Multi-level grounding and self-organization of behaviour through evolution, development, learning and culture

Diego Federici
Norwegian University of Science and Technology
Department of Computer and Information Science
N-7491 Trondheim, Norway

NTNU

NTNU Trondheim
Norges teknisk-naturvitenskapelige universitet
Doktor ingeniøravhandling 2005:43
Institutt for datateknikk og informasjonsvitenskap
IDI-rapport 2005:43
ISBN 82-471-6953-3
ISSN 1503-8181
printed February 15, 2005

Contents

List of Figures

Abstract

The research topic of this PhD concerns the production of intelligent adaptive behaviour from the embodied perspective.

The intelligence seen in biological organisms arises as an emergent property of a complex system displaying multiple levels of adaptive mechanisms. At the bottom, slow changes to the genotype structure are mediated by natural selection. At the top, to cope with more transient factors, faster lifetime transformations are found.

Lifetime adaptation also takes place at various levels, from the processes that are responsible for the organism construction and maintenance, to the regulation of behaviour based on instinctive responses, cognitive abilities and social interactions.

The traditional GOFAI (Good Old Fashion Artificial Intelligence) stand is based on the view of intelligence as computation taking place on a internal symbolic representations of the outside world. Powered by deductive logic, the classical artificial reasoner is centered on the possibility to describe reality within a symbolic framework. Unfortunately, this approach fails to cope with the ambiguous nature of reality, where good decisions are often based on partial and inconsistent knowledge. In these cases, reasoning could appear irrational, while its rationale is actually found in the necessity to take action.

To solve this problem, New AI hypothesizes that intelligent behaviour must be understood within the framework provided by the agent's physical interactions with the environment: subjective sensations and bodily interactions.

The result is a bottom-up exploration, which starts from the lowest adaptive mechanisms to reach the topmost cognitive abilities. The work presented in this thesis follows this hypothesis, analyzing multiple levels of adaptive mechanisms: evolution, development, learning and culture.

Preface

This thesis is submitted to the Norwegian University of Science and Technology and is organized as a collection of papers. Papers are included in appendix A in their original publication format. The work presented was carried out at the Department of Information and Computer Science.

Acknowledgments

First and foremost, I wish to thank prof. Keith Downing for the continuous support and precious advice.

My thanks go to all the CAOS guys and especially to prof. Pauline Haddow. Thanks also to prof. Tom Ziemke and prof. Julian Miller for the interesting discussions. A special thank to Frode, Jörg and Zoran for being around to answer lots of questions. Also my gratitude goes to the ladies at the second floor, all the people at LAIR back in Columbus and ASL2 in Lausanne, with a honorable mention for the Indian connection and the Balelec breakfast club.

Finally to my family and all the socks back home, thanks for being there.

Chapter 1

Introduction

Among the characteristics that separate humans from most animals is our ability to design and use proper tools to delegate, at least partially, the burden of our work needs. Artificial Intelligence and Robotics are among the latest steps in this direction. In this case, the object of design is the design mechanism itself, as algorithms and machines are constructed so that they can manufacture and invent.

As the two fields evolve and tools are being constructed, new challenges for the applicability of known techniques emerge. Robotic applications require a robustness toward incomplete and inconsistent knowledge that makes the classical approach to artificial intelligence unpractical.

The classical view in artificial intelligence is based on logic and symbolic manipulation. Its shortcomings emerge when symbols are hard to recognize in a physical world made of real sensors, noise and hardware failures.

The standard solution is to restrict the operational conditions of these devices, in order to make the sensors capable of making all the necessary knowledge visible to the robot's brain. Unfortunately, when taken out of their context, these devices not only might fail to operate correctly but will fail to detect the source of their errors.

Opposed to this top-down design, embodied artificial intelligence proceeds from the bottom. Its basic assumption is that intelligence as we know it, must be conceived within a world of physical and social interactions. Results in this field have proven how relatively simple reactive mechanisms can produce complicated and intelligent behaviour.

Still, when faced with insufficient sensory information, reactive mechanisms are no longer sufficient. In this case, the decision making process must rely on internal mechanisms to provide additional contextual knowledge. Yet, the achievement of embodied artifacts capable of proactive and cognitive responses has proven challenging. These artificial cognizers need to be able to cope with problems that require the induction of unknown but relevant information (Beer 2003; Tani 2003; Ziemke 2005). The problem lies in the fact that induction is in general ill-posed[1].

[1] since it might have more than one solution

The work presented engages in this still open problem. Long from putting the final word on the subject, this thesis is the author's attempt to design scalable methodologies that can be used to design intelligent, autonomous and embodied agents.

To achieve this goal, the author has formulated three implicit requirements:

1. learning algorithms must be able to operate online and in real time.
2. even if preliminary tests may be performed on problems of reduced complexity, the methods are designed to scale up to realistic settings.
3. operation must be based on subjective sensory-motor experience

The remainder of the paper is organized in three sections. This first section contains a brief introduction to artificial intelligence and to common sub-symbolic techniques focusing on those adopted in the work presented: neural networks and genetic algorithms. The second section summarizes the contributions of the collection of papers, putting them in the perspective of the long term research goal. The third section includes the conclusions and the author's vision for further development.

1-1 Research statement

The research issues covered by this thesis are linked to the synthesis of autonomous robot controllers from the embodied perspective. In other words, about the adaptive mechanisms that agents interacting with their environment could use to produce intelligent behaviour.

The collected papers are seen as successive steps of problem definition refinement and analysis of various techniques that could bring to viable solutions, spanning over several layers of adaptive mechanisms: evolution, development, learning and cultural transmission.

1-2 Robots and intelligent behaviour

Robots are situated agents: they have a particular position in the environment and perceive the outside world only through their sensors. In this respect they are confronted by the same challenges as all living animals, humans included.

The fact is that, no matter how good the robot's sensors are, they can only provide a partial view of the environment. Without access to complete knowledge, each decision must rely on some apriori assumption about the world. The proper choice and exploitation of these assumptions is fundamental for the production of 'intelligent' behavior, and it is one of the biggest issues that arises in the design of situated agents.

Reactive and Deliberative agents

Imagine that we have to construct a robot for a packaging company. The robot will be placed next to a conveyor belt carrying empty boxes, its task to insert a manufactured product and close the box.

A first possibility is to design a robot equipped with two sensors. The first one detects the presence of a box (IsBoxPresent) and is used to trigger the 'fill-and-close-the-box' behavior. After completing this action, the package will be full, but as it is still lying in front of the robot, it will activate again the IsBoxPresent sensor for at least some time. Without memory, the activated sensor will immediately trigger the 'fill-and-close-the-box' behavior, and the robot will attempt to insert a second product, and a third, until the conveyor belt will eventually manage to drag away the (abused) package.

To overcome this problem, the second sensor is provided. It detects whether the box is open or closed (IsOpen), allowing the robot to recognize packages that have already been processed. For the task it has to fulfill, the robot has now complete knowledge of the environment and its program must just react to the information provided by its sensors:

```
1: repeat forever
2:     if (IsBoxPresent and IsOpen) → fill and close the box;
```

Figure 1.1: The program of a reactive packaging robot

Instead of providing a new sensor, another solution is to equip the robot with a memory storage. In this way, the robot can be made to remember if the package laying in front of it has been previously closed. The robot will then wait to insert a new product until a new (supposedly) empty box arrives. The program of this deliberative agent would be as shown in Figure 1-2.

With the use of an internal state (BoxOpen) the deliberative robot is capable of working with an otherwise incomplete perception of the environment. Adding contextual information, the memory provides the same functionality as an ad-hoc sensor.

```
1: memory: BoxOpen ← true;
2: repeat forever
3:     if (IsBoxPresent and BoxOpen) → fill and close the box;
4:                                     BoxOpen ← false;
5:     if (not IsBoxPresent)         → BoxOpen ← true;
```

Figure 1.2: The program of a deliberative packaging robot

On the other hand, the design of the robot has made additional assumptions about the environment structure: the contextual information provided by the robot's memory will be valid as long as these assumptions hold. These apriori postulates are formalized at lines 1 and 5 in the program of the deliberative agent, stating that all incoming packages are open. If a package were to arrive already closed, the deliberate agent could not tell.

Inductive Bias

If on one hand, internal contextual information is necessary to disambiguate otherwise perceptually identical states of the environment (aliasing), on the other, it could end up containing incorrect information of the outside world.

In addition, the latter is just a corollary of the far reaching inductive problem. First, as Kant pointed out, the phenomenic experience is logically segregated from the knowledge of the *noumenon*[2]. Therefore nothing can rigorously be affirmed about the real nature of the world and of its state. Additionally, Hume contested, any inference derived from current experience (induction) could always be proved wrong when additional evidence is collected. As a consequence, any attempt to explain how reality works is rationally based on an 'act of faith' that anything that will happen in the future shares the same properties of what happened in the past.

With the same conclusions, in machine learning, the 'act of faith' is replaced by the more formal concept of inductive bias:

> (...) a learner that makes no a priori assumptions regarding the identity of the target concept has no rational basis for classifying any unseen instances.
> ((Mitchell 1997), pg 42)

At first, this analysis would discourage the use of reasoning. In reality, it recognizes that in order to take a decision in a novel situation assumptions *must* be made.

If on the conveyor belt example, the range of possible situations were very limited, in realistic settings it is the exact opposite. As the context around us is in perpetual change, without 'reason' any situation would appear novel. It is the (biased) attribution of abstract properties to the current unique situation that allows the reuse of previous experience.

[2] *noumenon*: the thing as it is, opposed to *phenomenon*: the thing as it is perceived by the senses.

Abstractions and their relationships with the environment are the cornerstone of every decision making process. Representing the subjective knowledge of the environmental structure, these internal representations are necessarily segregated from the outside world and their quality can only be ascertained with the understanding of the process that created them.

Artificial Intelligence

The classical view of intelligence is that of a rational and analytical mental process: an abstract manipulation of information akin to symbolic computation performed by a computer program.

As a consequence, intelligence is understood largely as the ability to solve problems defined on internal representations that 'stand in' for the real external world.

Unfortunately, there is no universally accepted definition of intelligence. Most definitions tend to be tautologies, a sort of "intelligence is what intelligent does". In the middle of it stands the human brain, the only artefact which is globally recognized for being intelligent. From this, derives the (mistaken) acceptance of intelligence as an exclusive property of the human being. Not coincidentally the famous Turing test proposes to measure a computer's intelligence with its ability to be 'mistaken' as a human.

From the engineering perspective, intelligent artifacts are interesting for their ability to find good solutions to problems, i.e. to (rapidly) perform appropriate decisions given the available information. After all, this is what I.Q. tests seem to evaluate.

Unfortunately, intelligence then takes a teleological connotation, since its evaluation requires speculation of the subject's currently available information, goals and internal states. In other words, intelligence seems to lie in the observer's eyes.

For example, a short movie from the "Twilight Zone" series told a futuristic story in which all kids upon reaching age 15 were obliged to take an intelligence test. This helped decide what kind of future education was most appropriate for each of them. What was hidden from these kids under examination was that those achieving genius scores were actually going to be killed (to preserve the equilibrium of society!). In this context, who knows if some of the low scores are actually produced by the smartest kids?

Intelligence is a property of an information system, which receives various sensory data to produce overt behaviour. Intelligent behaviour is commonly interpreted as a result of the conjunction of external (sensory) and internal information sources.

The use and construction of the internal sources of information (goals, emotions, memories, etc.) is what defines cognition. It is very well known that without the central nervous system, or with specific damage to it, cognitive abilities can be lost, to the point that animals can be left in a vegetative state.

Somehow the mind emerges from the processes taking place inside the brain.

The classical approach to the synthesis of intelligence A computer works following the instructions contained in its program. It usually takes some inputs,

transforms them following its program and produces corresponding outputs. For an artificial intelligence (AI), the computer hardware plays the role of the brain while the program is its mind.

The classical approach to AI (GOFAI) concentrated on the inner symbolic manipulation that apparently takes place in the highest levels of consciousness. These include logic and inferences such as those based on deduction and induction. The reason behind this first choice is quite straightforward, since, in first analysis, logic and inference are what differentiate cognitive beings (humans) from the other *less intelligent* animals.

This approach sees the mind as a computation over internal representations of the real-world. As such, representations are abstract entities whose meaning can be arbitrarily attributed.

Early research concentrated on *pure* problems which did not require physical interactions (mainly because that was impractical at the time). For example, these problems included 'brain intensive' games such as Chess, Checkers and Go, and stochastic ones like Backgammon. Apart from the game of Go, computer programs have since managed to beat the current human world champions.

These systems are based on logic and symbolic manipulation. Symbols are accessible through the game interface, and are computed with to produce optimal winning strategies.

The grounding problem Unfortunately, when these AI techniques were applied to the domain of robot control, a general weakness associated with the ascertainment of the symbols being manipulated was highlighted (Harnad 1990).

A symbol is just an inner image of the world. Its insubstantial nature is what makes it appealing for internal manipulations, but the possibility to think about the world at a symbolic level requires an understanding (i.e. ' grounding') of the symbols being manipulated.

Without grounding, symbols are meaningless entities, empty tokens isolated from physical reality. Tokens can be computationally manipulated, but for a robot, the problem of resolving the gap between the internal mirror world and the external one, accessible only to actuators and sensors, remains unsolved.

Without being able to ground a symbol to its phenomenological counterpart, symbols are useless for interaction with the environment: it is like the robot asking "what makes the re-charging station a re-charging station?".

The frame problem The classic formulation of the frame problem expresses a technical difficulty for the use of mathematical logic as a modeling tool (McCarthy and Hayes 1969): how is it possible to formalize the effects of actions without also having to specify the large number of non-effects of these actions?

More general is the definition found in (Dennett 1978, pg 125), where the question is: how can a cognitive agent keep track of all its beliefs about the environment as it acts in it?

The frame problem also reminds the engineer that not only internal representations must be phenomenologically grounded, but they must be manipulated in re-

altime. If symbols are represented by independent structures (e.g. predicate logic), the frame problem can very easily become computationally intractable and logical inconsistencies cannot be ruled out (Pfeifer and Scheier 1999, pg 65-69).

But situated agents may often not have the time to sit down and think thoroughly about every possibility. The reasoner must be able to act based on incomplete and inconsistent knowledge, possibly reaching irrational conclusions, while their rationale is actually the necessity to take some immediate action.

***Nouvelle* Artificial Intelligence** Especially directed to the solution of these problems, *Nouvelle* AI (Brooks 1991) understands intelligence through situatedness and physical embodiment, i.e. the physical interaction with the environment.

In its strongest definition it claims that internal representations are superfluous and costly: instead of concentrating on ungrounded processes, Nouvelle AI should follow the incremental evolution of intelligence as seen in nature, i.e. from simpler reactive agents to more complex cognitive ones.

In fact, starting from the topmost abstract processes, AI risks to loose touch with real-world constrains and forget that intelligent behaviour must be based on sensors, actuators and timely responses.

To support this view, *Nouvelle* AI was set to produce complex intelligent behaviour with very simple control systems. Examples include systems producing phototaxis and phonotaxis (Webb 2002), clustering and routing (Bonabeau, Dorigo, and Theraulaz 1999).

In all the aforementioned examples, intelligent behaviour appears from the interaction of simple reactive agents with their environment. Apart for the current applications of these techniques, it is also very interesting that when looking at the agents' controllers, it is hard to identify exactly what constitutes intelligence: the behaviour looks intelligent but the robot brains, in the classical sense, do not.

Embodiment While GOFAI concentrates on abstract symbolic manipulation, *Nouvelle* AI highlights the importance of environmental interaction: intelligent agents must be situated so that they can manipulate and be manipulated by the external world.

This notion is usually referred to as weak or minimal embodiment. Strong embodiment requires a deeper coupling with the environment like the physical one displayed by living organisms.

It is argued that without strong embodiment, agents might be excluded from certain types of cognitive processes typical of living beings. This argument follows the central one about the grounding of abstract concepts by direct (bodily) interactions: can a robot without a gripper understand what it means to grip? (Ziemke 2001).

In other words, the agent's understanding of the external world depends on its physical embodiment, i.e. its sensory and actuator coupling with the environment.

Internal representations Focusing on embodiment and on reactive controllers, *Nouvelle* AI emphasizes how intelligence can emerge without the need of costly

representations. On the other hand, it is well known that cognitive beings possess the ability to represent the world internally and reason about it.

The distinction between Cognizers and non-Cognizers is more complex that the one between reactive and deliberative agents. Deliberative agents possess internal states but these are not necessarily representations of the world. A notable definition is found in (Clark and Grush 1999), where the authors distinguish between weak and strong internal representations.

Weak representations are internal states that play part in behaviour with a continuous linkage to perceptual inputs, similar to sensory enhancers as the BoxOpen variable of the deliberative agent of section 1-2. Weak representations are those that would allow 'minimally cognitive tasks' (Beer 2003) by storing information necessary for proper behaviour.

Also weak appear the representations produced by social interactions and language games (Steels 2003; Cangelosi 2004) and (Sugita and Tani, in press). In this case, agents must develop suitable internal states which link the language tokens to behaviour. The focus here is more on the information exchange process and not on internal simulation.

Strong representations are stand-ins which recapitulate external reality and are capable of playing their roles in the absence of on-going input. With these inner images, the cognitive agent can reason (recursively) about the world and anticipate the consequences of possible course of actions.

Examples of strong internal representations can be found in (Tani 2003) and (Ziemke 2005, in press) and in paper A. In these cases representations are constructed following the "simulation theory" (see for example (Hesslow 2002)) which is based on the assumption that cognition is based on the anticipation of consequences of covert (imagined) actions.

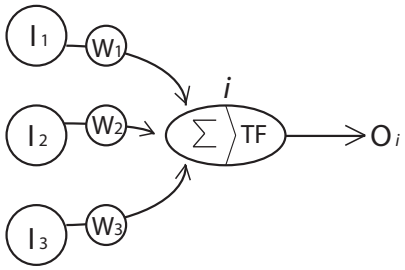


Figure 1.3: A simple Artificial Neural Network (ANN) with three input neurons (sensors) and a single output neuron. The inputs (I) are processed by the synapses (W) and integrated ($\sum$) by the output neuron. The neuron state is determined by its transfer function (TF) which is typically used to shrink the activation range into an appropriate interval.

1-3 AI Tools for Bottom-Up Design

With its focus on emergent behavior, New AI methods are often based on automated design tools. In particular, Artificial Neural Networks (Section 1-3.1) are extremely diffused both for their generality (function approximation), and for the large number of existing adaptive algorithms.

Among these, Genetic Algorithms (Section 1-3.2) are a frequent choice since they can be applied, as it is often the case, with minimal knowledge of the problem domain.

1-3.1 Neural Networks

More than as an algorithm, Artificial Neural Networks (ANN) should be regarded as a different paradigm for computing. Opposed to the Von Neumann vision of computation as the manipulation of a central memory module, ANNs are based on a high number of parallel and simple processing elements. Each neuron is interconnected to a set of neighbors with a set of synapses (often referred as weights), which indirectly specify its functionality.

The features that make ANNs interesting for computing are related to their generality, adaptivity and massive parallelism.

ANNs are often organized in layers, with inputs activating a hierarchy of hidden layers. The activity of a set of hidden layers (the output layers) specifies the output of the network. Connected in a cascade, several hidden layers increase the expressiveness of the ANN. Also, it is common to allow recurrent connections among the same or different layers. Recurrent connections enable history-dependent dynamics which can be regarded as a short term contextual memory.

Figure 1.3 shows a simple neural network. The output of neuron i is computed as follows:

$$O_i = \mathrm{TF}_i\Big(\sum_{j \in \text{ inputs of i}} W_j(I_j) \Big) \tag{1.1}$$

where I_j is the activation of input neuron j, and O_i the activation (output) of neuron i. The synaptic contribution to the net input of a neuron (W_j) is often modeled by a simple multiplication of a specific synaptic efficacy (weight) and the input neuron activity. In other cases the synaptic efficacy can contain a memory of recent activity. The transfer function (TF) takes the neuron's net input to produce its activation. Traditionally sigmoid functions, such as the hyperbolic tangent, are used. Radial basis function networks (see (Haykin 1999, pg 256-281)) use symmetrical transfer functions, such as a Gaussian, to model receptive fields. In other cases, it is possible to obtain more complex activation dynamics by embedding a short term memory or recent activation, this is the case for the Leaky Spiking Neural Networks (Gerstner and Kistler 2002; Maass and Bishop 1998) and Continuous Time Recurrent Neural Networks (Beer 1995).

supervised learning

Supervised learning allows the proper configuration of an ANN (usually by assigning its weights) by providing training examples in the form of an input set and its correspondent output set. During training, the network functionality converges to one among the continuum of functions implicitly defined by the training examples.

A powerful supervised learning method for ANN without recursive connections, is provided by the back-propagation algorithm (see (Werbos 1989) and (Haykin 1999, pg 161-174)). The algorithm operates by setting up an ANN with random weights and altering their value to minimize the expected error. The error is computed as the distance between the network and expected outputs. For layers other than the output one, an estimate of the error is computed by propagating the error backward through the synapses. Since the method is based on gradient descent over the error surface, it can be trapped in local optima.

For recursive connections, the error back-propagation cannot be computed exactly and some estimate must be used instead. This may lead to instabilities and prevent convergence (see paper A for a broader discussion).

Other forms of supervised learning are based on Hebbian learning. Hebb's rule sets the efficacy of a synapse as the correlation between the activities of the pre-synaptic and post-synaptic neurons. This method is often used to produce associative memories, like in the Hopfield net, Correlation Matrix Memories and the Predicate Network (see paper B). Hebbian learning is more biologically plausible (Markram et al. 1997) but does not solve the weight assignment problem of intermediate layers (other than input and output layers). More details about ANNs can be found in (Haykin 1999).

reinforcement learning

Sometimes a quality assessment of a network performance can only be obtained at the end of a sequence of inputs and outputs. This makes supervised techniques

difficult to use.

Algorithms based on Temporal Difference Learning (Sutton and Barto 1998) can be used to bridge the gap between the moment of reinforcement and the application of changes, for example see (Tesauro 1995).

It is also possible to apply supervised techniques to this problem. By storing all the network's history, the error gradient calculated at the moment of the reinforcement can be propagated backward in time (Back Propagation Through Time, see (Werbos 1990) and (Haykin 1999, pg 751-761)). This and similar methods have been applied with promising albeit quite limited results, see for example (Kwansy and Kalman 1994).

Another possibility is provided by Evolutionary Computation (EC, see Section 1-3.2), using the collected reinforcement as a measure of implicit fitness. ANN parameters are stored in a genotype subjected to selection. Selection allows genotypes that produce better performing networks to survive and eventually produce new fitter variants.

unsupervised learning

A different class of algorithms work in an unsupervised fashion. In this case no evaluation of the system behavior is necessary. These methods produce intermediate representations based on some statistical information contained in the input set.

For example, Self Organized Maps (Kohonen 2001) can produce topological maps of high dimensionality input vectors. Back-propagation networks trained as Auto Associative Memories can efficiently extract the principal components of the input set in a hidden bottleneck layer, providing a projection to a lower dimensionality space which maximizes information content.

Unsupervised learning is often used as a paradigm to explain and reproduce how implicit learning in biological organisms takes place. This includes structures in the cortex such as phonetic and orientation maps.

1-3.2 Genetic Algorithms

Inspired by the Darwinian theory of evolution by natural selection, Genetic Algorithms (GA, (Holland 1992; Goldberg 1989)) were proposed to the AI community as a versatile and general search algorithm.

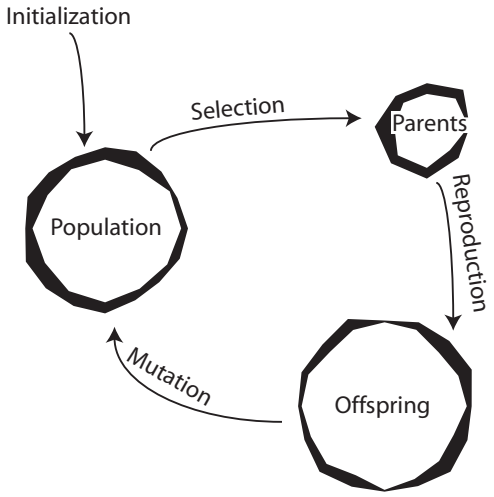


Figure 1.4: The cycle of evolutionary search in Genetic Algorithms (GA). At the beginning, the population is initialized at random. The best individuals are selected in base of their performance and reproduce. Their offspring can be mutated to produce the next generation's population.

GAs search solutions to a problem with a recursive process of exploration and evaluation. Exploration of the search space is based upon the assumption that the characteristics that make good solutions can be (1) inherited and (2) refined by small incremental changes (mutations).

Good solutions are bred to generate new, slightly different, variants. Since all solutions compete for survival, the best will replace the worst performing ones. Overall, as generations go by, better solutions will be generated until a suitable one is found.

The most interesting aspect of GAs is that they do not require any in-depth knowledge about the problem they are set to solve. They require only (1) a measure of the fitness of the solutions at hand and (2) a way to encode them in a replicating persistent medium (e.g. DNA, bit string, etc.). This makes GAs particularly suited for the automatic generation of artifacts, optimal parameters or whole computer programs in domains in which analytical methods are either unpractical or unknown.

This is possible because creativity in GAs is obtained by random changes of

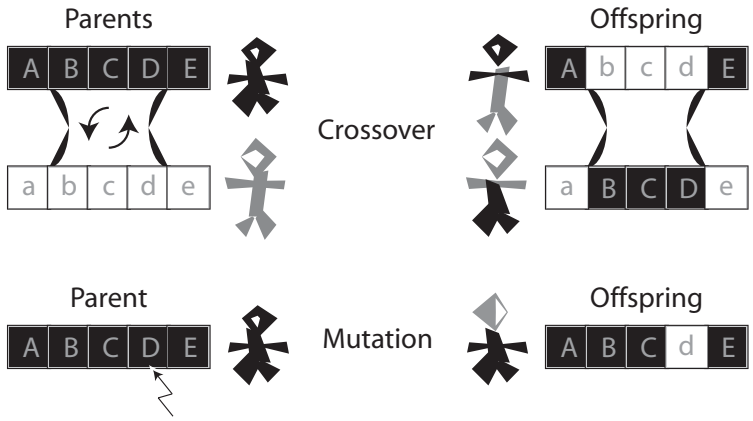


Figure 1.5: Depiction of the crossover and mutation. Crossover shuffles the parents' building blocks spreading 'good' genes in the population. Mutation introduces new genotype variants.

selected solutions. With creativity proceeding blindfolded, directedness is provided by fitness based selection.

Genotype

Solutions must be described in a language which makes them suitable for search. As for biological organisms, these descriptions are called genotypes.

In nature, genotypes are physical entities. The DNA is a double helix of deoxyribonucleic acids in which specific sequences of bases indirectly specify the organism's structure and behavior. Similarly, artificial genotypes are a sequence of numbers (a data structure). As DNA bases are translated to amino-acids and proteins, genotypes are used in GAs to synthesize the solutions under scrutiny.

While for biological organisms fitness is implicitly defined by the sustained ability to survive and reproduce; for GAs fitness is usually some direct measure of the quality of the solutions.

In GA, to allow the exploration of new solutions, replicated genotypes are subjected to random crossover and mutations (Figure 1.5).

Crossover is a recombination of the genetic material of two parents. With crossover the genotype's building blocks (genes) are exchanged and their contribution to the overall fitness can be implicitly estimated. It can be shown (Schema Theorem, (Goldberg 1989)) that, with low epistasis[3], good genes will geometrically proliferate (stochastically) in the population. Since epistasis reduces the perfor-

[3]the interaction between two or more genes to control a single phenotypic trait

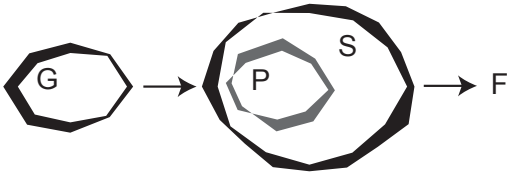


Figure 1.6: Transformations from genotype space to fitness values. The space of possible genotypes (G) is mapped to the phenotype space (P) which, in general, is a subset of the solution space (S). Solutions are then evaluated to produce a fitness value (F). While crossover and mutation operate in the G space, selection operates on fitness.

mance of the evolutionary search, the design of the genotype must take care to minimize its impact.

Optimal genotype representations are problem specific and require domain specific knowledge. Unfortunately, it is the lack of domain knowledge that makes evolutionary search interesting in the first place.

It is also interesting to notice that, in nature, crossover is one of the major sources of gene duplication. Because of a misplaced crossover insertion, entire sections of DNA can be replicated from parent to offspring. In these cases, duplicated genes increase the size of the genotype, often without altering the organism they produce. On the other hand, duplicated paralogous genes offer a redundancy that can eventually be exploited to produce organisms of increasing sophistication. It is currently estimated that 90% of eukaryote genes were introduced by gene duplication (Teichmann and Babu 2004), see also paper G.

Crossover does not introduce new alleles in the population. This role is covered by the mutation operator, which stochastically alters the chemistry of a gene. It is mutation that prevents the search from stagnating, counterbalancing the homogenization of selection.

From genotype to phenotype

In order to evaluate their fitness, genotypes must be converted to phenotypes. The evolutionary task requires the exploration of the space of possible genotypes ($g \in G$) in order to find the phenotype ($\mathrm{P}(\ .\ )$) with optimal fitness value ($\mathrm{F}(\ .\ )$):

$$\arg\max_{g \in G} \ \mathrm{F}\big(\mathrm{P}(\ g\ )\big) \tag{1.2}$$

As a result, the task requires the resolution of two levels of indirection: the genotype to phenotype map (G-P map) and the fitness evaluation (see also Figure 1.6).

It is generally assumed that little is known about the relationship between fitness values and solutions; otherwise direct search methods could be applied. Therefore,

great care must be taken in the design of the G-P map. Without additional knowledge, it is generally assumed that similar phenotypes will produce similar fitness values. Therefore good G-P maps are those that preserve a good correlation between the phenotype and genotype spaces. With small changes to the genotype often producing small changes to the phenotype, evolution can proceed by small differential refinements and optimal performance is expected.

G-P maps can be as simple as bijective transformations (e.g. encoding the variables for function optimization), or very complex constructs requiring stochastic learning phases and environment interaction (e.g. robot controllers).

Chapter 2

Research Description

The papers collected in this thesis can be partitioned in four successive phases, characterized by a common research sub-topic and methods.

Overall, the work has touched all three types of adaptive mechanisms: learning, genetic evolution and development. Because of its relationship with knowledge acquisition, cultural evolution, which is usually considered a sub-class of life-time adaptation, has also been explored.

Figure 2.1 depicts the path followed in the production of this thesis from the point of view of the relevant adaptive mechanisms.

Initially the problem regarding the scalable development of intelligent behaviour for situated and embodied agents was formulated as a long term objective. The objective includes overcoming two well known obstacles: the Frame and the Symbol Grounding problems.

Phase 1: Online learning Aimed at the prediction of sensory consequences of performed actions, anticipatory learning methods are based on the acquisition of contextual information trough the interaction of an agent with its environment. Sub-symbolic methods, such as the Simple Recursive Network (Elman 1990), seem to offer a viable solution to the grounding and frame problems, but the shortcomings of known training techniques make them unpractical for our goal (see paper A). Current techniques either show poor on line acquisition, or unfeasible space and time requirements. The attempts to produce a novel training technique based on associative memory capable of compressing information during operation (see paper B) lead to the realization that only with a loss of generality, in other words with a stronger learning bias, could the requirements be met.

Phase 2: Cultural transmission In order to reach a better understanding of what can be considered a minimal and yet useful unit of behavioral knowledge, in a following phase, we produced a model containing communicating agents undergoing evolution (see section 2-2). Agents had to perform a simple harvesting task, also being capable of exchanging minimalistic functional information about the world. These memes (Dawkins 1976) are acquired by Operant Conditioning and are shown to instantiate a process of cultural evolution atop of the genetic one (see papers C

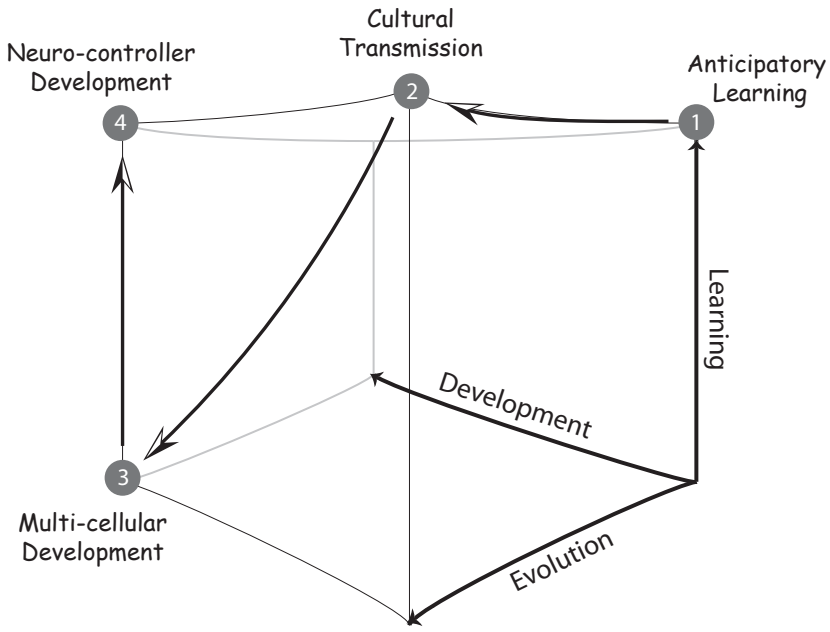


Figure 2.1: The four successive phases that lead to the production of this thesis, plotted over an evolution, learning and development space

and D). On the other hand, in these papers the agent episodic knowledge is stored in a localistic memory model. With a neuron dedicated to each meme, such model can not be applied to more realistic settings because of its space requirements.

Phase 3 and 4 In mammals, episodic memory is believed to be stored in specific structures of the brain. For example, the Hippocampus is associated with spatial and short term memory retention (Carew 2000, 382-399), while the Basal Ganglia performs reinforcement and anticipatory sequence learning, see (Downing 2005, submitted), (Kandel, Schwartz, and Jessell 2000, pg 853-867) and (Gazzaniga, Ivry, and Mangun 2002, pg 449-451). The exact way that these structures operate is still matter of debate, and although mathematical models have been proposed, the complexity at the neural level makes actual implementations very difficult.

As a possible tool for the design of such structures Evolutionary Computation (EC) could be used. Unfortunately, even if artificial evolution has been shown to solve minimally cognitive tasks requiring the use of memory (Beer 2003), (Ziemke 2005, in press), such solutions are not believed to be feasible for problems requiring rich contextual information. Evolutionary techniques are often good at finding minimalist solutions, but also show poor scalability.

Brains of biological organisms are indeed very complex. On the other hand,

they also show a good degree of modularity. For example, the neocortex shows two levels of organization with repeated local micro-circuits (see (Gazzaniga, Ivry, and Mangun 2002, pg 635-636)). In EC, indirect encoding methods can take advantage of modularity to increase scalability. In order to develop large neural structures, the third research phase (section 2-3), investigated an indirect encoding based on multi-cellular development. A first analysis concentrated on the evolution of the ontogeny of simple multi-cellular patterns (see papers E, F and G). In the following phase (section 2-4) plastic neuro-controllers were also evolved (see papers H, I and J).

2-1 Anticipatory learning

Considered from an algorithmic perspective, animal are unsupervised learners. Even when they show the ability to learn from reinforcements and supervision, the brain still operates autonomously within an unsupervised framework.

Adaptive mechanisms based on reinforcement, such as Classical and Operant conditioning, are quite well understood at the microscopic level as processes based on Hebbian learning (Markram et al. 1997; Abbott and Nelson 2000) and (Kandel, Schwartz, and Jessell 2000, pg 1260-1262). But since synaptic plasticity operates within small time frames (10-100ms), from the system perspective, the problem of memory retention is still open. How does the brain select what is important to remember to drive the learning process?

Also challenging is bridging the gap between unsupervised and supervised learning. The problem is that supervision requires the ability to interpret third-person behaviors in correlation to self behaviors. In other words, in order to be able to construct causal relationships from goals to actions, the learner must interpret other's behaviors and assign them a motivational frame of reference.

In the primate brains, the "mirror neurons" structure (Rizzolatti et al. 1996) is supposed to be responsible to fill the imitational gap. Yet its organization, set up and exact role is still a subject of debate.

One common characteristic among the neural structures that supposedly implement these mechanisms is that they show many recursive connections (see Figure 2.2).

A first possible application of feedback is to allow the compensation of the noise of sensors and actuators (Delcomyn 1998, 263-272). This would be similar to the use of retroaction in PID controllers (Haykin 2001), as found in the coupling between muscles and spindles (stretch sensors), see Figure 2.2A.

At a higher level, feedback can be used to refine partially unknown motor tasks. A good example is provided by experiments on song learning in birds (Carew 2000, 231-265), where the recurrent connection from vocalization to hearing is mediated by the environment (Figure 2.2B). Birds are capable of learning how to produce proper songs by (1) listening to other birds, (2) producing their own chirping and (3) comparing their melody with the one they heard form others. In this case, this type of tuning mechanism could resemble the one provided by supervised learning neural networks or equivalent Kalman filters (Haykin 2001).

The role of the cerebellum as a shortterm predictor of action consequences can be seen as direct extension of the song-tuning mechanism, with tuning being focused on the anticipation of sensory feedback (Figure 2.2C).

With a properly tuned anticipatory system, it is then possible to create grounded sensory motor simulations that can be used for planning, for example see (Tani 1996; Meeden, McGraw, and Blank 1993) and (Ziemke, Jirenhed, and Hesslow 2005, to appear).

Neural networks can be trained with standard supervised techniques (i.e. back-

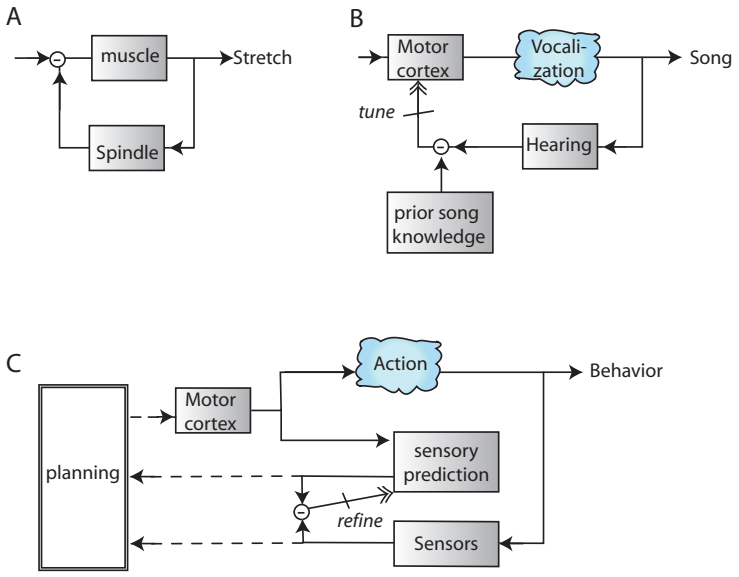


Figure 2.2: Adaptation through retroaction. A) the spindle immediately regulates the actual stretching of muscles (Delcomyn 1998, 263-272). B) tuning of bird singing is based on the sensory feedback from vocalization to hearing (Carew 2000, 231-265). C) anticipatory learning is based on the comparison of expected and observed consequences of motor actions (see for example Paper A).

propagation) in auto-associative tasks (Figure 2.3 on the left). With neural activation proceeding layer by layer, all the information necessary for the reconstruction of the current input is forced through a bottleneck layer (PC). The activation pattern of the bottleneck is a unique internal description of the input. Such method is often used to compute the principal components of a set, providing a projection of each member of the set to a lower dimensional space.

Analogous but used to produce the next input vector, anticipatory associative neural networks (Figure 2.3 on the right) can extract the principal components from a set of sequences. In this case, the activation of the bottleneck layer (C) provides contextual information useful to disambiguate similar sub-sequences. This contextual information is considered a first level of functional grounded representation (Elman 1990; Pollack 1990; Chalmers 1990; Chrisman 1991; Stoianov 2000). Also, using these internal representations as abstract sensors, additional layers of anticipatory networks can be stacked to produce higher level representations.

An interesting aspect of these representations is that they emerge through the exploration of causal relationships in the input sequences: they are a consequence

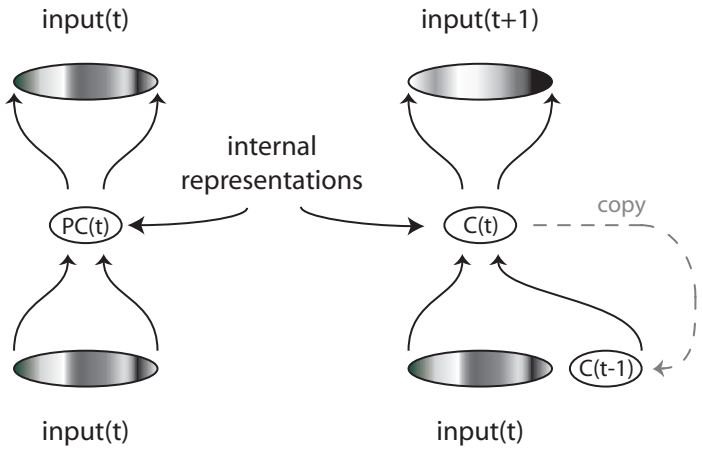


Figure 2.3: Auto-Associative Neural Networks (left) and a Simple Recursive Network as an implementation of an anticipatory associative neural network (right). The middle bottleneck layer contains all the available information for the auto-associative (PC) and anticipatory (C) tasks. For the auto-associative network, these internal representations contain the principal components (descriptions) of the specific input in relation with all the other possible inputs. For the recursive anticipatory network the internal representations provide a suitable context to uniquely predict future inputs.

of the function implicitly defined in the sequence itself. From this perspective, they attempt a theory building task, to represent the causal links between input subsequences.

Concerning the problem of memory retention, such internal representations have the advantage of being compact and conserving all the functionally relevant information. Additionally the anticipatory system can be used recursively to recreate a whole sequence from a starting seed-context. In this aspect, it would match the theory of constructive memory recollection (Suddendorf and Corballis 1997; Schacter 1996).

In a similar way but used backward, the anticipatory system can be used to reconstruct preconditions from consequences. As such, it can be used to recursively recover the set of assumptions that led to an unexpected event or, in other words, the motivations of a sequence of actions. As a consequence it could drive a process of learning by imitation.

Limitations of sequence learning

A widespread approach to the implementation of anticipatory systems for sequence learning is based on an extension of the back-propagation algorithm. Because of the presence of recursive connections, back-propagation can only produce a partial estimation of the error surface. As a consequence training is both very slow and it typically gets stuck in local optima (Kwansy and Kalman 1994). Both limitations make the system infeasible for real world applications.

As pointed out in Paper A, the problem is that the back-propagation algorithm does not make an appropriate use of the knowledge that is already stored in the network.

A second approach is based on one-shot associative memories, capable of retaining input-output patterns without losses. Unfortunately, as far as we know, there is no such memory that can produce compact representations. This excludes the possibility to obtain holistic representations (Chrisman 1991), and with the consequence that space and time requirements necessary for robot implementations are not met.

To solve this problem, the memory model described in paper B was produced. The Predicate Network is capable of storing on line and in real-time an arbitrary function $F : B^n \rightarrow B$, compressing the content of the hidden layer as patterns are being presented. Its extension to outputs of arbitrary size, unfortunately does not appear trivial.

Conclusions

The design of a constructive memory system suffers form the lack of an online memory retention which is both scalable and general.

The scalability constraint is fundamental toward the deployment of real world applications, which should deal with both complex environments and high dimensionality inputs and outputs.

The generality constraint derives from the fact that the patterns that need to be stored do not have features that are known beforehand. If such features were present, the memory system could exploit them to reduce the complexity of the acquisition task.

For example, there is abundant biological evidence that confirms the presence of topological maps both at the sensory and motor levels. Also, recordings of hippocampal activity suggest the presence of a hierarchical structure in the organization of place cells.

Therefore, limiting the operation to structured instead of totally distributed activity patterns, an ad hoc retention system could be made that is both sufficiently general and capable of online acquisition.

2-2 Cultural Transmission

Without considering the imitation mechanism, but giving it for granted, it is possible to allow the horizontal transmission of information among a group of agents. By imitation, agents can learn to replicate behaviors without having to go through a trial and error phase: a phase which can be long and possibly dangerous.

An horizontal exchange of traits in a population goes against the basic principles of Darwinian inheritance, allowing an epidemic spread of information. This process is usually referred to as cultural evolution.

In (Dawkins 1976) it was suggested that individual's knowledge would be transmitted by discrete entities named memes. Memes are defined as the basic units of cultural transmission in the same way as genes are for genetic transmission.

Memes are subjected to a selection process, with characteristics that are both endogenous (its own ability to be transmitted) and exogenous (the host preference for a meme). New meme variants are generated with first hand experience, transmission errors and internal creative processes. With selection, replication and innovation it is argued that an evolutionary (cultural) process is created.

From our perspective, memes are interesting since they carry subjective knowledge from one host to another. Therefore, a simulation of cultural evolution based on memetic transmission can provide clues about (1) a minimal grounded knowledge representation model, (2) the eventual advantage of memes from an evolutionary perspective.

Characteristics of cultural evolution

Even if models have already been produced, there is still little agreement on their validity as there is strong criticism regarding the memetic approach (Gould 1991; Fog 1999; Boyd and Richerson 2000).

We find that it is possible to summarize the most common critiques in the following:

- Selection is irrelevant since cultural innovation is not stochastic as genetic mutation but complex and directed. Without a complex substrate, i.e. the human mind, there cannot be cultural evolution.

- Unlike the biological one, cultural evolution is not powered by natural selection.

The first critique is connected to creativity. New ideas, memes, are formed by some directed and systematic process of intelligence (Cavalli-Sforza and Feldman 1981; Hull 1982). With innovation having little or no stochasticity, selection is irrelevant. Without selection, no real evolutionary process can take place.

Still, it is argued that as long as variation is present selection *is* relevant (Gil-White 2004). The characteristics of the specific innovation process affect the dynamics of evolution but do not rule it out, since memes that replicate better will still diffuse better.

