A. W. Tucker, editors, *Contributions to the Theory of Games*, volume II of *Annals of Mathematics Studies 28*, pages 307-317. Princeton University Press, Princeton, 1953.
- [28] R. Shipman and M. Shackleton. Issues in designing a neutral genotype-phenotype mapping. *2002 Congress on Evolutionary Computation*, 2002.
- [29] M. Toussaint and C. Igel. Neutrality: A necessity for self-adaptation. *Proceedings of the Congress on Evolutionary Computation (CEC 2002)*, 2002.
- [30] D. Whitley and C. Bogart. The evolution of connectivity: pruning neural networks using genetic algorithms. *Proceedings of the International Joint Conference on Neural Networks, Vol. I*, pages 134-137, 1990.

- [31] D. Whitley, T. Starkweather, and C. Bogart. Genetic algorithms an neural networks: optimizing connections and connectivity. *Parallel Computing*, 14: 347-361, 1990.
- [32] X. Yao. Evolving artificial neural networks. *Proceedings of the IEEE*, 87(9).pages 1423-1447, 1999.
- [33] B.T. Zhang and H. Muhlenbein. Evolving optimal neural networks using genetic algorithms with Occam's Razor. *Complex Systems*, 7, pages 199-220, 1993.

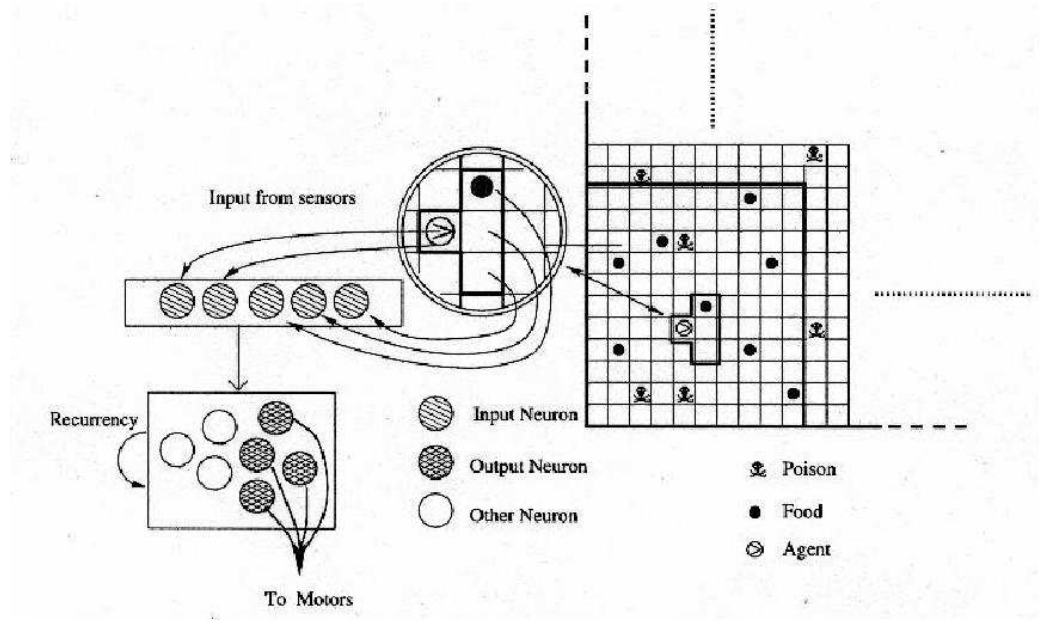


Figure 1: The agent and the environment. The agent is marked by a small arrow on the grid, whose direction indicates its orientation. (adopted from Aharonov-Barkai, 2001).