Regarding the complexity of the innovation process, it is possible to use simple algorithms and still provide a directed innovation process. For example in papers C and D we used operant conditioning.

In reality, the argument seems to revolve around the opportunity to refer to these as models of 'Human Culture'. To this respect any model is necessarily reductionist, but still can capture some basic properties of the real system.

Also, we are not only interested in human culture, and we believe that cultural evolution does not only occur in humans. Other primates also show cultural inherited traits and even bacteria conjugation has strong similarities to memetic transmission.

The second argument derives from the observation that selection in cultural evolution is subjected to the host's decision making (Cavalli-Sforza and Feldman 1981). Therefore, it is not enough for a meme to be transmitted for it to be able to replicate. Replication is accomplished only if the receiver accepts the content of the meme.

Acceptance is a non trivial mechanism that works as a filter against novel information. It can be seen as a mechanism of defense against the spread of harmful traits and it constitutes a second level of selection distinct from the natural one.

There are several psychological mechanisms that characterize memetic acceptance. First of all age is a crucial factor, since at earlier ages imprinting is easier. At a second level memes are accepted with higher degrees if they come from authoritative sources. Third, already possessed memes have regulatory effects on the acquisition of novel ones.

Interesting enough, conjugating bacteria also possess an enzymatic acceptance mechanism. This filtering process has evolved to provide a line of defense against the spread of malicious genes.

We will not discuss each of these in further detail since it will be beyond the scope of this thesis, but we will argue that acceptance does not rule out selection. Acceptance must be seen as an inclusive aspect of fitness and, as such, it just imposes a different scenario on the evolution of memes.

In addition, meme-meme and gene-meme arms races will also be present. In fact, not only memes can compete and exclude each other, but also, since acceptance has genetic components, it allows the control of memetic acquisition at the genetic level. The result is that, on one hand, good memes can increase the reproductive chance of the genotype and, on the other, good genes will try to inhibit the acquisition of malicious memes (Dennett 1995).

Memes

In the work we have presented, memes are simple behavioral entities in the form $\{\text{precondition} \oplus \text{action} \rightarrow \text{consequence}\}$. A meme could be mapped directly to an agent's internal representation as defined in the anticipatory system of Figure 2.3.

Also, it can be used as a training exemplar. Still, missing precise contextual preconditions, which are necessarily subjective and therefore cannot be contained in a meme ($C(t-1)$ in Figure 2.3), the instructions will be loosely grounded. This

could be interpreted as the difference between knowing in theory and knowing in practice.

With such meme implementation, a process of cultural evolution is shown to take place on a heterogeneous population of harvesting agents (paper C). When, in terms of fitness, the 'value' of culture reaches a certain critical point, agents tend to evolve a genetic learning/social strategy as opposed to the harvesting one. In this sense, the evolutionary advantage of the cultural process is self sustaining.

Also, with memes being discovered by Operant Conditioning, culture is shown to acquire fit behaviors faster even if its learning bias prevents it from developing the optimal harvesting strategy (paper D).

Conclusions

Given a population of communicating agents it is shown that a grounded cultural process can produce populations of genetic learners. This effect appears modulated by, but quite independent from, the cost of learning.

These results suggest that within a similar framework, it is possible that genetic evolution will produce, or at least refine, a memetic acquisition system with realistic constraints of simulation time and complexity.

On the other hand, when considering neural networks, evolution must be able to search in a space of adaptive networks of variable topologies and interconnectedness. Such spaces have already proved challenging and often produced limited results (Cangelosi, Nolfi, and Parisi 1994; Gruau 1994). A major constraint appears linked to the evolvability of large networks.

2-3 Evolution of development

The ontogeny of the human brain involves the accurate setup of about 100 billion neurons and 100 trillions synapses. Traditional evolutionary approaches are based on the direct encoding of the phenotype, requiring the use of one gene for each phenotype trait. With the dimension of the search space given by the size genotype space (G-space), the direct evolution of neural structures comparable to the human brain is clearly intractable.

Even for humbler tasks, Artificial Neural Networks (ANNs) contain a number of connections which typically grows quadratically with the size of the layers, while the possible connections among layers grows combinatorially. As a result the phenotype space (P-space) grows rapidly to intractable sizes.

In nature, organisms' ontogeny is the resultant of an evolved development process and its interactions with the environment. Instead of a blueprint, the genotype appears as a developmental recipe of DNA sequences. DNA bases, interacting with in a physical world made of polymerases, ribosomes and proteins, only indirectly generates the mature organism.

If on one hand, the level of indirection between genotype and phenotype adds complexity to ontogeny, on the other it allows the reuse of genetic information. With reuse, the number of genes can be much smaller than the number of phenotypic traits.

The effect of these indirect encodings is that search proceeds in a space of compressed genotypes. As a result, generated phenotypes are bound to lower levels of Kolmogorov complexity[1] since, in general, the 'shortest program lengths' that represent them cannot be bigger than the genotype (see Equation below).

$$C_{Kolmogorov}\,(\text{phenotype}) \leq \|\text{genotype}\| \leq \|\text{phenotype}\|$$

This does not prevent indirect encoding from producing interesting phenotypes. For example, all Cellular Automata (CA) have very simple generative rules and therefore low Kolmogorov complexity. Still Wolfram class 4 (complex) CA are still produced (see (Wolfram 2002) or Paper G for an example of such a CA).

The fact is that Kolmogorov complexity defines an 'algorithmic complexity' which does not directly relate to the 'interesting complexity' displayed by living organisms.

The hope is that indirect encoding can produce interestingly complex phenotypes within a tractable space of short genotypes. As long as a better definition is not found, indirect encoding will require empirical tests.

Evolvability of development

The factors that allow good evolvability for indirect encodings are still quite unclear. Here, we will focus on four characteristics that describe the quality of the mapping, the genetic search, and the level of sophistication of the phenotype.

[1] the Kolmogorov complexity of a sequence s in a language $\mathcal{L}$ is defined as the length of the shortest $\mathcal{L}$-program which produces s

Uniform Coverage
The indirection introduced by development allows for smaller search spaces but also transforms them into spaces that could be more difficult to search.

For example, the transformation M from genotype to phenotype is generally not injective:

$$\exists\ g_1, g_2 \in \text{G-space} \mid g_1 \neq g_2 \wedge M(g_1) = M(g_2)$$

If this genetic redundancy is not uniform across the P-space, phenotypes that have more genotype expressions will be, on average more easy to reach. The result is a search bias that can favor the evolution of only a restricted set of phenotypes.

Correlation
A second issue regards the correlation between G-space and P-space. The correlation is a property of the transformation M, as a measure of the expected difference between the distance of two genotypes and the distance of their respective phenotypes (Lehre and Haddow 2003). Since we are searching by moving in G-space, the basic evolutionary hypothesis is that small changes to a genotype induce small changes to the fitness values. As discussed in section 1-3.2, care must be taken to produce a genotype to phenotype map with high correlation.

More specifically we are interested in good correlation between G-space and P-space during search, as a direct effect of the innovation operators. For example, taking the stochastic mutation operator *mut* and the transformation M, we can define a statistical (average) measure of correlation as:

$$\text{Correlation}_{M,mut}(n) = 1 - E\Big[\delta\Big(M(g), M(mut^n(g))\Big)\Big] \tag{2.1}$$

where $mut^n(g)$ is the genotype obtained by the application of n mutations to genotype g, and δ is a distance metric of the P-space. The measure of correlation can be used in a exploratory phase to design good innovation operators, e.g. showing self-organized criticality: frequent minor changes that infrequently lead to intermittent changes of all sizes.

Incrementality of Innovation
Also, with development being de facto a recursive process of rewriting, it is easy to imagine that alterations affecting early phases of epigenesis will produce bigger phenotypic consequences. This leads to a conservative search, in which innovations that integrate over existing traits, as opposed to those replacing them, will have a higher probability to be benign. Supporting this view, in nature it is observed that 'ontogeny recapitulates phylogeny', i.e. the similarity of two species' embryogenesis is proportional to their phylogenetic relatedness (see also Paper G pg. 2), proving that ontogeny is mostly conserved during evolution.

As a consequence, evolvability could benefit from mechanisms that allow a protection of the genes that are activated early during development.

For example, using the best evolved group 1 32x32 French-flag individual from Paper G, in Figure 2.4 we plot the correlation as defined in Eq. (3) with mutation

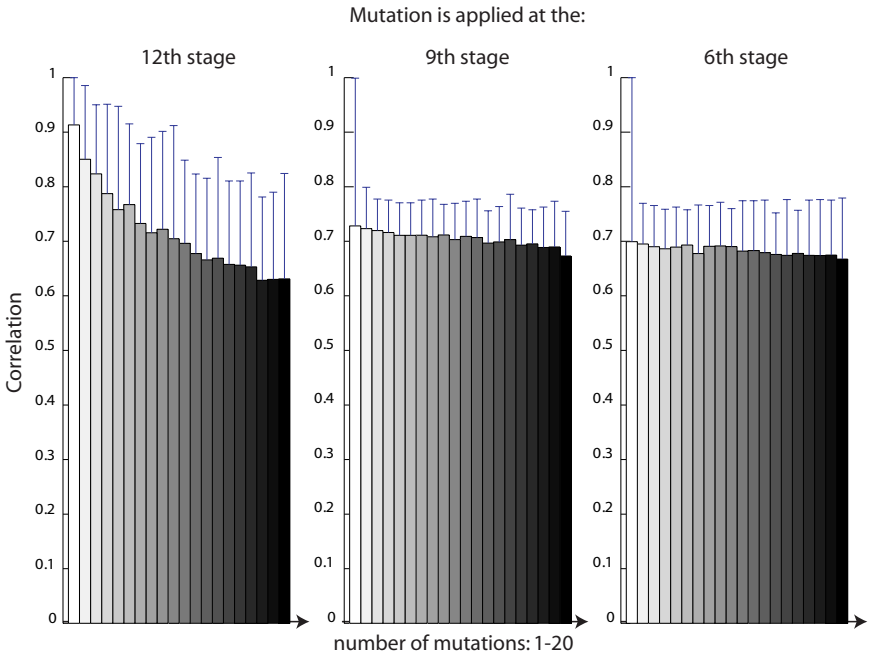


Figure 2.4: Phenotypic correlation as a function of the number of mutations. Average and maximum correlations from 100 repetitions of random mutations to the best 32x32 French Flag individual evolved with 12 embryonal stages (see Paper G for details). Correlation is lower for mutations affecting earlier phases of development.

being applied at different phases of development. In this case the metric δ is defined as the Hamming distance:

$$\delta(\text{Phen}_A, \text{Phen}_B) = \frac{1}{\|\text{Phen}\|} \sum_{\forall T_i \in \text{traits}} \Big(1 - \text{equals}(T_{A,i}, T_{B,i})\Big)$$

$$\text{equals}\,(\,a, b\,) \begin{cases} 0 & if \quad a \neq b \\ 1 & if \quad a = b \end{cases}$$

Phenotypic Sophistication

A final aspect concerns the issue of gene reuse. With a gene potentially controlling several aspects of the phenotype, the optimization carried on by evolution must deal with the eventuality that several independent phenotypic traits are bound by the expression of the same gene. Linked trait expressions introduce dependencies which may preclude the expression of optimal solutions.

In this perspective, the biological mechanism of gene duplication (Ohno 1970)

is understood as a neutral projection to higher dimensional G-spaces, allowing a stochastic disassociation of linked phenotypic traits.

Also, related to the concept of sophistication, genotypes that allow a timely regulation of developmental rules can achieve higher levels of differentiation over time (heterochrony) and space (heterotopy).

Model and first results

With the objective to eventually develop large ANNs, we have designed a development system based on cellular automata.

The system starts from a zygote cell that in a given number of developmental steps, replicates and differentiates to produce a mature multi-cellular organism (see Figure 2.5 for an example). Cells are immersed in a 2D Cartesian space, where released simulated chemicals diffuse and local membrane interactions take place. From this perspective, it is similar to the models found in (Bentley and Kumar 1999) and (Miller 2003; Miller 2004).

Differences lie in the fact that ontogeny is controlled by a recursive ANN, which serves as a model of a gene regulatory system. Also, the development program is allowed to complexify incrementally during evolution: new recursive ANN duplicates can be created, each one controlling different phases of development (embryonal stages).

For different developmental steps, embryonal stages make possible: (1) different genetic expressions, and (2) a differential sensitivity to innovation. Overall these methods are proved to increase evolvability of development.

Simulations also proved that the proposed developmental model scales well to large phenotypes, with average performance converging to values highly independent of the organisms' size. Additionally, the ontogeny displays emergent regeneration capabilities, which can be used to produce highly resilient fault-tolerant organisms. All the simulations detail can be found in Papers E, F and G.

Conclusions

The results obtained in this phase demonstrated that when compared to direct encoding, developmental systems appear more suited for the evolution of large phenotypes. This seems valid also for targets of 'interesting complexity' which are commonly considered very hard for ontogeny.

Our next objective is to apply the development system to the domain of neural networks.

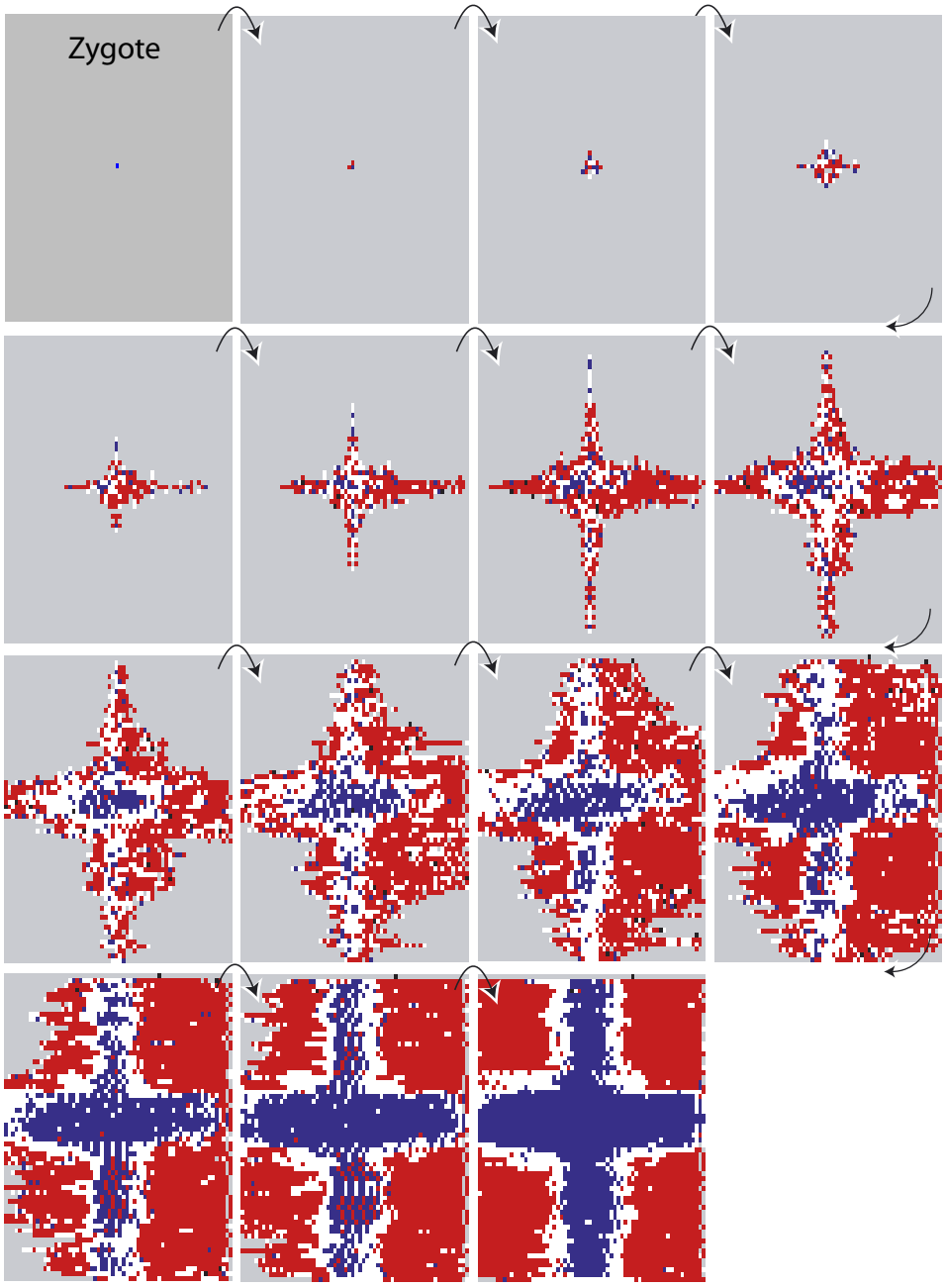


Figure 2.5: An example of development of 64x64 organism. In this case fitness is proportional to the resemblance of the mature organism to a Norwegian flag.

2-4 Evolution, development and neural networks

The evolved multi-cellular organisms produced in the work contained in Papers E, F and G are selected only after their topological properties. In other words, fitness evaluated the growth program only for the cellular structure it produced.

Conversely, neural structures require an additional process, a translation of the phenotype in a functional organism. To produce good results, the transformation from phenotype to network must be carefully designed to take advantage of what development can offer.

One straight-forward implementation of this 'network encoding', is to grow the weight matrices for a standard layered neural network. In this case, the cell types, internal metabolic and external chemical concentrations are translated one by one to produce each synaptic efficacy of the evaluated network. We will refer to the phenotype as the Kitano weight matrix encoding, since it was adopted in his matrix rewriting scheme (Kitano 1990).

Unfortunately, with the development model we have produced, the Kitano encoding in practice produces poor results. The problem is that weights matrices do not appear to have topological structures that can be exploited by development. For example, in Figure 2.6 we show how a typical weight matrix looks like when plotted as bitmap. If such a complexity is fundamental to solve the task properly, than we might expect a performance similar to the one of Wolfram CA 2D targets used in Papers F and G.

Spiking neural networks

Standard ANNs models were designed with supervised training in mind. Thus, it is possible that other models may be better suited for evolutionary purposes.

Recently, some new experiments have started to clarify the internal working of neural structures. One interesting conclusion of these experiments is that the assumption that information is encoded by the post-synaptic firing rate contradicts some biological evidence.

For example, acoustic localization in owls is shown to operate on phase information of train of post-synaptic potential pulses (Carr and M.Konishi 1990). Similarly evidence is collected from experiments on bats' echo detection, human skin sensors, and visual processing in flies (Rieke et al. 1997). This evidence produced an alternative ANN model based on pulsed information encoding (spikes).

For modeling purposes, Spiking neural networks (SNN) are interesting because they are shown to be more expressive than standard analog neurons (Maass and Bishop 1998; Maass 1995), and to maximize information transmission (Rieke et al. 1997). Also the internal dynamics of the leaky integrate and fire models can be used to produce interesting response dynamics (Gerstner and Kistler 2002).

From the perspective of evolutionary computation SNN, as well as Continuous Time Recurrent Neural Networks (CTRNN, (Beer 1995)), often produce simpler circuits for tasks requiring time-dependent pattern generation (e.g. locomotion).

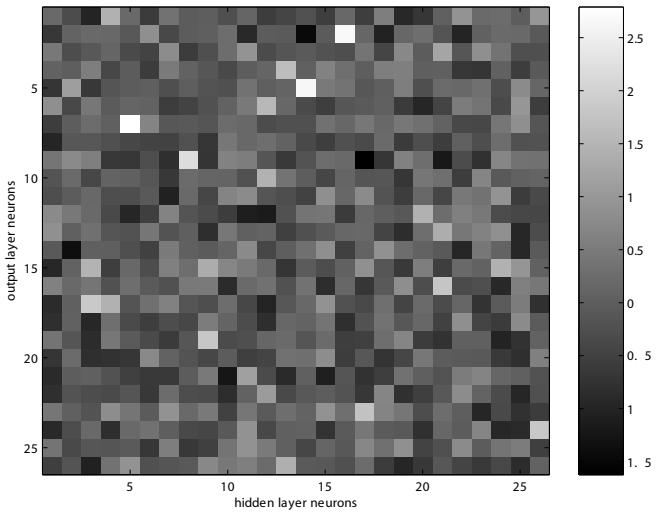


Figure 2.6: The output weights of an ANN used for a character recognition task. The ANN is a standard sigmoid network with one hidden layer evolved with direct encoding. The weight matrix connecting the single hidden layer to the output is plotted as a gray-scale bitmap. The fact that little structure appears evident in this high fitness solution, suggests a possible reason for the poor performance of the development system with the Kitano network encoding.

Compared to CTRNN, spiking networks are computationally more efficient but still allow rich non-linear dynamics. From the hardware point of view, spikes are easily implemented as voltage pulses, minimizing the effects of signal noise and reducing the complexity of the electronic circuits. Hardware implementations are clearly very interesting since ANN computation is extremely distributed.

Also, plasticity provides an additional motivation for the choice of pulsed models. When working with rate coding, correlation based synaptic plasticity usually takes the form of Hebbian or anti-Hebbian learning rules. There is other evidence that, probably, a large number of other rules exists (Abbott and Nelson 2000). With spikes, these rules can be easily produced with minimal computational requirements and few learning constants (see Paper I).

Multi-cellular neural encoding

Shifting away from the weight-centric Kitano network encoding, it is possible to use the local variables of a cell to encode neurons instead. Each cell type, metabolism and the chemical concentrations at its position are used do specify the network encoding, see Figure 2.7.

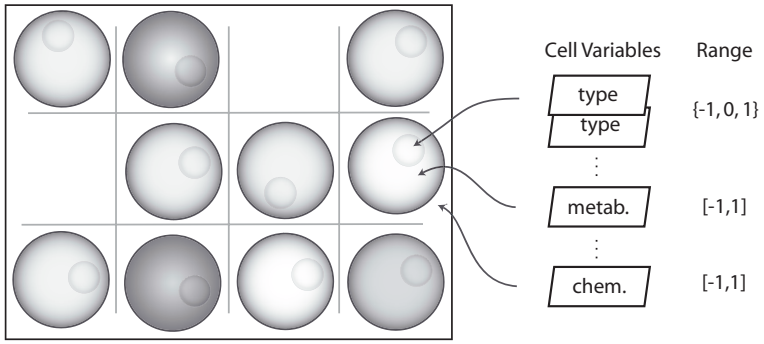


Figure 2.7: Depiction of the multi-cellular neural encoding. Each cell of the evolved multi-cellular organism is specified by a set of variables. The type of the cell is a vector of discrete ternary value. The metabolism and diffusing chemicals are vectors of real-valued concentrations in the range [-1,1]. Each cell variable is used to produce a distinct spiking neuron. The position of the cells defines the topology of the network.

The first implementation of this method is introduced in Paper H. In that case the network contains tripolar cells (one axon and two dendrite trees), with afferent connections coming from a single sensory input and the neurons in the local neighborhood, and the axon projecting to a specif output.

The cell type specifies the input and output connections, and their signs. Overall the network encoding requires 243 cell types, represented by a 5-dimensional vector of ternary values: 2 for the input and output signs (+,-,0), 2 for the input selection (one of the available 9 sensors) and the last for the output selection. An additional metabolic concentration is instead used to encode the neuron threshold, while the efficacies of the synapses to the neighboring neurons was computed as the difference in the local concentrations of diffusing chemicals.

The system was used to produce a network of integrate and fire neurons, applied to a harvesting task similar to the one used in Paper C and D. The results are summarized in Figure 2.8 showing that 75% of the runs produced nearly optimal controllers. The remaining 25% either lacked the ability to avoid poison or collect food.

The analysis of the best evolved neurocontrollers showed that the horizontal connections hardly ever influenced behaviour. The problem is that, based on the concentrations of diffusing chemicals, synaptic efficacies tend to vary very little across the network. As a consequence lateral processing was very limited (Paper H).

Another problem of this network encoding is the way that input and output connections are represented. Increasing the number of inputs and outputs also

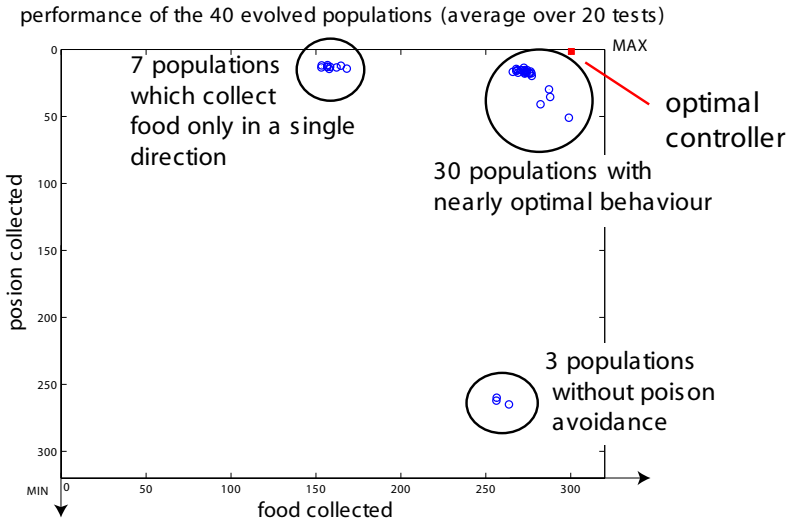


Figure 2.8: Harvesting behaviour of heterogeneous populations of evolved developing neuro-controllers (see Paper H). 30 out of 40 populations have a performance comparable to the one of a homogeneous population of optimal hand-written controllers. Notice that selection was handled at the individual level, so that competitive behavior was not excluded.

increases the number of required cell types. As a consequence, the number of inputs and outputs of the growth program also increase, and so does the search space. This would result in a limitation to the evolvability of networks with large input and output vectors.

In biological organisms, neuro-genesis is handled in a different way. Neurons do not encode their synaptic efficacies directly. There is increasing evidence that the proper configuration of many pathways involve plasticity. A well studied example is offered by the configuration of afferent connections to the lateral geniculate nuclei (LGN (Delcomyn 1998, pg 541-542)). In the LGN the information about the visual field is segregated so that the left side of field is processed and sent to the right side of the cortex and vice-versa. Although the connections from the retina to the LGN are well ordered and apparently would not require plastic mechanisms, it has been shown that the process goes through a prenatal Hebbian-based learning phase. Inhibition of synaptic plasticity blocks the development of the proper connections.

At first, it might seems that the introduction of unnecessary plastic mechanisms will just make the evolutionary task harder. On the other hand, if plasticity can be used to produce sufficient synaptic structures, by encoding each neuron's learning rule, the information of each single weight can be pruned away, decreasing dramatically the size of the genotype. It is not unusual, in fact, to find neurons having more than 10^4 synapses.

The specific topological structure of the neural assemblies, and their slight variations in terms of learning rules, internal dynamics and activation patterns, are exploited during the interaction with the environment to produce fine-tuned connections.

These effects have also been reported in the evolutionary computation community. In (Floreano and Mondada 1996) a Khepera robot was evolved in two different environments. In that case, plasticity was exploited in the earliest steps of fitness evaluation to affect the navigation strategy of the robot. A similar effect is shown in Paper I, where horizontal adaptive connections are used to produce subtle behavioral changes that increase the quality of the navigation task. It is also interesting to notice that in the latter case, plastic connections are totally unnecessary for the evolutionary task. Nevertheless, adaptivity causes an increase in evolutionary speed.

Concerning the connection to afferent inputs, neurons mostly connect to neighboring cells. For example, in the retina, the ganglions whose axons project to the optical nerve, receive inputs from the bipolar cells immediately below and from amacrine and other ganglion cells around them. This evidence suggests that the position of each cell can be used to determine its input-output connectivity.

The network encoding presented in Paper I therefore uses cell types only to encode the signs of the input and output connections. Metabolic concentrations are used to encode the value of the threshold and of the learning constants. The position of each cell is finally used to determine the synaptic connections.

Obtained results suggest that this topological encoding is both more efficient and scalable.

Self-Healing

As observed in (Miller 2004) and in Papers F and G, developing organisms display a sustained resistance to faults: ontogeny appears able to deal with inconsistencies in the phenotype.

In this cases, the resilience of developing organisms appears as an intrinsic property of ontogeny. On one hand this would help shred light on the amazing regenerative capabilities of living organisms. On the other, it offers a methodology to design fault-tolerant devices based on a self-healing process. With the regeneration of faulty components, self-healing artifacts could operate for very long periods without external technical assistance.

Previous related work concentrated on non functional organisms. In Paper J we have used the same Leaky Integrate and Fire spiking neural network of Paper I to produce self-healing neuro-controllers.

The results confirm the hypothesis formulated, suggesting that development based on local information has an intrinsic robustness. Quite interestingly, results show that the healing process is achieved by functional homeostasis: while the phenotype continuously grows new cells, the robot behaviour is stable. This is similar to what happens in biological organisms in the case of the epidermis and partially

also in the central nervous system (see for example (Zhao et al. 2003; Wu et al. 2001)).

Chapter 3

Conclusions

The work presented in this thesis is directed towards the construction of intelligent agents. Following the principles of the New AI wave, intelligent behaviour is evaluated in an ethological perspective.

This approach highlights the necessity of a holistic design, taking into consideration all sources of interactions between the agent, its behaviour, and the surrounding environment.

When the operation of biological organisms is analyzed, the picture that emerges is an engineer's nightmare.

Starting from the bottom, the cell functions are based upon a huge number of signaling pathways which are strongly intertwined. For example, the fundamental synaptic plasticity is based on second-order effects caused by the propagation of the neural signal, eventually triggering the activation of mRNA localized at synapses (Wu et al. 1998).

As it appears, complexity is already overwhelming in the basic building blocks of life. Considering this evidence, we have to consider how well traditional engineering approaches might be suited for our design objective.

The classical engineering approach is in fact based on the proper neglect of forces with lower order influences. For example, digital circuit design does not consider small voltage oscillations, and electronics neglects gravitational effects on electrons. Otherwise, an analytical treatise becomes impractical.

Compartmental models are therefore necessarily poorer. Even in the case of much simpler neural network models, it is argued that increased functionality derives by second-order effects of diffusing gases(Smith et al. 2002), or signal integration (Maass and Bishop 1998; Beer 1995).

While these or other lower order effects would be abstracted away in traditional design, evolution takes a holistic approach and solutions can exploit all the available subtleties.

An example is also provided by the number of adaptive mechanisms which operate simultaneously in biological organisms. Still, typical AI systems tend to include only one. We argue that, when trying to design grounded controllers for complex and autonomous robots, each of the four evolutionary, developmental, learning and cultural adaptive mechanisms, bring forward key characteristics. Each level provides

specific advantages either for design or for the organization of behavioral responses.

3-1 Evolution

As already argued, searches based on evolutionary methods have the advantage of allowing a very general exploration of the solution space. In fact, the search bias introduced by Genetic Algorithms is based on two fundamental assumptions:

- the solutions can be refined incrementally
- solution traits are phylogenetically inheritable

For our purposes, evolution is appealing as an aid in the design of robot controllers. Provided the possibility to judge the quality of behavior, evolved controllers will have a tendency to display a minimalistic elegance, i.e. to exploit every available resource: solutions often emerge from a complex composition of simple behaviours.

On the other hand, evolutionary search is very broad. As a consequence the dimension of the problem must be kept within a reasonable (small) size.

3-2 Development

Concerning search-space reduction, developing organisms can exploit intrinsic regularity in a solution to increase its evolvability. The various degrees of symmetry and modularity seen in living systems suggest that advantageous phenotypic traits could be discovered just once and reused multiple times.

In this sense, development allows the evolution-aided design of controllers with larger (but regular) sensory and motor connections (see Paper I).

Also, ontogeny could be exploited as an adaptive mechanism, both to recover functionality after faults and to produce functional stages during operation (see Paper J).

3-3 Learning

For controllers, plasticity offers a means to adapt to features which are unknown at design time. These may include component faults (e.g. fatigue) in which case the control strategy can adapt to the new conditions, or ad hoc aspects of the agent task requiring the use of novel relevant information (e.g. planning).

When considered from the evolutionary design perspective, another benefit of plasticity is that the inter-connections between controller modules, instead of being directly specified, can be dependent upon learning rules.

Since the number of connections among modules grows combinatorially with the number of modules, large systems should benefit from an encoding based on learning rules.

3-4 Culture

Here we adopt a broad definition of culture, simply as the pool of information that can be exchanged by non-genetic means.

The obvious advantage of a horizontal transmission of innovations is that interesting behaviors have a faster diffusion (see Paper C). Discoveries are made more readily available to the population, avoiding the necessity to 'reinvent the wheel' for phylogenetically unrelated species.

Still, the memetic diffusion process must be engineered carefully in order to avoid over-exploitative (Paper D) and epidemic effects. In Section 2-2, we argued that a crucial aspect is played by the genetic predisposition to the acceptance of novel information. Also in this concern, the innovation mechanisms provided by the learning substrate appear fundamental.

Culture also offers an advantage associated with the development of cognitive functions. If on one hand it is argued that language structure is derived from the structure of the brain (Arbib 2001), on the other, it is also argued that the necessity to acquire information from other agents produces an increased level of knowledge organization in the brain (Calvin and Bickerton 2000). The structure of retained knowledge can be interpreted as an emergent internal representation. For example, in (Tani 2003; Cangelosi 2004) and (Sugita and Tani 2005, in press) more structure is shown to emerge when language and imitation are allowed to interact with the production of behavioral responses.

3-5 Design of Multi-Level Adaptive Agents

Putting all these arguments together, in Figure 3.1 we depict the relationships among the four analyzed adaptive mechanisms aimed at the production of autonomous intelligent agents.

At the base we imagine the designer specifying a selection criterion (inclusive fitness) for the evolutionary process. Evolution conducts a broad holistic search allowing the emergence of relationships among the available adaptive mechanisms.

To increase scalability, the controller structure (topology) is produced by a developmental process and plastic rules are used to set up the internal connections among modules. Also development stages and learning interact to produce online (lifetime) adaptation to the environment.

At the top, horizontal transmission allows a fast and directed information acquisition process. By exchanging information, agents collaborate to reduce the necessity of exploring the environment (bootstrapping). To achieve efficient transmission, evolution will facilitate the refinement of the acquisition dynamics based on stigmergy, imitation or language.

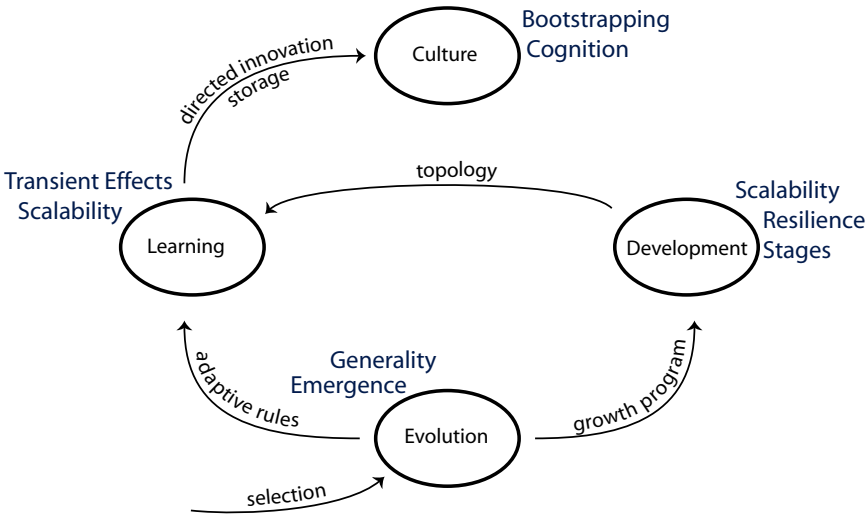


Figure 3.1: Hypothesis for the construction of an autonomous intelligent controller: inter-connections among different adaptive mechanisms. Each mechanism provides particular benefits (shaded), and participates to the construction of higher level mechanisms (arrows).

Appendix A

Papers

A Limitations of gradient methods on sequence learning

Author: Diego Federici. Published in proceedings of ICONIP 2002 9th International Conference on Neural Information Processing.

Abstract:

Recurrent neural networks, such as the well-known Simple Recurrent Network, trained to predict their own next input vector offer a promising framework for developing internal representations of environmental structure. Current training techniques focus on the use of different gradient methods and genetic search. These techniques have the advantage of being general, to develop distributed representations and to achieve holistic computation. On the other side their generality does not pay off in terms of learning speed, accuracy or flexibility. In this paper a temporal learning problem is analyzed with respect to traditional on-line learning approaches. The results show that gradient methods do not offer a way to identify and correct the actual cause of misclassifications and so are prone to be stuck on local maxima.

Objective:

Investigate Simple Recurrent Networks as an autonomous and grounded inductive methodology.

Conclusions:

Limitations appear connected to the misuse of information derived by reinforcement. The overcoming of these problems require a more directed (episodic) retention system, but current models do not seems to be able to scale up to realistically complex problems.

LIMITATIONS OF GRADIENT METHODS IN SEQUENCE LEARNING

Diego Federici

Norwegian University of Science and Technology
Department of computer and information science
N-7491 Trondheim, Norway
Federici@idi.ntnu.no

ABSTRACT

Recurrent neural networks, such as the well-known Simple recurrent Network (SRN,[1]), trained to predict their own next input vector offer a promising framework for developing internal representations of environmental structure. Current training techniques focus on the use of different gradient methods and genetic search. These techniques have the advantage of being general, to develop distributed representations and to achieve holistic computation. On the other side their generality does not pay off in terms of learning speed, accuracy or flexibility. In this paper a temporal learning problem is analyzed with respect to traditional on-line learning approaches. The results show that gradient methods do not offer a way to identify and correct the actual cause of misclassifications and so are prone to be stuck on local maxima.

1. INTRODUCTION

Training recurrent networks on an anticipatory task (prediction of the next input vector) offers a way to develop unique distributed representations of input sequences. Such representations are considered to be the first step towards a process of abstraction and symbolic reasoning [2].

The continuous input-output pairing permits the use of traditional supervised gradient methods [1]. On the other side, the recurrent connections prevent the computation of the exact error gradient for infinite sequences.

Back propagation through time (BPTT, [3]) is a gradient training technique that can compute the error gradient for finite sequences only. Some experiments have used BPTT splitting the input sequence in subsequences both with SRNs [1] and RAAMs [4] as in [5, 6, 2, 1]. Reducing the input stream in subsequences requires a certain supervision that does not appear plausible in the context of autonomous learning.

SRNs and RAAMs have typically been applied to handling symbolic sequences like in language acquisition. One of the reasons is that language is a well established metaphor for every kind of input-output pairing. On the other side, in written language, words are clearly separated by spaces and sentences by punctuation, simplifying the creation of subsequences. But this appears to be an exception to the general case.

Let us consider a simple task of orientation. The autonomous agent is equipped with four proximity sensors and it is allowed to wander in a closed maze. The proximity sensors are really feeble and allow the agent to perceive walls only in the immediate neighborhood.

Robotic Navigation learning using ANNs is not a new task in the literature [7, 8, 9]. In this case, the orientation task is interpreted as a metaphor of a general cognitive task that involves the discrimination among long deterministic input sequences. There is evidence that recurrent ANNs perform well on short sequences but they show a decreasing performance on long ones [1, 4, 2]. It is still not clear whether this is caused by the stochastic nature of learning tasks or by deficiencies of the learning model.

Here, the cognitive task consists of building an anticipatory system, capable of predicting the effects of action movements on the sensory input, formally:

$$CT : sensors_t \times action_t \rightarrow sensors_{t+1}$$

CT is deterministic and requires learning the map of the maze and the effects of movement from a subjective perspective, i.e. in relationship to the input vectors.

This task is similar to what is continuously done by animals in order to know, for example, the fastest escape routes and food locations. Map learning is a topic that has been deeply studied in psychology [10, 11] and several investigators demonstrated that the rat learning process appears to be performed continuously and in the absence of rewards. Rats are apparently acquiring a cognitive map from sensory experiences [12].

From the point of view of the prediction system, map learning and language acquisition are equivalent tasks. The difference lies only in the interpretation of the tokens that build up the training sequences (see figure 3).

In the case of language, tokens represent words, phonemes or letters. In the case of map learning, tokens are the presence or absence of obstacles in the different directions. Also, for orientation, the input sequence is not composed of independent sub-sequences, so that online learning techniques are required.

In order to predict the next token the extended SRN (eSRN, see figure 1) must learn to distinguish similar input vectors, implicitly keeping track of the robot's location.

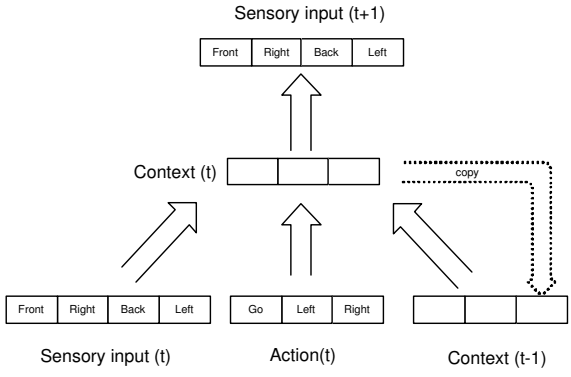


Figure 1: Predictor structure: eSRN

The contextualized input vectors are encoded in the hidden context layer in a form suitable to the production of correct outputs. In this case, a functional representation of the maze's map.

1.1. Function

When properly trained, an anticipatory system, can be useful for simulating the effects of actions. In this context, the controlling system of a robot can use it to perform a simulation of possible action strategies and deliberate among the possible alternatives [9, 7, 8].

Also, the context state encoded in the hidden layer of a eSRN is a compact representation of what the robot can perceive of the environment. The representation is minimal with respect to the learning task, because it will only contain those features useful for its fulfillment.

In this sense, this representation is viewed as the first step toward a process of abstraction.

1.2. Requirements

There are three major requirements that a training algorithm should provide:

$F1$ *Content*: encode arbitrary input sequences

$F2$ *Space*: develop distributed representations

$F3$ *Time*: encode and decode in real-time

$F1$ expresses a requirement of universal computation. If there are some meaningful sequences that cannot be encoded, then the system is clearly not complete.

$F2$ requires that the number of states representable in the hidden layer grows exponentially with the number of nodes used. As the information encoded in the context layer is a representation of the current and past input vectors, the extensive use of localistic representations are not fit to real-world problems, as the complexity of the environment requires the use of a huge number of representational states.

$F3$ is necessary for the anticipatory system to be useful. The requirement is both for the system's usage, encoding and decoding of the representations, and for its acquisition speed.

2. EXPERIMENTAL SETTINGS

The task is to train an anticipatory system, realized by a SRN, ideally mounted on a simulated robot. The simulated robot is immersed in a 2D environment and it is free to move along corridors. It can only perceive the presence or absence of neighboring walls in the four cardinal directions. To reduce the environmental clues, which allow a greater part of the task to be solved reactively, the active spot of the sensors is minimal (see figure 2). Its movement is determined by an external controlling module which decides one of the three possible actions: move ahead, rotate left, rotate right. The controlling module simply follows a corridor and when faced with several possible alternatives selects one at random.

The predictor network receives as input a seven bit vector. Four bits are set according to the state of the four proximity sensors, and each of the remaining three encode one of the possible actions (figure 1). Notice that, even if the controller is stochastic, the eSRN knows what action is being performed and so its task is deterministic. The predictor network is expected to produce as output the following state of the four proximity sensors.

The eSRN is trained in the map shown in figure 2.

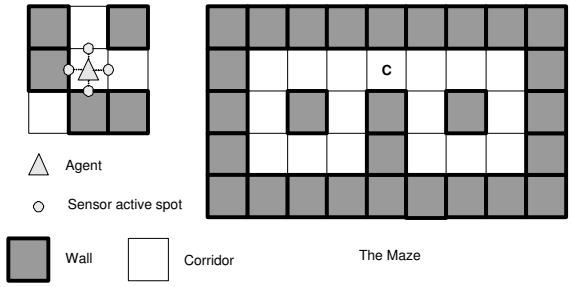
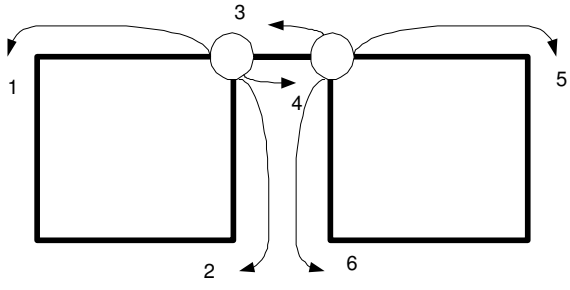
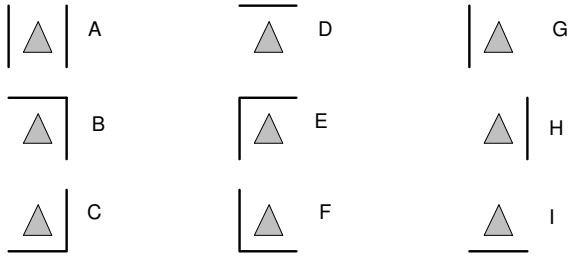


Figure 2: The maze used in the experiments

Given the control strategy explained above, there are 6 possible paths that the robot can follow (see figure 3). The current path determines the experienced input sequence as shown in table 1. This control strategy maximizes the need of contextual memory because of the long sequences it generates.



(a) possible pathways



(b) possible sensory inputs as tokens

Figure 3: paths and input tokens

2.1. Training of the predictor

The eSRN has been trained with different gradient methods: standard back propagation (BP), back propagation through time (BPTT), truncated BPTT with approximation for small weight change (aBPTT{κ}) [1] and standard BPTT{κ} (BPTT{κ}) [2]. A reactive ANN (R), without recurrent connections, has been trained for performance comparison. For a reference on the different gradient methods see [13, 14, 3].

2.2. Environmental settings

Each input bit is represented by a floating number, a high bit is represented by 0.5 and a low bit by -0.5. In the output layer a positive value represents a high bit. White noise $\theta \in [0.05, -0.05]$ is added to the inputs to increase stability. The nodes use a hyperbolic tangent function.

Different training strategies have been applied. Variations include the use of different learning rates (η) and momentum (μ), 5 or 10 steps of error back-propagation (κ), the use of incremental learning and of a trophic wave of plasticity.

Incremental learning [15] is a technique that consists in perturbing the contextual layer(randomizing the node's activation) every τ iterations. τ is slowly increased to infinity. This method modulates the time span of the working memory, limiting the algorithm to the extraction of simple relationships at first. As the working memory span increases with τ, more complex relationship can be acquired.

The trophic wave of plasticity is a method adapted from Sharanger and Johnson [16]. Plasticity is regulated by a wave that moves across the context layer. The wave regulates a node-specific learning rate.

3. RESULTS

The results in table 2 show the minimum and average misclassification statistics over 10 different runs for each parameter settings. eSRNs have been trained for one million iterations and performance is checked over an additional 1000. By default the steps of error back-propagation κ is set to 5, and the context size CS to 10. iBP stands for BP using incremental learning, while tBP using the trophic wave of plasticity.

3.1. Analysis

Given that the prediction task is deterministic, it is possible to solve it without misclassifications. None of the eSRNs managed to solve it completely even if they performed substantially better than the reactive

[1] aBPTT{κ} is similar to BPTT but injects only the error of the last iteration and propagates it back κ times

[2] BPTT{κ} is similar to aBPTT{κ} but it keeps a memory of the weight history to be used in the back-propagation phase

Path	*Sequences of input vectors* for actions S= go straight, L= turn left, R= turn right, ?= more than one possible											
1	A+S	B+L	C+S	A+S	B+L	C+S	A+S	B+L	C+S	A+S	D+?	
2	I+S	A+S	E+R	F+S	A+S	E+R	F+S	A+S	E+R	F+S	A+S	G+?
3	H+S	A+S	H+?									
4	G+S	A+S	G+?									
5	A+S	E+R	F+S	A+S	E+R	F+S	A+S	E+R	F+S	A+S	D+?	
6	I+S	A+S	B+L	C+S	A+S	B+L	C+S	A+S	B+L	C+S	A+S	H+?

Table 1: possible input sequences

Training parameters		R	BPTT	aBPTT{κ}	BPTT{κ}	BP	iBP	tBP
$\eta = .1\ \mu = .5$	Best	20.8	19.6	9.0	13.5	12.2	11.1	12.8
	Mean	25.2±3.8	27.0±7.5	13.6±2.9	15.5±1.7	15.1±4.4	14.0±1.9	23.7±10.9
$\eta = .05\ \mu = .5$	Best	19.9	12.8	10.1	9.4	10.0	11	14.1
	Mean	25.0±4.8	22.7±7.2	13.5±2.8	13.0±2.4	13.7±3.4	15.3±5.8	21.5±9.2
$\eta = .01\ \mu = .75$	Best	21.9	10.3	10.5	**7.5**	12.5	9.7	19.0
	Mean	27.9±4.6	17.1±5.2	13.0±2.0	**13.3±1.9**	17.1±4.6	13.9±3.4	25.2±5.7
$\eta = .01\ \mu = .75$	Best		10.5	10.7	11.5			
$\kappa = 10$	Mean		31.0±25.3	15.6±3.8	13.4±1.2			
$\eta = .01\ \mu = .75$	Best	23	8.3	10.5	10.8	9.8	10.5	13.5
$CS = 15$	Mean	31.3±13.9	16.7±6.7	13.5±2.3	13.2±1.5	14.1±4.0	16.2±5.2	26.6±8.1
$\eta = .05\ \mu = .5$	Best	20.7	9.9	8.8	10.0	10.7	9.3	17.5
$CS = 20$	Mean	23.5±2.8	18.3±5.4	11.4±1.9	12.1±1.1	17.8±9.8	13.1±3.2	27.7±5.8

Table 2: misclassiffication statistics in percentage: best, mean ± std. In emphasis the best performing eSRN

solutions. Incremental learning showed a faster convergence without altering the quality of the predictor. The best results has been obtained with BPTT{κ}, method that also shows the least performance variance. The most interesting result is that all the best performing eSRNs convergence to the same suboptimal strategy, which involves a typical misclassification pattern around the central tile (marked with a c in figure 2). As the suboptimal strategy emerges contantly, it highlights a limitation of the training algorithms.

3.2. The source of the problems

Consider the input sequence from path 1 (or similarly paths 2, 5 and 6). The sequence is composed of three identical turns to the left that lead to a T-junction in proximity to the central tile (c). Notice that the internal representation should be different for every left turn. The first left turn leading to the second, the second to the third and the third to the T-junction.

The recording of the activation patterns during testing, show that in the trained solutions this does not happen. Some or all the three turns share the same internal representation. Because of this, while the robot moves along the corridor the predictor system is stucked in an imaginary infinite loop.

In fact, in this context, the repetition of an internal representation works as a recursive call without a termination condition.

When the robot reaches the end of the corridor (path 1), the predictor system is still in the imaginary loop and can not anticipate the incoming of the central tile opening. It erroneously anticipates a wall instead of the corridor.

The central tile prediction errors are typically small. Usually a wall, in a trained network, is predicted by an output of $\sim$0.35 while in this cases it is only $\sim$0.05. The meaning of it being that the gradient methods treat the prediction error as if it was noise from the sensors.

The problem is that the manifestation of the error depends on a mistake committed seven iterations in the past, which is the repetition of the internal representation. The problem could be fixed by changing the representation encoding, but the error gradient contains no clue about this possibility. This is true also for training algorithms with $\kappa = 10$, which propagate the error 10 steps in the past.

An other source of ambiguities is that similar sequences (paths 1 and 6, and paths 2 and 5) often share the same internal representations. The reason being the same.

The decoding subnet behaves as when faced with

the problem of approximating a stochastic function. For the central tile a wall is predicted because is the most frequently experienced.

This appears to be the limitation of this *"moving target learning strategy"* [4], for which it is impossible to roll back to the real causes of error.

3.3. Conclusions

Gradient based training on eSRN presents two major sources of problems.

The first one is connected to the acquisition speed of the method. The extensive amount of training required to learn complex sequences with back-propagation is a well known problem [17].

The second problem is the quality of the developed representations. Representations do not stand for unique contextual information (maze's loci) and contextual information does not have a unique representation.

While the latter simply states that there are multiple definitions for the same concept, the first one addresses a semantical problem. The same internal representation has several possible meanings, leaving the robot lost in the maze.

Notice that the internal representations emerge as a side effect of the error back-propagation due to the fluctuations in the unbound contextual layer. When a representation mistake is committed it is difficult to recover (escape the local optimum) because of the difficulty to recognize the actual source of error.

Instead of fixing the encoding error, it acts to correct a decoding failure as if the output contained noise. This is a result of the inductive bias of back-propagation.

4. ACKNOWLEDGMENT

I wish to thank Tom Ziemke for the constructive comments in the process of writing this paper.

5. REFERENCES

[1] J.L. Elman, Finding structure in time, *Cognitive Science*, 14:179-211, 1990.

[2] I. Stoianov, Recurrent Autoassociative Networks and Holistic Computations, *International Joint Conference on Neural Networks*, Como (Italy), 2000.

[3] S. Haykin, *Neural Networks. A Comprehensive Foundation*, Prentice-hall, New Jersey, second edition 1999.

[4] J.B. Pollack, Recursive Distributed Representations, *Artificial Intelligence*, 46:77-105, 1990.

[5] D.J. Chalmers, Syntactic transformations on distributed representations, *Connection Science*, 2:53-62, 1990.

[6] L. Chrisman, Learning recursive distributed representations for holistic computation, *Connection Science*, , 3:345-365, 1991.

[7] J. Tani, Model-based learning for mobile robot navigation from the dynamical systems perspective, *IEEE Trans. system, man, and cybernetics Part B*,26(3):421-436, 1996.

[8] L. Meeden, G. McGraw, D. Blank, Emergent control and planning in an autonomous vehicle, *Proc. 15th annual conference of the cognitive science society*, Lawrence Erlbraun, Hillsdale NJ, 1993.

[9] D.A Jirenhed, G. Hesslow, T. Ziemke, Exploring internal simulation of perception in mobile robots, *Fourth European Workshop on Advanced Mobile Robots*, Lund (Sweden), 2001.

[10] L.M. Barker, *Learning and behavior*, Prentice Hall, New Jersey, 1996.

[11] H.C. Blodgett, The effect of the introduction of reward upon maze performance of rats, *University of California publications in psychology*, 4:113-134, 1929.

[12] E.C. Tolman, Cognitive maps in rats and men, *The psychological Review*, 55(4), 189-208, 1948.

[13] P. Campolucci, A. Uncini, F. Piazza, A Unifying View of Gradient Calculations and Learning for Locally Recurrent Neural Networks, *Proc. of WIRN97, Italian Workshop on Neural Networks*, Vietri sul Mare (Italy), 1997.

[14] M. Chester, *Neural Networks*, Prentice-hall, New Jersey, 1993.

[15] J.L. Elman, Origins of language: a conspiracy theory, *The emergence of language*, Hillsdale NJ, Lawrence Erlabraun, 1999.

[16] J. Shranger, M.H. Johnson, Dynamic plasticity influences the emergence of function in a simple cortical array, *Neural Networks*, 8:1-11, 1996.

[17] S.C. Kwansy, B.L. Kalman, Tail-recursive representations and simple recurrent networks, *Connection Science*, 7(1):61-80, 1994.

B Implicant network: an associative memory model

Author: Diego Federici. Published in proceedings of IJCNN 2003 International Joint Conference on Neural Networks.

Abstract:

The Implicant Network is a neural network model capable of storing an arbitrary boolean function $F : 0, 1^n \rightarrow 0, 1$. The difference from previous one-shot learning models is that the training algorithm compresses the positive set online with linear time and space requirements. The algorithm works by building a Sum Of Products (SOP) representation of the positive set as it is presented to the network. Since the minimum coverage of implicants is an NP-Hard problem, the compression rate is not optimal at first but it is shown to increase rapidly as the positive set is shown over again.

Objective:

Produce a realtime episodic retention system which also compresses information during operation.

Conclusions:

The systems works and additionally uses only local learning (Hebbian) rules. Compression is achieved by a recursive process. Further work should focus on the extension of the system to multi-dimensional outputs. Analytical and evolutionary approaches have been attempted, but based on the time constraint imposed by the project, it was strategically decided to postpone the investigation.

Implicant network: an associative memory model

Diego Federici
Norwegian University of Science and Technology
Department of computer and information science
N-7491 Trondheim, Norway
Email: Federici@idi.ntnu.no

***Abstract*— The Implicant Network is a neural network model capable of storing an arbitrary boolean function $F : \{0,1\}^n \rightarrow \{0,1\}$. The difference from previous one-shot learning models is that the training algorithm compresses the positive set online with linear time and space requirements. The algorithm works by building a Sum Of Products (SOP) representation of the positive set as it is presented to the network. Since the minimum coverage of implicants is an NP-Hard problem, the compression rate is not optimal at first but it is shown to increase rapidly as the positive set is shown over again.**

I. Introduction

When designing an autonomous agent, a short term memory storage is often a necessary mechanism, also found in animals and humans. Its functioning is associated with many cognitive tasks [1], not only for remembering a phone number or the name of a street but also more general memory tasks, like that a certain street is blocked for works or that John just left the office. When a robot has to navigate a given environment it has to remember several clues in order to orientate. Rats have proven to be really good at these tasks [2]. The number of clues can be considerable in the general case so it is interesting to investigate to which extent it is possible to compress information without altering the memory content. In order to fit a mobile agent, the acquisition mechanism must operate online and be capable of recalling and compressing in real-time.

A. Different approaches

There is no associative memory model that we are aware of that compresses information online and in real-time. Those algorithms that achieve good compression need the complete training set in order to operate. Those that can store patterns as they are presented, do not compress much at all.

In the following discussion, the algorithms are encoding an arbitrary $F : \{0,1\}^n \rightarrow \{0,1\}^m$.

BiAssociative Memories (BAM,[3]) belong to the first family. They are capable of storing $p \leq min(n,m)$ patterns[1]. Other algorithms, such as Hamming Associative Memory [5] Flash and Solar systems [6], also belong to the first group. They achieve optimal storage given assumptions of good orthogonality of the input patterns.

For online algorithms, a simple solution consists of adding a node for each new pattern, see for instance [7]. This can be implemented in a simple way, but it does not compress information at all. Correlation Matrix Memory (CMM), is a binary perceptron approach. Perceptrons are unable to distinguish between non-linear separable classes. CMM solve this problem by projecting the input space with some non-linear function into a higher dimension space. As the new input space increases, the probability that the classes became linearly-separable increases too. CMM is easily implemented in hardware [8], which makes this approach feasible. Still, the algorithm does not really compress information. It is also possible that the projecting function needs to be changed as new patterns are presented.

The Implicant Network is capable of storing an arbitrary function with $m = 1$. The information is acquired as it is presented to the network and compressed in no more than n activation phases. In a mobile setting, the same pattern is likely to be experienced several times and this proves to be beneficial to the system as the compression rate is steadily improved. The algorithm compresses information by optimizing a SOP formula.

Any boolean function can be represented with a SOP formula such as $G(a,b,c) = abc \vee bc \vee (\neg b)c$. Each of the constituents of a SOP formula (in this case 'abc', 'bc' and '$(\neg b)c$') is called implicant. An implicant containing every possible variable (or its complement) is called minterm (such as 'abc'). A minimal cover is the set of implicants with minimal cardinality that represents a function. In the previous example, the minimal cover of G is 'c'.

Finding the minimal cover of a SOP formula has been proven to be NP-Hard [9].

II. The Implicant Network

The network of Figure 1 implements an arbitrary function $F : \{0,1\}^n \rightarrow \{0,1\}$. At the beginning the network is empty and F maps every input to 0. As positive exemplars (minterms) are presented the training algorithm rearranges the Predicate Neurons (PN) that connect to the output. Each PN is equivalent to an implicant of the SOP formula under construction.

Every time a pattern is presented during training, a new PN (PN_{new}) is temporarily created. The PN_{new} is going to be kept only if either:

1) it encodes new information, or
2) it increases the information compression

In this model, properties 1-2 are checked with a feedback phase that propagates backward the activation of the PN to

[1]Given constrains on the input pattern set, the storage capacity can be much higher, see also [4]

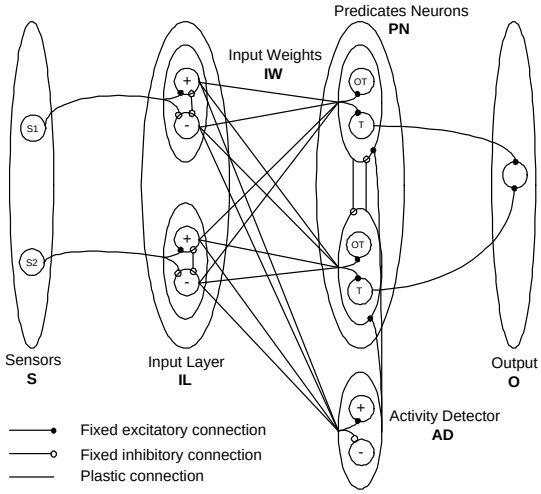


Fig. 1. The Implicant Network

the input layer (IL) and then forward to the PNs again.

This feedback-feedforward phase is used to identify the relationships between PN_{new} and the PNs that were contained in the network. In particular, the feedback-feedforward phase identifies:

1) the set of PNs that PN_{new} covers[2] in all but a single dimension. We will refer to this set as the set of Sub-Threshold neurons ($SubT\ PN$s).
2) the set of PNs that cover PN_{new}. We will refer to this set as the set of Over-Threshold neurons ($OT\ PN$s).

$SubT\ PN$s are covered by PN_{new} in all but a single dimension. In other words, given a $SubT\ PN$ it exists a dimension $i \in [1, n]$ such as without that PN_{new} would cover entirely the $SubT\ PN$, but with that dimension it does not.

For every minterm m_k covered by the $SubT\ PN$ it exists a minterm $\overline{m}_j$ covered by PN_{new} such as m_k and $\overline{m}_j$ differ only in dimension i. It is then possible to extend the $SubT\ PN$ by dropping the constrains on variable i.

Example: let PN_{new} = 'ab'. PN_{new} covers minterms m_1 = 'abc' and m_2 = '$ab(\neg c)$'. Consider a node PN_1 = '$a(\neg b)c$'. PN_1 is a $SubT\ PN$ of PN_{new} because without the dimension of variable 'b', PN_{new} covers $SubT\ PN$. It is then possible to expand the coverage of PN_1 by removing the constrain on the variable 'b'. PN_1 becomes 'ac'. The extended coverage of PN_1 does not add minterms/patterns to the network, but by adding redundancy it might allow the removal of some other PN.

Redundancy is checked by the activation of the set of Over-Threshold neurons ($OT\ PN$s). The existence of a $OT\ PN$ signals that some node in the network already covers all the information covered by PN_{new}. PN_{new} can be removed without altering the information contained in the network.

OT and $SubT\ PN$s are activated in the feedforward phase thanks to the modulation of the Activity Detector (AD). AD is activated only during training, see equation 2.

Basically, the training algorithm searches for the $SubT$ PNs of the input pattern. If any $SubT$ exists, the PN_{new} merges[3] with the $SubT$, it is fed-back to the input, and the process continues.

Once no $SubT$ is found, the presence of an active OT signals wether the PN_{new} is redundant and must be discarded.

These processes do not actually require a standard search, but occur with a single activation of the predicate layer in a totally parallel way. The use of this parallel search is the major contribution of this paper.

A. Layers details

An input pattern is a vector $P \epsilon \{0,1\}^n$. The sensors layer (S) is set to P, each node representing a bit of the vector.

The Inpuit Layer IL contains $2\,n$ units. For each unit a_i in S, there is a couple $[a_i^+, a_i^-]$ in IL, where a_i^+ is equal to a_i and a_i^- is the negation of it. For example the S vector $[1\ 0\ 0]$ gives the IL vector $[1\,0\ \ 0\,1\ \ 0\,1]$. Notice that, in this case, the total number of high nodes in IL is always n.

Let's define this function as:

$$IL = F_{FF}(S) \tag{1}$$

The functions governing the activity of the other layers follow:

$$\begin{aligned} AD &= F_{AD}(IL) \\ F_{AD}(IL) &= \begin{cases} n - (\mathcal{I} \cdot IL) + 1 & if\ training \\ 0 & otherwise \end{cases} \end{aligned} \tag{2}$$

$$\begin{aligned} PN &= F_{PN}(IL, AD) \\ F_{PN}(IL, AD) &= \theta_n((IW \cdot IL) + AD) \end{aligned} \tag{3}$$

$$\begin{aligned} OT &= F_{OT}(IL, AD) \\ F_{OT}(IL, AD) &= \tilde{\theta}_n((IW \cdot IL) + AD) \end{aligned} \tag{4}$$

$$O = F_O(PN) = \begin{cases} 1 & if\ any\ PN\ is\ 1 \\ 0 & otherwise \end{cases} \tag{5}$$

where O is the output of the Implicant Network and

$$\theta_n = \begin{cases} 1 & if\ n \\ 0 & otherwise \end{cases} \quad ; \quad \tilde{\theta}_n = \begin{cases} 1 & if > n \\ 0 & otherwise \end{cases}$$

$$\begin{aligned} &IW \epsilon \{0,1\}^{\kappa \times 2\,n}; \quad \kappa = \|PN\| \\ &\mathcal{I} = [1,\ 1,\ \ldots,\ 1] \end{aligned} \tag{6}$$

Equations 1–5 can be used to compute the activation of the network: $O \leftarrow activate(S)$.

[2] as every PN stands for an implicant, we will often refer to an implicant as the PN it is represented from

[3] merging is the process of extending coverage, see subsection II-D for further details.

B. The input weights

The input weight matrix (IW, see equation 6) has a row for each PN. When PN_{new} is created, an empty row is added to IW. During merging, IW will change with the following formula (Hebbian learning):

$$IW_{PN_{new}} \leftarrow IW_{PN_{new}} \vee IL^2 \tag{7}$$

so that, following equation 2, the new node will be active when IL is further presented. IL^2 is computed from IL removing activation from pairs of corresponding units a_i^+ and a_i^- when they are active at the same time (see also below).

C. Feedback

The feedback deserves some extra attention. The backward activation of IL is computed as follows:

$$IL = F_{FB}(PN) = IW^T \cdot PN \tag{8}$$

Also, IL has a delayed lateral inhibition between each opposite unit a_i^+ and a_i^-, so that if they are both high, they will inhibit each other. Let IL^1 define the activity in IL before lateral inhibition, IL^2 the one after lateral inhibition.

When equation 1 is used, lateral inhibition is negligible because $IL = IL^1 = IL^2$. But if the PN feeding back covers more than a single pattern then $\exists\ i\ \epsilon\ [1, \kappa]$ such as a_i^+ and a_i^- that belong to IL and are both high.

Example: if PN_i is active and covers patterns [0 0] and [0 1] then IW_i = [0 1 0 1] $\vee$ [0 1 1 0] = [0 1 1 1] and equation 8 yields IL = [0 1 1 1]. IL^1 is then equal to [0 1 1 1], while IL^2 is [0 1 0 0].

AD from equation 2, also has two values, AD^1 from IL^1 and AD^2 from IL^2. Similarly PN^1, PN^2, OT^1 and OT^2 are defined.

In this case, a PN is defined high when both PN^1 and PN^2 are high. The same applies to OT.

D. Merging and Deleting

When a new node PN_{new} is added to the network, it is possible that it finds a $SubT$ node to merge with. In this case, $IW_{PN_{new}}$ is updated. The $SubT$ node feeds back (see equation 8) yielding $IL_{SubT} = IW^T SubT$ and IW is updated as follows:

$$IW_{PN_{new}} \leftarrow IW_{PN_{new}} \vee IL^2_{SubT} \tag{9}$$

Equation 9 derives directly from equation 7.

Example: suppose that PN_{new} covers pattern [0 1] so that $IW_{PN_{new}}$ = [0 1 1 0]. $SubT$ covers instead [0 0] and [1 0] so that IW_{SubT} = [1 1 0 1]. Then, IL_{SubT} = [1 1 0 1] which implies IL^1_{SubT} = [1 1 0 1] and IL^2_{SubT} = [0 0 0 1]. Using equation 9 we find:

$$IW_{PN_{new}} \leftarrow [0\,1\,1\,0] \vee [0\,0\,0\,1]$$

$$IW_{PN_{new}} \leftarrow [0\,1\,1\,1]$$

which implies that PN_{new} now covers patterns [0 1] and [0 0]. Notice that, as it was the $SubT$ node feeding back, OT now signals whether $SubT$ must be deleted or not.

E. training pseudo code

$Output \leftarrow activate(S)$;
$IL \leftarrow F_{FF}(S)$;
Create a new predicate neuron (PN_{new});
Set its input weights to $IW_{PN_{new}} \leftarrow IL$;
loop-again $\leftarrow$ 1;
A-PN-Needs-To-Be-Deleted $\leftarrow Output$;
while loop-again = 1;
  loop-again $\leftarrow$ 0
  $IL \leftarrow F_{FB}(PN_{new})$; //**feedback**
  $AD \leftarrow F_{AD}(IL)$
  $SubT \leftarrow F_{PN}(IL, AD)$
  $OT \leftarrow F_{OT}(IL, AD)$
  if at least one $SubT$is high, //$SubT$**and** PN_{new}
      would be adjacent on a karnagh map
    $SubT_i \leftarrow$ pick one high $SubT$at random
    $IL \leftarrow F_{FB}(SubT_i)$; //**feedback**
    $IW_{PN_{new}} \leftarrow IW_{PN_{new}} \vee IL^2$; //**merge**
    $AD \leftarrow F_{AD}(IL)$
    $\widehat{OT} \leftarrow F_{OT}(IL, AD)$
    if at least one $\widehat{OT}$ is high and is not $SubT_i$,
        //$SubT_i$ **is redundant because of** $\widehat{OT}$
      remove $SubT_i$;
      A-PN-Needs-To-Be-Deleted $\leftarrow$ 0;
    end if
    loop-again $\leftarrow$ 1;
  else if (there is an high OT and is not PN_{new}) OR
      A-PN-Needs-To-Be-Deleted
      //PN_{new} **is redundant because of** OT **OR**
      //**It does not increase compression as it**
      //**did not remove any** PN
    remove PN_{new}
  end if
end while

It is possible to download the Matlab source code, see footnote[4].

F. A training example

In this section, we give a step by step trace of a simple training run of the Implicant Network.

Let the positive set for a function of 2 variables be {[00], [10]}. This is equivalent to the function $H(a, b) = (\neg a)(\neg b) \vee a(\neg b)$.

A pattern ([00]) is selected at random.
S = [00] implies IL = [01 01].
The input weights $IW_{PN_{new}}$ are set to IL = [01 01].

[4] http://www.idi.ntnu.no/~federici/code/implicant

PN_{new} is the first node in Implicant Network so
$IW \equiv IW_{PN_{new}}$ and $PN_{new} \equiv PN_1$.
Feed-Back Phase:
$IL = IW_{PN_{new}}$ = [01 01].
no lateral inhibition takes place.
$AD = 2 - 2 + 1 = 1$.
compute the $SubT$ set
$(IW \cdot IL) + AD = 3$.
$\Theta_2(3) = 0$.
$SubT \leftarrow \emptyset$.
compute the OT set
$(IW \cdot IL) + AD = 2$.
$\tilde{\Theta}_2(3) = 0$.
$OT \leftarrow \emptyset$
there is no $SubT$.
there is no OT.
no PN needs to be deleted.
so keep PN_{new} as PN_1.
no merging occurred: training is finished.

A new pattern is selected at random: [10].
S = [10] implies IL = [10 01].
The input weights $IW_{PN_{new}}$ are set to IL = [10 01].
PN_{new} is the second node in the network so $PN_{new} \equiv PN_2$
and $IW = \left| \begin{array}{lcccc} IW_{PN_1}: & 0 & 1 & 0 & 1 \\ IW_{PN_2}: & 1 & 0 & 0 & 1 \end{array} \right|$
First Feed-Back Phase:
$IL = IW_{PN_{new}}$ = [10 01]
no lateral inhibition takes place.
$AD = 2 - 2 + 1 = 1$.
compute the $SubT$ set
$(IW \cdot IL) + AD = [2\ 3]^T$
$\Theta_2([2\ 3]^T) = [1\ 0]^T$.
$SubT \leftarrow \{PN_1\}$.
compute the OT set
$(IW \cdot IL) + AD = [2\ 3]^T$
$\tilde{\Theta}_{[2\ 3]^T}(3) = [0\ 1]^T$
$OT \leftarrow \{PN_2\}$
there is a $SubT$.
$SubT_i \leftarrow PN_1$
$SubT_i$ (PN_1) Feed-Back phase:
$IL = IW_{PN_1}$ = [01 01]
no lateral inhibition takes place.
Merge PN_{new} and PN_1:
$IW_{PN_{new}} \leftarrow$ [10 01] $\vee$ [01 01]
$IW_{PN_{new}} \leftarrow$ [11 01]
$AD = 2 - 2 + 1 = 1$.
compute the $\widehat{OT}$ set
$(IW \cdot IL) + AD = [2\ 3]^T$
$\tilde{\Theta}_2(3) = [0\ 1]^T$
$\widehat{OT} \leftarrow \{PN_2\}$
there is an high $\widehat{OT}$
delete PN_1, remove IW_{PN_1}
As Merging occurred, repeat feedback of PN_{new}
Second Feed-Back Phase:
At this moment $IW = \left| \begin{array}{lcccc} IW_{PN_2}: & 1 & 1 & 0 & 1 \end{array} \right|$
$IL = IW_{PN_{new}}$ = [11 01]
lateral inhibition takes place.
IL^1 = [11 01]; IL^2 = [00 01]
$AD = [AD^1\ ,\ AD^2] = [2 - 3 + 1\ ,\ 2 - 1 + 1] = [0\ ,\ 2]$
compute the $SubT$ set
$(IW \cdot IL) + AD = 3$
$\Theta_2(3) = 0$.
$SubT \leftarrow \emptyset$
compute the OT set
$(IW \cdot IL) + AD = 3$
$\tilde{\Theta}_2(3) = 1$
$OT \leftarrow \{PN_2\}$
there is no $SubT$.
there is an high OT but it is PN_{new}.
recall that $PN_2 \equiv PN_{new}$
no PN needs to be deleted.
so keep PN_{new} as PN_2.
no further merging occurred: training is finished.

The network contains a single PN
$IW = \left| \begin{array}{lcccc} IW_{PN_2}: & 1 & 1 & 0 & 1 \end{array} \right|$

Notice that, at this moment, the network contain a single node which covers both patterns presented. In this case this is equivalent to the minimal cover of H, consisting of the single implicant '$\neg b$'.

III. Simulations

In order to test its compression rate, the Implicant Network has been trained on random functions. In figures 2–4, it is possible to see how many nodes are required in average to store a function (Y axis) depending on the dimension of the positive set (X axis). For each positive set size, results in figures 2 and 3 are the average over 50 randomly generated functions, while in figure 4 over 20 randomly generated functions. The positive set of each function is presented in random order, several times.

The order in which minterms are presented affects the compression ratio achieved by the network (number of PN per minterm). That is why, the compression of the positive set steadily increases as it is presented over again, see also figure 5. The compression achieved by an optimal minimum cover algorithm is plotted in a dotted line.

The average compression progression for a full positive input set of 10 variables (2^{10} minterms) is given in figure 6. The input set is presented in random order each of the 100 passes. The average is computed over 20 runs. The number of required PNs monotonically decreases to reach the optimal solution. The optimal SOP formula requires a single predicate.

IV. Conclusions

The Implicant Network is an online algorithm capable of compressing and remembering patterns with linear time and space requirements.

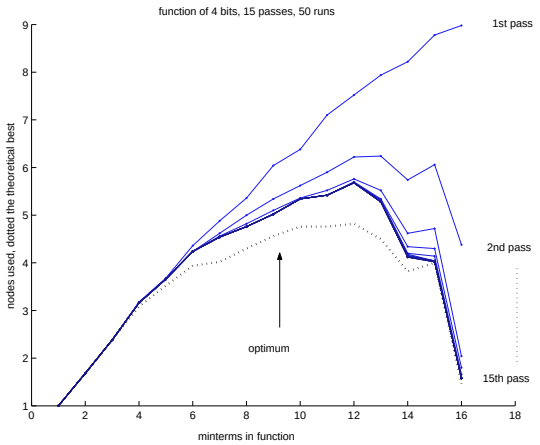


Fig. 2. Average number of nodes required for 50 random functions of 4 bits

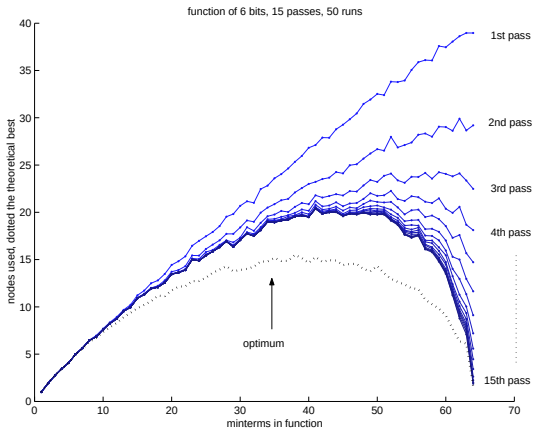


Fig. 3. Average number of nodes required for 50 random functions of 6 bits

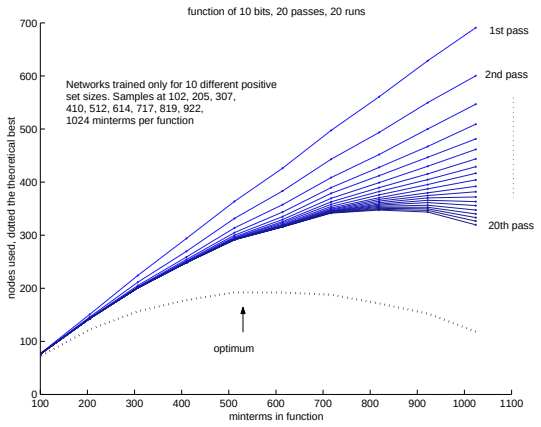


Fig. 4. Average number of nodes required for 20 random functions of 10 bits

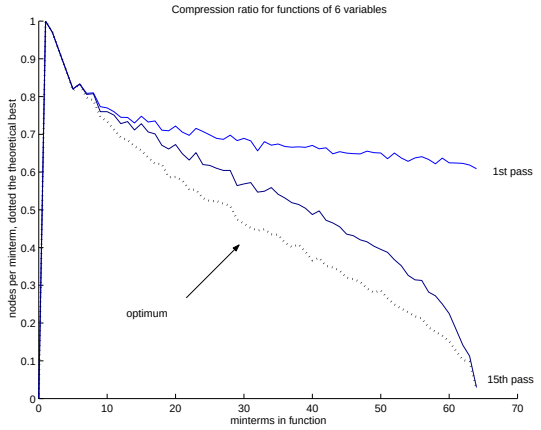


Fig. 5. Compression increases as the positive set is shown over again

The compression happens in terms of number of nodes in the predicate layer of the network, and it is totally non-destructive. Also, the compression algorithm requires only information already stored in the network without decompressing it. This is a fundamental feature of the algorithm, as information decompression would be intractable in the general case.

An n-input, single-output boolean function can have up to $3^n/n$ prime implicants and 2^n minterms (patterns, [9]).

The compression is possible because of a feedback phase, during which, an active predicate neuron is able to find similar nodes to merge with, adjoint to finding those nodes that are already covering its information.

This use of the feedback phase is a major contribution of this paper.

The feedback phase is similar to the feed forward activation. It requires $O(n \cdot \kappa)$ multiplications in a serial implementation, κ being the size of the predicate layer. In a parallel implementation, the activation of each of the κ nodes could be computed in parallel, in linear time.

The compression ratio improves monotonically as the input set is shown over and over again.

This is a nice feature for autonomous agents, as representation of the input set can merely occur as the agent wonders in its environment. The improvement takes place just by being up and running, so it is achieved for free.

The compression ratio seems to approach the theoretical optimum for SOP formulae, which is computable solving the minimal cover problem and is proven to be NP-Hard.

Still, it is often the case that convergence stops on a local optimum. This happens most often for highest complexity input sets, whose number of minterms is $\approx 2^n/2$. Among those sets there is the parity function set, which is not compressible by any mean with a SOP formula.

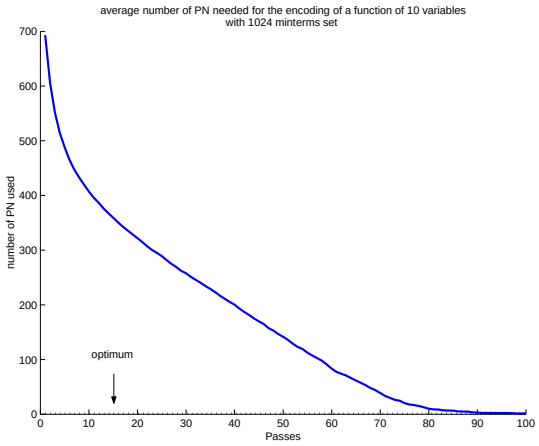


Fig. 6. Average number of nodes required for a full positive input set of 10 bits

The number of passes required for the system to converge to the local optima appears to be a not-trivial function of the input space dimension, of the positive set size, of the function represented by the positive set and of the order in which patterns are presented.

When the whole positive set is presented in random order at each pass, convergence is reached in $\approx n^3/5$ passes in the worst case. The worst case is associated to high cardinality positive sets, where all minterms but a few map to 1.

V. Further work

At the moment, we are investigating the possibility to extend the algorithm to allow a ρ node representation, with ρ fixed a priori and bigger than 1. In this model, the activation of the output layer would require ρ high nodes in the hidden layer.

In fact, the Implicant Network uses a localistic representation $(\rho = 1)$ in the hidden predicate layer.

To be able to encode arbitrary functions with m-outputs, non-localistic representations seem to be the only efficient way.

On the other side achieving a storage capacity which grows exponentially with hidden layer size requires distributed representations. This is unfortunately a very hard task in one-shot learning algorithms.

References

[1] T. Carew, *Behavioral Neurobiology: The Cellular Organization of Natural Behavior*. Sinauer Assoc, 2000.

[2] E. Tolman, "Cognitive maps in rats and men," *The Psychological Review*, vol. 55, no. 4, pp. 189–208, 1948.

[3] B. Kosko, "Bidirectional associative memories," *IEEE Transactions on Systems, Man and Cybernetics*, vol. 18, no. 1, pp. 49–60, 1988.

[4] K. G. Haines and R. Hecht-Nielson, "A bam with increased information storage capacity," *Proceedings of the IEEE 1988 International Conference on Neural Networks*, vol. 1, pp. 181–190, 1988.

[5] M. H. Hassoun and P. B. Watta, "The hamming associative memory and its relation to the exponential capacity dam," *IEEE International Conference on Neural Networks, ICNN*, 1996.

[6] K. Demura, "One shot learning and adaptive similarity," *PhD Thesis*, 1996. [Online]. Available: http://www.his.kanazawa-it.ac.jp/ demura/research/papers/demura_thesis.pdf

[7] Y. Wu and S. N. Batalama, "Improved one-shot learning for feedforward associative memories with application to composite pattern association," *IEEE Transactions on Systems, Man and Cybernetics*, vol. 31, no. 1, pp. 119–125, 2001.

[8] J. Austin and K. Lees, "A search engine based on neural correlation matrix memories," *Neurocomputing*, vol. 35, no. 1, pp. 55–72, november 2000.

[9] G. D. Micheli, *Synthesis and Optimization of Digital Circuits*. McGraw-Hill, 1994.

C Culture and the Baldwin Effect

Author: Diego Federici. Published in proceedings of ECAL 2003 European Conference of Artificial Life.

Abstract:

It is believed that the second phase of the Baldwin effect is basically governed by the cost of learning. In this paper we argue that when learning takes place the fitness landscape undergoes a modification that might block the Baldwin effect even if the cost of learning is high. The argument is that learning strategies will bias the evolutionary process toward individuals that genetically acquire better compared to individuals that genetically behave better. Once this process starts the probability of experiencing the Baldwin effect decreases dramatically, whatever the learning cost. A simulation with evolving learning individuals capable of communication is set to show this effect. The set of acquired behaviors (culture) competes with the instinctive one (genes) giving rise to a co-evolutionary effect.

Objective:

Produce and analyse a minimal model of cultural evolution.

Conclusions:

Horizontal information exchange produces a significant alteration of the evolutionary dynamics. The discovery of social strategies accelerates the evolution of culture, which in turns increases the advantage of social strategies. As a result, it appears that the adopted meme implementation provide a minimal, yet sufficient, model of cultural evolution.

Culture and the Baldwin Effect

Diego Federici[1]

Norwegian University of Science and Technology
Department of computer and information science
N-7491 Trondheim, Norway
federici@idi.ntnu.no,
http://www.idi.ntnu.no/~federici

Abstract. It is believed that the second phase of the Baldwin effect is basically governed by the cost of learning. In this paper we argue that when learning takes place the fitness landscape undergoes a modification that might block the Baldwin effect even if the cost of learning is high. The argument is that learning strategies will bias the evolutionary process towards individuals that genetically acquire better compared to individuals that genetically behave better. Once this process starts the probability of experiencing the Baldwin effect decreases dramatically, whatever the learning cost. A simulation with evolving learning individuals capable of communication is set to show this effect. The set of acquired behaviors (culture) competes with the instinctive one (genes) giving rise to a co-evolutionary effect.

1 Introduction

1.1 The Baldwin effect

In the context of the debate between Darwinism and Lamarckism, James Mark Baldwin (1896) proposed that phenotypic plasticity might be regarded as "a new factor in evolution" [1]. Phenotypic plasticity allowing adaptation, would smooth the fitness landscape increasing the efficiency of the evolutionary process [2, 3]. However, phenotypic plasticity has inherent costs associated with the training phase in terms of energy, time and eventual mistakes. For these reasons, in a second phase, evolution may find a way to achieve the same successful behaviors without plasticity.

Thus the Baldwin effect has two phases. During the first phase, adapting individuals can, in same cases, acquire behaviors that help them achieving higher fitness scores. But because of the costs of adaptation, there is an evolutionary advantage towards the discovery of equivalent instinctive behaviors. Thus in this second phase, a behavior that was once learned may eventually become instinctive (see also below and [1–5]) In computer science, the phenotypic plasticity is analog to a local search strategy. The evolutionary process and the local search may be used in combination, often achieving higher efficiency than either of the methods alone [5, 6].

There are three basic requirements for the second phase to take place. First there must be a cost for the local search. In this way, the evolutionary process will have a reason (in terms of inclusive fitness) for the genetic assimilation to take place. This also means that in some settings, i.e. in a fast changing environment, genetic assimilation will never take place. With those setting, plasticity would be *the* optimal strategy. We will refer to this characteristic by *Assimilation Advantage*.

Also, genetic assimilation requires for the optimal strategy, acquired first through local search, to be expressible by the genotype. This might be impossible under some genotype-phenotype mapping strategies in which, the phenotype plasticity is required as part of the developmental process [1]. We will refer to this characteristic as *Genotypic Expressibility*.

In addition, the probability of the assimilation depends on the distance between the genotype using plasticity and the one not using it. The distance would be measured using the metric imposed by the genetic operators. A small distance is possible if there is a strong neighborhood correlation in the transformation from genotypic to phenotypic space. Where the distance is too high, the probability of genetic assimilation could be so little to be considered actually impossible. We will refer to this as *Genotypic-Phenotypic Correlation*

1.2 How learning effects evolution

In a now famous paper, Hinton and Nowlan [6] proved that with the help a local search mechanism it is possible to speed up evolution in a hard fitness landscape.

In the Hinton and Nowlan example, the adaptive solution is ideally placed in the middle between the lowest fitness solutions and the single high one, hence smoothing the fitness landscape. Adaptation in this case is a step towards the discovery of the best non-adaptive solution. The same considerations apply to other examples such as [2, 8] among others.

In these cases adaptation success is not affected by any genetically coded learning strategy. We argue that the search for good learning strategies might distract the evolutionary process from the discovery of fit non-adaptive behaviors. In other words, co-evolution of learning and the non-learning strategies modifies the fitness landscape. The quality of these modifications is an other factor that governs the second phase Baldwin effect.

2 Learning, culture and fitness

Learning can be seen as the process of acquiring behaviors. The difficulty and time lost acquiring behaviors constitute a cost of learning. We have to introduce a clear distinction between instinctive and acquired behaviors. Instinctive are those behaviors that emerge steadily and directly from the genotype, while acquired ones are those that emerge through the interaction with the environment.

[1] Like in the development of the retina [7]

If we consider individuals belonging to a population sharing the same genotype, their individual fitness can be considered the sum of shared population fitness (PopFit), fitness change due to local environmental characteristics (LEFit) and fitness change due to individual specific behavior (IFit): $Fitness = PFit + LEFit + IFit$.

The LEFit can be considered as noise and could be absent in ideal experimental settings (all individuals having the exact same initial conditions or long fitness tests). Fitness deriving from acquired behaviors (IFit) constitutes the value of the learning process and incorporates the cost of learning.

When PFit $\gg$ IFit, the advantage for plasticity is negligible. Otherwise acquired behaviors may provide an advantage to genotypically similar individuals (see figure 1). In this case, there is strong evolutionary pressure towards the discovery of better acquisition mechanisms.

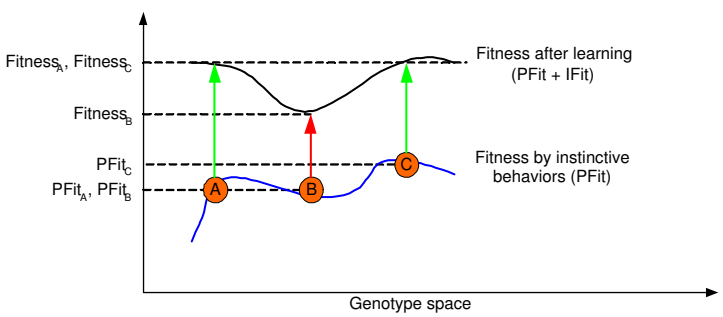


Fig. 1. Two similar individuals (A and B), share similar genotype and PFit values. By acquiring different behaviors they achieve different fitness scores. On the other side, two genotypically different individuals (A and C) reach the same fitness values because the same acquired behavior shadows the instinctive ones.

2.1 Memes

As genes form the transmission medium of biological systems, memes [9] do for acquired behaviors. Memes will be considered behavioral information blocks[2]. Basically memes are those things that "leap from brain to brain" [9] carrying a behavioral content. To strike a comparison to human society, we will call the set of transmittable behaviors *Culture*.

3 Simulation details

We set up a population of learning individuals. Each individual/agent is equipped with a single layer neural network (NN) subject to an evolutionary process and a

[2] Memes usually have a wider definition, but considering only the behavioral ones, the discussion is simplified

classifier-like system (memes), see Figure 3. Agents perceive resources and other bots from all tiles in a hamming distance of 2 (see figure 2), this constitutes the input vector.

The NN produces an output vector with the expected reinforcement for each of the possible actions: don't move, go north, west, south and east.

Memes remind the agent the reinforcement experienced in the past. They are constituted by an input pattern P, an action a and an expected reinforcement R. If the pattern P matches the present input vector, then the meme replaces the output of the genetically evolved NN with R for the given action a. Basically the meme can recognize a particular sensory context (P) and reminds the agent that in the past he had performed a certain action (a) and the action yielded a given reinforcement (R).

The four expected reinforcements, generated by the NN and eventually modified by the memes, are used to stochastically select the action performed by the agent.

Agents score fitness by collecting resources spread at random in a toroidal map. Each resource type gives a fixed amount of reinforcement. If an agent does not move or collides with another agent it receives a small negative reinforcement[3]. Fitness is the sum of all reinforcements received over a fixed number of iterations. Agents undergo a steady state selection with a replacement fraction of 25%. Surviving individuals keep their memes. The offspring is placed in the proximity of a parent.

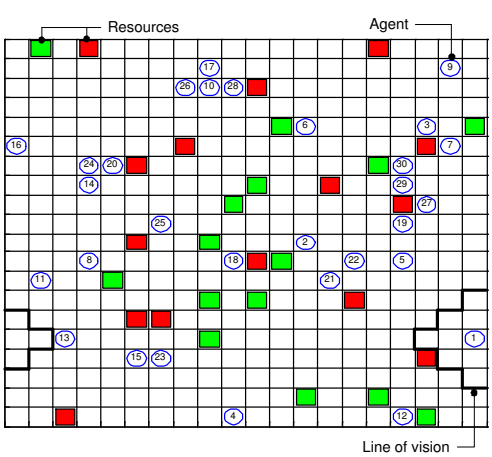


Fig. 2. Simulated Environment. The vision range of agent 1 is printed in a thick line. Two different types of resources are present. The resources represented by a darker color give a fixed negative reinforcement and fitness value, while the others give a fixed positive value. Resource types never change value and when consumed are regenerated on a random tile. Newly generated agents are placed in proximity of a parent.