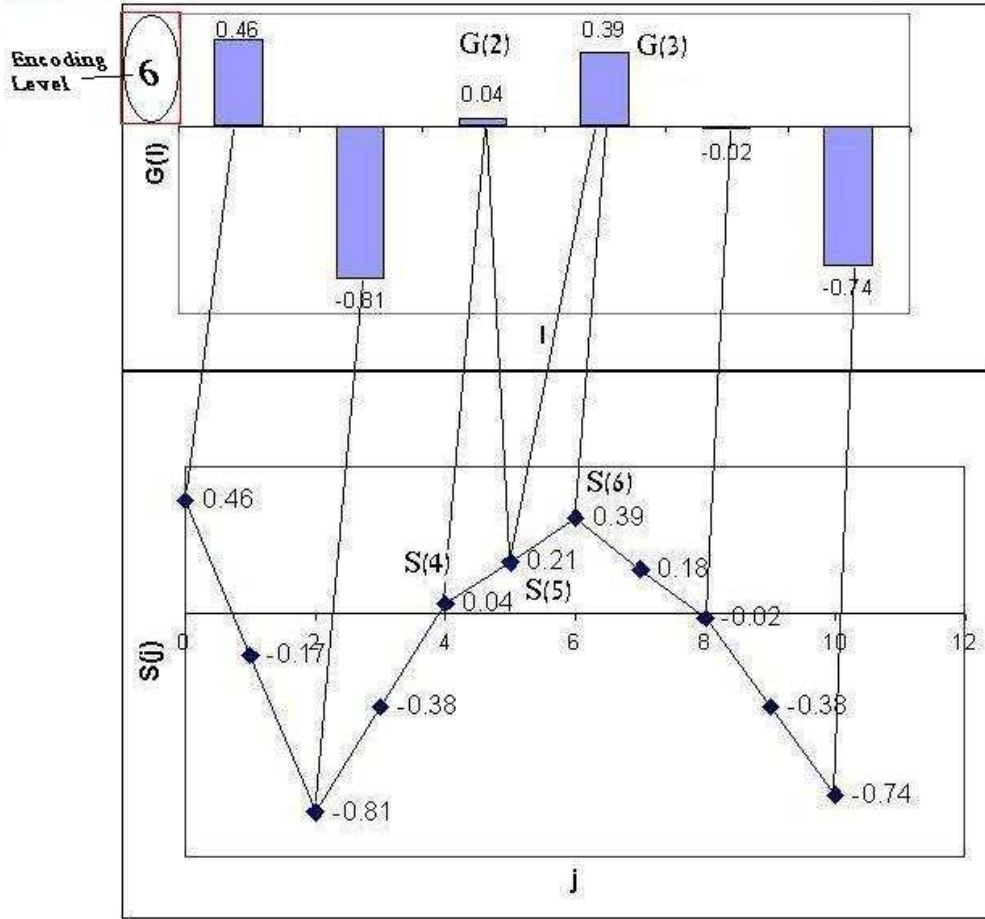


Figure 2: An example of the SOCE genotype-to-phenotype transcription transform T_K : The top figure shows the 6 encoding parameters of a neuronal gene segment (one of many in the agent's genome) that are transcribed to form its 11 synaptic weights shown in the bottom figure. For example, synapses $S(4)$ and $S(6)$ receive their values directly from the corresponding encoding parameters $G(2)$ and $G(3)$, while $S(5)$ is determined by interpolating $G(2)$ and $G(3)$. The encoding level of the neuron is 6 (encoded in the initial part of the segment). Just a small fraction of the 'edges' designating genotype-to-phenotype transcription are shown in the figure, for clarity.