[3] These penalties were added to speed up the evolutionary process

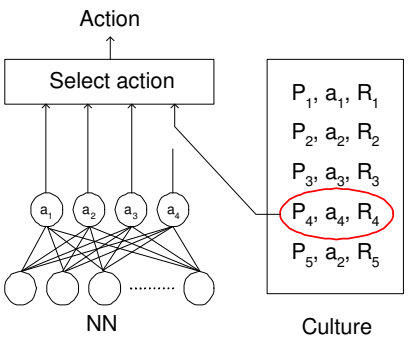


Fig. 3. Agent controller. The genetically evolved NN and the acquired culture are activated in parallel. When a pattern P matches the current input, the corresponding meme is activated (encircled in the figure). Its expected reinforcement R_4 replaces the NN output for action a_4. The performed action is selected stochastically giving a higher probability to higher expected reinforcements.

3.1 Cultural evolution

An agent culture is built by a certain number of memes (20 at maximum in these simulations). Memes can be acquired in two different ways.

First by transmission, whenever two individuals are next to each other. Every iteration a fixed number of memes can be transmitted. These memes are probabilistically selected from the transmitter culture according to their estimated value. The number of memes transmitted when two agents are in contact (communication speed, CS) is varied from 0 (no transmission) to 20 (all the meme pool is transmitted in a single iteration). As the CS increases, the agents can acquire fit behaviors earlier during the fitness evaluation, hence the *Assimilation Advantage* is reduced.

The second way is through operant conditioning.

Operant conditioning is a learning mechanism that has been observed in a variety of animals. When an animal experiences a reinforcement, its brain tries to explain what caused the reward. The effect is that the behaviors that are thought to be responsible of the reinforcement are rewarded. Learning appears to build a relationship between behavior and reinforcement based on two general assumptions: the behavior that steadily is followed by reinforcement is held responsible for it, behavior and reward must fall into a certain time window. These assumptions have strong biological and psychological support [10–12].

Whenever an agent experiences an unexpected reinforcement a meme is generated from this situation. The value of the meme changes as it is used, increasing when it helps predict the expected reinforcement. This inhibition/enhancement is an explicit measure of the memes fitness, and is used to drive the memetic evolutionary process. Unfit memes can be explicitly identified and dropped, fit ones will proliferate through transmission and new variants will eventually be generated.

Memes variants are generated by merging, a stochastic generalization mechanism. Merging is the memetic equivalent of crossover and mutation in genes. It can occur if two memes code the same action and expected reinforcement. In this case, the merging probability is proportional to the hamming distance

between the memes input matching patterns. Those parts that are different in the two input matching patterns are replaced by *don't care* symbols.

Merging is a weak simplification of a boolean function:
given $(P_1 \wedge a \mapsto R)$ and $(P_2 \wedge a \mapsto R)$ then with probability $\sim d_H(P_1, P_2)$ replace them with $((P_1 \circledast P_2) \wedge a \mapsto R)$; where $P_i \in$ pattern, $a \in$ action, $R \in$ reinforcement, d_H is the Hamming distance, and $\circledast$ is a bitwise operator $\circledast(b_i, b_j) = \{\ b_i$ if $b_i = b_j$, *don't care* if $b_i \neq b_j\ \}$.

If it does not merge, a meme can be added only if the meme pool size does not exceed the maximum. If the maximum is exceeded a meme is dropped, the less general being dropped with higher probability. Because merging of memes can sometimes produce unfit memes, if the expected reward does not match the one experienced, the meme responsible for the error is removed.

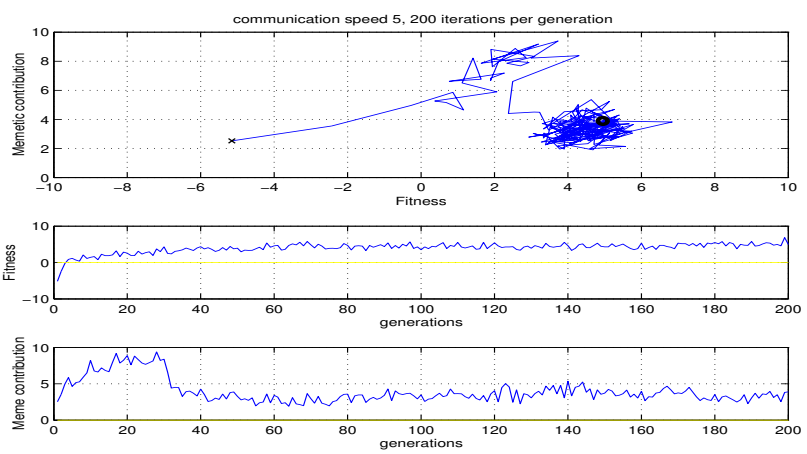


Fig. 4. run showing the Baldwin effect, convergence to the $\mathbb{MG}$ attractor. The memetic contribution to the fitness is at first high. As it decreases, the memetic behavior is partially assimilated in the genes.

4 Results

Agents can transmit some of their memes whenever their are next to each other. The number of memes transmitted when two agents are in contact (communication speed, CS) is varied from 0 (no transmission) to 20 (all the meme pool is transmitted in a single iteration).

As the CS increases, the agents can acquire fit behaviors earlier during the fitness evaluation, hence the *Assimilation Advantage* is reduced.

It is possible to evaluate the amount of fitness generated through acquired behaviors by stopping the simulation every generation and recording how much

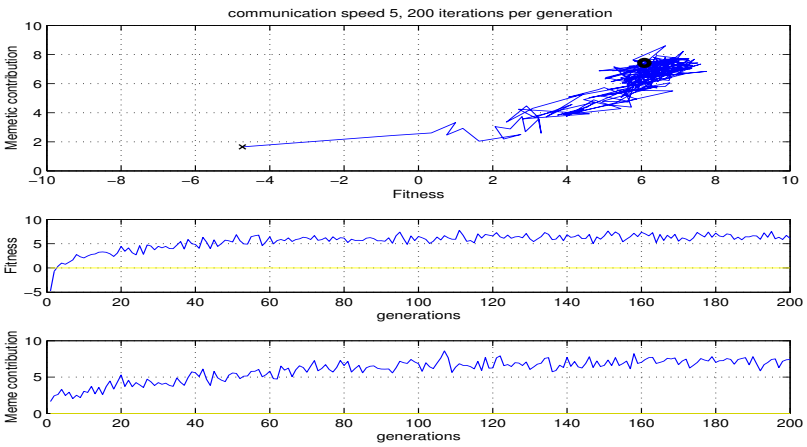


Fig. 5. run not showing the Baldwin effect, convergence to the $\mathbb{M}$ attractor. Genetic assimilation does not take place.

fitness the agents score with and without the help of memes. It is so possible to plot how much the memes contribute to the total fitness score (memetic contribution). Figures 4 and 5 show three different plots. The first one is a state diagram, memetic contribution vs. fitness, showing the trajectory that a population undergoes during evolution. The second one displays the amount of average fitness scored by the population, and the third the quantity of memetic contribution.

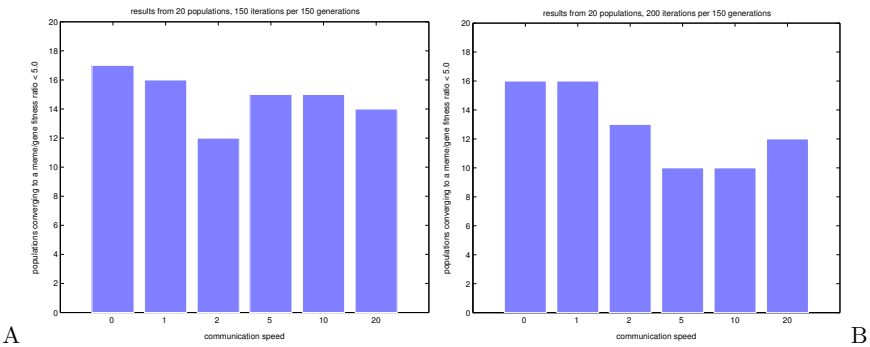
The state diagram is particularly useful because it shows the attractors of the evolutionary process. Figure 4 shows the attractor for a typical population that went through the second phase of the Baldwin effect, figure 5 one that doesn't show it.

In almost every setting two attractors, such as those in figure 4 and 5 are present. The two attractors show the convergence basin of different strategies. The first relies both on memes and genes ($\mathbb{MG}$) with the instinctive behaviors capable of scoring some fitness. The second relies on memes only ($\mathbb{M}$).

It is then understandable that while the dynamic towards $\mathbb{MG}$ is associated to the second phase of the Baldwin effect, the path to $\mathbb{M}$ is not.

Figure 6 shows with which frequency a population falls in the $\mathbb{MG}$ attractor.

One would expect to experience a decreasing number of populations falling in the $\mathbb{MG}$ attractor as CS is being increased. In fact, increasing CS reduces the cost of learning and the advantage for genetic assimilation. Instead the frequency decreases to a minimum and then increases again. This proves that the Baldwin effect is influenced by other factors apart the cost of learning.



A B

Fig. 6. Populations falling into the $\mathbb{MG}$ attractor with each CS setting

4.1 Different strategies

An agent can acquire fit behaviors in three possible ways: evolving the proper NN, building memes by operant conditioning, and acquiring memes from other agents. CS can affect only the latter.

Being born next to its parent without any meme, an agent is faced with a dilemma. It can either try:

1. first receive as many memes as possible from other agents, and then exploit them to score fitness
2. start scoring fitness immediately and acquire memes by itself

Either choice requires a genetically coded strategy, a social one in the first case, and an asocial one in the latter. The social strategy, scoring fitness mainly by memes, is the one that converges to $\mathbb{M}$. $\mathbb{MG}$ is instead the attractor of the asocial strategy.

CS does not only change the cost of learning, modulating the acquisition speed. It also changes the nature of the two strategies $\mathbb{M}$ and $\mathbb{MG}$.

As CS decreases the social strategy becomes more difficult. In other words it requires a more committing strategy and a more specialized genotype. In fact, it must strongly avoid any asocial behavior, otherwise it will risk to interrupt the acquisition phase. On the other side when CS is maximal, the parent's culture is acquired in a single shot. The social strategy is achieved simply by being born. In this case $\mathbb{M}$ and $\mathbb{MG}$ will be maximally overlapping because an asocial individual will receive the same amount of culture as a social one.

The difference between the two strategies determines the probability of moving from one to the other. The higher the difference the lower the probability.

A second aspect is that the quality of culture (the amount of fitness it can guarantee) is not fixed. As genes evolve every generation, memes do every iteration. If CS is zero, agents cannot share their memes and the culture quality improves in a given generation but not across generations. If CS is high, even if an agent dies some of its memes might survive and continue evolving on other hosts.

The fitness advantage of a strategy or the other, is subjected by the actual level of cultural evolution. When many agents do not socialize, culture improves slowly and the social strategy is less attractive. With many social agents, cultural evolution is faster and offers a greater prize for adaptivity.

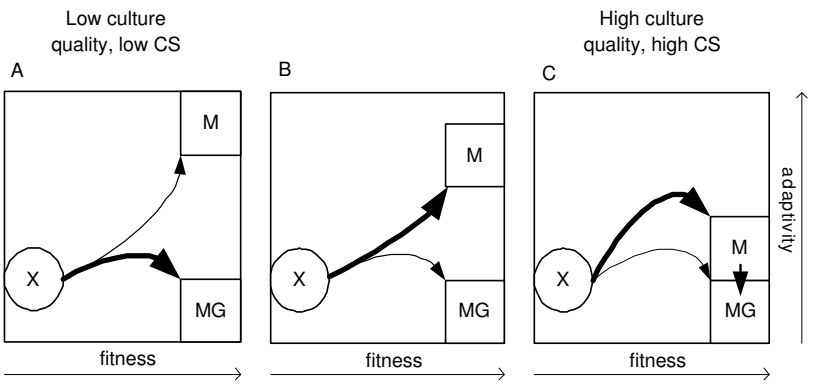


Fig. 7. effects of CS to evolvability

As the cultural quality evolves or fails to evolve, the fitness landscape changes. If cultural evolution proceeds steadily, the $\mathbb{M}$ strategies will be able to dig into their attractor increasing their stability. Still, if $\mathbb{MG}$ lies too close to the $\mathbb{M}$ attractor, this is insufficient to prevent the second phase of the Baldwin effect. Figure 7 summarize these concepts.

Figure 7A With low CS, the quality of culture is not enough so that the asocial strategy $\mathbb{MG}$ is more convenient (the thicker line indicates a higher transition probability).

Figure 7B CS increases and the $\mathbb{M}$ attractor becomes stronger. As the attractors are far away in genotypic space, the probability to move from $\mathbb{M}$ to $\mathbb{MG}$ is very low. The frequency of the second phase of the Baldwin effect is minimal.

Figure 7C CS continues to increase, the social and asocial strategy are very similar and passage from $\mathbb{M}$ to $\mathbb{MG}$ is more probable.

5 Conclusions

We have provided an example in which genetic assimilation cannot be explained by the Baldwin effect alone.

It is suggested that the Baldwin effect considers cases in which the genotype cannot modify directly the adaptation mechanism. Under these conditions, the genetic search can evolve only the non-adaptive part of the phenotype, and only

that can provide a continuous fitness improvement. Examples for this case can be found in [6, 2, 8, 13].

We argue that the adaptive behavior must be somehow expressed in the genotype, so that evolution affects also the adaptation mechanism and its quality.

Evolution can than proceed in at least two directions, one towards the discovery of better adaptive strategies (in this case the social behavior), the second towards the discovery of fitter instinctive behaviors (the asocial behavior).

The state of the evolution of adaptivity affects both the cost of learning and the correlation between genotype and phenotype. This can cause that both the cost of learning and the correlation decrease, so that the probability of observing the Baldwin effect is not a monotonic function.

Even in a static environment[4] as the one provided in this paper, the fitness landscape will undergo a dynamic. Under these circumstances, genetic assimilation is ruled more by the quality of the fitness landscape dynamic than by the assimilation advantage.

The simulations presented in this paper are very computational expensive and have been run on ClustIS[5] cluster.

References

1. J.M. Baldwin: A new factor in evolution. American Naturalist **30** 441–451 (1896)
2. K. Downing: Reinforced Genetic Programming. Genetic Programming & Evolvable Machines. Kluwer Publishers 259–288 (2001)
3. P. Turney: Myths and legends of the Baldwin Effect. Proceedings of the ICML-96 (13th International Conference on Machine Learning) 135–142 (1996)
4. P. Turney, D. Whitley, R. Anderson: Evolution, learning and instict: 100 years of the Baldwin effect. Evolutionary Computation **4:3** 206–212 (1996)
5. R.K. Belew, M. Mitchell: Adaptive individuals in evolving populations: models and algorithms. addison-wesley (1996)
6. G.E. Hinton, S.J. Nowlan: How learning can guide evolution. Complex Systems **1** 495–502 (1987)
7. L.M. Chalupa, B. L. Finlay: Development and organization of the retina : from molecules to function. Plenum Press, New York (1998)
8. R. French, A. Messinger: Genes, phenes and the Baldwin effect. In Proceedings of Artificial Life IV 277–282 (1994)
9. R. Dawkins: The selfish gene. Oxford university press (1976)
10. E.R. Kandel, J.S. Schwartz, T.M. Jessell: Principles of neural science. Elsevier Science Publishing Co., New York (2000) 4th edition
11. H. Markram, J. L'bke, M. Frotscher, B. Sakmann: Regulation of Synaptic Efficacy by Coincidence of Post-synaptic APs and EPSPs. Science, **275** 213–215 (1997)
12. T.J. Carew Behavioral Neurobiology: The Cellular Organization of Natural Behavior Sinauer Associates (2000)
13. T. Arita, R. Suzuki Interactions between learning and evolution: The outstanding strategy generated by the Baldwin effect. In Proceedings of Artificial Life VII, 196–205 (2000)

[4] where the best strategy does not require adaptation
[5] information can be obtained from http://ClustIS.idi.ntnu.no/

D Combining genes and memes to speed up evolution

Author: Diego Federici. Published in proceedings of CEC 2003 Congress on Evolutionary Computation.

Abstract:

It is recognized that the combination of genetic and local search can have strong synergistic effects. In same cases though, the local search mechanism can be too aggressive, mislead the evolutionary search and produce premature convergence. We set up a population of evolving agents also capable of learning by operant conditioning and communicating acquired behaviors (memes). The diffusion and discovery of memes gives rise to a second process of evolution atop of the genetic one. Memes are shown to have both guiding and hiding effects on baldwinian and lamarckian evolution. In contraposition to previous models, simulations show that back-coding of acquired behaviors is highly beneficial only at the beginning of the evolutionary search. This result arises because of the different nature of the guiding provided by memes and the hiding effect that they generate. To minimize the negative influence of the hiding effect but still benefit from the memetic guidance, we decrease the maximum number of memes that an agent can acquire as evolution proceeds. Agents can then develop the optimal harvesting strategy in incremental steps with a great performance advantage.

Objective:

Exploit cultural evolution and grounded directed innovation processes to increase the efficiency of the evolutionary search.

Conclusions:

Lamarckian back-coding of culturally acquired traits is beneficial towards the speed up of evolutionary search. Still, innovation becomes more directed and limits the exploration of the search space. Best performance is obtained when back-coding is gradually inhibited.

Combining genes and memes to speed up evolution

Diego Federici
Norwegian University of Science and Technology
Department of computer and information science
N-7491 Trondheim, Norway
federici@idi.ntnu.no

Abstract- It is recognized that the combination of genetic and local search can have strong synergistic effects. In same cases though, the local search mechanism can be too aggressive, mislead the evolutionary search and produce premature convergence.

We set up a population of evolving agents also capable of learning by operant conditioning and communicating acquired behaviors (memes). The diffusion and discovery of memes gives rise to a second process of evolution atop of the genetic one. Memes are shown to have both guiding and hiding effects on baldwinian and lamarckian evolution.

In contraposition to previous models, simulations show that back-coding of acquired behaviors is highly beneficial only at the beginning of the evolutionary search. This result arises because of the different nature of the guiding provided by memes and the hiding effect that they generate.

To minimize the negative influence of the hiding effect but still benefit from the memetic guidance, we decrease the maximum number of memes that an agent can acquire as evolution proceeds.

Agents can then develop the optimal harvesting strategy in incremental steps with a great performance advantage.

1 Introduction

Even though Lamarck's theory of evolution [1] has been disproved there has been quite an interest in its application in artificial evolution.

As local search strategies can be more directed than genetic, back-coding of acquired characteristics can operate as a smart mutation operator yielding faster convergence to optima.

Since local search focuses on the most promising parts of the search space, it can increase the evolutionary speed in two ways: at the beginning by overlooking low fitness zones and at the end by climbing local optima. Of course, this more aggressive strategy can often produce premature convergence [3, 4].

The type of local search modulates the pro and cons of back-coding. This can be summarized by the *guide or hide* dichotomy [9, 10]). If on one side, learning can guide the evolutionary process by smoothing the fitness landscape (Baldwin effect [2, 5]) or through back-coding (lamarckianism [1, 3, 4]), on the other it can mask the selection pressure for certain characteristics hiding genetic differences and slowing down the entire process (hiding effect [9, 10]).

In this paper we adopt a learning mechanism based on acquisition through operant conditioning and communication. Individuals have a genetically encoded neurocontroller that outputs the expected reinforcements for the different possible actions. Individuals that experience an unexpected reinforcement build a meme (a reminder) that will allow them to avoid the same error in the future. At the same time, when two individuals fall into the communication range, they can exchange memes.

As memes are acquired, exchanged and dropped during a fitness evaluation, they give rise to a second evolutionary process atop of the genetic one. We will refer to this process as cultural evolution.

Previous work has already introduced models of cultural and memetic evolution. In [11, 12] culture is a population shared memory that acts as a global blackboard that individuals can read and write. The model of social exchange used in [13] is implemented by a crossover operator that combines the candidate with an individual of high fitness.

In this model memes are stand alone behavioral entities that reside on a single host and can be acquired and transmitted. If an agent perceives two memes to be similar enough, it will be merge them, generating a more general variant that can eventually spread in the population. The set of memes available to an individual specifies its culture and modifies its instinctive behavior. Since behavior determines the fitness scored by individuals, there is an evolutionary advantage in the development of fit cultures and therefore fit memes.

In this framework, the lamarckian back-coding of an individual's culture is shown to have a positive guiding effect in a first phase of the evolutionary search but is also shown to mask refinements of instinctive behaviors that do not yield immediate reinforcements. Because of the way that they are built, memes can encode only sources of immediate reinforcement while optimal control policies should also consider long term effects. Acquired behaviors have precedence over instinctive ones resulting in a censorial action of culture.

To minimize the negative effects of this cultural masking, we show that it is possible to decrease the number of memes that an agent can possess as evolution proceeds.

Although not applicable in every context, a good feature of this hybrid evolutionary system, is that it does not require additional fitness evaluations since memes are acquired online during the single fitness test.

2 Background

Lamarck's theory of evolution states that adapted traits are inheritable [1]. The discovery of germ cells disproved Lamarck's theory, still Baldwin suggested that there could have been a "new factor" that might operate in a similar

way.

The Baldwin effect [2] states that phenotypic plasticity would allow adaptation to partially successful mutations, smoothing the fitness landscape and increasing the efficency of the evolutionary process.

However, phenotypic plasticity has inherent costs associated with the training phase in terms of energy, time and eventual mistakes. For these reasons, in a second phase, evolution may find a way to achieve the same successful behaviors avoiding plasticity. Thus a behavior that was once learned may eventually become instinctive.

In computer science, the phenotypic plasticity is analog to a local search strategy. The evolutionary process and the local search may be used in combination, often achieving higher efficency than either of the methods alone [6, 4, 7].

Hinton and Nowlan [7] were the first to prove the benefits of the Baldwin effect in a computer simulation. In a *needle in the haystack* function optimization problem, they showed that a local search mechanism could speed up evolution. The difficulty of the fitness landscape is dampened by the local search strategy, but since each step of the local search requires an additional fitness evaluation, the speed up of the evolutionary search is paid by the increased time required for each generation.

A different picture appears when considering the evolution of systems that require long fitness tests, for example controllers for situated agents. To get a good evaluation of the agent's fitness, it is often necessary to run several hundreds activations of its controller, see [16, 14, 15] among others.

In this context it is possible to add learning without requiring additional fitness evaluations. For example we can suppress behaviors that lead the robot to immediate negative reinforcements, such as when it hits an obstacle during the fitness test.

3 The model

We set up a population of 30 learning individuals that move in a 20×20 toroidal world. The world contains 30 of each of the two different types of resources: food and poison (see figure 1). When an agent visits a tile containing a resource it consumes it, receiving an immediate reinforcement. When consumed, the resource is removed and regenerated at random in the world.

The fitness is defined as the sum of the accumulated reinforcements over 150 simulation steps.

Each simulation step, an agent's reinforcement is computed as the sum of any of the following:

+0.8	if	visiting a tile containing food
-0.8	if	visiting a tile containing poison
-0.1	if	colliding with another agent
-0.1	if	the agent did not move

Each individual/agent is equipped with two different controllers. The first, a single layer neural network (NN) with hyperbolic tangent transfer function, is subject to an evolutionary process. The second is a classifier-like system (memes) and models the individual culture.

Agents perceive resources and other bots from all the 13 tiles within a hamming distance of 2 (see figure 1), this constitutes the input vector for both controllers.

For each of the 13 tiles, the 39 element boolean input vector contains a triplet T_{1-3} such as

T_1	1 if the tile contains food,	0 otherwise
T_2	1 if the tile contains poison,	0 otherwise
T_3	1 if the tile contains an agent,	0 otherwise

The neuro-controller also receives an additional input, always set, to provide the network bias.

The action performed by an agent is computed as follows:

- The NN produces 5 outputs. Each output is interpreted as the anticipated reinforcement (R_A) for each of the possible actions: don't move, go north, west, south and east. This constitutes the agent's instinctive response.
- The memes produce a set of reminders. Each reminder contains an action a and an experienced reinforcement R_E. These tell the agent that it seems to recognize the current input and that if action a is performed it will yield a reinforcement equal to R_E. This set constitutes the acquired responses.
- Acquired responses R_E replace the corresponding instinctive ones R_A (see figure 2), this constitutes the vector of expected reinforcements. The action with the highest expected reinforcement is selected with .7 probability. Otherwise a random action is selected.

3.1 genetic evolution

Given that the NN receives 40 inputs (a triplet for each of the 13 tiles within an agent's vision range, plus a bias), and that it produces 5 outputs (R_A), the weight matrix $\in \Re^{\{(39+1)\times 5\}}$.

The neuro-controller genotype is a linear gray-coded representation of its weight matrix.

Each generation, the best scoring 25% of the population survives and reproduces. Three quarters of the offspring are generated by the crossover of two randomly selected reproducing individuals; the remaining are generated by mutation of a single parent.

Mutation modifies each weight with a .2 probability by adding to it Gaussian noise with .25 variance.

As each NN can be considered composed of 5 independent sub-nets, one for each output, crossover produces two new individuals shuffling the parents sub-nets.

3.2 memetic evolution

Memes ideally remind the agent of the reinforcement experienced in the past[1]. They consist of an input pattern P, an action a, a value V and an experienced reinforcement R_E. If the pattern P matches the present input vector, then

[1] a meme could also have been acquired by communicating with other agents

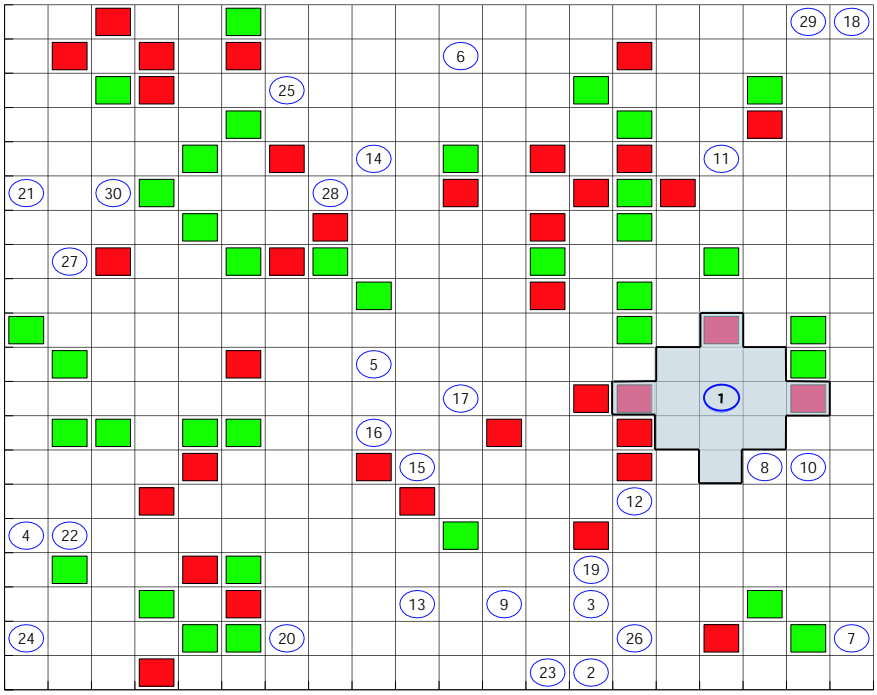


Figure 1: Simulated Environment. The vision range of agent 1 is shadowed and surrounded by a thick line. The two different types of resources are represented by squares of different colors. The resources represented by a darker color give a fixed negative reinforcement and fitness value (-0.8), while the others give a fixed positive value (+0.8). Resource types never change value and when consumed are regenerated on a random tile.

the meme replaces the output of the genetically evolved NN with R_E for the given action a. Basically the meme can recognize a particular sensory context (P), and it reminds the agent that in the past he had performed a certain action (a) and the action yielded a given reinforcement (R_E). If two memes match the current input, the one with highest value is used.

The input pattern P is a $\{-1, *, 1\}^{39}$ vector. Each element of P matches an element of the input vector. $*$ is a *don't care* symbol and matches any value of the input element.

An agent culture consists of up to 20 memes. Memes can be acquired either by transmission or by operant conditioning.

Transmission occurs whenever two individuals are next to each other. In such a way, the two agents can acquire each other's memes.

When an agent experiences an unexpected reinforcement a meme is generated through an operant conditioning mechanism. A reinforcement (R) is unexpected if the instinctive anticipation (R_A) is too different from the actual one:

$$R\, is\, unexpected\, if \;\mid R_A - R \mid \geq 0.075$$

The meme's pattern P is set to match the input vector proceeding the reinforcement, a is set to the performed action, R_E to the reinforcement and V equals $\mid R \mid$.

Memes variants are generated by merging, a stochastic generalization mechanism. Merging can occur if two memes code the same action and expected reinforcement. In this case, the merging probability (P_M) is inversely proportional to the hamming distance (d_H) between the memes input matching patterns:

$$P_M(meme_i, meme_j) \;=\; 1 - \frac{d_H(P_{meme_i}, P_{meme_j})}{39}$$

where the distance between elements containing a $*$ is 0.

Merging is seen as a weak simplification of a boolean functions:
given ($P_1 \wedge a \mapsto R$) and ($P_2 \wedge a \mapsto R$) then with a probability proportional to the similarity of P_1 and P_2 replace them with $((P_1 \circledast P_2) \wedge a \mapsto R)$; where $P_i \in$ pattern, $a \in$ action, $R \in$ reinforcement and $\circledast$ is a bitwise operator:

$$\circledast(b_i, b_j) \;=\; \begin{cases} b_i & if \quad b_i = b_j \\ * & if \quad b_i \neq b_j \end{cases}$$

The value of the new meme is set to $\mid R \mid \cdot N_*$ where N_* is the number of *don't care* symbols in the new meme.

If it does not merge, a meme can be added only if the meme pool size does not exceed the maximum. If the maximum is exceeded a meme is dropped, the one with lower value being dropped with higher probability. Because merging of memes can sometimes produce unfit memes, if the expected reward does not match the one experienced, the meme responsible for the error is instantly removed.

3.3 Lamarckianism

Since memes are generated when the neuro-controller makes a prediction error, the situations that their patterns represents are a source for possible improvements of the genotype. A meme can then be used as a training example with which increase the neuro-controller performance.

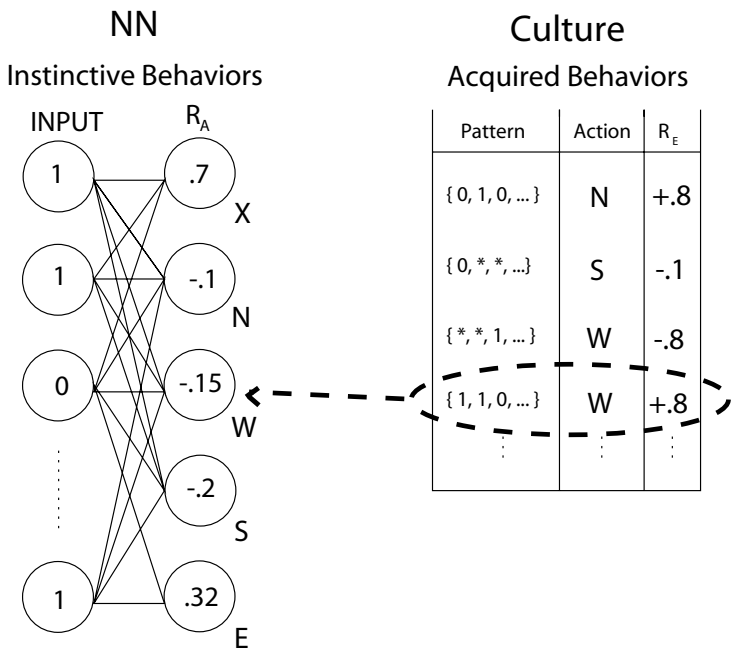


Figure 2: Agent controller. The genetically evolved NN and the acquired culture are activated in parallel. When a pattern matches the current input, the corresponding meme is activated (encircled in the figure). The meme's R_E value ($+.8$) replaces the NN output ($-.15$) for the corresponding action (go west). This modified output vector is interpreted as the agent's expected reinforcements when performing each of the possible actions. The action that is actually performed is selected stochastically giving a .7 probability to the one with highest value in the modified output vector. If, after performing the an action, the actual reinforcement received is too different from the expected one, the agent's culture is modified (refer to section 3.2).

In this way, acquired behaviors are coded back into the genotype in a lamarckian process.

At the beginning of each new generation, all the population undergoes a training phase based on its acquired behaviors. Each individual's culture is used to train its neural network in 10 steps of back-propagation. The gradient is computed in batch mode using each meme as a training exemplar. $*$'s are replaced by zeros for this purpose. A learning rate of .25 is used.

The offspring uses one of the parent's culture for training (but does not inherit the culture itself).

This method cannot compute the exact gradient as each $*$ symbol distorts the error back-propagation. In fact, patterns containing N_* *don't care* symbols, should have each of them replaced by $\{-1, 1\}$. But this would give rise to the expansion of 2^{N_*} training exemplars for each pattern, with the result being too computationally expensive.

Because of this distortion, the training mechanism is actually performing only an incomplete back-coding of the acquired behaviors.

4 Results

We have tried four different simulation settings:

- Genetic: standard genetic evolution without memes.
- Memetic: no evolution, individuals continue acquiring memes through all the simulation. All instinctive outputs are set to zero.
- Baldwinian: both memes and genes evolve. No back-coding takes place.
- Lamarckian: both memes and genes evolve. Memes are used to modify the genes.

Figure 3 shows the average over 10 runs of the average population fitness with and without the help of memes. Selection is performed on fitness values plotted in figure 3A, but since the task is to optimize the evolution of the genetically encoded NN, its performance is given in figure 3B.

Memetic performance increases more quickly than the genetic one. The average fitness scored at generation 25 by memetic populations is at least as high if not better than any of the others even after 500 generations.

But since memes can suggest only actions that yield an immediate reinforcement, the genetically evolved NN could take advantage of the two squares vision range, also approaching distal resources.

Tests performed on a population with an optimal one-square vision controller gave a score of ~ 14, while with an optimal two-square vision controller the score was ~ 20. This means that a genetically evolved NN can score 40% higher fitness.

The baldwinian simulations are the most penalized from the use of memes. The acquired behaviors appear to mask the pressure towards the evolution of the appropriate neuro-controller.

An interesting picture emerges in the comparison between lamarckian and genetic simulations. During the first 250 generations, lamarckian runs outperform standard genetic evolution. After that the *hiding effect* takes over and while the standard genetic evolution keeps on improving the neuro-controller performance, lamarckian populations

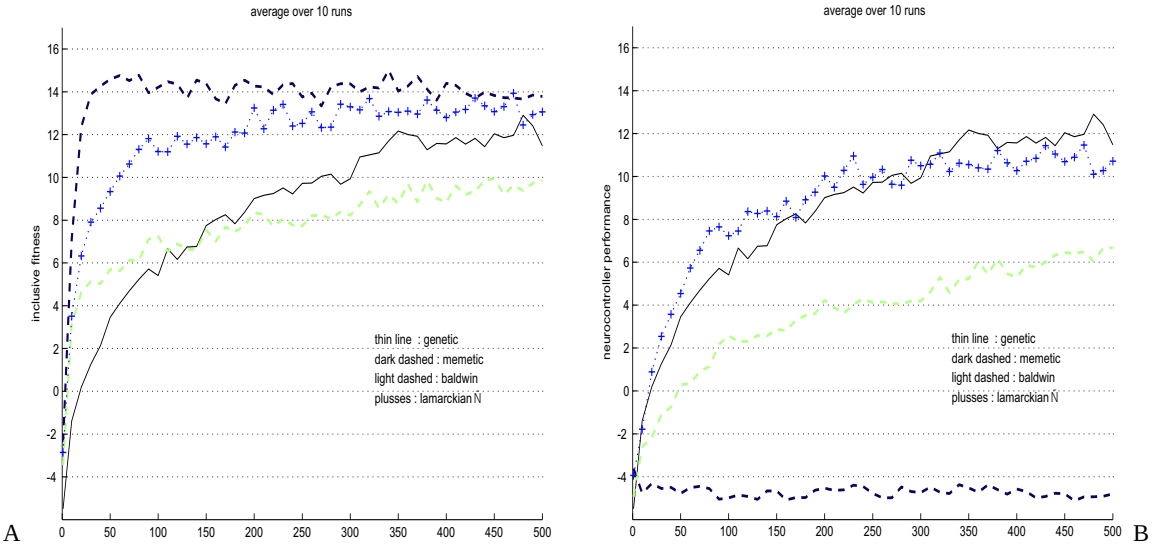


Figure 3: Inclusive Fitness and neuro-controller performance plots. A: inclusive fitness. B: neuro-controller performance. The performance of the neuro-controller is computed without the mediation of agent's cultures, in this way only the evolved NN is responsible of the agent's fitness. Population averages over 10 runs.

show a vary slow increase.

The problem is that their behavior is still very dependent on culture. Genetic assimilation and back-coding of memes, should make the genotype less prone to mistakes and hence reduce cultural acquisition (see section 3.2). This effect takes place but is shown to be slow, see figure 4.

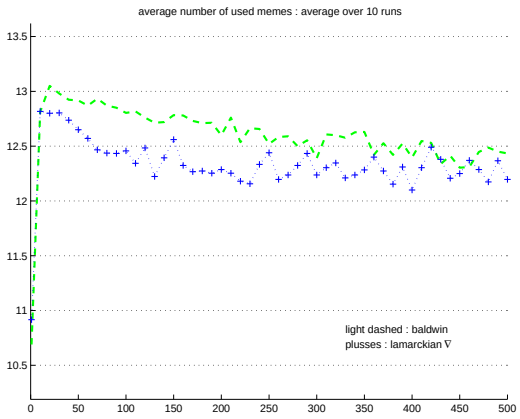


Figure 4: Population average of the number of memes used calculated over 10 runs. Both genetic assimilation and back-coding are slowly reducing the need of memes.

Partial lamarckianism, i.e. only a fraction of the population undergoes the back-coding phase, cannot solve this problem, see figure 5. The stagnation of the evolutionary process is not caused by convergence to a local optima, but by the hiding effect cause by the individuals' cultures.

To reduce the hiding effect, it is possible to progressively decrease the maximum number of memes. This will unmask the advantage for the refinement of the neuro-controller in a second phase of the evolutionary process.

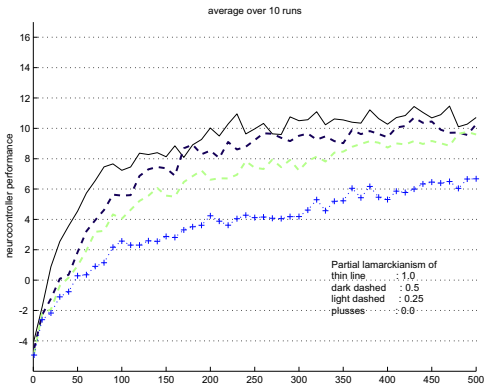


Figure 5: Neuro-controller performance plots for populations with partial lamarckianism. The fraction of the population to which back-coding is applied is varied from 0 to 1 (lamarck share).

The maximum number of memes is reduced every 50 generations from 20 to 0, and at the same time, each individual culture is reset.

As the inclusive fitness must increasingly rely on the genetically evolved NN, pressure is gradually shifted from memes to genes. Results show that populations using lamarckianism have both a higher average and steeper increase in performance, see figure 6.

Figure 7 presents a summary of the performance for the best populations with different experimental settings. Every population with a neuro-controller capable of scoring a fitness greater than 15 is represented by a dot. Populations scoring more than 19 contain almost only optimal

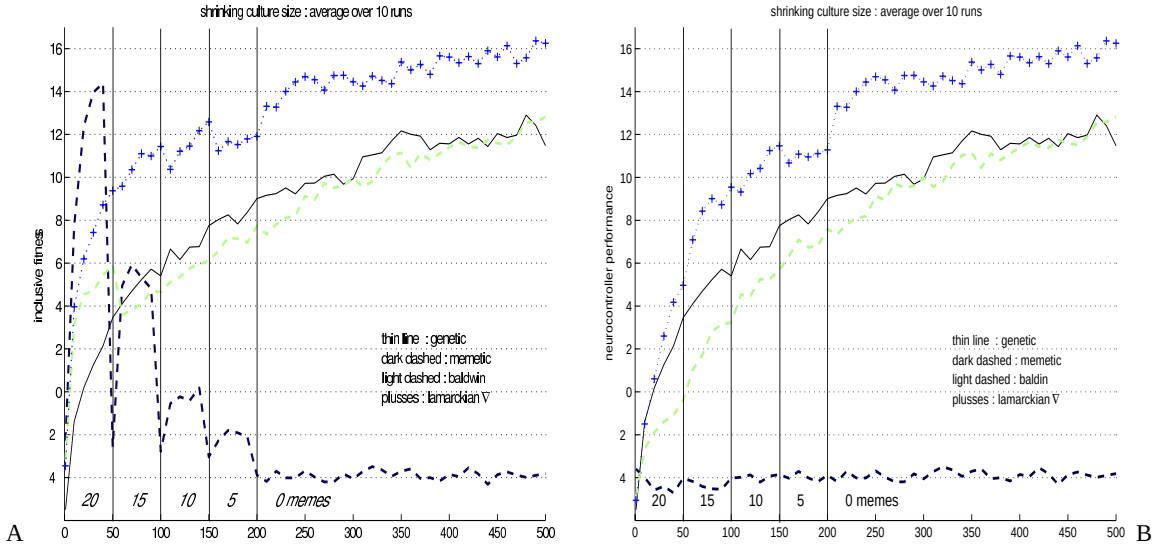


Figure 6: Inclusive Fitness and neuro-controller performance plots with a decreasing maximum culture size. Vertical lines indicate when the meme pool is reset and shrunk. The average performance of the genetic simulations is plotted for comparison. A: inclusive fitness. B: neuro-controller performance. The performance of the neuro-controller is computed without the mediation of agent's cultures, in this way only the evolved NN is responsible of the agent's fitness. Population averages over 10 runs.

controllers. The plot shows the performance enhancement given by lamarckianism with a decreasing culture size.

5 Conclusions

This paper introduces a hybrid model of genetic and cultural evolution. The objective is to develop a neuro-controller for situated agents performing a harvesting task.

Cultural evolution is based on a simple mechanism of operant conditioning and memetic transmission. Given that there are 2^{39} possible inputs and that there are 3^{39} different matching patterns, developing general and fit memes without a priori knowledge is not a trivial task. Cultural evolution is shown to quickly develop behaviors that constitute an incomplete but fit strategy.

The incompleteness of the memetic strategy arises from its innovation process, which considers only immediate sources of reinforcements. Nevertheless, back-coding of memes allows individuals to quickly assimilate behaviors, speeding up the development of an optimal controller.

After this phase of increased evolutionary speed, culture appears to mask any further development. Being based on expectation, the acquisition mechanism was designed to allow un-masking of the complete strategy. This effect takes place but is very slow.

To accelerate it, the maximum number of memes is externally reduced, thus forcing individuals to rely more and more on instinctive behaviors. The method is shown to increase the performance of baldwinian and lamarckian simulations, with the latter capable of outperforming the other evolutionary strategies both in convergence speed and fitness score.

Lamarckian populations seem to benefit from the incremental refinement of the harvesting strategy. Mediated by cultural evolution, at first individuals develop a one-square vision optimal strategy and only afterwards, with the disappearance of memes, the two-square vision strategy is obtained.

These results suggest a methodology for an incremental development of control strategies. By assigning a reinforcement to causes of immediate fitness change (i.e. hitting a wall, entering a target zone or moving to full speed) and with culture back-coding, individuals will quickly learn to perform well in the most trivial cases. Individuals can thereafter discover the complete control strategy building upon the incomplete memetic one, thus saving evolutionary time.

As a final remark, the benefits of the lamarckian process must be inclusive of the overhead introduced by the local search mechanism. As in Houck et al. [4] it is necessary to compare performance taking as a reference the number of function evaluations and not generations alone. To this respect, cultural evolution does not require additional effort. In spite of this fact, populations with a maximum of 20 memes ran in simulation up to 4 times slower than those that did not use memes.

Acknowledgments

The simulations presented in this paper are very computational expensive. Results have been produced by the use of the (inexpensive) ClustIS cluster [20]. All the code has been written in Matlab.

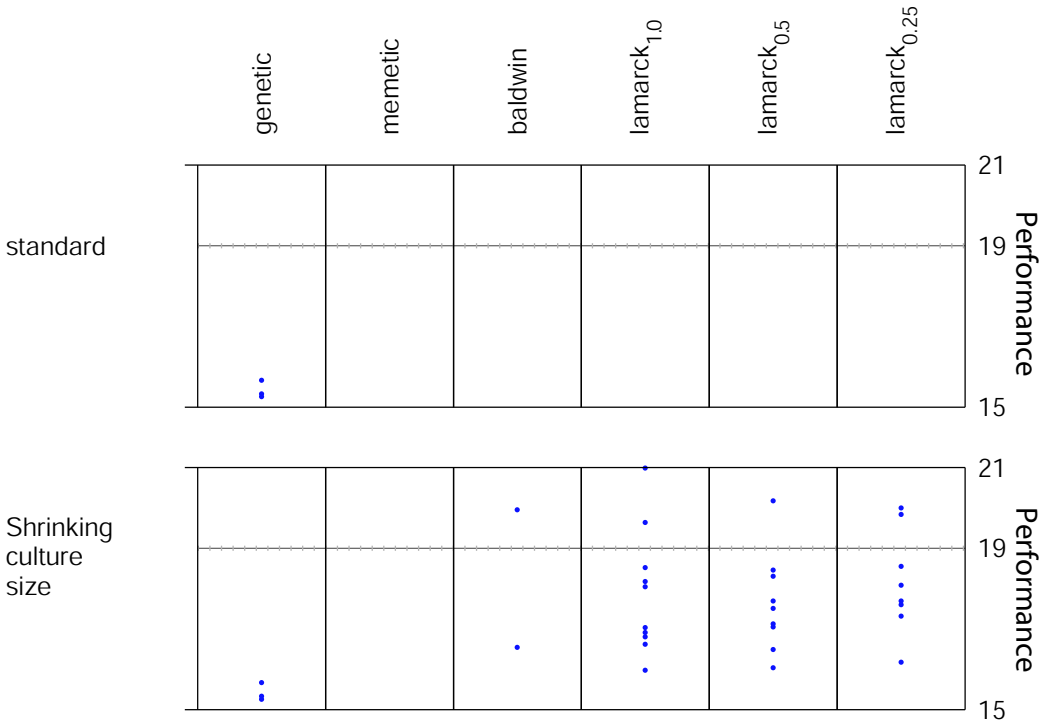


Figure 7: Neuro-controller performance plots for all the different simulations. Each dot represents the final average performance of one population (only those with a score greater than 15 are plotted).

Bibliography

[1] J.B. Lamarck: Philosophie zoologique, 1809

[2] J.M. Baldwin: A new factor in evolution. American Naturalist **30** 441–451 (1896)

[3] D. Whitley: Modeling Hybrid Genetic Algorithms. in Genetic Algorithms. Wiley and Sons, 191–201 (1995)

[4] C.R. Houck, J.A. Joines, M.G. Kay and J.R. Wilson: Empirical Investigation of the Benefits of Partial Lamarckianism. Evolutionary Computation 5(1), 31-60 (1997)

[5] P. Turney: Myths and legends of the Baldwin Effect. Proceedings of the ICML-96 (13th International Conference on Machine Learning) 135–142 (1996)

[6] R.K. Belew, M. Mitchell: Adaptive individuals in evolving populations: models and algorithms. Addison-Wesley (1996)

[7] G.E. Hinton, S.J. Nowlan: How learning can guide evolution. Complex Systems **1** 495–502 (1987)

[8] R. French, A. Messinger: Genes, phenes and the Baldwin effect. In Proceedings of Artificial Life IV 277–282 (1994)

[9] G. Mayley: Landscapes, Learning Costs and Genetic Assimilation. In Evolutionary Computation, Evolution, Learning and Instinct: 100 years of the Baldwin Effect (1996)

[10] G. Mayley: Guiding or hiding. In the Proceedings of the Fourth European Conference on Artificial Life (ecal97) (1997)

[11] L. Spector and S. Luke: Cultural Transmission of Information in Genetic Programming. in Genetic Programming 1996: Proceedings of the First Annual Conference, 209–214 (1996)

[12] L. Spector and S. Luke: Culture Enhances the Evolvability of Cognition. in proceedings of Cognitive Science 1996 (1996)

[13] C. Wellock and B.J. Ross: An Examination of Lamarckian Genetic Algorithms. 2001 Genetic and Evolutionary Computation Conference Late Breaking Papers, 9-11, 474–481 (2001)

[14] S. Nolfi, J.L. Elman and D. Parisi: Learning and Evolution in Neural Networks. Adaptive Behavior, Vol. 3, 1, 5–28 (1994)

[15] D. Floreano and F. Mondada: Evolutionary Neurocontrollers for Autonomous Mobile Robots. Neural Networks, 11, 1461–1478 (1998)

[16] J. R. Koza: Genetic Programming II. MIT Press, Cambridge (1994)

[17] J.H. Holland: Adaptation in natural and artificial systems. MIT press, Cambridge (1975)

[18] E.R. Kandel, J.S. Schwartz, T.M. Jessell: Principles of neural science. Elsevier Science Publishing Co., New York (2000) 4th edition

[19] H. Markram, J. L'bke, M. Frotscher, B. Sakmann: Regulation of Synaptic Efficacy by Coincidence of Post-synaptic APs and EPSPs. Science, **275** 213–215 (1997)

[20] J. Cassens and Z. Constantinescu Fülöp: It's Magic: SourceMage GNU/Linux as HPC Cluster OS. to appear in Proceedings Linuxtag 2003, Karlsruhe, Germany, July 2003. http//clustis.idi.ntnu.no/

E Increasing the evolvability of development with Embryonal Stages

Author: Diego Federici. Published in proceedings of WORLDS Workshop on Regeneration and Learning in Developmental Systems, hosted by GECCO 2004

Abstract:

Indirect encoding methods are aimed at the reduction of the combinatorial explosion of search spaces, therefore increasing the evolvability of large phenotypes. These so called Artificial Embryogeny systems have so far shown increased scalability for problems involving solutions of low complexity. This leaves open the more general question about the evolvability of complex phenotypes. In this paper, we introduce a novel method of cellular growth regulated by a developmental program. Genotypes are selected for their ability to develop organisms of specific shape and cell types. Results show that the use of Embryonic Stages, which involves the incremental addition of growth programs, displays positive effects on the evolvability of development.

Objective:

Development appears very interesting for the evolution of large phenotypes, still previous work has produced only limited results. Here we introduce a general method to increase the heterochrony of development and, as a consequence, evolvability.

Conclusions:

The performance of the evolutionary systems is promising. Still, further investigations of the general properties of the model are required.

Increasing the evolvability of development with Ebryonal Stages

Diego Federici

Norwegian University of Science and Technology
Department of computer and information science
N-7491 Trondheim, Norway
federici@idi.ntnu.no

Abstract. Indirect encoding methods are aimed at the reduction of the combinatorial explosion of search spaces, therefore increasing the evolvability of large phenotypes.
These so called Artificial Ebryogeny systems have so far shown increased scalability for problems involving solutions of low complexity. This leaves open the more general question about the evolvability of complex phenotypes.
In this paper, we study the evolvability of a model of cellular growth regulated by a developmental program. Genotypes are selected for their ability to develop organisms of specif shape and cell types.
Results show that a particular method involving Embryonal Stages displays positive effects on the evolvability of developmental programs.

1 Introduction

The evolution of large phenotypes is one of the most serious problems in the field of evolutionary computation (EC). With each characteristic of the phenotype encoded by a single gene, the increase of the phenotipic size imposes for direct encoding strategies a combinatorial explosion of the search space.

On the other side, biological systems develop into mature organisms with a complex process of embryogeny. Embryogeny is mediated by the interaction of DNA, RNA and proteins to produce the cell regulatory system. This sort of interaction does not permit a one to one map from gene to phene, since each gene influences several aspects of the phenotype.

Motivated by the development of biological systems, several authors have proposed indirected encoding schemes. With indirect encoding, each phenotype is developed by a process in which genes are reused several times. The term 'Artificial Embryogeny' (AE) has been recently proposed to describe these evolutionary systems [1].

In AE, development is de facto a decompression of the genotype. Since compression is generally higher for regular targets, a serious question is how much these methods will prove viable for the evolution of high complexity phenotypes.

Hints in this direction, also come from a recent study on Matrix Rewriting [2], showing how the genotpye-pheotype correlation decreases with the complexity of the phenotype [3].

In this paper we present a model of cellular growth which is targeted to the development of multi-cellular organisms of specific two dimensional shapes and colors. These organisms must be intended as a metaphor of functional devices, in which each color represents the specific function of the cell and the 2D displacement encodes their local connectivity.

For example, such organisms could develop decentralized locally connected digital circuits [4] or layers of artificial neural networks [5].

In AE, growth methods are either based on rewriting rules or cell chemistry models. The firsts, like the well known Cellular Encoding [6], evolve the rules of a grammar used to produce the mature organisms. The seconds proceed by evolving the cell metabolism thus controlling the state and development of the phenotype.

The model presented in this paper belongs to this second category. Phenotypes are multi-cellular organisms in which each cell shares the same growth program. The growth program regulates the cell type, chemical production and replication based only on the state of the particular cell and of its neighborhood.

Also belonging to this cathegory, Bentley and Kumar proposed a model which develops 2D tiling patterns [7]. Cells can only be of a single type and the aim is to produce perfect tessellating phenotypes. The growth program is composed by a set of rules which upon matching the state of the local neighborhood activate a specific cellular response. Results showed that the systems performed and scaled better than a direct encoding method. On the other side, the best solutions developed had very regular phenotypes.

Miller extended Bentley and Kumar's model and developed more complex patterns [8]. He allowed 4 different cell types (colors) and a chemical undergoing diffusion. The growth program is a boolean network evolved with the Cartesian Genetic Programming. Results showed evolved phenotypes resembling the target with only very few misplaced cells.

Additionally, Miller analyzed the behaviour of the evolved phenotype after the developmental step in which the fitness was computed and when subjected to severe mutilations. Phenotypes were shown to regrow the missing parts regaining qualitative resemblance to the target. The self repair feature is very interesting since it was not selected for during evolution.

Additional references can be found in the work of Stanley and Miikkulainen which have proposed a survey and a taxonomy for AE systems [1]. Another survey, addressing more specifically AE in control systems, can be found in [9].

Albeit that AE is showing promising results, it suffers from a general difficulty connected to the evolvability of the genotypes. Miller reported that in the development of a specific 'french flag' pattern few runs produced satisfiable results. The other tended to be stucked in local optima.

One of the reasons for this is intrinsic to the idea of gene reuse. In fact, if we immagine an individual of high fitness with only a few misplaced phenes, a direct encoding method could allow the tweaking of the few corresponding uncorrect genes allowing the cumulative refinement of the phenotype. In the case of indirect encoding, the change of the same few phenes may require a complete redesign

of the genotype. In fact, the corrisponding genes might be resposible for other features of the phenotype in other developmental stages. Their change may cause interferences in the maturation of the organism with catastrophic effects.

Therefore AE models may be prone to create deceptive fitness landscapes as the results in [3] seem to suggest. To reduce this effect and increase evolvability, in this paper we have adopted three strategies.

1. An Artifical Neural Network (ANN) encodes the growth program. Compared to discrete rules, the space of continuous functions representable by ANNs allows a finer tuning of cellular responces. In this case, escaping local optima should be easier.
2. Population diversity is increased rewarding fitness to individuals with rare phenes. This reduces the chances that innovation, which in developmental systems have saltatory charactarestics, may favor a single strain of genotypes.
3. Development may happen in more than one Embryonal Stage. The single growth program is sub-divided in several stages each one governing the development at subsequent times. Stages are evolved incrementally, the first being evolved before. Embryonal Stages resemble but are capable of differentiate from the previous ones, therefore allowing genetic refinement without interference and a 'zoom-in effect' in the search space.

The remaining of the paper is organized as follows: section 2 contains a description of the evolutionary task, section 3 the developmental model, section 4 the details of the genetic algorithm, section 5 the results of the simulations and section 6 the conclusions.

2 The evolutionary task

Yet an other issue in AE is the proper choice of the targets used for benchmarking. In [1] the authors suggest 4 different tasks: evolution of pure simmetry, of specific shapes, of specific conenctivity patterns and of a simple controller.

In the simulations presented we have selected four shapes with various levels of simmetry. Fig.1A is a pattern composed of three colored stripes similar to the one used in [8]. Fig.1B has a bounding layer which insulates the internal cell type from the outside. Fig.1C contains repetitions of a simple 'plus' pattern. These sort of regularities should be exploitable by AE systems. Fig.1D is a more complex Norwegian flag pattern and can be seen as a vertical and horizontal insulated wiring.

Fitness is proportional to the resemblance of an individual to the target, and is computed as shown in equation 1.

$$FIT(P,T) = \Big(\sum_{x,y} \; EQUALS\,(\,P,T,x,y\,) \cdot PheneValue(x,y) \Big) \;/\; ||T|| \qquad (1)$$

$$EQUALS\,(\,P,T,x,y\,) = \begin{cases} 0 \; if \; P(x,y) \neq T(x,y) \\ 1 \; if \; P(x,y) = T(x,y) \end{cases}$$

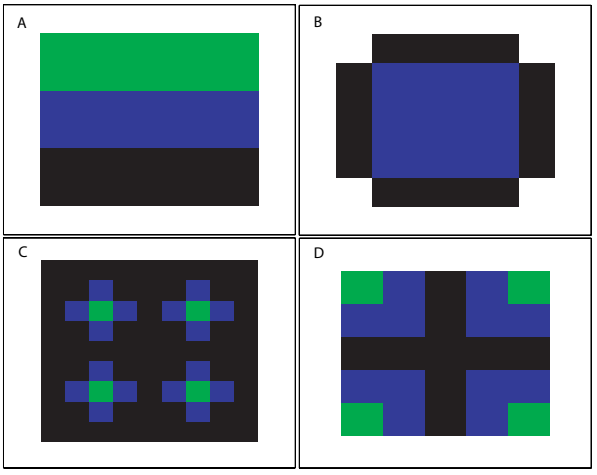


Fig. 1. The four targets of the evolutionary task. A) a three-stripes pattern, B) a bounded pattern, C) a group of pluses, D) a Norwegian flag pattern. Development could take advantage of the various degrees of simmetry and modularity of the targets.

where P is the phenotype to evaluated, T the target, and $PheneValue(x, y)$ is the frequency in the population of the phene in position x, y. $PheneValue$ is used to increase population diversity (see also section 4.1).

3 The developmental model

Phenotypes are developed starting from a single egg (zygote) placed in the center of a fixed size 2D grid. Morphogenesis proceeds in discrete developmental steps, during which the growth program is executed for each cell. The execution order is determined by age, older cells being taken first.

Cells (see figure 2) are characterized by internal and external variables. Internal variables define the cell state and move with it, while external ones (chemicals) belong to the environment and follow a simple diffusion law.

At each developmental step, any existing cell can release chemicals, change its own type, alter the internal metabolism and produce new cells in the cardinal directions North, West, South and East. If necessary, existing cells are pushed sideways to create space for the new cells (see figure 3). When a cell is pushed outside the boundaries of the grid it is permanently lost.

Morphogenesis is governed by an Artificial Neural Network (Morpher) defined by the genotype. The genotype is a direct gray-code representation of the Morpher. The hyperbolic tangent is used as transfer function.

The Morpher (figure 4) receives in input the current cell internal and external variables, and the cell types of the neighboring cells in the four cardinal directions. Its output determines the new internal and external variables of the cell

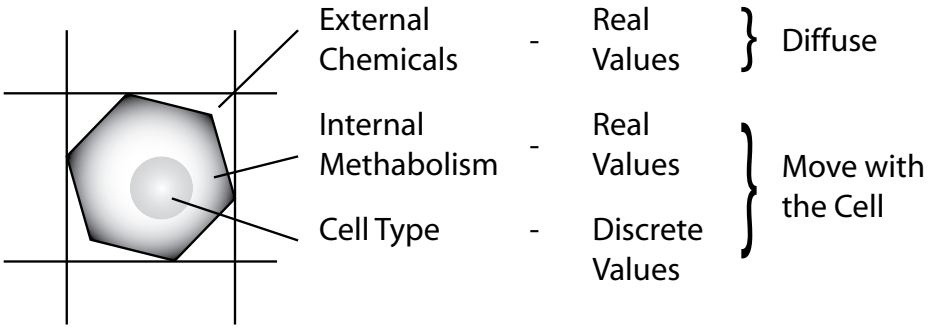


Fig. 2. Description of the variables used for development. External variables follow a diffusion law, while internal ones move with the cell. While chemical concentrations, internal and external, vary in the range $[0, 1]$, the cell type can take one of 4 discrete values

and, in case, the internal variables of the newly generated cells. An additional local variable, the cell age, is set to 1 at birth and decays exponentially to 0.

Chemical production and internal metabolism values are read directly to and from input and output lines, one line being dedicated to each different internal or external chemical. The cell type is encoded / decoded from a single value as shown in Eq.2 and 3.

$$Encode(value) = \begin{cases} 0 \;\; if \;\; v < -2/3 \\ 1 \;\; if \;\; v \in (-2/3 \; 0) \\ 2 \;\; if \;\; v \in [0 \; 2/3) \\ 3 \;\; if \;\; v > 2/3 \end{cases} \tag{2}$$

$$Decode(type) = \begin{cases} -1 \;\;\; if \;\; t = 0 \\ -1/3 \;\; if \;\; t = 1 \\ 1/3 \;\;\; if \;\; t = 2 \\ 1 \;\;\;\;\; if \;\; t = 3 \end{cases} \tag{3}$$

In these simulations, feedforward networks with four hidden units have been used.

4 The evolutionary model

Each population in the simulations presented is composed by 400 individuals undergoing elitarian selection with a survival share of 1/8. A tenth of the new individuals are produced by crossover, while all the offspring undergoes mutation. The mutation operator takes each weight with a P_{mut} probability and adds to it Gaussian noise with V_{mut} variance.

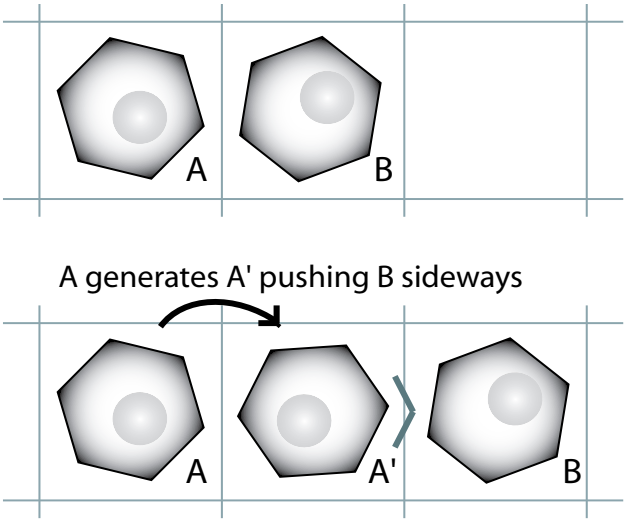


Fig. 3. Placement of new cells. If necessary, space is created by pushing sideways existing cells. Cells falling off the development grid are lost.

P_{mut} and V_{mut} vary in the ranges $[.01, .2]$ and $[.035, .1]$ respectivelly. Their value is proportional to the time passed from the last increase in the top fitness score, reaching the maximum in ten generations. These values have being selected after a preliminary study on short evolutionary runs, proving to be more effective than fixed ones.

4.1 population diversity

Often, in AE systems, evolutionary improvements have saltatory characteristics. Under these conditions a positive innovation can increase the reproductive chance of a particular strain reducing the chance of survival of all others. This increases the chances of convergence to local optima.

To increase population diversity, fitness scores are modified looking at the frequency of the phenes that individuals possess, counteracting homogenization and favoring individuals with rare characteristics:
(1) the population is first ordered by fitness values before modification (in case of ties younger individuals go first).
(2) fitness scores are recomputed following this order, but the value for each phene ($PheneValue$ in Eq.1) decreases linearly with use from 1 to 1/100.

4.2 embryonal stages

Biological organisms have the interesting property that embryonal developmental stages of philogenetically related species share similarities:

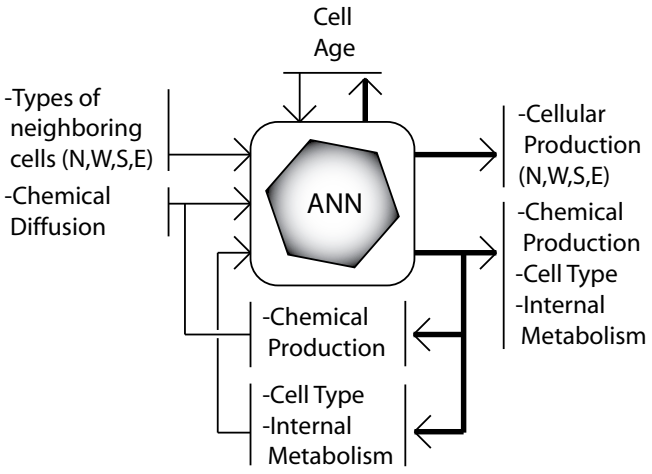


Fig. 4. Inputs and outputs of the growth program, the Morpher, implemented with a feedforward ANN. Each cell internal variables, cell type and metabolism, implement a direct feedback pathway, while chemical production and diffusion offer a channel for longer range communication.

> It is generally observed that if a structure is evolutionary older than another, then it also appears earlier than the other in the embryo. Species which are evolutionary related typically share the early stages of embryonal development and differ in later stages. [...] If a structure was lost in an evolutionary sequence, then it is often observed that said structure is first created in the embryo, only to be discarded or modified in a later embryonal stage. Wikipedia [10]

This apparent relationship between ontogeny and philogeny, which should not be confused with the discredited Recapitulation theory, suggests that multicellular organisms evolve new traits incrementally over older phenotipic characteristics.

In fact, the modification of early embryonal stages may disrupt development with catastrophic results. Therefore, mutations affecting later stages of the ontogenetic process will have a higher probability to be useful.

This suggests, also for AE, that decreasing the chances of modification of the early steps of development will increase evolvability. Although such preservation mechanism could be found by means of evolution, to simplify the evolutionary task we propose an explicit mechanism for it.

We allow growth to be controlled by a set of programs. Phenotypes are developed in subsequent embryonal stages, each one governed by a different program.

At the beginning of the evolutionary search, organisms develop in a single stage. When a certain performance or generation are reached a new stage is

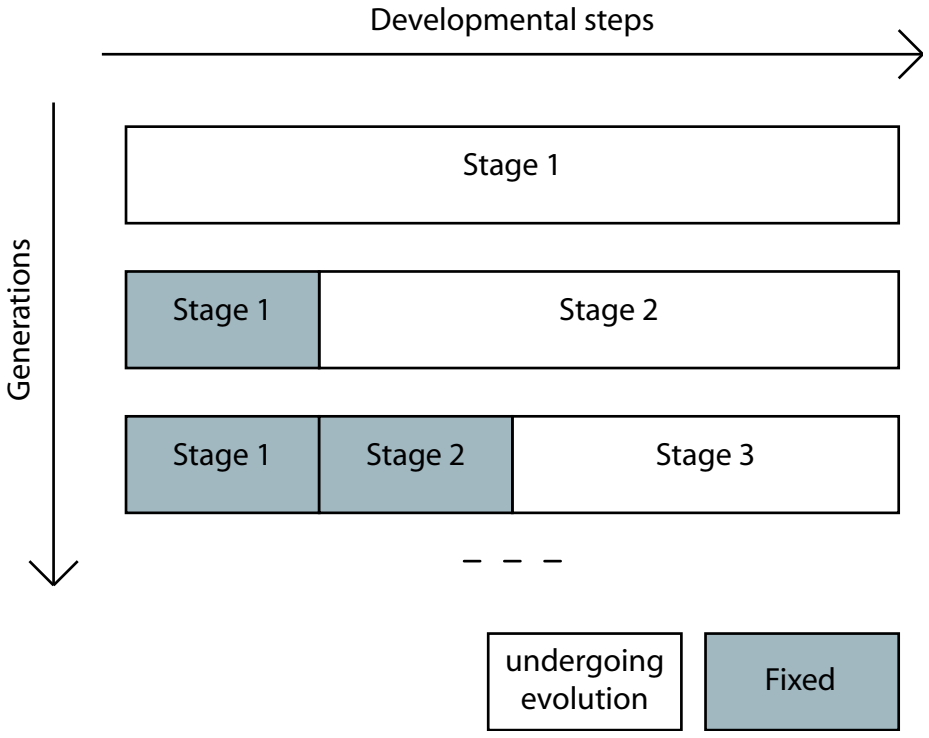


Fig. 5. Embryonal stages. Stages are introduced incrementally, and they govern ontogeny from a pre-determined developmental step. Only the latest step is subject to evolutionary operators, the others are fixed.

added. While the older stage starts the ontogenetic process as usual, the new one will assume control at a pre-determined developmental step completing the maturation of the organism (see Figure 5).

The new stage developmental program is initialized as a copy of the previous one. In this way, at first, the introduction of new stages does not alter ontogeny. On the other side, the innovation operators are allowed to modify only the program of the latest embryonal stage without affecting any of the previous ones.

Additional stages, being built upon the previous ones, add resolution to the specific spot in the search space. This allows an incremental refinement of the ontogenetic process, helping escaping local optima due to interference effects.

In the case of complex phenotypes, this positive effect should be even more visible, since the amount of information required to produce them is higher, as the chances of gene's interference.

5 Results

The performance of runs with and without embryonal stages is compared. Simulation specific details are given below:

- 1000 generations maximum.
- fitness is computed at developmental steps 7 and 8. Fitness values are expressed in % of target resemblance.
- simulations can have either 1 or 5 embryonal stages. Where used, new stages are introduced at generations 500, 750, 875 and 938, or when a maximum population fitness of 75%, 83%, 92% and 100% is reached. Each stage takes development from steps 0, 4, 5, 6 and 7.
- cells can release one external and one internal chemicals.

Statistics from 10 runs with each parameter settings (target and number of stages) are shown in the following table.

		target shape (see fig.1)			
phases		bounded	3 stripes	flag	pluses
1	max	95%	84%	86%	83%
	avg	83%	76%	78%	79%
	min	70%	68%	72%	78%
5	max	99%	98%	94%	88%
	avg	91%	88%	88%	83%
	min	89%	77%	76%	81%
fitness change		+4%	+16%	+12%	+5%

From the results above, it is evident that embryonal stages have a positive effect in all cases. Also, performance is somewhat proportional to the regularity of target phenotypes.

In figures 6–9 is possible to see the development of the best evolved individuals. Steps 7 and 8 (highlighted in the figures) are the only ones used to calculate fitness. This means that the following steps, 9 and 10, are not relevant for selection purposes. Even though, individuals maintain a resemblance to the target, with a few of them, such as the one in figure 7, reaching a developmental stasis.

The modular target of figure 9, which is also the less regular, proved to be the most difficult to evolve. Development is very chaotic until it reaches the steps in which fitness is checked. This lack of regularity during development suggests that, under such conditions, modular decomposition will be hardly achieved by means of gene reuse. In fact, neighbors and chemical concentrations will generally share little similarity in the places where modules should develop.

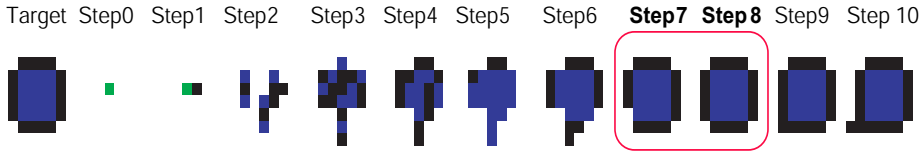


Fig. 6. development of the best 'bounded' individual

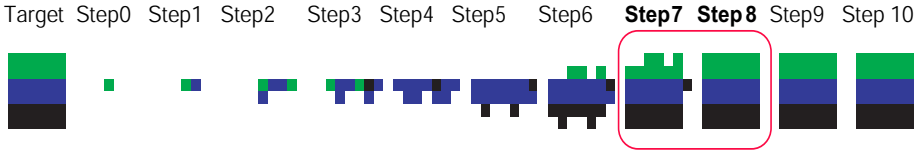


Fig. 7. development of the best '3 stripes' individual

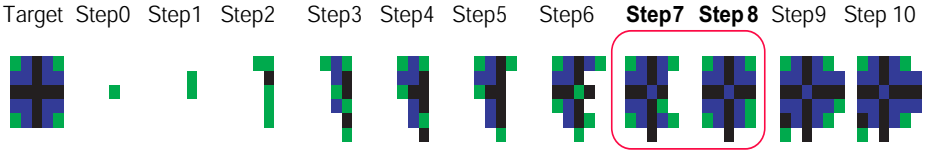


Fig. 8. development of the best 'norwegian flag' individual

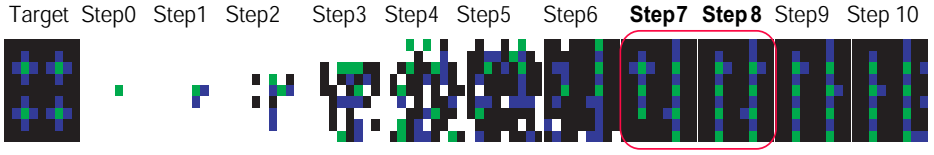


Fig. 9. development of the best 'pluses' individual

We have also checked how well these organisms responded to damage. The best evolved individuals have been developed with various degrees of cellular necrosis. At selected developmental steps, a number of cells have been removed from the phenotype (but at least one cell was left). The average fitness of the final organisms are plotted in figure 10.

Since cells in early developmental steps are more important for development, one would expect a domino effect on fault propagation. Still performance degrades linearly the earlier damage is applied. Organisms seem to limit the effect of faults during development. This behavior was not selected for and is similar to what reported in [8].

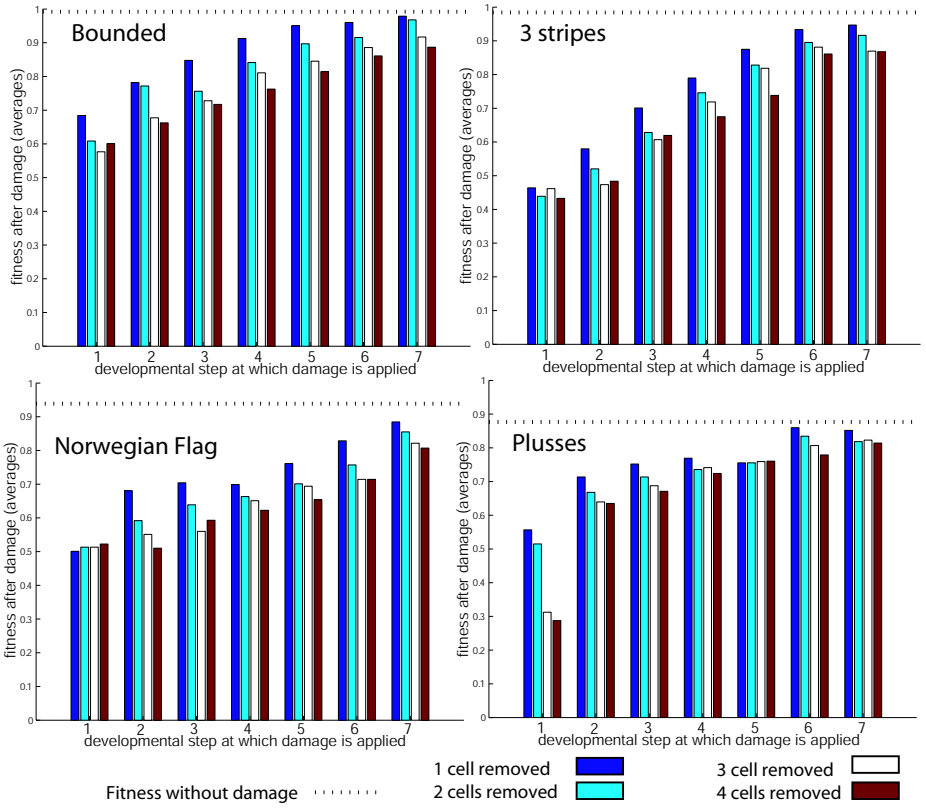


Fig. 10. Impact of cell death at different developmental steps. Average fitness computed over 20 random development faults consisting of 1 to 4 cell deaths. Countrary to intuition, faults do not propagate exponentially during development.

6 Conclusions

An artificial embryogeny (AE) model has been proposed and tested in the development of four two dimensional targets of specific morphology. The system is based on the evolution of a growth program that regulates the ontogeny of multicellular organisms starting from a single egg cell.

This model is aimed at the construction of an evolutionary platform for the development of functional organisms, such as locally connected digital circuits [4] or neural networks [5].

The primary aim of the paper is to investigate the effects of multiple embryonal stages on evolvability. This method of incremental development is devised to reduce the catastrophic interference caused by the change of genes that regulate ontogeny in different growth phases.

Results show that embryonal stages have positive effects on performance, even if specific targets still prove hard to evolve.

Also, the behavior of phenotypes undergoing different degrees of damage during development was analyzed. Similar to results in [8] individuals showed a good resistance to faults. This is particularly interesting since there was no selection for this characteristic.

6.1 further work

The role of chemicals, both internal and external, deserves additional investigation. For example, we have noticed that the evolution of the 'bounded' target benefits from the presence of an external chemical, while the 'pluses' performs better when only internal chemicals are present. For the other two targets, performance increases when chemicals are not present at all.

Also, the scalability of these development systems should be put to the test, searching for the relation between evolutionary effort and size of the search space.

Acknowledgements
I wish to thank Julien Miller for the useful discussions that inspired the work presented in this paper, and Keith Downing and Gunnar Tufte for the valuable suggestions. The simulations presented in this work were run on the inexpensive ClustIS cluster [11].

References

1. Stanley, K., Miikulainen, R.: A taxonomy for artificial embryogeny. Artificial Life 9(2):93–130 (2003)
2. Kitano, H.: Designing neural networks using genetic algorithms with graph generation system. Complex Systems, 4(4):461–476 (1990)
3. Lehre, P., Haddow, P.C.: Developmental mappings and phenotypic complexity. Proceeding of CEC 2003 (2003)
4. Tufte, G., Haddow, P.C.: Insertion of functionality into development on an sblock platform. Proceeding of CEC 2003 (2003)
5. Federici, D.: Evolving a neurocontroller through a process of embryogeny. submitted to SAB conference 2004 (2004)
6. Gruau, F., ed.: Neural Network Synthesis using Cellular Encoding and the Genetic Algorithm. PhD Thesis, Ecole Normale Sup'erieure de Lyon (1994)
7. Bentley, P., Kumar, S.: Three ways to grow designs: A comparison of embryogenies for an evolutionary design problem. Proceeding of GECCO 1999, 35-43 (1999)
8. Miller, J.: Evolving developmental programs for adaptation, morphogenesys, and self-repair. Proceeding of ECAL 2003, 256–265 (2003)
9. Kodjabachian, J., Meyer, J.: Evolution and development of control architectures in animats. Robotics and Autonomous Systems, 16(2-4) (1995)
10. Wikipedia: Ontogeny and phylogeny. http:// en2.wikipedia.org /wiki /Ontogeny_and_phylogeny (2003)
11. Cassens, J., Fülöp, Z.C.: It's magic: Sourcemage gnu/linux as hpc cluster os. in Proceedings Linuxtag 2003, Karlsruhe, Germany (2003)

F Multi-cellular development: is there scalability and robustness to gain?

Authors: Daniel Roggen and Diego Federici
Published in proceedings of PPSN VIII 2004 The 8th International Conference on Parallel Problem Solving from Nature

Abstract:

Evolving large phenotypes remains nowadays a problem due to the combinatorial explosion of the search space. Seeking better scalability and inspired by the development of biological systems several indirect genetic encodings have been proposed. Here two different developmental mechanisms are compared. The first (POEtic), developed for hardware implementations, relies on simple mechanisms inspired upon gene regulation and cell differentiation. The second, inspired by Cellular Automata, is an Artificial Embryogeny system based on cell-chemistry. This paper analyzes the scalability and robustness to phenotypic faults of these two systems, with a direct encoding strategy used for comparison. Results show that, while for direct encoding scalability is limited by the size of the search space, developmental systems performance appears to be related to the amount of regularity that they can extract from the phenotype. Finally the importance of comparing different genetic encodings is stressed, in particular to evaluate which key characteristics are necessary for better scalability or fault-tolerance. The lack of standard tests or benchmarks is highlighted and some characterizations are proposed.

Objective:

Compare the scalability of two different indirect encoding approaches against direct encoding.

Conclusions:

In a task that is favourable for direct encoding, both indirect methods produce more efficiently large phenotypes. Also, the distributed nature of the encoding can be exploited to achieve high levels of fault-tolerance.

Multi-cellular development: is there scalability and robustness to gain?

Daniel Roggen and Diego Federici*

Autonomous Systems Laboratory, EPFL, Lausanne, Switzerland
*Norwegian University of Science and Technology, Trondheim, Norway
http://asl.epfl.ch *http://www.idi.ntnu.no/∼federici
daniel.roggen@epfl.ch *federici@idi.ntnu.no

Abstract. Evolving large phenotypes remains nowadays a problem due to the combinatorial explosion of the search space. Seeking better scalability and inspired by the development of biological systems several indirect genetic encodings have been proposed. Here two different developmental mechanisms are compared. The first, developed for hardware implementations, relies on simple mechanisms inspired upon gene regulation and cell differentiation. The second, inspired by Cellular Automata, is an Artificial Embryogeny system based on cell-chemistry. This paper analyses the scalability and robustness to phenotypic faults of these two systems, with a direct encoding strategy used for comparison.
Results show that, while for direct encoding scalability is limited by the size of the search space, developmental systems performance appears to be related to the amount of regularity that they can extract from the phenotype. Finally the importance of comparing different genetic encodings is stressed, in particular to evaluate which key characteristics are necessary for better scalability or fault-tolerance. The lack of standard tests or benchmarks is highlighted and some characterisations are proposed.

1 Introduction

The evolution of large phenotypes is one of the most serious problems in the field of evolutionary computation. With each characteristic of the phenotype encoded by a single gene, the increase of the phenotypic size imposes for direct encoding strategies a combinatorial explosion of the search space.

On the other hand, biological systems develop into mature organisms with a complex process of embryogeny. Embryogeny is mediated by the interaction of DNA, RNA and proteins to produce the cell regulatory system. This sort of interaction does not permit a one to one map from gene to phenotypic trait (phene), since each gene influences several aspects of the phenotype (pleiotropy).

Motivated by the development of biological systems, several authors have proposed indirected encoding schemes. With indirect encoding, each phenotype is developed by a process in which genes are reused several times.

In this case, development is de facto a decompression of the genotype. But since compression is generally higher for regular targets, a serious question is how

much these methods will prove viable for the evolution of arbitrarily complex phenotypes. For example, the correlation between genotype and phenotype space may decrease as the complexity of the target increases [1].

In other words, when looking at system evolvability, it appears that there is a tradeoff between the combinatorial gain achieved by searching in a restricted genotypic space and hindrances of a more complex fitness landscape caused by gene reuse.

Additionally, the restriction on the search space implies that a part of the solution space becomes unreachable, and some targets (such as those of high regularity) might be more viable than others.

These considerations imply that in the analysis of such systems, performance benchmarks play a fundamental role. Still, there is little agreement on a set of evolutionary targets that can be used for assessing their quality. On the contrary, it appears that the tasks are often selected ad hoc to highlight the strengths of a particular model.

In this paper we want to compare two different developmental models, the first used in the POEtic circuit [2, 3], the latter an Artificial Embryogeny system [4] based on cell chemistry [5, 6]. The comparison is carried out for varying phenotypic sizes, against a direct encoding strategy in a task that should favour the latter.

The intention is to investigate the viability of these two indirect encoding methods without leaving doubts about the generality of the results.

To this end, we have tried to set up a 'worst case scenario' for developmental systems, pushing for results that do not depend on particular features of the targets.

The selected task is the evolution of specific 2D patterns of various complexity (figures 3 and 4) and sizes (from 8x8 to 128x128), with fitness being proportional to the resemblance to the target. In the case of the direct encoding strategy, with a gene representing a single pixel, the fitness landscape is a simple unimodal function. On the contrary, in the case of development, gene reuse may imply a multimodal deceptive fitness landscape. Thus, the comparison of the methods will allow to address the influence of search space and pleiotropy on evolvability.

Development systems also provide internal dynamics which are absent in direct encoding strategies. These dynamics may provide a way to withstand phenotypic injuries. This aspect is explored by comparing the tolerance to faults of both systems with the linear deterioration typical of direct encodings.

The remaining of the paper is organized as follows. Section 2 gives an overview of the development systems, section 3 describes the evolutionary task, section 4 presents the results on fault resistance and section 5 concludes.

2 Multi-cellular growth and differentiation mechanisms

2.1 Morphogenetic System (MS)

The morphogenetic system [3] (**MS**) is a developmental model designed for multi-cellular systems and focusing on simplicity and compact hardware imple-

mentation, initially developed for the POEtic circuit [2]. It uses signalling and expression mechanisms which are remotely inspired by the gene expression and cell differentiation of living organisms [7], notably by the fact that concentrations of proteins and inter-cellular chemical signalling regulate the functionality of cells. Related works include the use of L-Systems [8] and various cell-based developmental processes [9, 10], and biologically plausible development models [11].

The **MS** assigns a functionality to each cell of the circuit from a set of predefined functionalities. Here functionalities are the colours necessary to draw the patterns. It operates in two phases: a *signalling phase* and an *expression phase*.

The signalling phase uses inter-cellular communication to exchange signals among adjacent cells to implement a diffusion-like process. A signal is a simple numerical value (signal intensity) that a cell owns. Special cells, called *diffusers*, own a signal of maximum intensity and start the diffusion process. Diffusion rules rely on the four neighbours of a cell to generate signal intensities which decrease linearly with the Manhattan distance from the diffuser. They do so by taking the smallest value for which the signal gradient with all the initialized neighbours is -1, 0 or 1. Figure 1 shows an example of the signalling phase in the case of a single type of signal, with two diffusers placed in the cellular circuit.

The expression phase finds the functionality to be expressed in each cell by matching the signal intensities in each cell with a corresponding functionality stored in an expression table.

The genetic code contains the content of the expression table and the position of the diffusers. A genetic algorithm is used for evolution. 16 diffusers and 4 functionalities (colours) are used. The population is composed of 400 individuals, selection is rank selection of the 300 best individuals, the mutation rate is 0.5% per bit, one-point crossover rate is 20% and elitism is used by copying best individuals without modifications into the new generation.

2.2 Embryogeny Model based on Cell Chemistry

Introduction Another way to develop the phenotype is to proceed with a recursive process of rewriting, which starts from a single egg cell to produce the mature organism. Among these Artificial Embryogeny (AE) systems [4], there are two main approaches.

The first is aimed at the evolution of a grammar which is repeatedly applied to the phenotype. Examples include the Matrix Rewriting scheme [12], the Cellular Encoding [10], Edge Encoding [13] and the GenRe system [14].

The second evolves the regulatory system of a cell with its metabolism and its ability to duplicate. Ontogeny results of the emergent interaction of neighboring cells and the chemical concentrations in the environment.

The model used in this paper belongs to this second category, and is an extension of the one presented in [6]. An extensive description on the model can be found in [5].

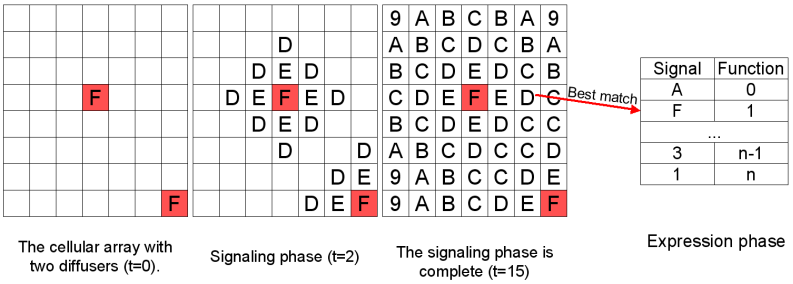


Fig. 1. The arrays on the left are snapshots of the signalling phase with one type of signal and two diffusers (gray cells) at the start of the signalling phase, after two time steps, and when the signalling is complete. The number inside the cells indicates the intensity of the signal in hexadecimal. The expression table used in the expression phase is shown on the right. The signal D matches the second entry of the table with signal F (smallest Hamming distance), thus expressing function F_1.

Description Phenotypes are developed starting from a single egg (zygote) placed in the center of a fixed size 2D grid. Morphogenesis proceeds in discrete developmental steps, during which the growth program is executed for each cell, one cell at a time.

Cells are characterized by internal and external variables. Two internal variables (cell type and internal chemical concentration) define the cell state and move with it, while the external one belongs to the environment and follows a simple conservative diffusion law.

At each developmental step, any existing cell can release a chemical to the environment, change its own type, alter its internal metabolism and produce new cells in the cardinal directions North, West, South and East.

The growth program is governed by a feedforward Artificial Neural Networks (Morphers) without hidden layers. Each Morpher is specified by 144 genes (floating values), one for each of the 8 inputs, 16 outputs and bias weights (see figure 2).

Ontogeny is governed by multiple morphers, each one defining an Embryonic Stage which spans one or more developmental steps. Stages are introduced incrementally, those controlling earlier developmental steps being evolved first and only the last one undergoing evolution (please refer to [5] for the full details of the model). This system has the advantage of increasingly adding resolution to promising areas of the search space while excluding the others, also reducing pleiotropy among different developmental steps. New stages are introduced if the performance did not increase for the last 100 generations.

The population is composed of 400 individuals, the 100 best individuals survive and reproduce. Crossover is set at 10%. All the offspring undergo mutation: each of the weights of the evolving Morpher being changed with a .01 probability by adding Gaussian noise with .035 variance.

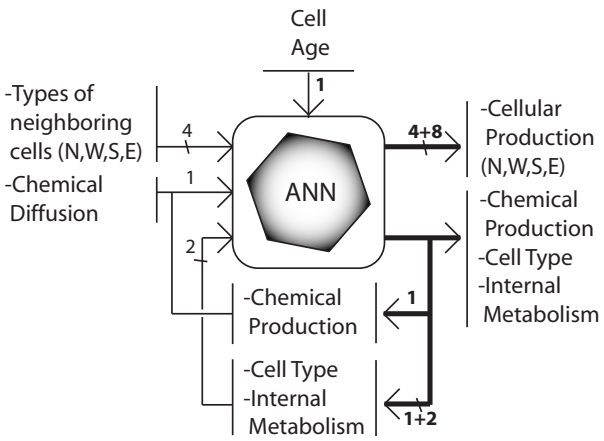


Fig. 2. The growth program (Morpher) input and output lines with their respective sizes. Each line is a floating point value $\in [-1, 1]$. The Morpher is implemented by a feedforward ANN, even though each cell internal variables (cell type and metabolism), implement a direct feedback pathway. Chemical production and diffusion offer a channel for inter-cell communication.

3 Evolution of patterns and scalability

The evolutionary task consists in evolving phenotypes resembling specific 2D patterns of increasing size. This type of problem has been selected in order to simplify the analysis of the results and avoid that the developmental models might benefit from embedded "*tricks*", which will not be applicable in other settings.

The targets are 8x8, 12x12, 16x16, 32x32, 64x64, 96x96 and 128x128 multi-cellular arrays. Each cell can take one of four possible types (colours). Two different typologies of targets are considered. The first one is a more regular 'Norwegian flag' pattern (figure 3) which presents a high degree of symmetry that should be exploitable by developmental systems. The latter, is a very complex pattern generated from a Cellular Automata using Wolfram's rule 90 and starting from random initial conditions (figure 4). Wolfram's rule 90 has been selected because it steadily produces patterns of high complexity, which are supposedly very difficult targets of developmental systems. In the case of direct encoding, the target patterns have equivalent difficulty.

Fitness is proportional to the resemblance of the individual to the target. In order to avoid premature convergence, individuals with rare phenotypic traits (pixels) are rewarded (please refer to [5] for further details).