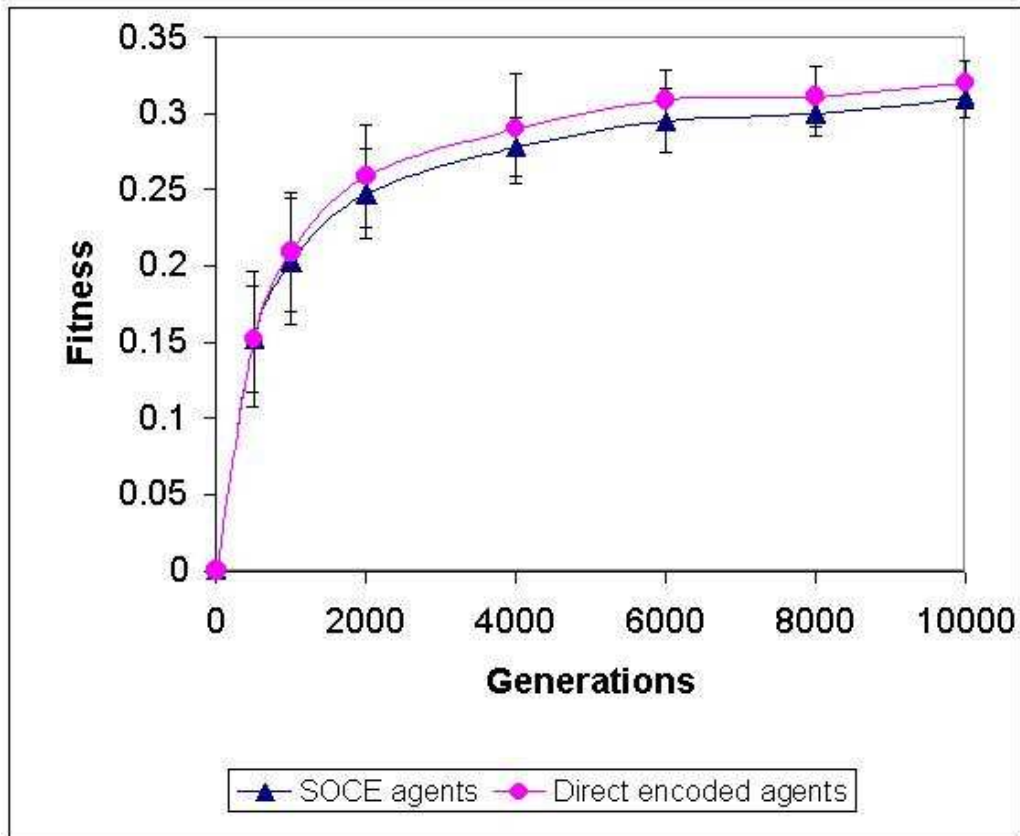


Figure 3: Fitness of direct-encoding versus SOCE agents, plotted across generations of evolution. The mean and standard deviation of 20 experiments are shown.

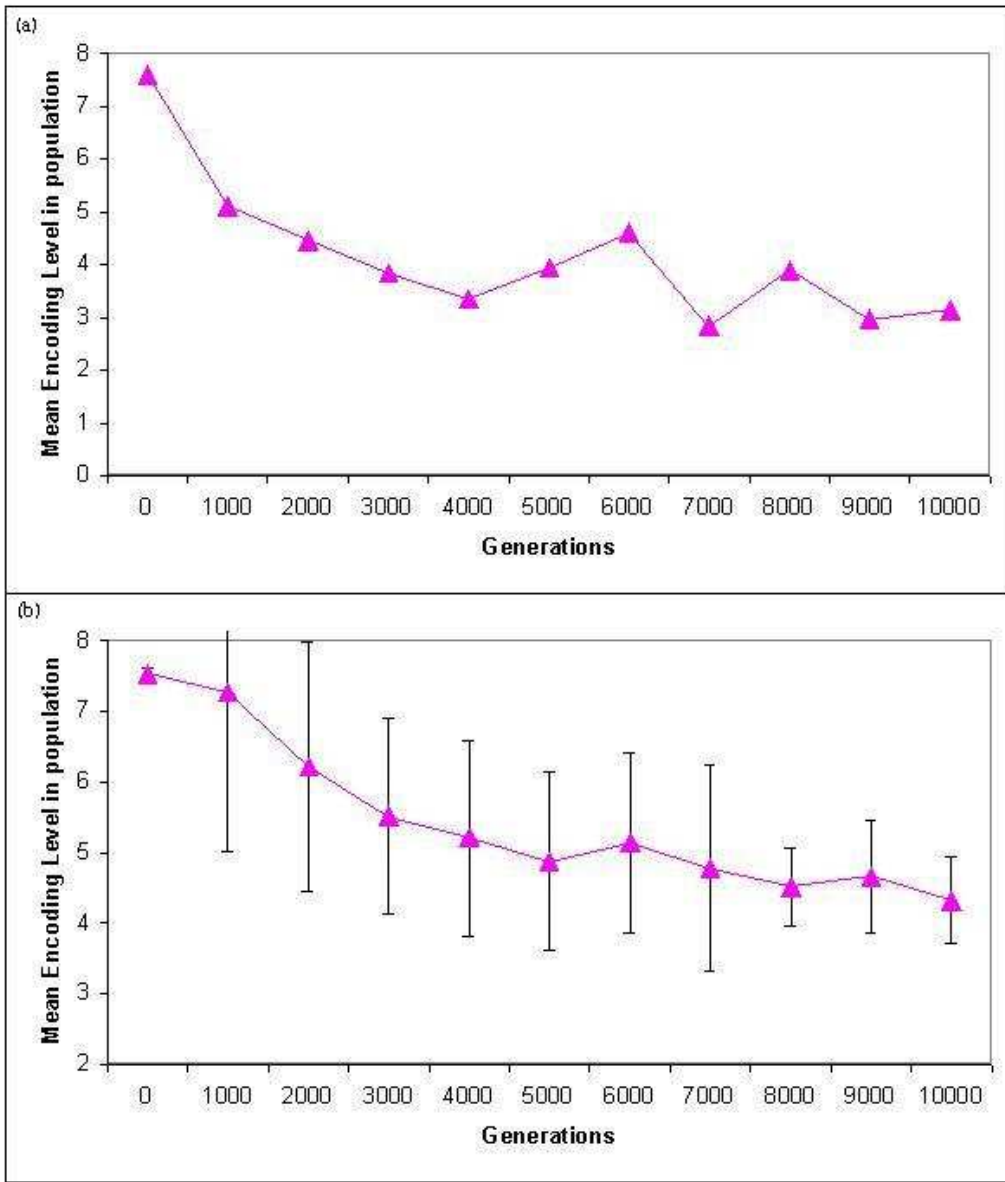


Figure 4: Mean encoding level in a population of 100 agents throughout simulation runs of the SOCE method. The best run (a) and averages over 10 runs (b) are shown.