The experiments were conducted 20 times for each target size. The population is composed of 400 individuals, undergoing elitism selection for 2000 generations. Model specific GA parameters are listed in section 2. For direct coding, the GA parameters are 10% single point crossover, mutation rate of 0.5% per gene, and each gene represents one of the 4 possible colors.

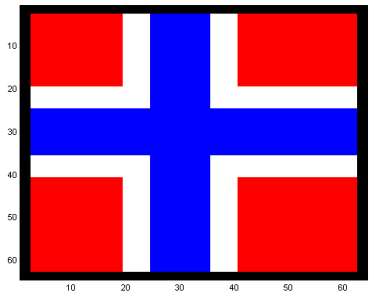


Fig. 3. Norwegian Flag Target (64x64).

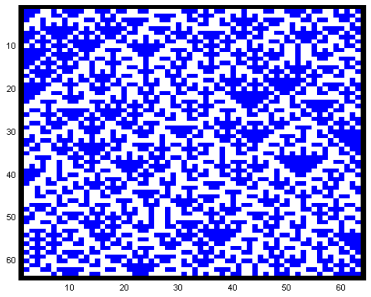


Fig. 4. Target generated using a cellular-automata with rule 90 of Wolfram starting from a random initial line (64x64).

The genotype dimension for the various encodings and target sizes are listed in table 1. The size of the genetic code with the **MS** scales with the logarithm of the size of the array because the number of bits used to encode the position of the diffusers depends on the size of the array. The size of the genetic code using the embryogeny model remains constant because the morpher neural network relies only on the state of immediate neighboring cells to update the state of the current cell and hence needs no information about the size of the array. Size of direct encoding scales with the size of the array.

Encoding	Search space by target dimension						
	8x8	12x12	16x16	32x32	64x64	96x96	128x128
Direct coding	64	144	256	1024	4096	9216	16384
MS	192	224	224	256	288	320	320
AES	144	144	144	144	144	144	144

Table 1. Search space size of the 3 encoding methods presented for each target size. Genes, in the Direct Encoding determine the color at a given position, in the Indirect Encodings regulate the ontogeny of the phenotype. In the **MS** each gene is a bit, in the Embryogeny model is a floating point number in the range $[-1, 1]$.

Scalability is shown in figure 5 for the Norwegian flag and the CA-generated pattern. Direct encoding steadily reaches 100% fitness for arrays up to size 32x32. For larger targets, the explosion of the search space limits the overall performance.

Both development approaches perform similarly for small target sizes where they tend to get high fitness scores. Larger targets show a reduced performance which tends to stabilize around a certain level. In the case of the Norwegian flag, this level is determined by the complexity of the target pattern, which is constant

with its size. In the case of the CA-generated target, complexity increases with size and solutions tend to exploit more the spatial frequency of the colours than their exact position.

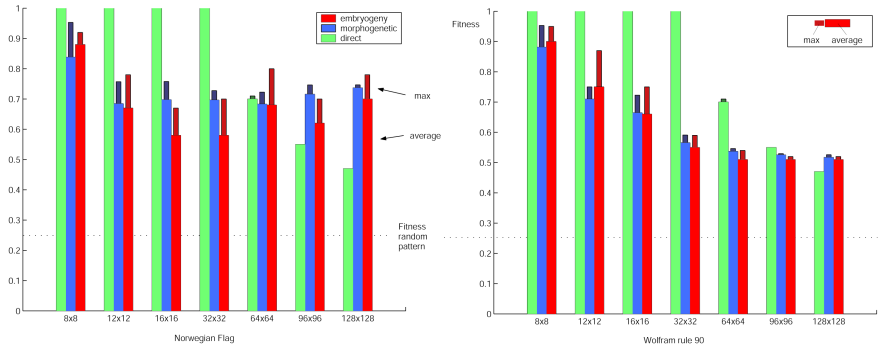


Fig. 5. Scalability for the morphogenetic system (**MS**), embryonic model (**AES**) and direct encoding on evolution of the Norwegian flag and CA-generated pattern.

Figure 6 shows the best evolved Solutions for 64x64 targets with the different encoding schemes. Notice that the solutions generated by development systems show artifacts, due to their decoding scheme (diamond-shaped patterns for the **MS** and regular repetitions for the **AES**). On the other hand, direct encoding exhibits Salt and Pepper noise.

4 Robustness

Natural organisms exhibit recovery capabilities, for example in case of injuries. In this section, we explore how these models behave when subjected to faults.

In order to have a meaningful deterioration mechanism for both developmental models, we consider here transient events which damage the state of the cell (e.g. by means of radiation corrupting memory elements). As development continues to operate normally, cell functionality could be recovered. Notice that individuals were not selected for their fault resistance.

In the case of the **MS**, faults modify the chemical content of a cell[1]. For the embryogeny model, faults kill selected cells, while for direct encoding they alter their colour.

Robustness is tested on the best evolved phenotype of the Norwegian flag on the 64x64 array. This pattern and size has been selected because the fitness of the three genetic encodings is very similar and higher than the trivial solution

[1] It is assumed that no faults occur in the expression table, since in any case it can be recovered from neighboring cells with a majority voting scheme.

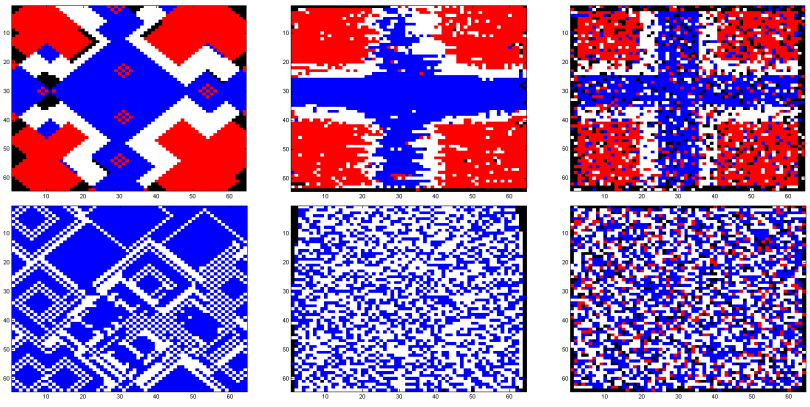


Fig. 6. Best evolved 64x64 solutions using, from left to right, **MS**, **AES** and direct coding. Norwegian flag above, CA-generated pattern below. Please refer to figures 3 and 4 for the actual targets.

consisting of exploiting only the frequency of colours as is the case with the CA-generated target. The damage rate (percentage of faulty cells) is varied between 0% and 100%. The damage process is repeated 100 times for each damage rate.

Figure 7 illustrates the results. While direct encoding is subject to a linear decrease in fitness, both developmental systems show a superior resistance to faults.

The **MS** benefits from the fact that chemical concentrations vary with continuity, and can be reconstructed with little effort. Also, evolution assigned the most frequent colour in the target to the default cell type, which explains the fitness value with 100% faults.

In the case of the **AES**, fault recovery is a byproduct of ontogeny. These results are in support to what was previously observed in [6, 5].

5 Conclusions

We have tested the scalability of two developmental and one direct encoding strategy on a minimal task involving the evolution of specific target phenotypes.

Results show that the selected task, which was intended to be favorable to the direct encoding scheme, is easily solvable by the latter only for reasonably small target sizes. In these cases direct encoding greatly outperforms the developmental systems. Direct encoding benefits from the fact that each gene contributes independently to the fitness and therefore its landscape is both unimodal and non-deceptive.

On the other side, developmental systems suffer from the pleiotropy introduced by gene reuse. Also, as there is a single optimal solution, this utterly

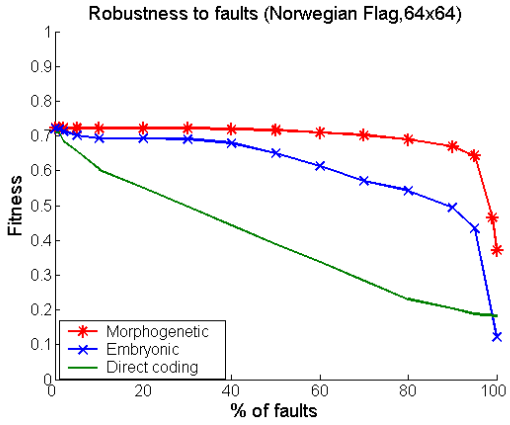


Fig. 7. Robustness of the **MS**, the **AES** and the direct coding on the Norwegian flag (size 64x64). Average over 100 tests. The fitness with 0% faults, is the one of the evolved solutions (.72 for development systems, .71 for direct encoding). Phenotypes with similar fitness scores were selected for better comparison. Fault recovery was not selected for.

complicates the evolutionary task, since it is not guaranteed to fall within the space of expressible phenotypes.

In any case, with bigger targets, development systems can take advantage of their reduced search spaces, and this is reflected in their performance levels starting at the 64x64 Norwegian flag and 96x96 CA-generated targets.

Developmental systems seem 'smarter' at finding exploitable regularities in the targets, such as shape and most frequent cell types. On the other side, this is impossible for direct encodings, so that errors in the evolved phenotypes must take the shape of high frequency noise. This gives a different 'psychological' perception of incomplete phenotypes and seems to affect performance for larger targets.

Finally, both systems behave very well against phenotypic faults and are capable of recovering from significant amount of damage, even if the tested individuals were not selected for this characteristic.

As a last remark, we want to stress the lack of standard tasks usable for benchmarking developmental systems. In [4] the authors suggest 4 different tasks: evolution of pure symmetry, of specific shapes, of specific connectivity patterns and of a simple controller.

We believe that some of these, albeit interesting to demonstrate capabilities of a model in principle, leave doubts about the generality of the results. For example, the evolution of a controller is a task that imposes complex fitness landscapes with usually many optimal and suboptimal solutions, and therefore is difficult to analyse in relation to system evolvability.

On the other side, testing a developmental model against targets of various phenotypic complexity[2], from pure symmetry to total lack of it, may offer a good indication of the system strengths and weaknesses in more general settings.

[2] possibly calculating phenotypic complexity as its compressibility with standard algorithms

6 Acknowledgments

Daniel Roggen is funded by the Future and Emerging Technologies programme (IST-FET) of the European Community, under grant IST-2000-28027 (POETIC). The information provided is the sole responsibility of the authors and does not reflect the Community's opinion. The Community is not responsible for any use that might be made of data appearing in this publication. The Swiss participants to this project are funded by the Swiss government grant 00.0529-1.

References

1. Lehre, P., Haddow, P.C.: Developmental mappings and phenotypic complexity. Proceeding of CEC 2003, 62–68 (2003)
2. Tyrrell, A.M., Sanchez, E., Floreano, D., Tempesti, G., Mange, D., Moreno, J.M., Rosenberg, J., Villa, A.: POEtic Tissue: An Integrated Architecture for Bio-Inspired Hardware. In Tyrrell, A.M., et al., eds.: Proc. of the 5th Int. Conf. on Evolvable Systems (ICES 2003), Berlin, Springer (2003) 129–140
3. Roggen, D., Floreano, D., Mattiussi, C.: A Morphogenetic Evolutionary System: Phylogenesis of the POEtic Tissue. In Tyrrell, A.M., et al., eds.: Proc. of the 5th Int. Conf. on Evolvable Systems (ICES 2003), Berlin, Springer (2003) 153–164
4. Stanley, K., Miikulainen, R.: A taxonomy for artificial embryogeny. Artificial Life 9(2):93–130 (2003)
5. Federici, D.: Using embryonic stages to increase the evolvability of development. to appear in proceedings of WORLDS workshop at GECCO 2004 (2004)
6. Miller, J.: Evolving developmental programs for adaptation, morphogenesys, and self-repair. Proceeding of ECAL 2003, 256–265 (2003)
7. Coen, E.: The art of genes. Oxford University Press, New York (1999)
8. Haddow, P.C., Tufte, G., van Remortel, P.: Shrinking the Genotype: L-systems for EHW? In Liu, Y., et al., eds.: Proc. of the 4th Int. Conf. on Evolvable Systems (ICES 2001), Berlin, Springer (2001) 128–139
9. Eggenberger, P.: Cell interactions as a control tool of developmental processes for evolutionary robotics. In Maes, P., Mataric, M.J., Meyer, J.A., Pollack, J., Wilson, S.W., eds.: From Animals to Animats 4: Proceedings of the Fourth International Conference on Simulation of Adaptive Behavior, Cambridge, MA, MIT Press-Bradford Books (1996) 440–448
10. Gruau, F.: Automatic definition of modular neural networks. Adaptive Behavior **3** (1994) 151–183
11. Kumar, S., Bentley, P.J.: Biologically inspired evolutionary development. In Tyrrell, A.M., et al., eds.: Proc. of the 5th Int. Conf. on Evolvable Systems (ICES 2003), Berlin, Springer (2003) 57–68
12. Kitano, H.: Designing neural networks using genetic algorithms with graph generation system. Complex Systems, 4(4):461–476 (1990)
13. Luke, S., Spector, L.: Evolving graphs and networks with edge encoding: Preliminary report. In Koza, J.R., ed.: Late Breaking Papers at the Genetic Programming 1996 Conference. (1996) 117–124
14. Hornby, G.S., Pollack, J.B.: The advantages of generative grammatical encodings for physical design. In: Proceedings of CEC 2001, 600–607. (2001)

G Evolution and development of a multi-cellular organism: scalability, resilience and neutral complexification.

authors: Diego Federici and Keith Downing. technical report: submitted to Artificial Life Journal

Abstract:

To increase the evolvability of larger search spaces, several indirect encoding strategies have been proposed. Among these, multicellular developmental systems are believed to offer great potential for the evolution of general, scalable and self-repairing organisms. In this paper we reinforce this view, presenting the results achieved by such a model and comparing it against direct encoding. Extra effort has been made to make this comparison both general and meaningful.
Embryonal stages, a generic method showing increased evolvability and which can be applied to any developmental model, are introduced. Development with embryonal stages implement what we refer to as direct 'Neutral Complexification': a direct genotype complexification mechanisms by neutral duplications of expressed genes.
Results show that, even for high complexity evolutionary targets, the developmental model proves more scalable. The model also shows emergent self-repair, which is used to produce highly resilient organisms.

Objective:

Finalize the analysis of the DES system (Development with Embryonal Stages), focusing on the advantage provided by the embryonal stages, the role of various parameters, its scalability, and its self-healing capabilities.

Conclusions:

Results highlight the viability of the proposed method. Further work should focus on embedding functionality in the evolved organisms.

Evolution and Development of a Multi-Cellular Organism: Scalability, Resilience and Neutral Complexification

Diego Federici* and Keith Downing
Norwegian University of Science and Technology
Department of computer and information science
N-7491 Trondheim, Norway
Email: {federici | keithd} @idi.ntnu.no

* corresponding author
Phone: +47 73551018

February 15, 2005

keywords: genetic algorithms, development, scalability, fault tolerance, neutral complexification

Abstract

To increase the evolvability of larger search spaces, several indirect encoding strategies have been proposed. Among these, multicellular developmental systems are believed to offer great potential for the evolution of general, scalable and self-repairing organisms. In this paper we reinforce this view, presenting the results achieved by such a model and comparing it against direct encoding. Extra effort has been made to make this comparison both general and meaningful.

Embryonal stages, a generic method showing increased evolvability and which can be applied to any developmental model, are introduced. Development with embryonal stages implement what we refer to as direct 'Neutral Complexification': a direct genotype complexification mechanisms by neutral duplications of expressed genes.

Results show that, even for high complexity evolutionary targets, the developmental model proves more scalable. The model also shows emergent self-repair, which is used to produce highly resilient organisms.

1 Introduction

Without apriori specific knowledge to use, it is well understood that the bigger the search space the less efficient the search will be. This makes the evolution of large phenotypes one of the most serious problems in the field of evolutionary computation (EC).

On the other hand, biological systems seem to have come to terms with it since living organisms can easily contain trillions of cells (with each cell being itself a structure of baffling complexity). These systems rely upon an artifice, the emergent process during which a single replicating cell develops into the mature organism. Inside each cell, an identical restricted set of genes interact to provide the instructions for development. Morphology emerges as initially identical cells interact with their local environment assuming specific roles.

Ontogeny is the result of the distributed process during which a relatively small genotype decompresses into a large phenotype. For example, it is estimated that there are only $30K$ genes in the human genotype ($45M$ DNA bases), while $50T$ (Tera) cells constitute a mature phenotype.

However, even if evolving direct representations of structures of similar complexity is inconceivable, is not yet well understood under which circumstances a developmental process can be beneficial to EC.

Specifically open questions regard:

- the evolvability of complex phenotypes
- the scalability of methods based on Artificial Embryogeny (AE)
- the properties of such systems

This paper addresses these issues with an empirical study of a model of multicellular development, which is compared to direct encoding. In addition we include a discussion and propose a test suited for this analysis.

Results show that:

- Development with Embryonal Stages (DES), a method based upon the biological mechanisms of gene duplication and diversification, has significant benefits for evolvability. DES implements neutral complexification and operates by preventing pleiotropy among different stages of development.
- Development results are more scalable than direct encoding for both regular and high complexity targets.

The remainder of the paper is organized as follow: section 2 contains an introduction on artificial and biological development and describes neutral complexification. Section 3 discusses the evolutionary task. Section 4 describes the adopted multicellular development model, while section 5 gives details of the evolutionary methods for Direct Encoding (DE) and Artificial Embryogeny (AE). Section 6 presents results from the simulations concerning the use of embryonal stages, scalability comparison between DE and AE, performance under

various AE settings and emergent and evolve regeneration features of the AE model. Section 7 contains the conclusions.

2 Development

Since the search space grows exponentially with the genotype size, the evolution of large phenotypes should benefit from parsimonious encodings. Among these indirect encodings, development uses a genetically encoded growth program in several recursive steps. Matrix Rewriting [20] and Cellular Encoding [14] are emblematic approaches to such a definition; both are based on grammatical rules.

Rewriting rules can be applied an arbitrary number of times, so that the genotype size should be highly independent of the phenotype size. On the other hand, development introduces a level of indirection between genotype and phenotype. The level of indirection is proportional to the amount of gene reuse, which is often linked to the number of developmental steps. For example, consider the following grammars:

$$\begin{array}{l} S \rightarrow aSa \\ S \rightarrow aSb \end{array} \quad \text{producing} \quad \begin{array}{c} aaaaa...S...aaaaa \\ aaaaa...S...bbbbb \end{array} \tag{1}$$

A single change in the rule/genotype induces a phenotypic change proportional to the number of substitution/developmental steps. The result is that the correlation between genotype and phenotype is reduced with each rule reuse. A decreasing correlation normally leads to rougher fitness landscapes, which means harder searches. Ultimately, the impact of the search space compression on the fitness landscape is what rules the performance of artificial developmental systems.

In AE, genes activated early in development are very costly to change. With a sort of Domino Effect, modifications affecting early phases of development will have huge phenotypic consequences. Big leaps in the phenotypic space go against the basic evolutionary assumption of small incremental refinements.

Interestingly, this is valid also in natural evolution. Observing the development of embryos, it is seen that the more related two species the more similar their ontogeny. It is actually possible to judge phylogenetic relatedness by looking for how long embryos of different species share a common developmental dynamic:

> It is generally observed that if a structure is evolutionary older than another, then it also appears earlier than the other in the embryo. Species which are evolutionary related typically share the early stages of embryonal development and differ in later stages. [...] If a structure was lost in an evolutionary sequence, then it is often observed that said structure is first created in the embryo, only to be discarded or modified in a later embryonal stage. Examples include:

> Whales, which have evolved from land mammals, don't have legs, but tiny remnant leg bones lie buried deep in their bodies. During embryonal development, leg extremities first occur, then recede. Similarly, whale embryos (like all mammal embryos) have hair at one stage, but lose most of it later. All land vertebrates, which have evolved from fish, show gill pouches at one stage of their embryonal development. The common ancestor of humans and monkeys had a tail, and human embryos also have a tail at one point; it later recedes to form the coccyx. The swim bladder in fish presumably evolved from a sac connected to the gut, allowing the fish to gulp air. In most modern fish, this connection to the gut has disappeared. In the embryonal development of these fish, the swim bladder originates as an out pocketing of the gut, and the connection to the gut later disappears. Wikipedia [34]

This conservative nature of ontogeny is a direct consequence of the aforementioned 'Domino Effect' since redesign is harder than incremental changes. Therefore, mutations causing big phenotypic effects are counterproductive because, with good approximation, they never produce better individuals.

Another issue relates to the regularity of the phenotypes that AE produces. Development is de facto a decompression of the genotype and since compression is generally higher for regular targets, a serious question is how much these methods are viable for the evolution of targets of high complexity.

Hints in this direction, also come from a study [21] on Matrix Rewriting [20], showing how the genotype-phenotype correlation decreases with the complexity of the phenotype. Again, low correlation decreases the chances of evolution by small incremental steps.

Micro-array analysis and gene sequencing are providing new data to understand how molecular evolution took place in organic systems. It is now clear that the most genetic innovations were originated by Gene Duplication [27] (see also Figure 1):

- Genotypes are not of fixed size. Notably, entire groups of genes can be duplicated (figure 1:B). Duplicated genes may not alter development at first since their expression is controlled by position invariant promoters and regulators. Evidence suggests that 90% of eukaryotic genes was produced by gene duplication [31].

- Duplicates can diverge assuming different roles (figure 1:C–E). Phenotypic traits that were linked by the expression of the same gene, can now evolve independently. This phase is often referred to as complexification, and can take two different forms: neofunctionalization (figure 1:C) and subfunctionalization (figure 1:D) [36].

The steps depicted in Figure 1 can be translated to the following steps of a

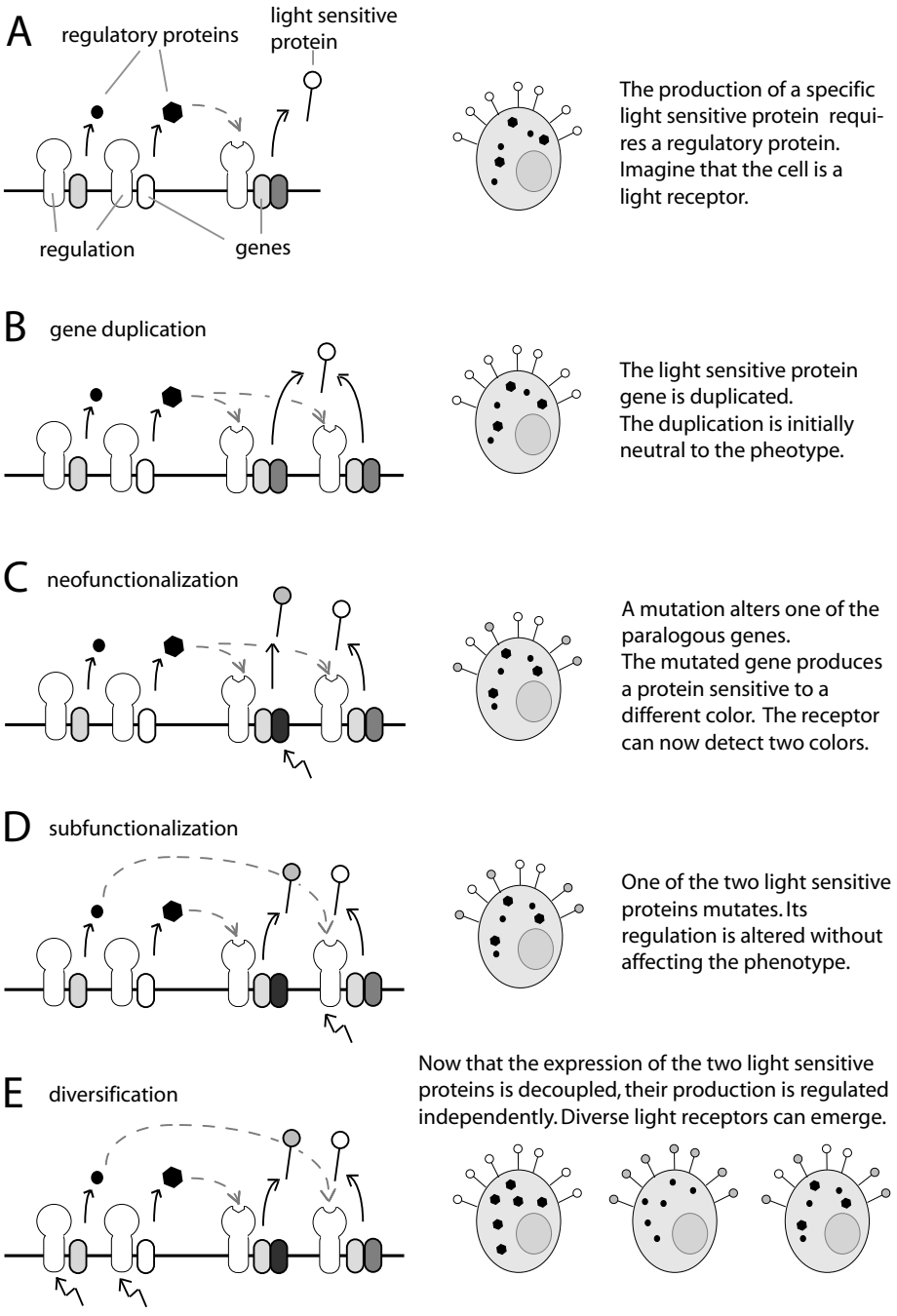


Figure 1: An example of phenotypic complexification by gene duplication. In this case gene duplication is initially neutral to the development of the phenotype. The genotypic redundancy introduced can eventually be exploited to increase the complexity of the phenotype.

grammar evolution:

	genotype	phenotype	type	
A)	$\{S \rightarrow AS\|\epsilon;\ \ A \rightarrow a\}$	aaaa...aa		
B)	$\{S \rightarrow AS\|\epsilon;\ \ A \rightarrow a\|a\}$	aaaa...aa	duplication	(2)
C)	$\{S \rightarrow AS\|\epsilon;\ \ A \rightarrow a\|b\}$	abba...ab	neofunc.	
D)	$\{S \rightarrow AS\|BS\|\epsilon;\ A \rightarrow a;\ B \rightarrow b\}$	abba...ab	subfunc.	

Neofunctionalization regards duplicated (paralogous) genes which assume a novel functionality. Example include the color specific opsins found in the light receptors of the retina and the Hox genes responsible for the vertebrate body structure. Subfunctionalization is the possibility that regulation of paralogous genes gets changed while the genes retain the same function. An example is found in the Zebra-Fish, where the genes engrail1 and engrail1b are expressed in different parts of the body, pectoral and hindbrain respectively, while they both produce the same protein. The engrail1b gene, originated from a duplication of engrail1, has diverged in its regulation while conserving the same functionality. It is possible to imagine that in the future they could eventually assume different specific functions.

The case of the engrail1 genes, appears as a Neutral Complexification (NC) of the genotype: an increase of genetic material which is functional (expressed) but neutral to the phenotype.

NC makes possible an increase of sophistication with small incremental steps consisting of neutral duplication events followed by divergence of paralogous genes.

For evolution, NC allows an initial fast exploration of a restricted search space, followed by an enlargement of the genotype which can achieve higher levels of specialization. It is possible to imagine that the two phases will repeat over and over, as individuals compete for higher values of fitness.

The great advantage of this incremental approach is that it induces an ordering of the search space, leading to an incremental increase of complexity from the initial genotypes. In this simple to complex re-ordering of the solution space lies its promise of higher performance.

2.1 implementing complexification

There are a number of genotype phenotype mappings that implement, either directly or indirectly, complexification.

One approach is based on artificial Gene Regulatory Networks (aGRN) with variable length genotypes [23, 5]. Genes are identified by a promoter site and therefore can occur anywhere in the genotype. Additionally, gene's activation regulate (and is regulated by) the expression of other genes. Activated genes are then used to define the morphology of the evolved organisms. Since gene functionality is position invariant, gene duplication can be achieved simply by replicating genotype sub-strings, while divergence is obtained by traditional mutation operators.

Notice that complexification is not generally possible for fixed-size string based aGRN models, such as those presented in [11, 3, 18, 8].

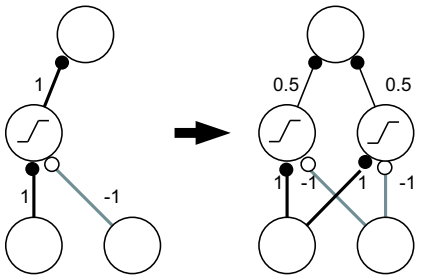
The advantage of this indirect implementation of NC is that it can achieve diversification of both the phenotypic traits (as in figure 1:3-4) and the regulatory network (by duplicating regulatory genes).

On the other hand, the approximate matching required for the identification of activated genes is quite expensive and grows linearly with the size of the genome and the number of regulatory genes/proteins[1]. This might impose a limit to the scalability of multicellular organisms, since the determination of each cell behaviour requires a separate activation of the aGRN.

Also, in Evolutionary Hardware there is a nascent interest for multicellular development [32, 25, 33], but given the state of technology, these models do not appear suited for hardware implementation.

Finally, current models do not protect against mutations affecting early phases of development, which we believe reduce system evolvability. This however could be easily avoided by using a mutation operator which takes into consideration the age of each gene.

A second, more abstract, approach is to increase sophistication by direct NC steps, adding specific active elements which do not alter functionality.



Depiction of Neutral Complexification for an artificial neural network. The network on the left is augmented horizontally by adding an active element without altering its functionality. Changes to the weights of the new network can produce more sophisticated functions than for the old one. Without altering functionality, complexification can be achieved incrementally. This mechanism, among others, is presented in the NEAT system [30]

Figure 2: Neutral Complexification

As far as we know, apart form the mechanism presented in this paper, there is no developmental system that implements direct NC. On the other hand, the NEAT system [30], a variable length direct encoding for Artificial Neural Networks (ANN), allows new neurons to be added almost neutrally with minimal changes to functionality, producing complexification by small incremental evolutionary steps. Also in this case, system evolvability, can benefit from the smaller-to-bigger reordering of the search space. Figure 2 gives an example of how such mechanism can be implemented for ANNs. Unfortunately, it is not

[1]The computational complexity for the approximate matching of a genome of length n with a single substring of length m is $O(mn)$

always possible to achieve NC, especially when non-linearities are present.

2.2 Development with Embryonal Stages

The regulatory system controls gene expression over two orthogonal dimensions: time and space. Development with Embryonal Stages (DES) implements a direct mechanism of Neutral Complexification (NC) for the temporal dimension.

As development spans over several consecutive steps, the idea is to start evolution with a single growth program (chromosome) which controls all the steps. As evolution proceeds, a new chromosome can be added by gene duplication.

The developmental steps are therefore partitioned into two groups. The first, controlling the initial steps of embryogenesis, is associated with the old chromosome. The latter, completing growth, is associated with the new, identical, duplicated chromosome.

Being exact copies, new chromosomes do not alter development, and are therefore neutral. But eventual mutations can independently affect each paralogous gene.

To avoid the Domino Effect caused by mutation altering early phases of development, the original chromosome can have a lower mutation rate or be arbitrarily excluded from evolutionary operators (see Section 6.2). In the latter case, only the new chromosome, which completes the maturation process, is allowed to change.

Likewise, new chromosomes can be added one by one, each one controlling the partition of the last development steps (see Figure 3).

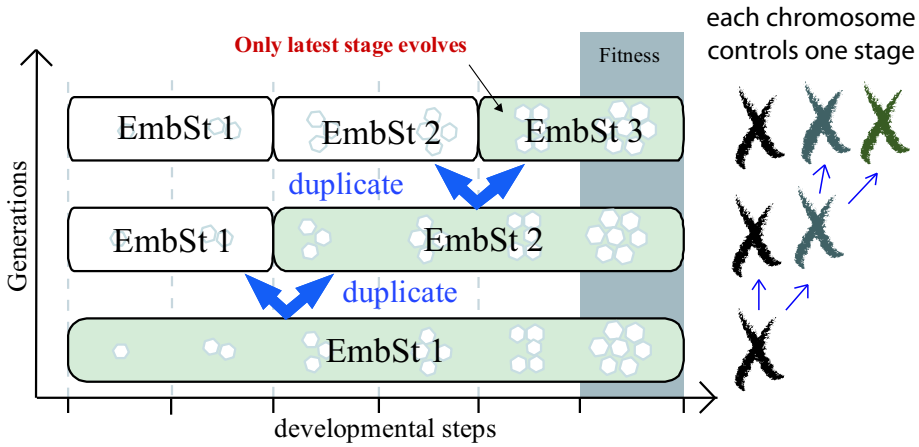


Figure 3: Embryonal stages. New chromosome duplicates are added incrementally without altering development. Evolution can thereafter alter only the growth program/chromosome controlling the latest steps of development. Without changing its size, new embryonal stage alters the search space by adding resolution around the current genotype and locking distal regions.

When only the latest embryonal stage is allowed to change, the advantage of DES is that each new stage restricts the evolutionary task to the optimization of fewer developmental steps. The new chromosome must take care of the maturation of an already partially developed phenotype. This new starting phenotype, as opposed to the zygote, is the result of the original chromosome evolution and provides a flying start for the second stage.

New stages do not modify the size of search space, but by controlling fewer developmental steps, they increase the resolution around the area represented by the current genome. Overall, the effect is an increase in genotype-phenotype correlation which leads to higher evolvability (see section 6.1).

2.3 Related development models

Several indirected encoding schemes have been proposed. All these 'Artificial Embryogeny' (AE, [29]) methods define the phenotype with a mapping which allows recursive gene reuse.

It is possible to distinguish two major evolutionary approaches to development. There are grammar-based approaches in which the genotype defines the substitution rules which are repeatedly applied to the phenotype. Examples include the Matrix Rewriting scheme [20], the Cellular Encoding [14], and Edge Encoding [22].

The major advantage of these methods is that compact grammar rules can generate a high number of organized patterns. For example, L-systems can easily produce interesting structures for locomotion. On the other hand, our target may contain many different phenotypic traits, each requiring a different production rule and symbol. Therefore the space of possible grammars is bound to grow with the complexity of the phenotype.

Some models include additional contextual information in each rule definition [17, 16], so that phenotypic traits variations can be generated. Also, it is possible to implicitly define the grammar by means of an aGRN [5] and use the accumulated concentrations of simulated chemicals to modulate the characteristics of morphological constituents.

In this direction, and inspired by Cellular Automata (CA), a second approach is to evolve the rules by which cells alter their metabolism and duplicate. Cells are usually capable of sensing the presence of neighboring cells [9], releasing chemicals which diffuse in simulated 2D or 3D environments [25], and moving and growing selective connections to neighboring cells [6].

Closely related to the one presented in this paper, Bentley and Kumar proposed a model which develops tiling patterns [4]. Ontogeny starts from a single cell which duplicates following rules expressed in the genotype. Rule preconditions are regular expressions matching the local North-West-South-East neighborhood (NWSE) of each cell and their absolute position.

Miller extended the previous model and evolved specific patterns [25]. He allowed 4 different cell types (colors) and a chemical undergoing diffusion. Cells based their growth program on their type and the types and chemical concentrations from the 8 neighboring cells. The growth program itself was encoded in

a boolean network that was evolved with the Cartesian Genetic Programming.

Miller also analyzed the behaviour of the evolved phenotypes when subjected to severe mutilations. Phenotypes were shown to regrow the missing parts achieving a striking resemblance to the target. The self repair feature is very interesting since it was not selected for during evolution. A similar resistance to damage was also reported in [28].

An important feature common to these last AE models, is that their growth program is based only on local variables. The localization of the inputs is fundamental to achieve a distributed control of development, which allows practical hardware implementation, failure recovery and, since the size of the program does not change with the size of the phenotype, for scalability issues.

3 The evolutionary task

When looking at system evolvability, it appears that there is a trade off between the combinatorial gain achieved by searching in a restricted genotypic space and hindrances introduced by gene reuse.

The fact that gene reuse constitutes an advantage when searching for regular patterns is quite established. For example, in [16] a generative (grammar-based) and non-generative systems where tested in the evolution of a program defining the construction of a 3D table. The generative mapping achieved much higher performance.

Similarly, in [4], the task was to evolve tessellating patterns. Individuals growing from a single cell to the mature organism showed higher scalability when compared to direct encoding. On the other hand, the generated solutions were always very regular.

A valid solution to this problem is a rectangle covering half of the allowed growth area. Since the zygote is placed in the center, such rectangle can be produced by growing in three over the four possible directions. The model presented in this paper, for example, can often solve the tessellation problem with the initial population of randomly generated individuals.

The problem of the complexity achievable by developed phenotypes is fundamental since high complexity targets are more difficult to refine incrementally and therefore to evolve [21].

Many authors have tested their system for developing Artificial Neural Networks (ANN). ANN are a natural choice for several reasons. First their proper set up is a central issue of AI. In addition, the number of synapses grows combinatorially with the size and interconnectedness of the network. Finally, the organization of the brain shows repeated structures that should favor gene reuse.

Even if the problem is tempting and surely deserves attention, we believe that it could prove misleading for initial investigations. In fact, there are several network configurations with equivalent functionality. Searching in a restricted search space, AE could stumble on a good solution at random. This is particularly true in the case of simple neuro-controllers, where often the problem is

as simple as finding the proper connection from one point to another (e.g. from an IR sensor to the motor output).

In [25], in order to show the capabilities of a cell-based developmental model, a 9x12 French flag pattern was evolved. The reason for this choice is that the uniqueness of the desired shape guaranteed that the problem was hard to solve. In fact, it is not even possible to know apriori if a sufficient growth program exists.

The advantages of this test are plentiful:

1) By presenting the phenotypes as 2D pictures, the bias of the development system often emerges as repeated sub-patterns. Difficult to represent mathematically, such artifacts are easily observable.

2) The generality of the model can be ascertained by testing targets of various shapes and complexity.

3) The number of colors of the target and its dimension clearly specifies the size of the solution space.

4) For direct encoding this is one of the easiest problems to solve. With each pixel of the target pattern represented by a gene, not only every target is expressible, but this is equivalent to the One-Max problem[2], which is totally decomposable, non deceiving and has a strictly monotonic fitness landscape.

5) Fitness is an absolute measure of the individuals quality. This is not so easy for other tasks, such as the evolution of locomotion or robot navigation.

In this paper we have tested direct encoding and the AE system in the evolution of 4-color patterns of varying size and complexity (see figure 4). The targets were chosen with different degrees of regularity, with the Circle being the most regular and the Wolfram CA-generated pattern the most complex.

To test scalability, the targets size has been set to 8x8, 16x16, 32x32, 64x64, 128x128, 256x256. The tested solution spaces therefore range from 4^{64} to 4^{65536}. As the geometrical properties of the targets are size invariant the amount of regularity should be invariant as well.

As proposed in [21], a measure of complexity can be provided by general purpose compression algorithms. In this case, it was calculated using the well known ARJ compressor [24]. In figure 4, below each target shape, the compression ratio and the size of the compressed data segment[3] are plotted for all the used target sizes. The ratio of compression can be seen as a depiction of the target complexity (ARJ-complexity), the size of the compressed data gives an indication of quantity of information needed to reproduce the pattern.

Apart for the CA-generated pattern, the ARJ-complexity decreases geometrically with size. This is due to the fact that, while regularity is preserved

[2]where the fitness is the sum all bits in the genotype which are set to 1

[3]The ARJ headers, containing file name and other technicalities, are stripped for these computations

for all shapes, the size increases quadratically. For the high complexity CA-generated pattern, there is little regularity that can be exploited and therefore the ARJ-complexity converges to $\sim 15\%$.

4 The model of multicellular development

Organisms develop starting from a single cell to reach the mature multi-cellular organisms in a precise number of developmental steps. Cells replicate and can release simulated chemicals in intra-cellular and inter-cellular space.

Cell behaviour is governed by a growth program based on local variables, and represented by a simple Recursive Neural Network (RNN). The RNN is seen as a plausible model of the cell Gene Regulatory Network (GRN) [2, 10].

Compared to typical models of GRNs in EC [5, 18, 9, 11, 3, 23] RNNs are much faster to evaluate. Since the growth program is evaluated literally billions of times, for a multicellular organism this is a fundamental feature. Also, while the evolutionary properties of artificial GRNs are still subject to debate, neural networks are better understood.

Compared to discrete functions, such as Boolean Networks [19, 25, 9] or production rules [4], the space of continuous functions representable by RNNs allows finer tuning and richer neutral space. Neutral space is very useful to escape local optima [1, 15], which pleiotropy makes abundant in the search space.

4.1 cell growth

Phenotypes develop starting from a zygote placed in the center of a fixed size 2D grid. Morphogenesis proceeds in discrete developmental steps, during which the unique growth program is executed for each cell. The execution order is determined by position, proceeding from northeast to southwest.

Cells (see figure 5) are characterized by internal and external variables. Internal variables (metabolism) define the cell state and move with it, while external ones (chemicals) belong to the environment and follow a simple diffusion law. Diffusion is simulated by convolving the previous chemical concentrations with the following kernel:

$$\text{diffusion kernel} = \begin{vmatrix} 1/16 & 1/8 & 1/16 \\ 1/8 & 1/2 & 1/8 \\ 1/16 & 1/8 & 1/16 \end{vmatrix} \tag{3}$$

At each developmental step, existing active cells can release chemicals, change their own type, alter their internal metabolism and produce new cells. An active cell can also die or become passive. Each step, up to four new cells can be produced in any of the cardinal directions North, East, South and West (NESW). The mother cell specifies the new cell internal variables (type and metabolism) and whether they are active or passive. If necessary, existing cells are pushed

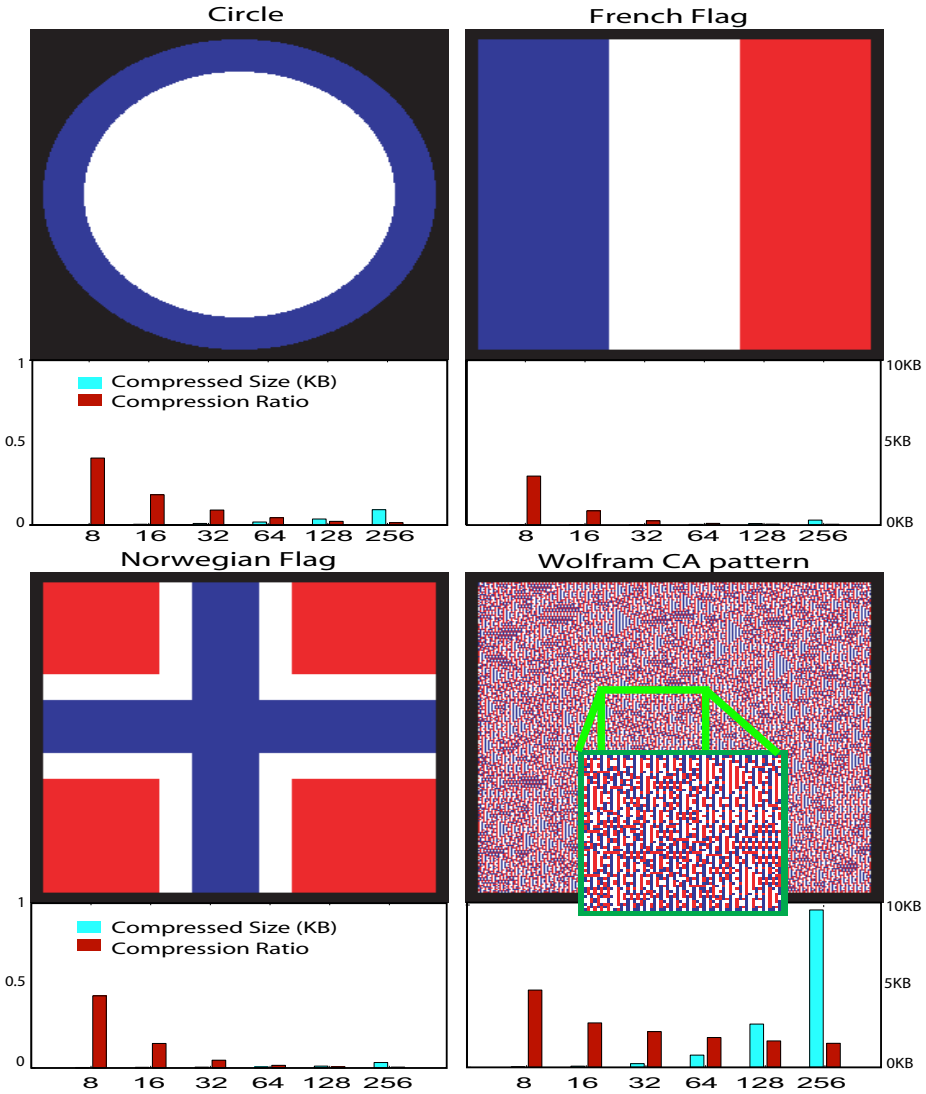


Figure 4: Shape of the evolutionary targets. A bounded circular pattern, a French flag, a Norwegian flag and a type 4 CA-generated pattern using the 3-color totalistic rule 1387 [35]. Below each image a graph reports the ARJ compression ratio (darker) and the size of the compressed data (lighter) for various target dimensions (from 8x8 to 256x256). The compression ratio can be seen as a measure of complexity, the size of compressed data as the measure of information. In the Wolfram CA pattern a zoom box is superimposed for clarity. The ARJ is a general purpose compression algorithm. Notice that each pattern also contains a frame, which also must be matched. The frame is an additional source of difficulty for the developmental system.

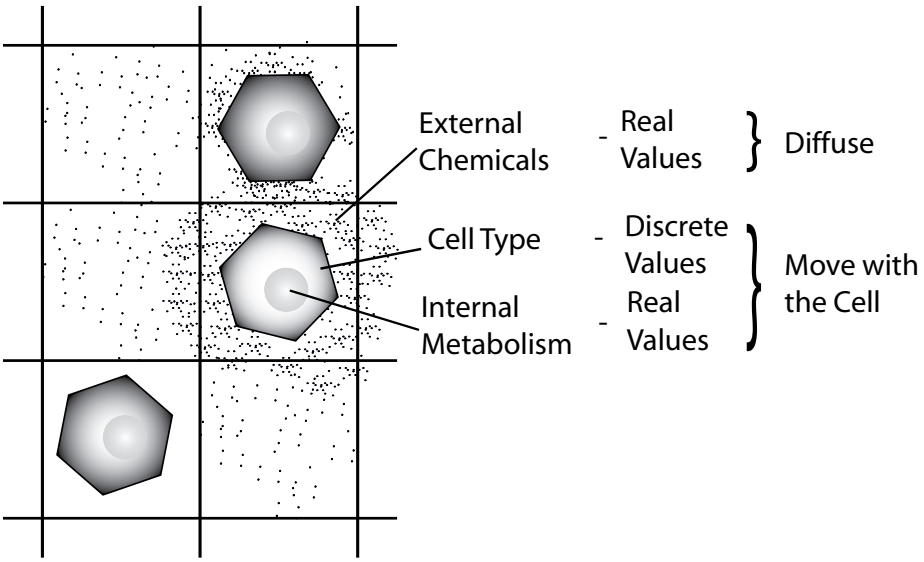


Figure 5: Description of the variables used for development. External chemicals follow a diffusion law, while internal ones move with the cell. While chemical concentrations, internal and external, take values in $[-1, 1]$, the cell type can take one of 4 discrete values, one for each phenotypic color.

sideways to create space for the new cells. When a cell is pushed outside the boundaries of the grid it is permanently lost.

Passive cells cannot release chemicals, change their state or produce new cells.

The cell behavior is governed by an Artificial Neural Network (Morpher) defined by the genotype. The genotype contains a floating point number for each synaptic weight. The Morpher receives as input the current cell internal and external variables, and the cell types of the neighboring cells in the four cardinal directions. Its output determines the new internal and external variables of the cell and, in case of replication, the internal variables of the newly generated cells. An additional local variable, the cell age, is set to 1 at birth and decays exponentially to 0.

The number of inputs and outputs to the growth program depends on the number of neighbors, cell types, external and internal chemical types (see figure 6). In most of the simulations presented, there are 4 cell types, 1 internal metabolism and 1 external chemical types. These sum up to 8 inputs (one being the bias) and 16 outputs which can take values $\in [-1, 1]$.

Chemical (production/concentration) and metabolism values are read directly to and from input and output lines. Since there is 4 possible cell types, each input/output line is quantized to four possible values $\{-1, -1/3, 1/3, 1\}$, each representing a different color in the phenotype: black, blue, red and white

respectively. When cell types are computed, the outputs take the nearest quantized value.

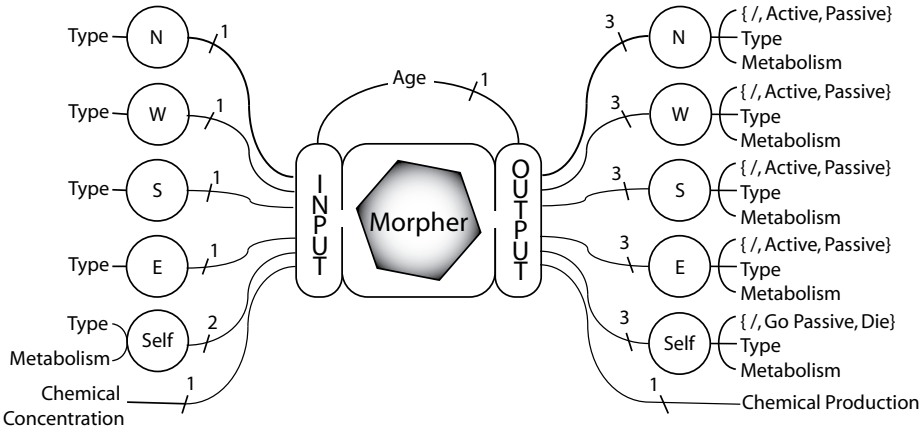


Figure 6: Inputs and outputs of the growth program, the Morpher, implemented with a feed-forward ANN. Each cell internal variables, cell's type and metabolism, implement a direct feedback pathway, while chemical production and diffusion offer a channel for longer range communication. A bias input neuron is also provided (not shown).

In picture 7, typical examples of phenotypes from random individuals are shown. The first phenotype (Random 1) is the most frequent.

5 Evolution for development and direct encoding

Every population is composed of 400 individuals. The best 1/8 survives and reproduces (elitism).

Fitness is proportional to the resemblance of an individual to the target, and is computed as shown in equation 4. For fitness computation, dead cells are assigned the default type 0 (black color)

$$\begin{aligned} \text{Fitness}(P,T) &= \left(\sum_{x,y} \text{Equals}\,(P,T,x,y) \cdot \text{TraitRarity}(x,y) \right) \;/\; ||T|| \\ \text{Equals}\,(P,T,x,y) &= \begin{cases} 0 & if \quad P(x,y) \neq T(x,y) \\ 1 & if \quad P(x,y) = T(x,y) \end{cases} \end{aligned} \tag{4}$$

where P is the phenotype, T the target, and $\text{TraitRarity}(x,y)$ is a measure of the rarity of the trait in $P(x,y)$ over the entire population.

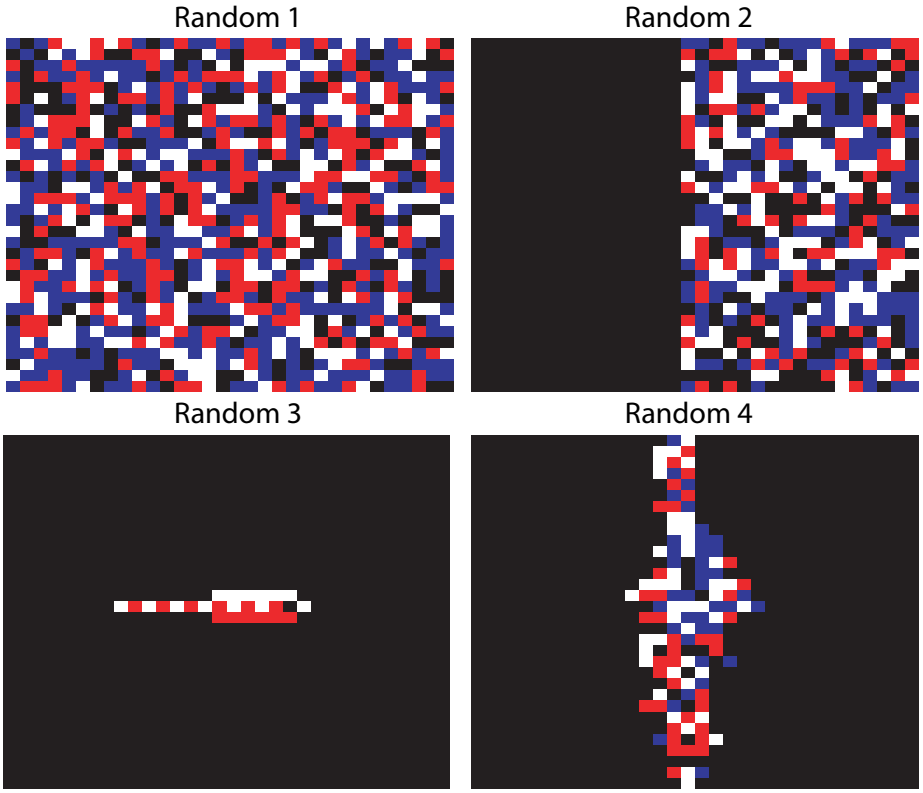


Figure 7: Typical phenotypes of first generation individuals. Dead cells have the same color of black cells.

TraitRarity is used to increase population diversity, counteracting homogenization and favoring individuals with rare characteristics. AE systems are in fact characterized by saltatory improvements (see figure 11). Under these conditions a positive innovation can increase the reproductive probability of a particular strain. TraitRarity helps reduce the crowding of a single genotype strain and is calculated as follows:
(1) The population is first ordered by fitness value before modification (in case of ties, younger individuals have priority). (2) Fitness scores are recomputed following this order, but the initial value of 1 of each trait (TraitRarity in Eq.4) decreases with use in steps of .1. Whenever a trait value goes to or below zero, it is assigned a value of 1/100.

The overall effect is that older individuals with common phenotype traits are replaced by younger, rarer individuals.

5.1 Mutation and Crossover

For direct encoding every gene (color) can mutate with a probability of 1/100 and 50% of the offspring are generated by single-point crossover.

In the case of AE, mutation takes each weight of the Morpher RNN with a P_{mut} probability and adds to it Gaussian noise with V_{mut} variance. P_{mut} and V_{mut} vary in the ranges $[1/20, 1/10]$ and $[1/30, 1/10]$ respectively. Their value is proportional to the time passed since the last increase in the top fitness score, reaching the maximum in ten generations. The reason behind these adaptive values is to increase exploration when the system begins to stagnate.

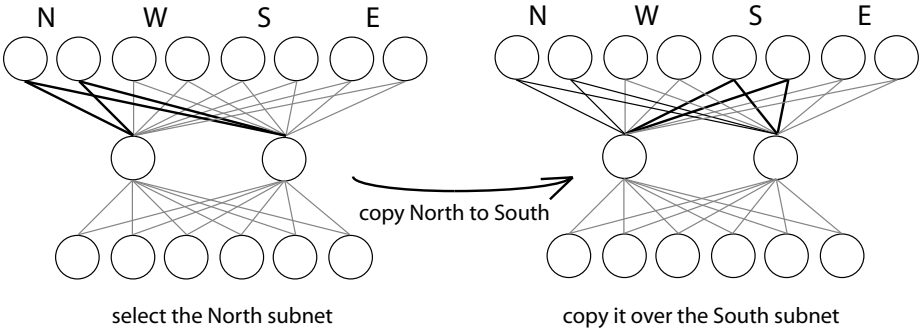


Figure 8: Symmetric mutation. The subnet responsible for the production of new cells in the north direction (on the left with a darker color) is copied over the subnet responsible for the south direction. Northward and southward cell production are now identical.

With a 1/20 probability an offspring undergoes an additional symmetric mutation (see Figure 8). The subnet responsible for the production of new cells in a chosen direction overwrites one or more of the other direction subnets. This operator should favor the evolution of phenotypes with various degrees of symmetry, but, since cells are not activated in parallel but follow an activation order, it does not produce perfect symmetrical phenotypes.

10% of the offspring are produced by crossover. If the Morpher RNN has a hidden layer, crossover shuffles the parents' hidden nodes (together with input and output weights). If no hidden nodes are present, it exchanges all the weights connected to inherited outputs units.

6 Results

The results present statistics over 20 independent evolutionary runs for each parameter setting (400 individuals and 2000 generations). Unless specified otherwise, the Morpher contains no hidden layers and each cells has 4 types, 1 internal and 1 external chemicals. Also, when more then one embryonal stage

is available, only the latest one is affected by the evolutionary operators. All other stages are fixed.

The results presented explore the following aspects of the multicellular development system:

i: Advantage of the embryonal stages
ii: Exploration strategies with embryonal stages
iii: Scalability of the growth model
iv: Performance of different Morpher configurations
v: Effects of the variables used by the growth program
vi: Emergent and evolved self-repair

The number of embryonal stages (N_{ES}) can be altered from a minimum of 1 to a maximum of one for every development step (devStep). The number of development steps (N_{dS}) has a strong influence on system evolvability: there must be enough steps to allow the emergence of good cell's configurations and not too many in order to minimize indirection (see section 2). Furthermore, the optimal N_{dS} often varies from target to target. In this paper we used the formula:

$$N_{dS} = 2 + \log_2(||\text{Target}||)$$

When a stage is available and the maximum fitness did not increase for a period of $1000/N_{dS}$ generations, a new stage is introduced. While the first stage is responsible for the ontogeny from the beginning of development. The i-th stage takes over at devStep $\big(N_{dS} - (N_{ES} - i)\big)$.

6.1 Advantage of the embryonal stages

16x16 and 32x32 targets (see figure 4) have been evolved with a single, half, and maximum number of stages (Group 0, $^1/_2$, 1 respectively). Their performance is shown in figures 9 and 10.

The average fitness scores demonstrate that embryonal stages are beneficial. Using an ANOVA test with $p = 10^{-3}$ the results always prove statistically significant for group 0. The difference between group $^1/_2$ and group 1 is only significant for the Wolfram CA targets, which are also the most complex.

With a single embryonal stage innovations require long periods of exploration and cause big phenotypic changes, more stages allows smaller and more frequent refinements (see figure 11). These results suggest that stages increase the correlation between genotype and phenotype.

6.2 Exploration strategies with embryonal stages

In most of the simulations presented in this paper, only the newest embryonal stage is modified by evolution. This strategy allows for a search space of constant size, with each stage increasing resolution around the current genotypes and excluding from search distal regions of the solution space.

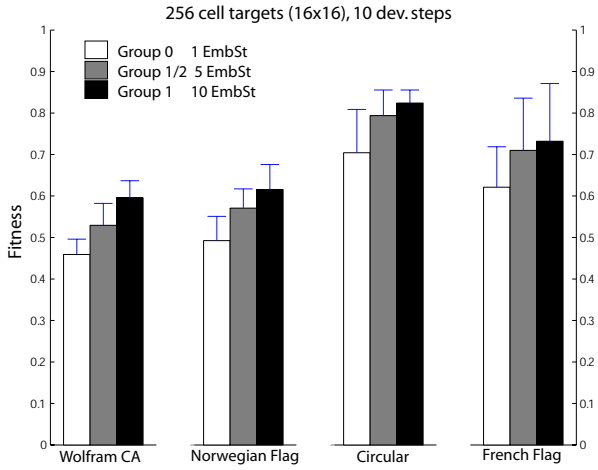


Figure 9: Effects of varying embryonal stages on performance. Mean and maximum performance from 20 runs for each of the 16x16 targets.

A premature introduction of embryonal stages restricts exploration and could affect overall performance.

To test this hypothesis, we allow every stage to be subjected to mutation. To offer a protection of older embryonal stages, the mutation variance (V_{mut} in Section 5.1) affecting the Morpher of a specific stage is reduced with age.

If the genotype currently contains N embryonal stages, the mutation variance affecting the Morpher of stage i is:

$$V_{mut}/(\text{variance decay}^{(N-i)})$$

thus, the older an embryonal stage the lower the mutation variance.

Figure 12 shows statistics from 5 different groups with various variance decay values. A value of 1 means that all stages change at the same rate.

The results show that, in some cases, locking old stages can reduce overall performance. The effect is more evident for the Norwegian flag target with variance decay values of 2 and 5. On the other hand, differences have no statistic significance ($p < 0.001$).

6.3 Scalability of the growth model

The most interesting property of developmental systems is the claim of high independence between evolvability and phenotypic size. Other papers compare the performance of development against direct encoding in specific tasks, for example [4, 16, 12]. Often though, the solution for the selected task can take advantage of particular features of the development system. This fails to highlight the trade-off between direct encoding and AE, often favoring the latter. In these cases, critics may argue that the better performance of AE derives from the exploitation of built-in implicit knowledge.

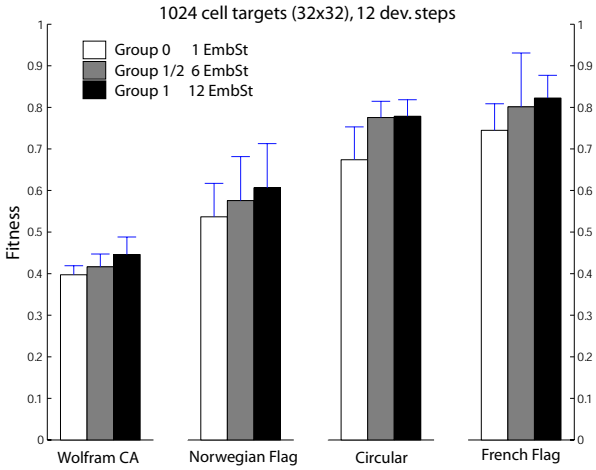


Figure 10: Effects of varying embryonal stages on performance. Mean and maximum performance from 20 runs for each of the 32x32 (right) targets.

Compared to previous work [28], we have tested more and bigger shapes. Also, the high complexity target presented in [28] contains mainly 2 colors/cell type, a property that could be exploited by minimalistic solutions.

The results for targets with 64, 256, 1024, 4096, 16384, 65536[4] cells are shown in figure 13. For direct encoding, performance is not influenced by the target shape, but for embryogeny the degrees of regularity have a great impact. The best evolved 256x256 individuals are plotted in figures 21-24.

Direct encoding steadily evolves perfect solutions for targets with up to 1024 cells. As expected, with bigger search spaces, performance decreases monotonically. This is not the case for development with embryonal stages. AE performance, after reaching a minimum for medium sized phenotypes, shows an increase and then levels off. The final fitness values appear to be proportional to the regularity of the specific target.

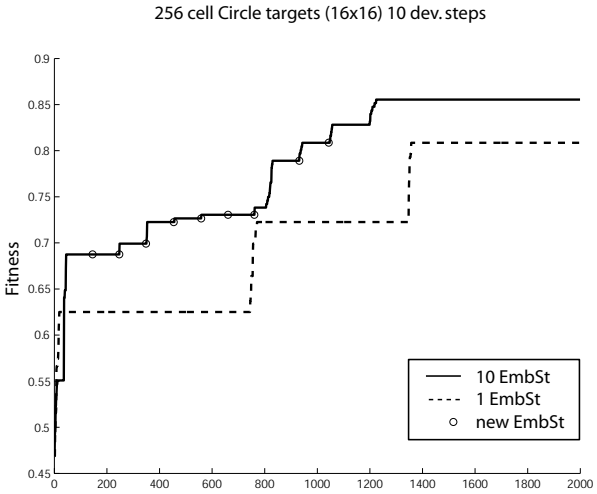
It is interesting to notice that, even the performance of the high complexity Wolfram CA patterns converges.

6.4 Performance of different Morpher configurations

It is possible to use various implementations of the growth program. The default model, a RNN with no hidden layers, has been selected for simplicity, evolvability and speed. In this section we have tested 32x32 targets with three different RNN topologies: with no hidden layers (NH), with 4 hidden nodes (H4) and 8 hidden nodes (H8). Results are plotted in figure 14.

It is possible to imagine that, given the complexity of the task, neural networks with more hidden units could better achieve an optimal growth program.

[4]for the 256x256 target, only 8 runs are available at the moment, others are still running



Fitness plots have saltatory characteristics. The introduction of new embryonal stages is frequently followed by a burst of fitness improvements. When evolving with one stage, improvements are less frequent and have bigger effects on the phenotype.

Figure 11: Maximum fitness plots of the best evolved populations for the 16x16 circular target from group 0 and group 1.

On the contrary, there is no significant difference ($P < 0.001$) among the various morpher configurations, as the higher representation flexibility is counterbalanced by bigger search spaces. These results advocate for growth program simplicity.

6.5 Effects of the variables used by the growth program

The growth program takes as input the neighbors cell types, its own internal metabolism state and the current chemical concentration at the cell's locus. To shed light on their role in development, the number of these variables can be altered.

In figure 15 populations were evolved with 0, 1 and 2 types of external chemicals. External chemicals diffuse in the environment allowing short range inter-cellular communication. Every external chemical adds one input and one output line to the growth program. Their number does not show a significant influence on evolvability.

In figure 16 populations were evolved with 0, 1 and 2 types of internal chemicals (cell metabolism). Internal chemicals implement the recurrent connections of the morpher. Every internal chemical adds one input and 5 output lines to the morpher. Also in this case, their number have no statistically significant influence on performance.

In figure 17 populations were evolved with 4 and 8 cell types. Additional types are redundant, so when 8 types are possible, every color is represented by 2 types. Additional cell types allow more information to be stored in the local neighborhood. Every additional 4 cell types requires 5 extra input and output lines. A statistically relevant performance increase is achieved only for

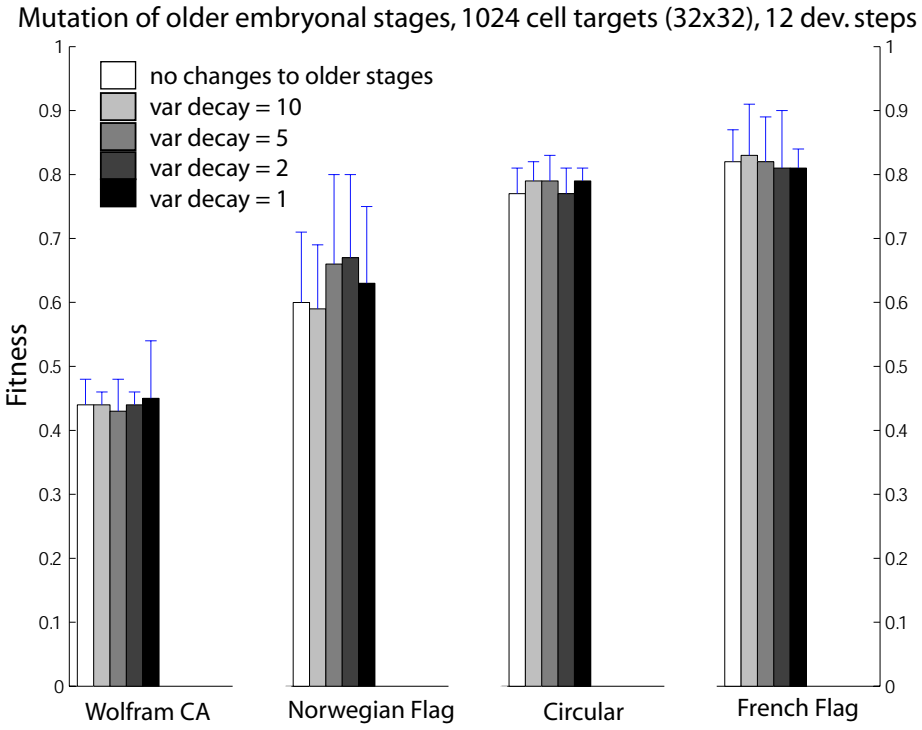


Figure 12: Average and maximum fitness scores for populations with different mutation strategies. On the contrary to other simulations presented, older embryonal stages are also affected by mutation. Higher values of variance decay (var decay, in figure) reduce the effects of mutation for older embryonal stages. Mutation affects the functionality of the Morpher.

the Wolfram CA pattern.

These results suggest that the exact number of variables used does not alter significantly the evolvability of development.

6.6 Emergent and Evolved self-repair

The regenerative capabilities of biological organisms have always been a source of inspiration for researchers. A very interesting property of this and similar multicellular development models is that they show emergent fault recovery [25, 26, 28]. In these cases, self-repair was not selected for and appears as an emergent byproduct of ontogeny. With each cell's behaviour based on the same growth program and only local variables, development achieves an intrinsic stability.

Self-repair is very interesting in the prospect of real world implementations, where random faults (e.g. radiation, fatigue) or substrate imperfections (e.g

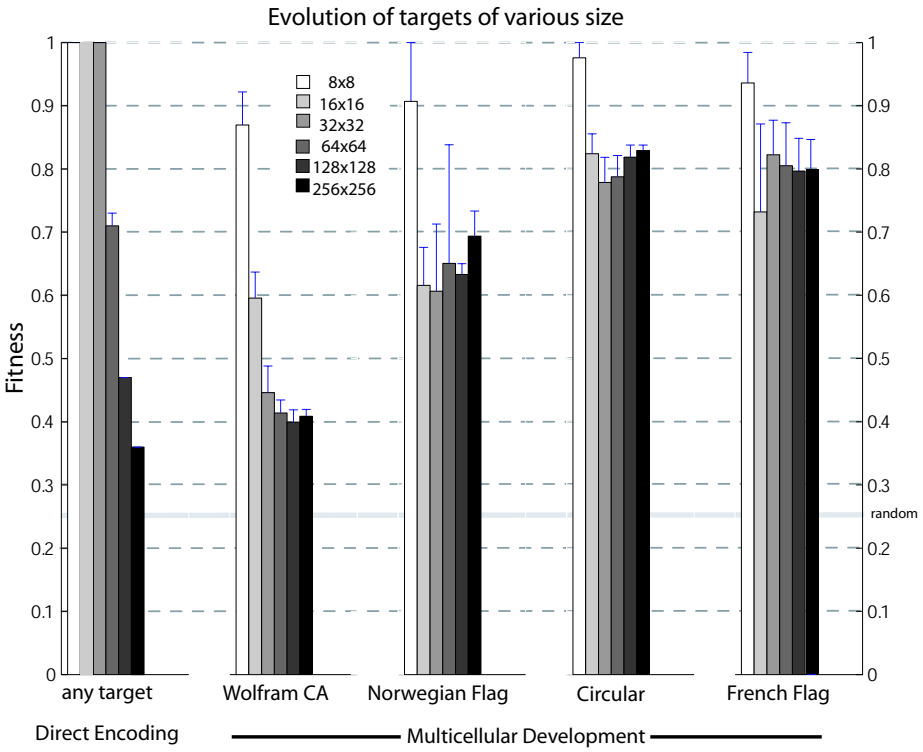


Figure 13: Average and maximum fitness scores for populations of various phenotypic size. Direct encoding performance decreases with bigger search spaces. On the contrary, multicellular development performance seems to converge. For comparison, the average performance of a random individual is also provided.

manufacture errors) are common. Contrary to traditional approaches, here fault-tolerance is achieved by regeneration without phenotypic redundancy.

In Figures 18-20, emergent and evolved fault tolerance are shown. To test the regenerative capabilities of the evolved organisms, the development of the best individuals has been repeated with random phenotypic faults, which cause the death of selected cells.

Each cell has been assigned a mortality rate $m_r \in \{0.005, 0.01, 0.05, 0.1\}$ per cell per developmental step. The probability for a cell sustaining a fault at any time during the development process (12 steps) is thus {0.06, 0.11, 0.46, 0.72} respectively.

Emergent fault tolerance (figure 18) reports the self-repair capabilities for individuals which were not selected for this characteristic.

On the contrary, evolved fault tolerance is achieved subjecting individuals to faults during evolution. Subjected to random faults, the stochastic growth process must be tested several times to guarantee a meaningful selection process.

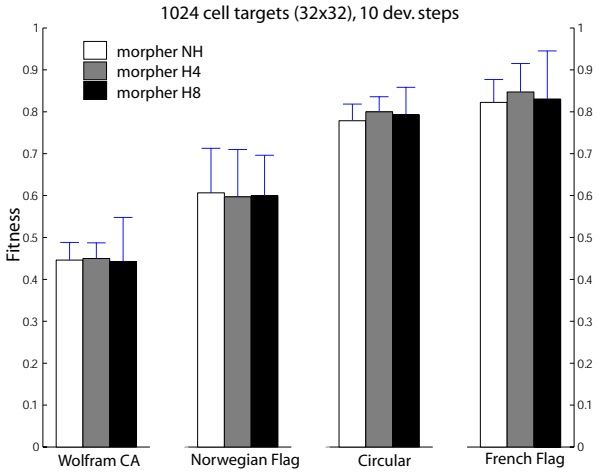


Figure 14: Average and maximum fitness scores for populations with different morphers: RNN without hidden layers (NH), and with one hidden layer of 4 (H4) and 8 (H8) neurons.

In particular, higher fault probability should require exponentially more repetitions. Since fitness evaluations are usually computationally expensive (for example in the evolution of robot controllers) here the number of repetitions is fixed to a reasonably small number. With $m_r \in \{0.01, 0.1\}$ fitness is computed as an average over only 5 independent growth processes.

Evolved individuals manage to reach high fitness scores and show a strong degree of fault tolerance.

7 Conclusions

We have tested a model of multicellular development showing that the method is both highly scalable and capable of evolving self-repairing organisms. As expected, performance is strongly dependent on the intrinsic regularity of the target. With larger search spaces the performance of the Direct Encoding (DE) steadily decreases, performance levels off for the presented Artificial Embryogeny (AE). This is valid also for the high complexity Wolfram CA target, even though both DE and AE performance is low for the largest phenotypes.

It is important to notice, that the presented test and especially the Wolfram CA target, are a 'worst case scenario' for the comparison of AE toward DE.

In fact, the problem of matching a string computing fitness as the Hamming distance to the target, is among the easiest for DE. Its difficulty is independent of the specific target's structure and it is equivalent to the One-Max problem.

For AE, difficulty is influenced by the target shape. In the general case it is not even clear whether a optimal solution exists. The fact that, even for

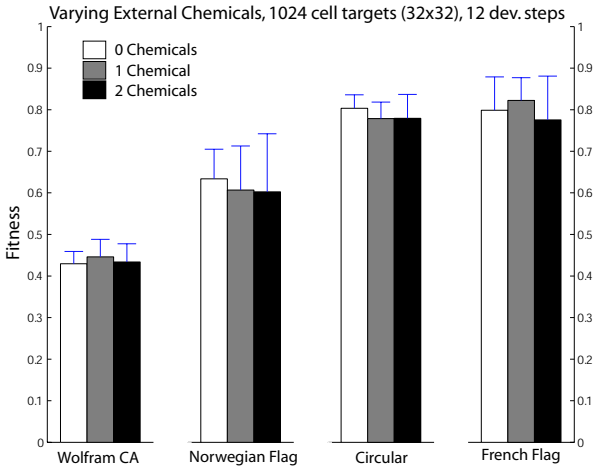


Figure 15: Maximum and average fitness scores from 32x32 targets with varying number of external chemical types. Differences are not statistically relevant ($p < 0.001$).

highly complex phenotypes, development ends up out-performing DE represents a strong evidence in favor of artificial embryogeny.

The analysis of random (Figure 4) and of evolved organisms (Figures 21-24) does not show developmental fingerprints, such as numerous repeated sub-patterns. These could prove a liability for the generality of the results. On the other hand, since cell genesis can only happen in the NWSE directions, horizontal and vertical sprouts are more common.

The development of organisms suffering stochastic cell deaths also shows good self-repair capabilities both when fault tolerance is neutral and selected for. Self-repair appears therefore as a by-product of ontogeny, with these results being consistent with those presented in [25, 26, 28].

The use of multiple embryonal stages has proved beneficial to evolution. Inspired upon gene duplication, development with embryonal stages implements a direct neutral complexification of an organism reducing the pleiotropy among different developmental phases. The method is very general and could be applied to any developmental model. While in general it is true that the more embryonal stages the better performance, on the other hand, if new stages are introduced too early during evolution they can be detrimental. When excluding older stages from search, new embryonal stages increase the resolution of proximal regions of the solution space, while locking distal ones. As a result, early introduction can show the drawbacks of over-exploitative search strategies. When all stages are allowed to evolve, these effects are weaker.

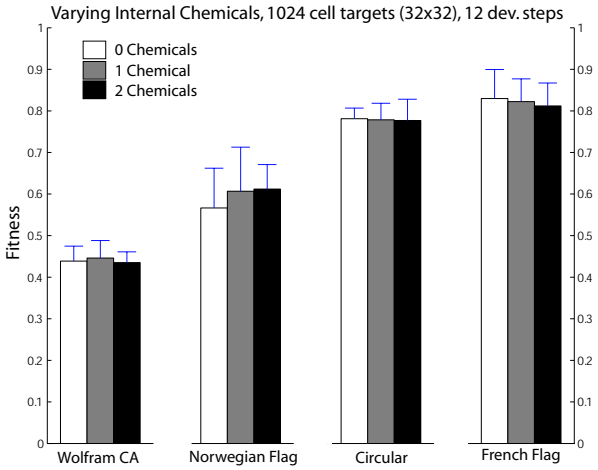


Figure 16: Maximum and average fitness scores from 32x32 targets with varying number of internal, metabolic, chemical types. Differences are not statistically relevant ($p < 0.001$).

7.1 Further Work

The evidence collected in this paper provides a good basis to start evolving more interesting artifacts. Preliminary results in this direction involve the evolution of locally connected spiking neural networks [13]. Given the high scalability of the model, future work will include the evolution of neuro-structures with functionality typical of peripheral brain regions, such as the retina and the motor cortex.

Also, embryonal stages provide neutral complexification only over the temporal domain. It is as well possible to design similar mechanisms operating on the spatial domain. Such method could prove particularly beneficial in the evolution of 3D organisms such as interconnected neural layers or body segments.

7.2 Computational Requirements

Locking older embryonal stages has a good effect on simulation speed. By caching old (fixed) stages, each new stages reduces the number of developmental steps required to produce a mature organism. Without this stratagem simulations are several times slower.

A single 256x256 cells evolutionary run may require 50 billions Morpher activations and take up to a week on a AMD XP 1800+ based computer. Without caching, the time needed would be 9 times as much. The over 1500 simulations presented have taken several weeks of computation time on the ClustIS Beowulf cluster [7].

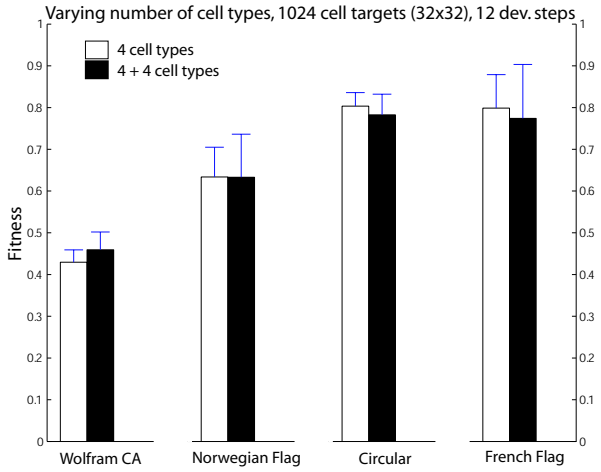


Figure 17: Maximum and average fitness scores from 32x32 targets with varying number of cell types. Differences are statistically relevant only for the Wolfram CA target ($p < 0.001$).

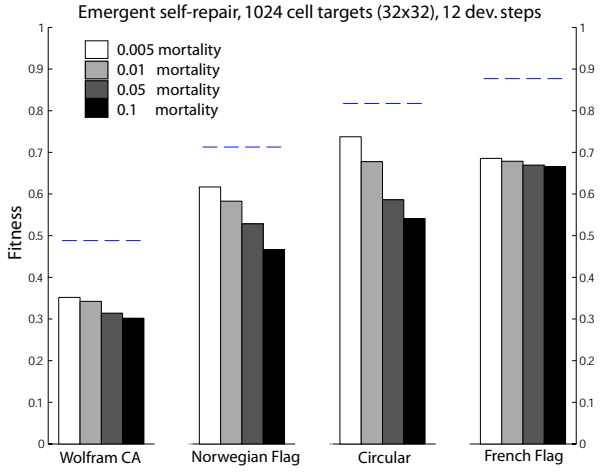


Figure 18: Emergent self-repair for the highest fitness 32x32 organisms with maximum number of embryonal stages (group 1). In this case, fault tolerance was not selected for and appears as a byproduct of ontogeny. Average performance over 100 runs with various mortality rates. The horizontal lines show the fitness score without faults.

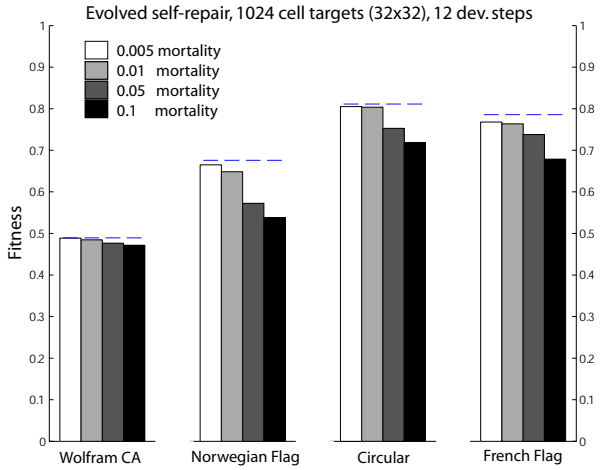


Figure 19: Evolved self-repair for 32x32 organisms with 12 EmbSt. Individuals evolved while subjected to a 0.01 fault probability, fitness being the average score over 5 runs. Average performance over 100 runs with various mortality rates. The horizontal lines show the fitness score without faults.

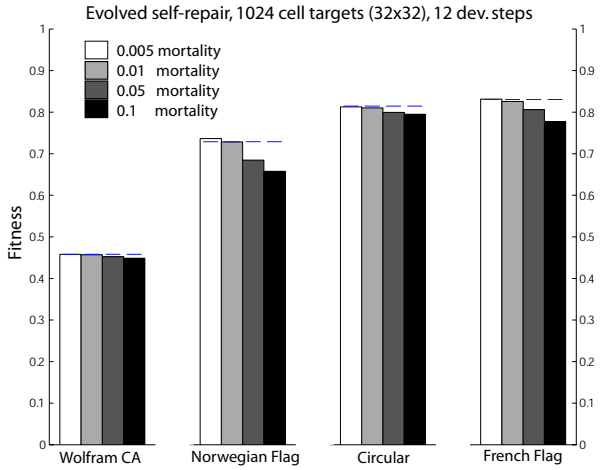


Figure 20: Evolved self-repair for 32x32 organisms with 12 EmbSt. Individuals evolved while subjected to a 0.1 fault probability, fitness being the average score over 5 runs. Average performance over 100 runs with various mortality rates. The horizontal lines show the fitness score without faults.

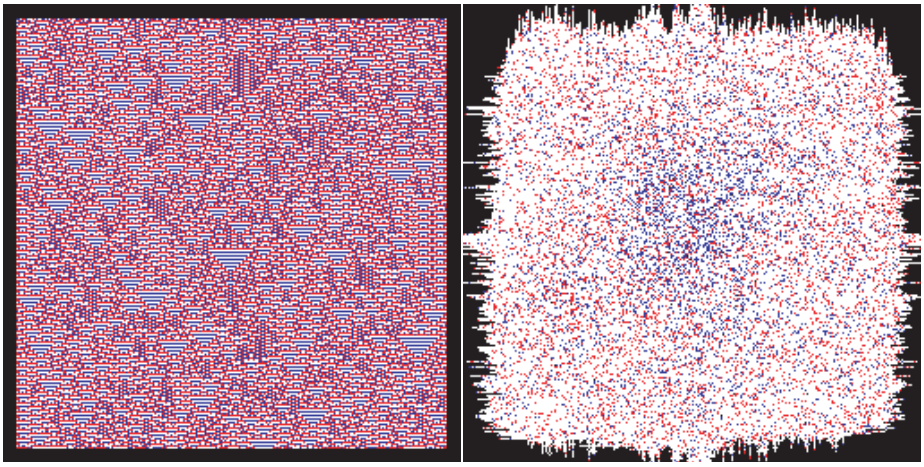


Figure 21: Wolfram CA target (left) and best evolved 256x256 individual (right).

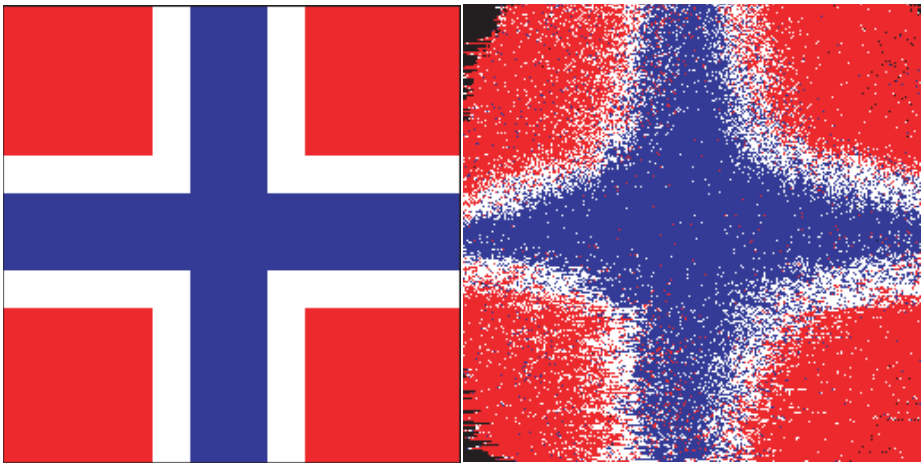


Figure 22: Norwegian Flag target (left) and best evolved 256x256 individual (right).

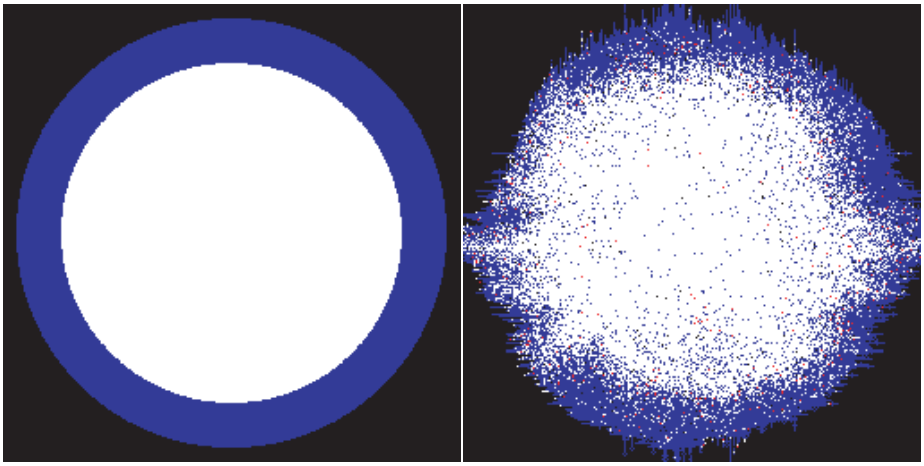


Figure 23: Circular target (left) and best evolved 256x256 individual (right).

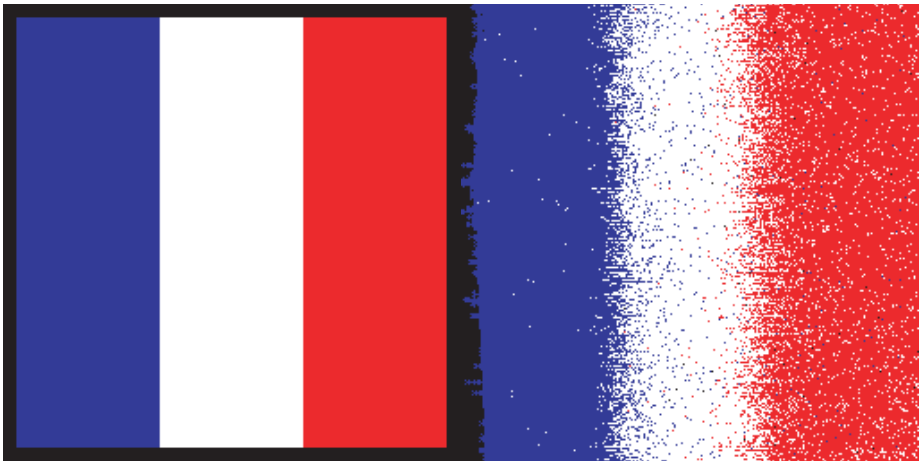


Figure 24: French Flag target (left) and best evolved 256x256 individual (right).

References

[1] W. Banzhaf. Genotype-phenotype-mapping and neutral variation - a case study in genetic programming. In *PPSN III: Proceedings of the International Conference on Evolutionary Computation. The Third Conference on Parallel Problem Solving from Nature*, pages 322–332. Springer-Verlag, 1994.

[2] Y. Bar-Yam. *Dynamics of Complex Systems*. The Advanced Book Studies in Nonlinearity. Westview Press, 1997.

[3] P.J. Bentley. Evolving fractal gene regulatory networks for robot control. In Wolfgang Banzhaf, Jens Ziegler, and Thomas Christaller, editors, *Proceeding of the European Congress of Artificial Life, ECAL 2003*, volume 2801/2003, pages 753–762. Springer-Verlag, 2003.

[4] P.J. Bentley and S. Kumar. Three ways to grow designs: A comparison of embryogenies for an evolutionary design problem. In Wolfgang Banzhaf, Jason Daida, Agoston E. Eiben, Max H. Garzon, Vasant Honavar, Mark Jakiela, and Robert E. Smith, editors, *Proceedings of the Genetic and Evolutionary Computation Conference*, volume 1, pages 35–43, Orlando, Florida, USA, 13-17 1999. Morgan Kaufmann.

[5] J. Bongard. Evolving modular genetic regulatory networks. In *Proceedings of the 2002 Congress on Evolutionary Computation (CEC2002)*, pages 1872–1877. IEEE Press, Piscataway, NJ, 2002, 2002.

[6] A. Cangelosi, S. Nolfi, and D. Parisi. Cell division and migration in a 'genotype' for neural networks. *Network: Computation in Neural Systems*, 5:497–515, 1994.

[7] J. Cassens and Z. Constantinescu Fülöp. It's magic: Sourcemage gnu/linux as hpc cluster os. In *in Proceedings Linuxtag 2003*, 2003.

[8] F. Dellaert and R. Beer. A developmental model for the evolution of complete autonomous agents. In Pattie Maes, Maja J. Mataric, Jean-Arcady Meyer, Jordan Pollack, and S W. Wilson, editors, *From Animals To Animats 4: SAB 1996*, pages 393–401, 1996.

[9] F. Dellaert and R.D. Beer. Toward an evolvable model of development for autonomous agent synthesis. In R.Brooks and P. Maes, editors, *Proceedings of Artificial Life IV*, pages 246–257. MIT Press Cambridge, 1994.

[10] J. Reinitz E. Mjolsness, D.H. Sharp. A connectionist model of development. *Journal of Theoretical Biology*, 152(4):429–453, 1991.

[11] P. Eggenbergen-Hotz. Evolving morphologies of simulated 3d organisms based on differential gene expression. In Phil Husbands and Inman Harvey, editors, *Proceedings of the 4th European Conference on Artificial Life (ECAL97)*, 1997.

[12] P. Eggenbergen-Hotz. Comparing direct and developmental encoding schemes in artificial evolution: A case study in evolving lens shapes. In *Proceeding of the Congress on Evolutionary Computation, CEC 2004*, pages 752–757, 2004.

[13] D. Federici. Evolving a neurocontroller through a process of embryogeny. In S. Schaal, AJ Ijspeert, A. Billard, S. Vijayakumar, J. Hallam, and JA Meyer, editors, *From Animals To Animats 8: SAB 2004*, pages 373–384, 2004.

[14] F. Gruau. *Neural Network Synthesis using Cellular Encoding and the Genetic Algorithm*. PhD thesis, Ecole Normale Superieure de Lyon, 1994.

[15] I. Harvey. Artificial evolution: A continuing saga. In Takashi Gomi, editor, *Proceedings of 8th Intl. Symposium on Evolutionary Robotics, ER2001*, pages 94–109. Springer-Verlag, 2001.

[16] G.S. Hornby and J.B. Pollack. The advantages of generative grammatical encodings for physical design. In *Proceedings of the 2001 Congress on Evolutionary Computation, CEC 2001*, pages 600–607. IEEE Press, 27-30 2001.