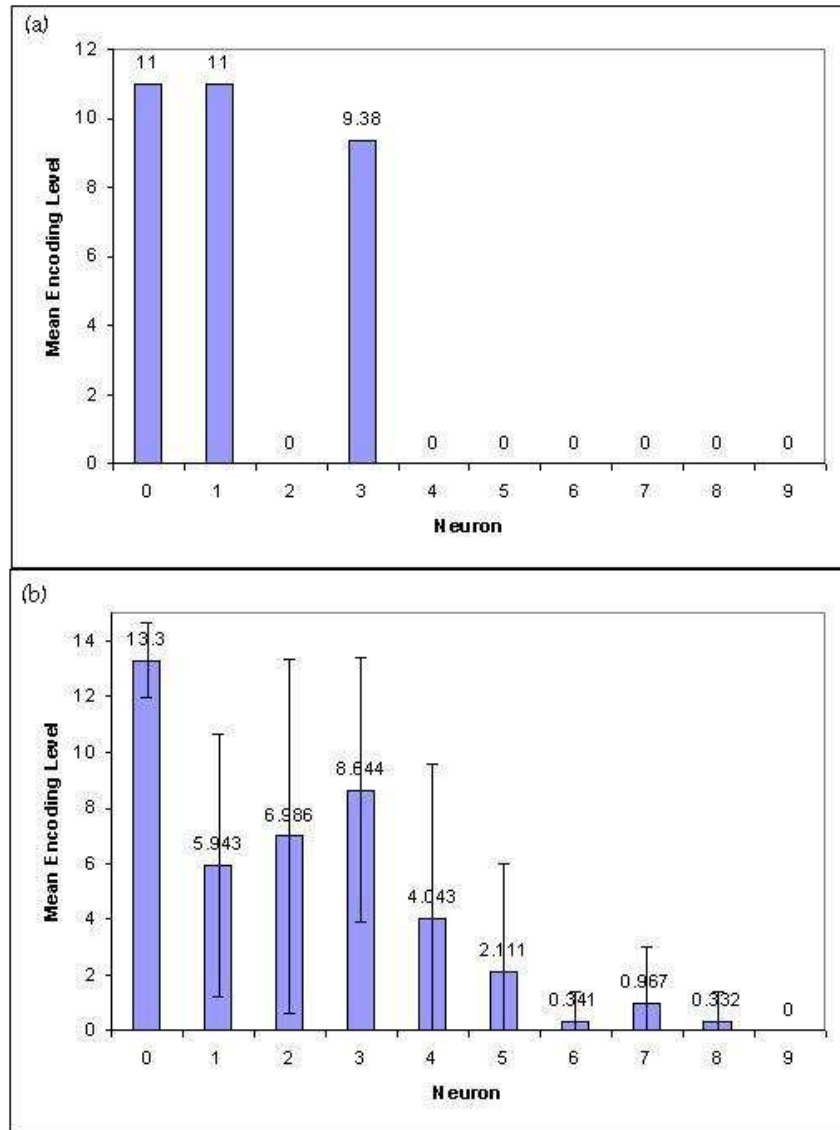


Figure 5: Distribution of encoding levels between neurons in the neurocontrollers of successfully evolved SOCE populations. The first 4 neurons are motor neurons. Results for the best population (a) and averages and standard deviations of 10 runs (i.e., from different lineages) (b) are shown. Each neuron receives 5 input synapses and 10 internal synapses, thus the maximal encoding level is 15.

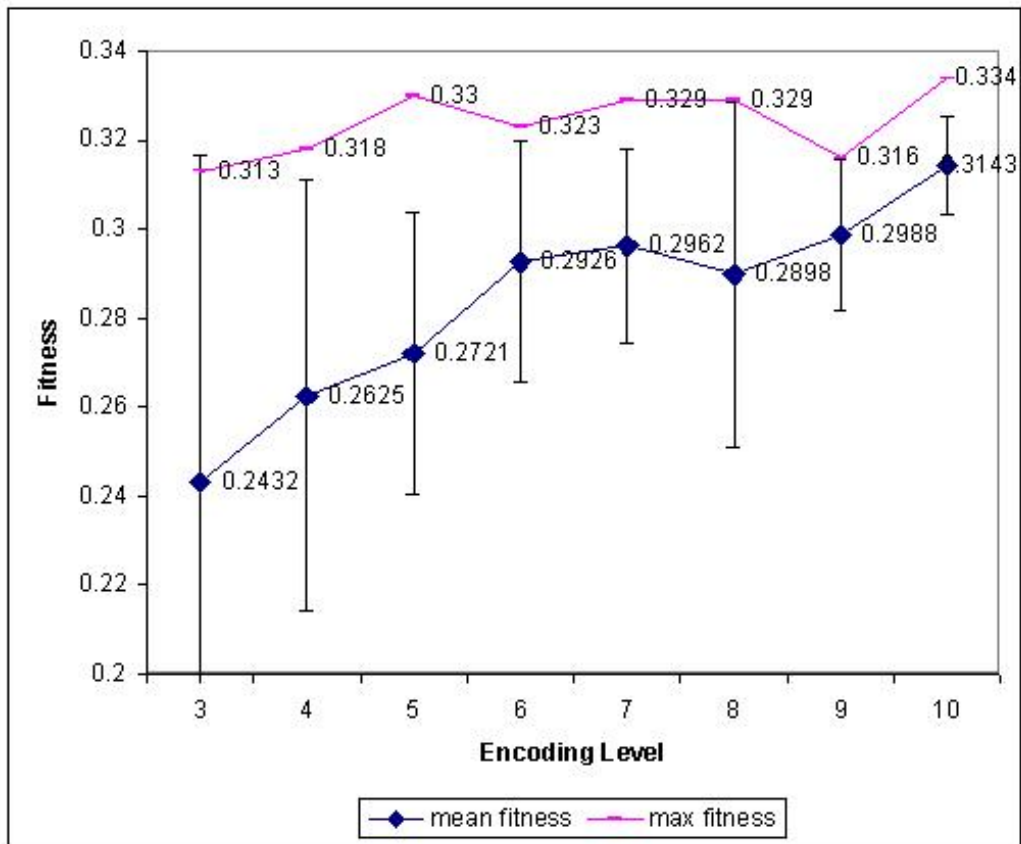


Figure 6: Fitness of agents using a compact encoding method with a fixed, *uniform* encoding level for all neurons in the population, as a function of the encoding level. Mean, maximum and standard deviations of 10 runs are shown.

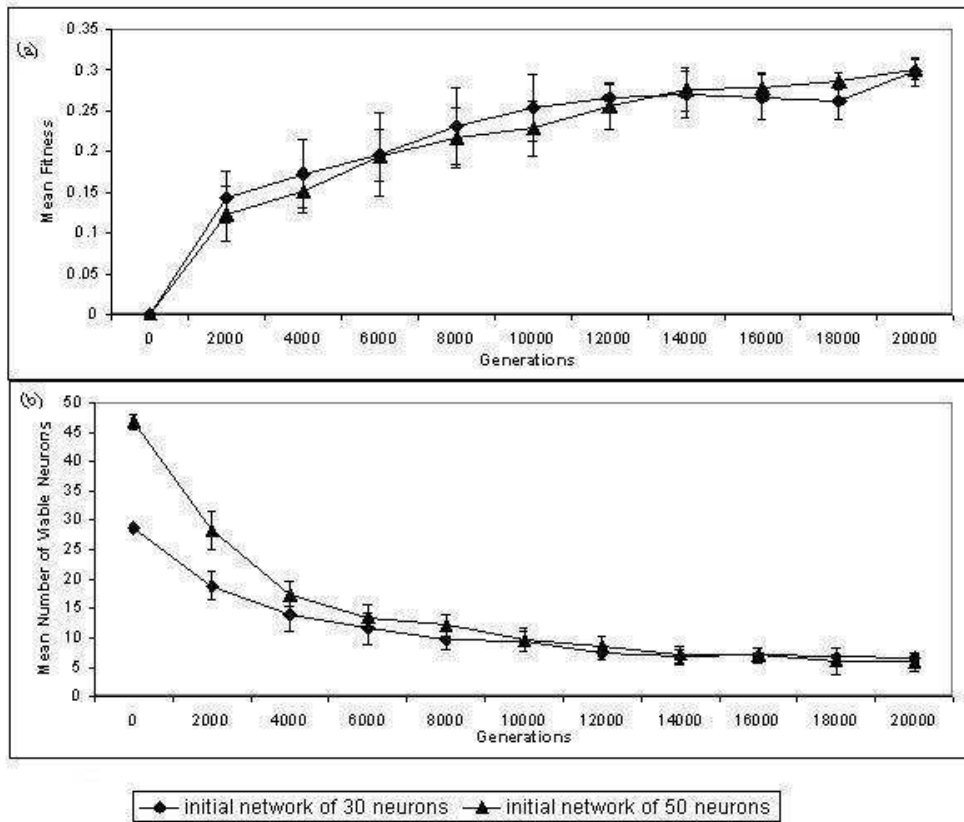


Figure 7: Evolutionary process in SOCE neurocontrollers starting from large initial networks containing 30 and 50 neurons. (a) Mean Fitness of 10 runs. (b) Mean number of viable neurons (neurons with a non-zero encoding level) averaged over 10 runs.

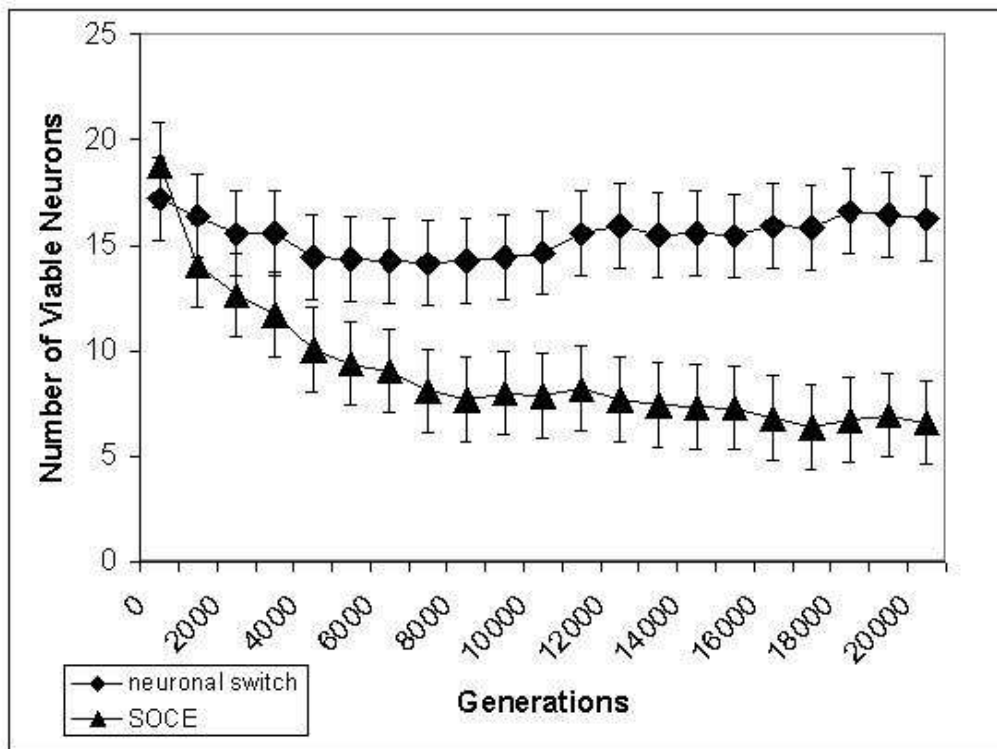


Figure 8: Mean number of viable neurons throughout the evolutionary process of SOCE and “neuronal switch” agents, averaged over 20 runs. In both cases, the initial size of the neurocontrollers is 20 neurons.

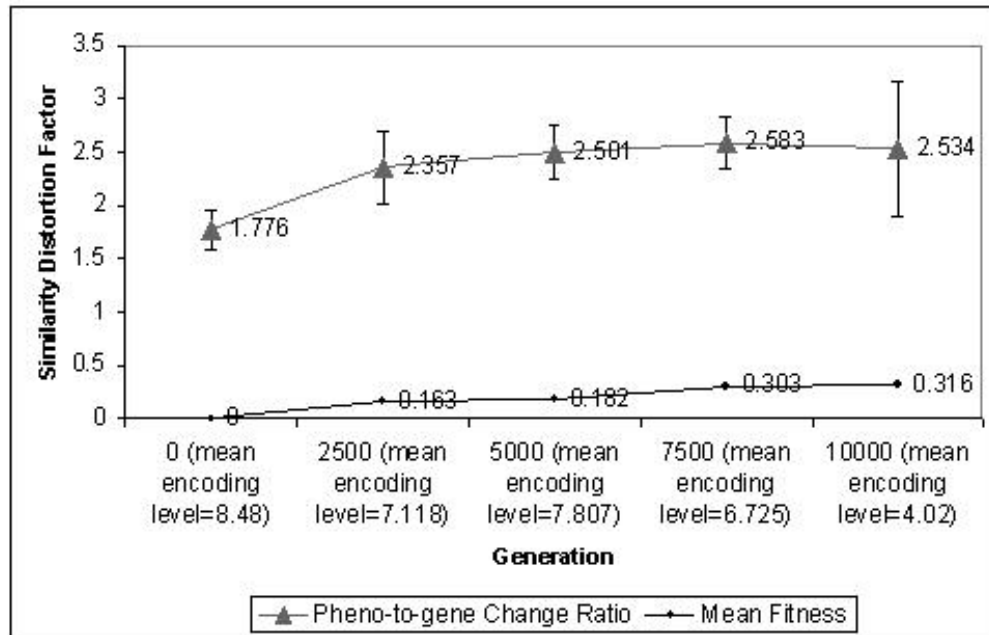


Figure 9: Similarity Distortion Factor in SOCE agents during the evolutionary process, given with the fitness at each stage. For each evolutionary stage the STD was calculated by applying 1,000 genetic mutations to the genome of each agent, averaged over the 100 different agents in that generation.

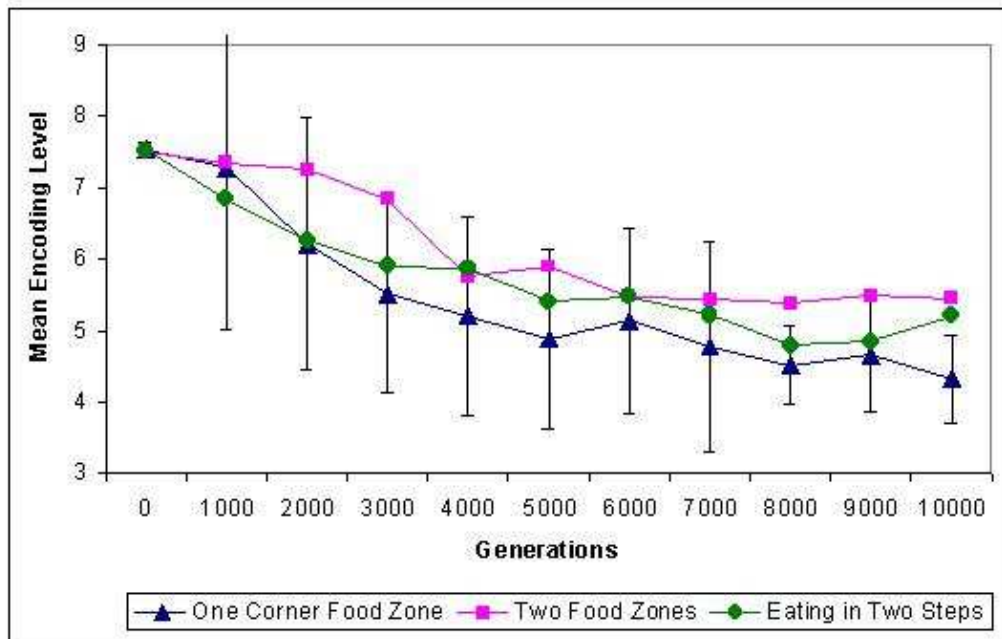


Figure 10: Mean encoding level of neurons in a SOCE population during the evolutionary process, in extended tasks. Means of 10 runs of the original task (with one food zone), an extended task with two food zones, and a task in which eating is performed only when the agent stays still for two steps on the food item are presented.

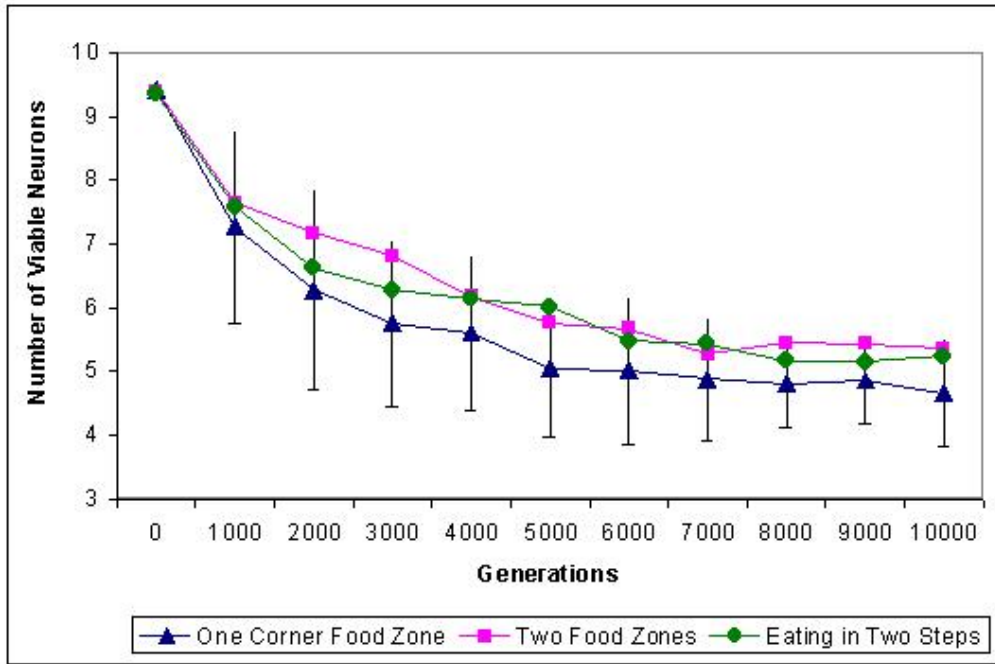


Figure 11: Number of viable neurons (with a non zero encoding level) in a SOCE population during the evolutionary process, in extended tasks. Means of 10 runs of the original task (with one food zone), an extended task with two food zones, and a task in which eating is performed only when the agent stays still for two steps on the food item are presented.

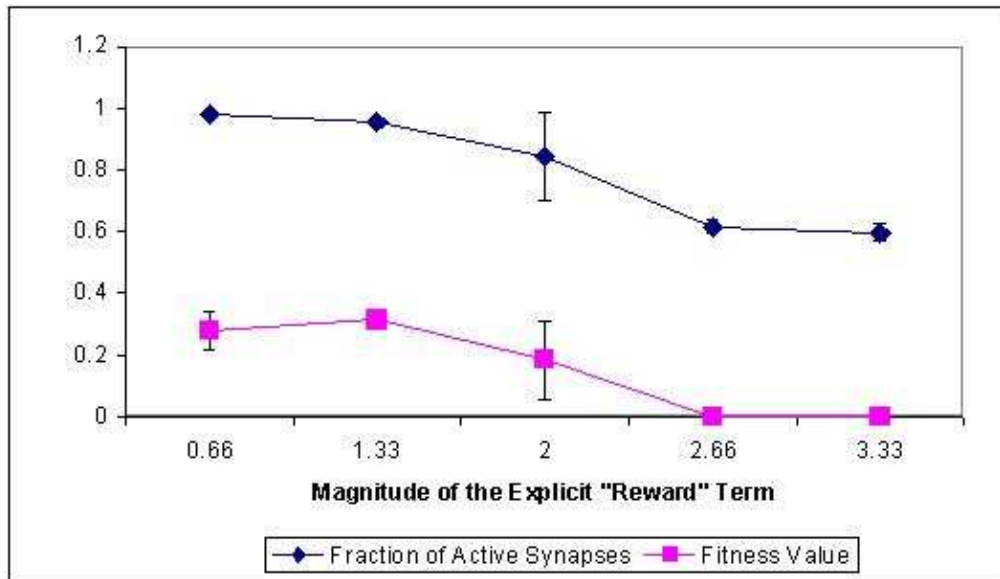


Figure 12: Direct encoding with explicit network minimization. The fraction of active synapses and fitness value of the best agent at the end of the evolutionary process (y-axis) is plotted as a function of the magnitude of the explicit “reward” term (x-axis). The mean values of 5 runs is presented for each value of reward term examined.

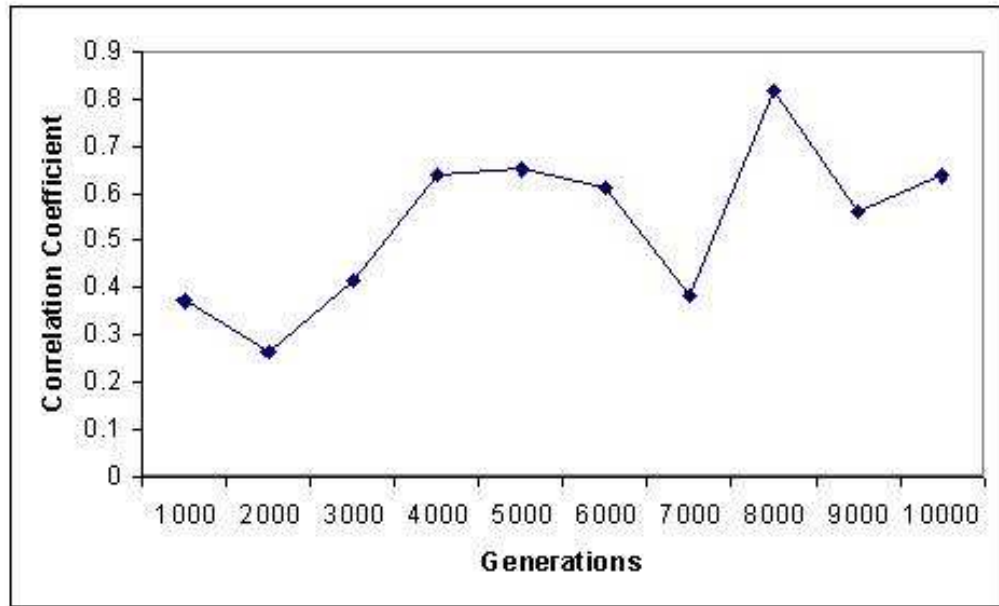


Figure 13: Correlation between encoding levels in SOCE and contribution values in the MSA algorithm during the evolutionary process. Values for best agent every 1000 generations are shown.