[17] G.S. Hornby and J.B. Pollack. Body-brain co-evolution using L-systems as a generative encoding. In Lee Spector, Erik D. Goodman, Annie Wu, W. B. Langdon, Hans-Michael Voigt, Mitsuo Gen, Sandip Sen, Marco Dorigo, Shahram Pezeshk, Max H. Garzon, and Edmund Burke, editors, *Proceedings of the Genetic and Evolutionary Computation Conference, GECCO-2001*, pages 868–875. Morgan Kaufmann, 7-11 2001.

[18] N. Jacobi. Harnessing morphogenesis. In P. Bentley and S. Kumar, editors, *On Growth, Form and Computers*. Academic Press, 2003.

[19] S.A. Kauffman. *The origins of order*. Oxford University Press, New York, 1993.

[20] H. Kitano. Designing neural networks using genetic algorithms with graph generation system. *Complex Systems*, 4:4:461–476, 1990.

[21] P.K. Lehre and P. C Haddow. Developmental mappings and phenotypic complexity. In Ruhul Sarker, Robert Reynolds, Hussein Abbass, Kay Chen Tan, Bob McKay, Daryl Essam, and Tom Gedeon, editors, *Proceeding of the Congress on Evolutionary Computation, CEC 2003*, 2003.

[22] S. Luke and L. Spector. Evolving graphs and networks with edge encoding: Preliminary report. In John R. Koza, editor, *Late Breaking Papers at the Genetic Programming 1996 Conference*, pages 117–124, 1996.

[23] C. Mattiussi and D. Floreano. Connecting transistors and proteins. In J.Pollack, M. Bedau, P. Husbands, T. Ikegami, and R. Watson, editors, *ALife9: Proceedings of the Ninth International Conference on Artificial Life*, pages 9–20, 2004.

[24] ARJ Software Inc. http://www.arjsoftware.com. ARJ specifications.

[25] J.F. Miller. Evolving developmental programs for adaptation, morphogenesys, and self-repair. In Wolfgang Banzhaf, Jens Ziegler, and Thomas Christaller, editors, *Proceeding of the European Congress of Artificial Life, ECAL 2003*, pages 256–265, 2003.

[26] J.F. Miller. Evolving a self-repairing, self-regulating, french flag organism. In Kalyanmoy Deb, Riccardo Poli, Wolfgang Banzhaf, Hans-Georg Beyer, Edmund K. Burke, Paul J. Darwen, Dipankar Dasgupta, Dario Floreano, James A. Foster, Mark Harman, Owen Holland, Pier Luca Lanzi, Lee Spector, Andrea Tettamanzi, Dirk Thierens, and Andrew M. Tyrrell, editors, *Proceeding of Genetic and Evolutionary Compuation, GECCO 2004*, pages 129–139, 2004.

[27] S. Ohno. *Evolution by Gene Duplication*. Springer, 1970.

[28] D. Roggen and D. Federici. Multi-cellular development: is there scalability and robustness to gain? In Xin Yao, E. Burke, J.A. Lozano, and al., editors, *proceedings of Parallel Problem Solving from Nature 8, PPSN 2004*, pages 391–400, 2004.

[29] K. Stanley and R. Miikulainen. A taxonomy for artificial embryogeny. *Artificial Life*, 9(2):93–130, 2003.

[30] K. Stanley and R. Miikulainen. Competitive coevolution through evolutionary complexification. *Journal of Artificial Intelligence Research*, 21:63–100, 2004.

[31] S.A. Teichmann and M.M. Babu. Gene regulatory network growth by duplication. *Nature Genetics*, 36(5):492–6, 2004.

[32] G. Tufte and P. C Haddow. Building knowledge into development rules for circuit design. In Torresen Tyrrell, Haddow, editor, *Proceedings of the International Conference on Evolvable Systems, ICES 2003*, pages 117–128. Springer-Verlag, 2003.

[33] A.M. Tyrrell, E. Sanchez, D. Floreano, G. Tempesti, D. Mange, J.M. Moreno, J. Rosenberg, and A. Villa. Poetic tissue: An integrated architecture for bio-inspired hardware. In Torresen Tyrrell, Haddow, editor, *Proceedings of the International Conference on Evolvable Systems, ICES 2003*, pages 129–140. Springer-Verlag, 2003.

[34] Wikipedia. Ontogeny and phylogeny. *http://en2.wikipedia.org/wiki/Ontogeny_and_phylogeny*, 2003.

[35] S. Wolfram, editor. *A new kind of science*. Wolfram Media, 2002.

[36] J Zhang. Evolution by gene duplication: an update. *Trends in Ecology and Evolution*, 18(6):192–198, 2003.

H Evolving a neurocontroller through a process of embryogeny

Author: Diego Federici. Published in proceedings of SAB 2004 Simulation of Adaptive Behavior.

Abstract:

We introduce a model of cellular growth that generates neurocontrollers capable of guiding simple simulated agents in a harvesting task. The morphogenesis of the neurocontroller is itself controlled by an evolved artificial neural network. The neural network operates only on local variables and chemical concentrations and is thought as a flexible model of a gene regulatory system and cell metabolism. The model is designed in order to increase the evolvability of the growth mechanism, which constitutes a serious issue in artificial embryogeny. Also, to increase the flexibility of development, organisms are grown in embryonal stages, which allow an incremental refinement of development. Neurocontrollers are organized in horizontal layers, with vertical input and output pathways. Within the same layer, neurons can have only local connections. On one side this limits the information needed for routing and on the other makes the system easy to implement in hardware. Results show that the system is capable of developing appropriate neurocontrollers in most of the evolutionary runs.

Objective:

Use the DES system (Development with Embryonal Stages) to produce a neuro-controller for situated agents.

Conclusions:

If on one hand results showed that efficient neurocontrollers are produced by development, on the other, the evolutionary speed is quite low. Analysis suggests a weakness of the proposed encoding for input, output and lateral connections.

Evolving a neurocontroller through a process of embryogeny

Diego Federici
Norwegian University of Science and Technology
Department of computer and information science
N-7491 Trondheim, Norway
federici@idi.ntnu.no

Abstract

We introduce a model of cellular growth that generates neurocontrollers capable of guiding simple simulated agents in a harvesting task.

The morphogenesis of the neurocontroller is itself controlled by an evolved artificial neural network. The neural network operates only on local variables and chemical concentrations and is thought as a flexible model of a gene regulatory system and cell metabolism.

The model is designed in order to increase the evolvability of the growth mechanism, which constitutes a serious issue in artificial embryogeny.

Also, to increase the flexibility of development, organisms are grown in ebryonal stages, which allows an incremental refinement of development.

Neurocontrollers are organized in horizontal layers, with vertical input and output pathways. Within the same layer, neurons can have only local connections. On one side this limits the information needed for routing and on the other makes the system easy to implement in hardware.

Results show that the system is capable of developing appropriate neurocontrollers in most of the evolutionary runs.

1. Introduction

The single most serious problem in the field of evolutionary computation (EC) is the combinatorial explosion of the search space as target designs grow in size.

Since the dimensionality of the search space in EC is given by the genotypic space (G-space) while designs reside in the phenotypic space (P-space), one possible conclusion is to reduce the first without loosing the possibility to produce the best solutions in the latter.

Concerning the evolution of artificial neural networks (ANN), the problem is that the number of weights typically grow quadratically with the size of the layers, while the possible connections among layers grows combinatorially.

Examples taken from biological organisms suggest that the solution may lay in a not trivial mapping from genotype to phenotype. When we look to the development of the human brain we can see that the ontogenetic and epigenetic processes decompress the estimated 10^4 human genes to 10^{14} synaptic connections.

Inspired by such considerations, several indirect encoding schemes have been proposed. By allowing multiple reuses of the same genes, indirect encodings decompress a smaller genotype in the final phenotype through a process of development.

The term 'Artificial Embryogeny' (AE) has been proposed to describe "the subdiscipline of evolutionary computation (EC) in which phenotypes undergo a developmental phase" (Stanley and Miikulainen, 2003).

In AE, it is possible to distinguish two major evolutionary approaches to development of ANN's. The first one, is aimed at the evolution of a grammar which is repeatedly applied to the phenotype. Examples include the Matrix Rewriting scheme (Kitano, 1990), the Cellular Encoding (Gruau, 1994), Edge Encoding (Luke and Spector, 1996) and the GenRe system (Hornby and Pollack, 2001).

A second approach evolves the regulatory system of a cell with its metabolism and ability to duplicate. Cells are usually capable of releasing chemicals which diffuse in a simulated 2D environment, grow selective connections to neighboring cells and sometimes move following some chemical gradient.

There are two different ways to model the cell's regulatory system. In biological organisms, DNA RNA and proteins interact in complex patterns which are still matter of debate. Still, it has been suggested that this gene regulatory system could be modeled by Boolean Networks (Kauffman, 1993). In (Dellaert and Beer, 1996) the authors have have developed a Gene Regulatory Network (GRN) capable of evolving a neurocontroller. Other models of GNRs have been proposed in (Jacobi, 1995) and (Bentley, 2003).

Inspired by cell automata, another approach is to regulate the development with a set of rules that, upon

matching the particular cellular state, activate specific responses. The model presented in this paper belongs to this category.

Also inspired by cell automata, Bentley and Kumar proposed a model which does not yet develop ANNs but tiling patterns (Bentley and Kumar, 1999). Cells can only be of a single type and the aim is to develop perfect tessellating patterns. Rule preconditions are composed of strings matching the local North-West-South-East neighborhood (NWSE) of each cell and their absolute position. *Don't care* symbols are also allowed. Results showed that the systems performed and scaled better than a direct encoding method. On the other side, the best solutions developed had very regular phenotypes. This left open the question whether the system could be used with targets of higher complexity.

Miller extended Bentley's model and developed more complex patterns (Miller, 2003). He allowed 4 different cell types (colors) and a chemical undergoing diffusion. Cells based their growth program on their type and the types and chemical concentrations from the 8 neighboring cells. The growth program itself was a boolean network that was evolved with the Cartesian Genetic Programming. Results showed evolved phenotypes resembling the target with only very few misplaced cells.

Additionally, Miller analyzed the behaviour of the evolved phenotype after the developmental step in which the fitness was computed and when subjected to severe mutilations. Phenotypes were shown to regrow the missing parts regaining a striking resemblance to the target. The self repair feature is very interesting since it was not selected for during evolution. A similar resistance to damage was also observed by this author in (Federici, 2004).

In this paper we extend this model of cellular development. Seeking a proof of the viability of these systems, we impose a functional target to the evolutionary search.

This, on one side imposes a bigger search space, more than 50 times bigger than previously tried, and on the other offers the opportunity to test an encoding strategy for the development of a given functionality.

In the genotype, instead of using lookup tables or boolean functions to control development, we adopted a recurrent neural network with a single hidden layer (neuromorpher). On one side, recurrent networks are an accepted model of genetic regulatory systems and on the other, by allowing continuous changes of their outputs, they allow a finer tuning of the program.

The mature phenotypes constitute themselves neurocontrollers for situated agents. The evolved neurocontrollers are composed of spiking neurons with only local connections. Compared to standard ANNs, the neurocontroller is more biologically plausible and offers a better hardware implementability, resistance to damage and scalability.

The following of the paper is organized as follows:
section 2 contains the agent's task description
section 3 the developmental system
section 4 the neurocontroller structure
section 5 details of the evolutionary model
section 6 results
section 7 conclusions

2. Agent's task and selection rules

Agent's are selected for their ability in a simple harvesting task. On a 60x60 toroidal map 240 agents compete for the collection of 320 food resources. While they search for food, agents must also avoid to crash against each other and avoid to collect poison resources (also 320). The fitness is computed as the sum of collected reinforcements over 100 activation steps. Reinforcements are calculated as follows:

+0.8	if	visiting a tile containing food
-0.8	if	visiting a tile containing poison
-0.3	if	colliding with another agent
-0.1	if	the agent did not move

Consumed resources are regenerated in random positions. At a given iteration, a tile can only contain either food or poison. Also, only one agent can occupy a given position. In case of a collision, the moving agent is unable to reach its destination.

The best 25% of the agent survive and reproduce. 10% of the new agents are produced by crossover and all the offspring undergoes mutation.

3. The development model

Phenotypes are developed starting from a single egg (zygote) placed in the center of a fixed size 2D grid. Cells (see figure 1) are characterized by internal and external variables. Internal variables define the cell state and move with it, while external ones (chemicals) belong to the environment and follow a simple diffusion law.

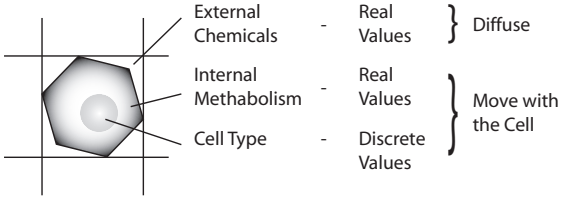


Figure 1: Description of the variables used for development

At each development step, any existing cell can release chemicals, change its own type and internal metabolism and produce new cells in the cardinal directions North, West, South and East. If necessary, existing cells are pushed sideways to create space for the new ones (see

figure 2). When a cell is pushed outside the boundaries of the grid it is permanently lost.

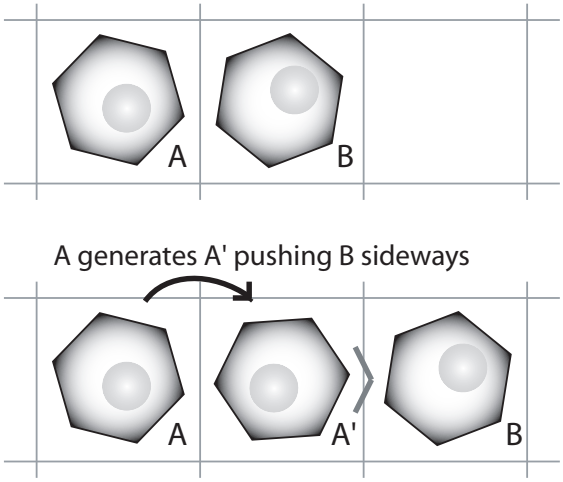


Figure 2: Placement of new cells

Morphogenesis is governed by the program, the neuromorpher, defined in the genotype. The neuromorpher (figure 3) receives in input the current cell internal and external variables, and the cell types of the neighboring cells in the four cardinal directions. Its output determines the new internal and external variables of the cell and, in case, the internal variables of the newly generated cells. An additional local variable, the cell age, is set to 1 at birth and decays exponentially to 0.

The neuromorpher is a model of a gene regulatory system, and it is implemented by a standard feedforward ANN[1], and the genotpye is a direct gray-coded representation of the ANN. The use of a neural network to control the development process allows, we believe, a greater degree of adaptivity for the definition of the appropriate growth rules. This same author have tested the same mechanism in the development of various 2D patterns (Federici, 2004).

4. The neural model

At every stage of development, a phenotype defines a neural network. For simplicity, in this paper the phenotype is first brought to maturation with a given number of development steps, and then used to control the movement of an agent.

The evolved neurocontroller (figure 4) is composed of a single layer. Each neuron can have only local connections to itself and to neighbors in the cardinal directions. Also it can specify a single input and a single output.

[1]even if, for an existing cell, the internal variables provide an immediate source of feedback

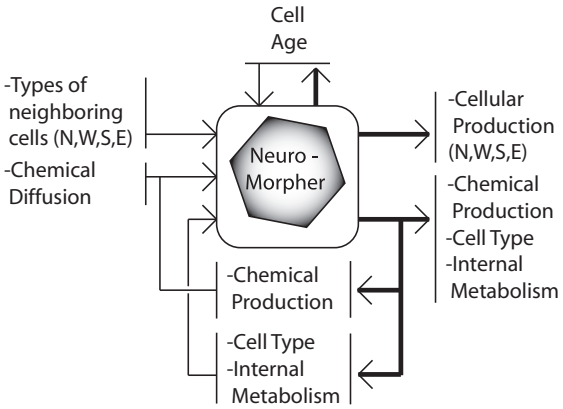


Figure 3: Inputs and outputs of the growth program

Neurons are spiking units. A spike is generated if the current net input exceeds the neuron's threshold.

The sensory input activates the hidden layer the first activation step only. The output of the neurocontroller is computed after a given number of activation steps.

Outputs are normalized in $[-1, 1]$. An output of 1 (-1) is possible if all excitatory (inhibitory) units connected to that output are spiking in the last step of activation of the neurocontroller. Otherwise the output assumes a value in the range, proportional to the neural activity among the connected neurons.

While intra-layer weights and thresholds take values in $[-2, 2]$, input and output weights can only have integer values in $\{-1, 0, 1\}$.

For the task presented herein, neurocontrollers have 9 inputs and 2 outputs. There are two possible types of sensors, one which detects food and poison (F-P) and the second the presence of an agent (A) on a particular tile.

The F-P sensor is set to 1 if food is present in the target tile, to -1 if poison is present, and 0 otherwise. The A sensor is set to 1 if an agent is detected, -1 otherwise.

In each of the cardinal directions, agents possess F-P and A sensors, while only a F-P sensor for the tile currently occupied (figure 5).

The two output nodes are used to select the agent's direction of movement. The direction taken is the one with strongest absolute output level. If both output levels are within a $[-.15, .15]$ region, no action is taken and the agent does not move.

4.1 construction of the neurocontroller

The position, internal and external variables of each cell select for their functionality in the organism:

1. The type determines input and output weights and to

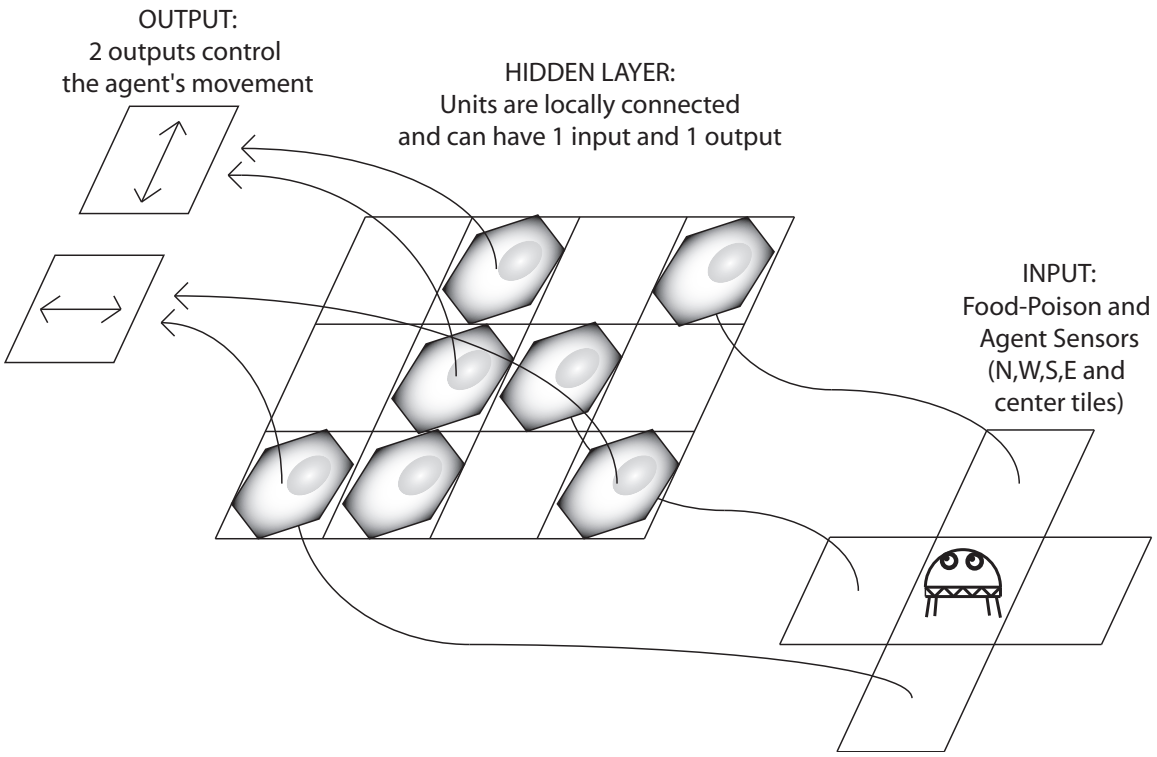


Figure 4: Organization of the neurocontroller

which single input and output the cell is connected.

2. The chemical gradients between neighboring cells are proportional to the strength of their connecting weights (eq.1). Only local connections in the cardinal directions are allowed.

3. The internal metabolism specifies the recurrent weight (eq.2) and the cell's threshold (eq.3).

The cell type is represented by a ternary vector of 5 dimensions. This gives rise to $3^5 = 243$ different cell types. The first and second dimensions contain the input and output weights, where a value of 0 is read as no connection, a value of 1 gives an excitatory connection (+1), and a 2 an inhibitory connection (-1). The following 2 dimensions contain the number of the input to which the cell is connected (9 combinations). The $5th$ dimension tells to which output the cell connects. In this case a value of 2 is interpreted as no connection.

The formulas to calculate weights and thresholds are given below.

$$W_{i,j} = Chemical_i^{Ext,1} - Chemical_j^{Ext,2} \quad (1)$$

$$W_{i,i} = Chemical_i^{Int,1} - Chemical_i^{Int,2} \quad (2)$$

$$\Theta_i = 2\left(Chemical_i^{Int,3} - \frac{1}{2}\right) \quad (3)$$

Where $Chemical_i^{Ext,k}$ and $Chemical_i^{Int,k}$ stand for the concentrations of the kth external or internal chemical for cell i.

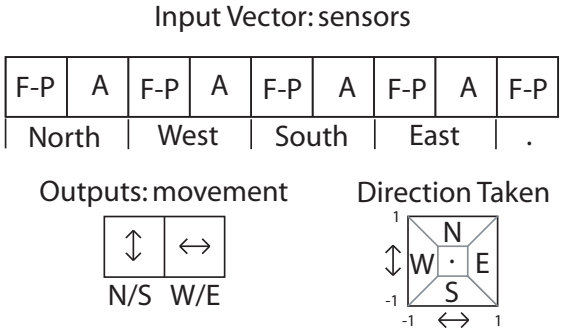


Figure 5: neurocontroller input and output. F-P stands for Food-Poison and A for Agent sensors. The direction marked with a '.' denoted the cell currently occupied by the agent. The N/S and W/E outputs specify the agent's direction of movement

4.2 genotype structure and operators

The neuromorpher is a feedforward ANN with 31 inputs, a single hidden layer with 8 units, and 47 outputs. The adopted transfer function is the hyperbolic tangent. The genotype is a direct gray-code representation of the weight matrices.

Mutation operates by selecting each weight with a .05 probability and shifting it by adding a value extracted by a normal distribution of zero mean and .01 variance. Crossover generates two new individuals shuffling the parent's hidden units.

5. The evolutionary model

*F*amily Grouping

One of the characteristics of AE systems is that evolutionary improvements have saltatory characteristics. Under these conditions a positive innovation can increase the reproductive chance of a particular strain reducing the chance of survival of all others. This often produce premature convergence to local optima.

To reduce these over-exploitative effects, we have adopted a family based grouping mechanism. Since small differences at genotype level can have a great impact on the development of the organisms, grouping is done at the phenotypic level.

Every generation, individuals are taken in random order and grouped by their phenotypes. For this task only the cell types and their positions are taken in consideration.

In order to belong to a family, individuals must have less than 4 phenotypic differences from the family prototype. If an individual does not fit in any family it will generate a new one, and its phenotype will be taken as new family prototype.

Within the same families, organisms are ranked by their fitness. In case of a tie, younger individuals are given a higher rank. The preference for younger individuals reduces the chances of stagnation (Miller, 2003). All those individuals which have a rank lower than the fourth have their fitness reduced:

$$fitness\ used\ for\ selection \quad = \quad fitness \cdot 0.9^{rank-3}$$

In figure 6, we show an example of how ranking counteracts the population convergence following an innovation.

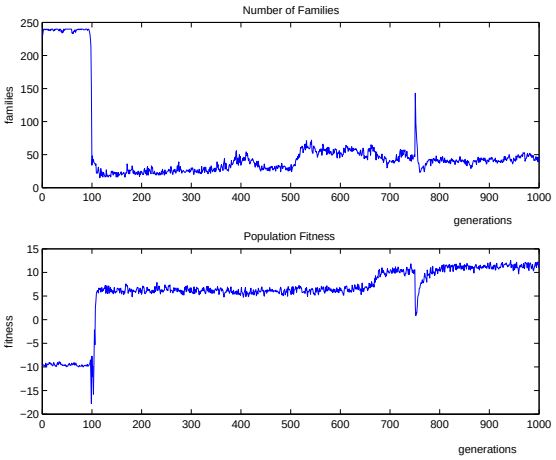


Figure 6: number of families and fitness. Around generation 100 an innovation favors a restricted group of genotypes. The number of families drops as these individuals are selected for reproduction. Counteracting this, ranking reduces the homogenization of the population to avoid stagnation.

*E*bryonal Stages

Another problem with the evolution of AE systems is that, because of gene reuse, occasional mutations may cause huge phenotypic changes. This effect reduces the correlation between G-space and P-space, which has well known negative effects on system evolvability.

A recent study conducted on the Matrix Rewriting model (Kitano, 1990) shows that this effect increases with the complexity of the phenotypes (Lehre and Haddow, 2003). Since organisms of high complexity are probable targets of evolutionary methods, these considerations pose serious questions on the viability of AE methodologies.

In biological organisms it is possible to observe a relationships between ontogeny and philology:

> It is generally observed that if a structure is evolutionary older than another, then it also appears

> earlier than the other in the embryo. Species which are evolutionary related typically share the early stages of embryonal development and differ in later stages. [...] If a structure was lost in an evolutionary sequence, then it is often observed that said structure is first created in the embryo, only to be discarded or modified in a later embryonal stage. (Wikipedia,)

Motivated by these considerations[2], this author has developed a method consisting of incremental embryonal stages (Federici, 2004).

Every embryonal stage is characterized by a single growth program. At the beginning of the evolutionary search, phenotypes are developed with a single embryonal stage. When a certain generation or fitness score are reached, a new stage is added. For this new stage, the novel growth program is initialized as a copy of the one from the previous stage. From now on, development will proceed in two successive phases.

At first it executes the growth program belonging to the first stage, while, after a predetermined developmental step, development switches to the second[3]. Additional stages are added likewise (see figure 7).

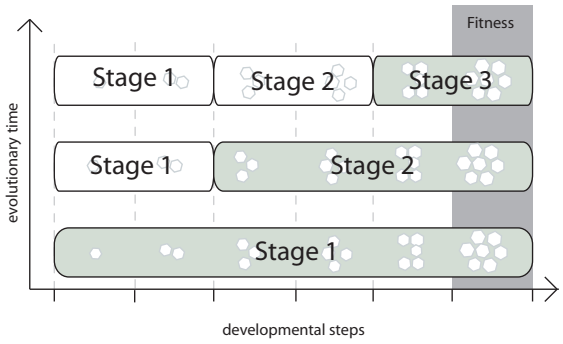


Figure 7: Embryonal stages. Stages are added incrementally, those controlling earlier stages of development being evolved first. Only the last stage is subjected to the innovation operators, the others are fixed. Fitness is computed always at the same developmental step.

Even if all the stages are used to grow the mature organism, evolution can only modify the program of the newest one, while the others are preserved intact.

This allows an incremental refinement of embryogenesis.

Even if the genotype size is increased linearly with the number of stages, as the search space is extended and explored incrementally, the evolvability is actually increased. This is because, 1) additional stages are created as a copy of the previous ones so that the evolutionary search will proceed without discontinuities, 2) the search is restricted to the refinement of the latest stage and therefor of fewer developmental steps, simplifying the task.

This effect should be particularly useful for high complexity phenotypes, since the change of the genotype of late embryonal stages will not affect early development causing catastrophic changes in development.

For the simulations presented in this paper, three embryonal stages have been used. The second stage is introduced at the 500th generation, and the third at the 750th. When all the stages are activated, the first grows the phenotype until the 5th developmental step, the second until the 7th and the third completes the maturation with step 8.

6. Results

We have evolved 20 populations with 240 agents. The average fitness of each population computed over the last 5 generations ranges from a minimum of 0.5 to a maximum of 13.4, with an overall average of 6.8.

To achieve a good fitness, individuals have to develop 3 different abilities.

1. Collect Food: when an agents is next to a tile containing food it must approach it.

2. Avoid Agents: agents must avoid to collide with other agents.

3. Avoid Poison: agents must not enter tiles containing poison.

Being that these are reactive agents, the optimal strategy would be to simply proceed in a preferred direction and steer only if confronted by an other bot, a tile containing poison or because detecting food in an other direction.

After evolution, we have taken the best individuals of each population and monitored their behavior in controlled enviromental conditions, such as with the presence of food, poison or other bots within the vision range.

Even if these controlled behavioral tests can check the agent reactions only on a subset of all the possible situations, they provide an objective description of the evolved control strategies. We were able to classify the collecting behavior in 7 categories of increasing quality (figure 8).

6.1 collecting behavior

No agent developed the ability to collect food in the direction opposite to the preferred direction of movement. On the other side, the advantage for the development of

[2] which should not be confused with the discredited Recapitulation theory

[3] notice that, as the two programs are identical, the introduction of new stages does not alter the development process.

	Description	grade	Nr
1	Always move in a single direction	low	3
2	As type 1 but also approach food in an additional direction	low	3
3	As type 1 but additionally avoid poison ahead	ok	3
4	As type 3 but also approach food in an additional direction	ok	2
5	As type 3 but also avoid agents ahead	good	4
6	As type 5 and collect food in an additional direction	good	4
7	As type 4 but approach food in three directions	good	1

Figure 8: the description of the 7 evolved behavioral strategies, a qualitative grade given to them, and the number of populations that developed them.

this ability is quite limited. By proceeding in a single direction, the tile left behind will be most often empty.

Also, the 60x60 toroidal world is quite crowded since there are 240 agents. In these conditions collecting food in all direction (strategy 7) can lead to globally unstable strategies. If an agent is surrounded by too many greedy individuals, it may be unable to collect food and score a low fitness score. Population-wise it might be as good to approach food only in two (strategy 6 and 4) or one (strategy 5 and 3) preferred directions, leaving not collected resources to individuals of the same specie.

6.2 agent avoidance

Similar considerations may also explain why only 4 populations (strategy 5) developed the ability to avoid agents in the preferred direction of movement. Since a homogeneous population will share the same preferred direction of movement, encounters with other agents will be very limited.

6.3 poison avoidance

Six populations (strategy 1 and 2) did not develop the ability to steer away from tiles containing poison, and therefore classified with a low grade. Still, the average population fitness of the two strategies is quite high, being 5.1 and 8.0 respectively. The reasons for this, probably found at the level of population behavior.

6.4 analysis of an evolved neurocontroller

To provide an example of an evolved neurocontroller, we have selected the best individual from a population of type 6.

Selected steps of the growth process are displayed in figure 10. Rectangles represent neurons, while synapses are plotted as cones of various dimensions (see figure 9). The bigger a cone the stronger the connection. White (black) cones stand for inhibitory (excitatory) connections.

A small circle at the right top of each neurons gives the amplitude of the recurrent weight.

Threshold values are represented by small vertical bars inside the neuron rectangle. Bars extending upwards have positive values.

Inside the neuron-rectangle two strings can be found. The topmost string indicates how and which input is connected to the neuron:

symbol	input to the unit
+AX	+1 only if agent is present
-AX	+1 only if agent is not present
PX	+1 only if poison is present
FX	+1 only if food i spresent

where X is either {N,W,E,S,.}.

The second string indicates towards which direction an activated neuron tries to move the agent.

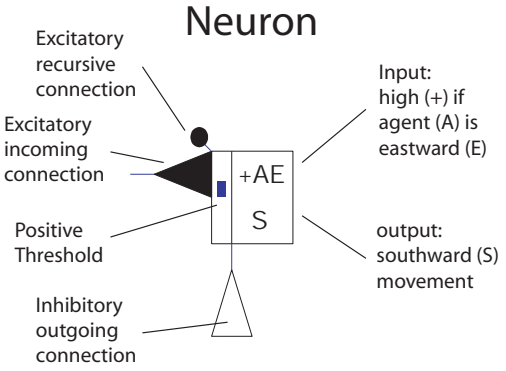


Figure 9: Explanation of the symbols used in the neurocontroller plots

The neurocontroller final activation in four important behavioral tasks is shown in figure 11.

Figure 11A contains the neural activity during the exploration task. In this case the agent proceeds in the default direction (east). The task is accomplished with the activation of 2 neurons for both the move south and move east actions. Since there are more connections to the north-south output, scaling favors the eastward direction.

In figure 11B, a single neuron is responsible for steering the agents towards the food. The extra activity sent to the north-south output suffice for the direction change.

In figure 11C, several neurons are activated by the presence of poison to the east. Only one though, is connected to the output and responsible for the avoidance behavior.

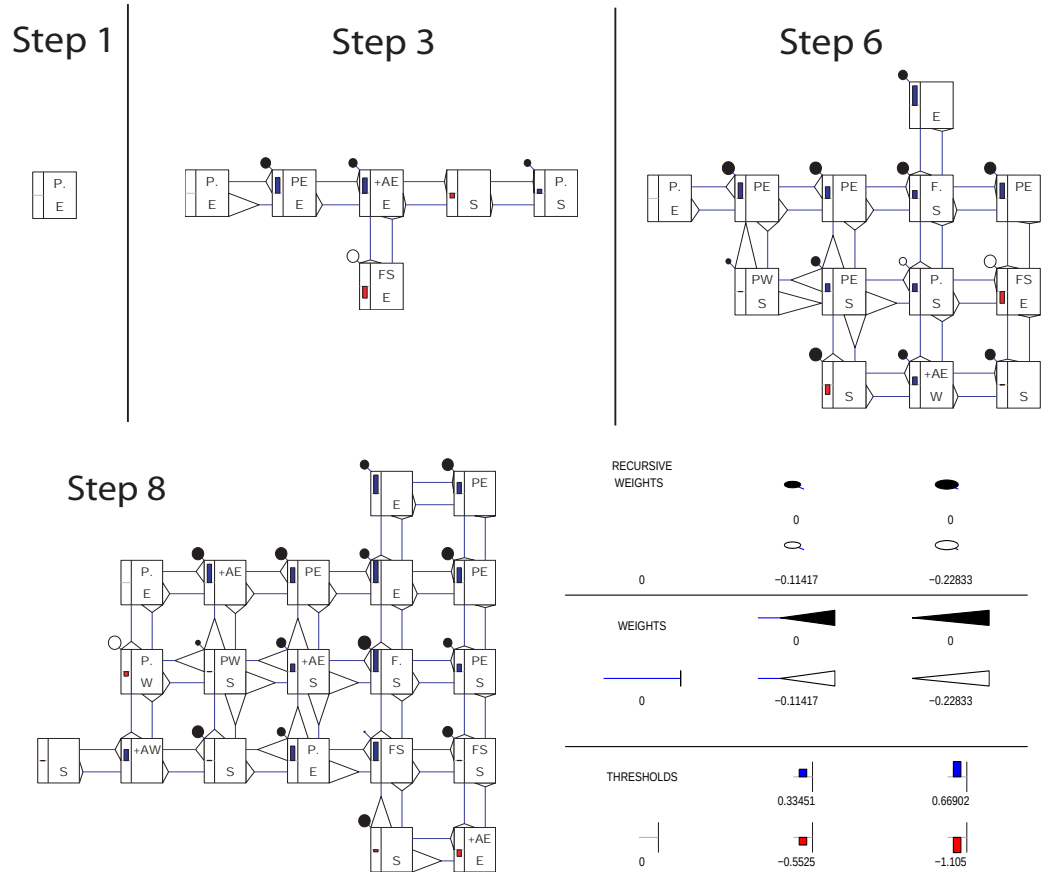


Figure 10: Development of the neurocontroller. The single zygote of step 0 grows into the functional organism by step 8.

In figure 11D, the agent must decide if it should enter a tile containing both food and an agent. In this case an agent detector correctly steers the bot away.

It is interesting to notice, that in this case, the only functional lateral weight goes from the neuron to the top left to the unit immediately below it. Without that connection the agent would start exploring southward loosing its ability to avoid agents and poison.

7. Conclusions

This paper contains an explorative work in the development of an Artificial Embryogeny model based on cell chemistry.

We have presented a novel model capable of developing a functional neurocontroller for a simple harvesting task of situated agents.

The optimal control policy is based on three behaviours: collecting a food resource, avoiding a poison resource and avoid other agents.

The quality of the results allows a cautious optimism towards the presented approach. Over the 20 simulations, 14 produced individuals with good control strategies. Among the other 6, 5 where still able to score acceptable average fitness scores. On the other side, behavioral tests showed their inability in the poison avoidance task.

Compared to similar previous work, these simulations have extended considerably the search space. Here there are 243 different types of cells and 5 internal and external chemicals.

The increased system's evolvability is given by three basic features: the ANN used to encode the growth program, the family based grouping, and the embryonal stages.

- Compared to other encodings, ANNs allows a smooth refinement of the morphogenesis process.
- Because of the saltatory characteristics of evolution in AE systems, grouping based on phenotype metrics plays a fundamental role avoiding the takeover of a

single genotypic strain.

- Adding several embryonal stages reduces the catastrophic effects of mutations affecting early developmental steps. This allows an incremental refinement of the growth program and increases the correlation between genotype and phenotype, expecially, we believe, for phenotypes of higher complexity.

The neuromorpher and neurocontroller introduced in this paper are organized in a bidimensional layer and present only local lateral connections. On one side, local connections decrease dramatically the routing information and therefore the search space. On the other, these characteristics make the system feasible for an hardware implementation such as the one presented in (Tufte and Haddow, 2003). This is true both for the evolutionary system and for the final evolved organisms.

From the analysis of the best scoring individuals, it appears that chemical gradients are not a good way to encode synaptic weights.

Since external chemicals diffuse in the environment, neighboring cells tend to have similar concentrations. This causes the connections to have mostly the same signs and values. Under such conditions, processing through lateral connections is very limited. Internal chemical concentrations seem more appropriate.

8. Further work

Probably the biggest limitation of the model presented is the way that input and output connections are encoded. The schema is very inefficient requiring a cell type for each possible input-output combination.

One possibility is to augment the encoding using the unique spatial disposition of each neuron. Also, this technique could permit the stacking of several neural layers one over the other.

A direction that is surely worth considering, is to activate and test the neurocontrollers also during development. The interaction with the environment would allow the possible integration of epigenetic processes.

Acknowledgements
I wish to thank Julien Miller for the useful discussions that inspired the work presented in this paper, and Keith Downing and Gunnar Tufte for the valuable suggestions.

References

Bentley, P. (2003). Evolving fractal gene regulatory networks for robot control. *Proceeding of ECAL 2003, 753–762.*

Bentley, P. and Kumar, S. (1999). Three ways to grow designs: A comparison of embryogenies for an evolutionary design problem. *Proceeding of the Genetic and Evolutionary Computation Conference 1999, 35-43.*

Dellaert, F. and Beer, R. (1996). A developmental model for the evolution of complete autonomous agents. In *proceedings the fourth international conference on Simulation of Adaptive Behavior*, pages 393–401.

Federici, D. (2004). Increasing evolvability for developmental programs. *Under submission to the Genetic and Evolutionary Computation Conference 2004.*

Gruau, F., (Ed.) (1994). *Neural Network Synthesis using Cellular Encoding and the Genetic Algorithm.* PhD Thesis, Ecole Normale Sup'erieure de Lyon.

Hornby, G. S. and Pollack, J. B. (2001). The advantages of generative grammatical encodings for physical design. In *Proceedings of the 2001 Congress on Evolutionary Computation*, pages 600–607. IEEE Press.

Jacobi, N. (1995). Harnessing morphogenesis. *Proceeding of information processing in cells and tissues, 29-41.*

Kauffman, S., (Ed.) (1993). *The origins of order.* New York: Oxford University Press.

Kitano, H. (1990). Designing neural networks using genetic algorithms with graph generation system. *Complex Systems, 4(4):461–476.*

Lehre, P. and Haddow, P. C. (2003). Developmental mappings and phenotypic complexity. *Proceedings of the 2003 Congress on Evolutionary Computation.*

Luke, S. and Spector, L. (1996). Evolving graphs and networks with edge encoding: Preliminary report. In Koza, J. R., (Ed.), *Late Breaking Papers at the Genetic Programming 1996 Conference*, pages 117–124.

Miller, J. (2003). Evolving developmental programs for adaptation, morphogenesys, and self-repair. *Proceeding of European Conference on Artificial Life 2003, 256–265.*

Stanley, K. and Miikulainen, R. (2003). A taxonomy for artificial embryogeny. *Artificial Life 9(2):93–130.*

Tufte, G. and Haddow, P. C. (2003). Insertion of functionality into development on an sblock platform. *Proceedings of the 2003 Congress on Evolutionary Computation.*

Wikipedia. Ontogeny and phylogeny. *http://en2.wikipedia.org /wiki /Ontogeny_and_phylogeny.*

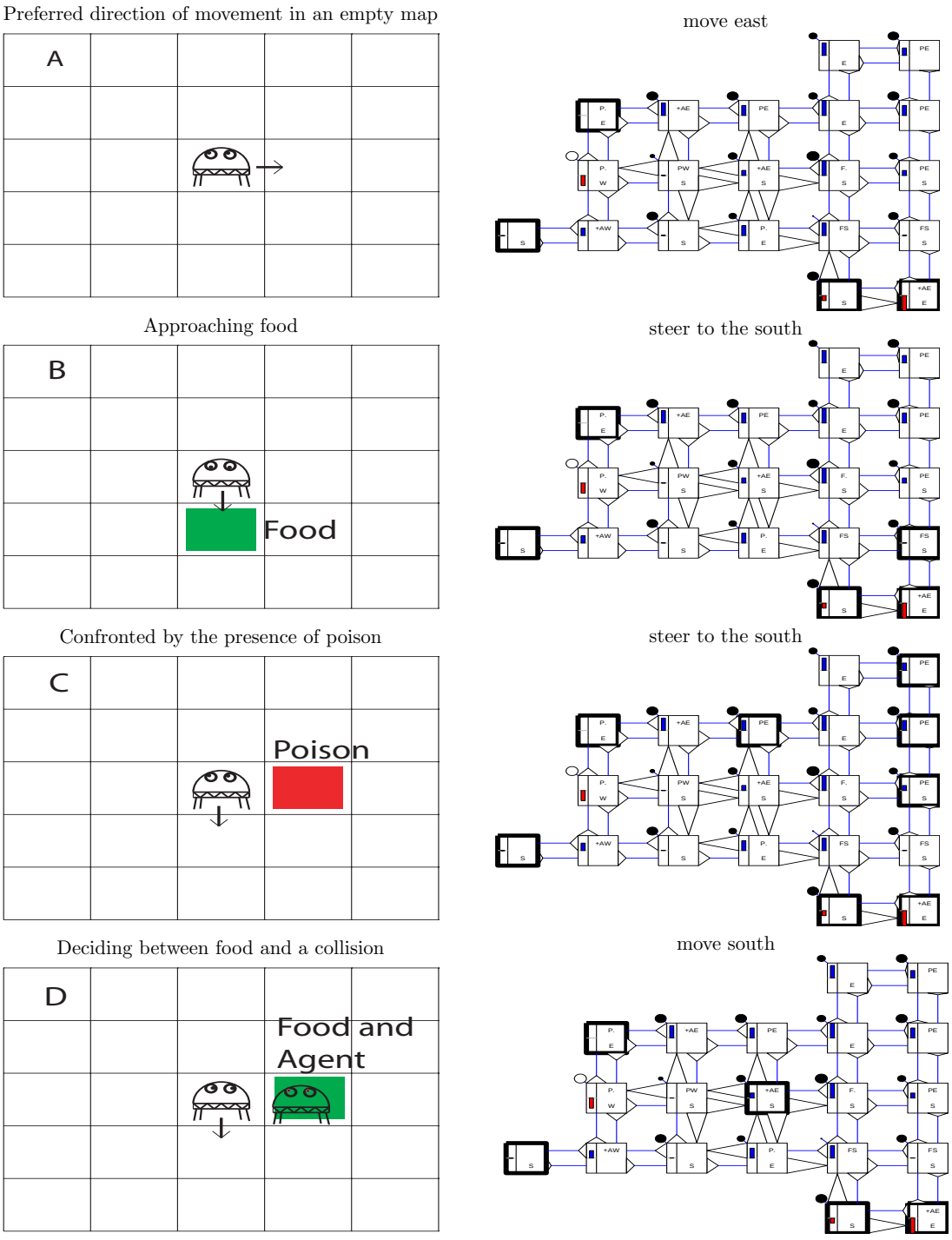


Figure 11: neural activity in various situations. A) without anything in sight, B) when spotting food, C) when the preferred direction of movement contains poison, D) when tempted by some food. Highlighted rectangles show activated neurons.

I A scalable evolutionary model for spiking neural networks: development with embryonal stages.

Author: Diego Federici. technical report: submitted to Genetic and Evolutionary Computation Conference - GECCO 2005

Abstract:

Evolutionary Computation based on indirect encoding strategies, aims to higher evolvability by reducing the dimensionality search space. If on one hand scalability is often improved for specific tasks, on the other the generality of these methods can be limited.
In this paper, we introduce a novel evolutionary friendly discrete-time leaky integrate and fire spiking neural model and evolve its topology, local connectivity and learning rules with both a multi-cellular Artificial Embryogeny system and direct encoding. The spiking networks are selected for their efficiency in a wall-avoidance task for simulated Khepera robots.
In previous work, to demonstrate its increased performance in the evolution of larger phenotypes, a developmental system was used to produce specific 2D patterns. In this paper, we use the same system to instead produce neural networks of increasing size, showing that similar conclusions can be drawn in a completely different domain: development increases the evolvability of large functional multi-cellular controllers for embodied agents.
Additionally, we introduce a learning mechanism based on local activity correlation. The mechanism is both very parsimonious and versatile and it has been especially designed for evolutionary applications. Even if the task does not require online adaptivity, simulations show that by allowing the evolution of the learning rules, performance is increased. This is valid only for the developmental system, where the increased phenotype size has a diminished impact on search efficiency.

Objective:

Develop an enhanced neural model and network encoding suited for multi-cellular development.

Conclusions:

> The proposed method evolves very rapidly proper neuro-controller for simulated Khepera robots. Especially with increasing phenotype sizes, development out-performs direct encoding. Also, in a task that does not require adaptivity, the proposed Hebbian-based learning model shows that plasticity increases overall evolvability.

A scalable evolutionary model for spiking neural networks: development with embryonal stages

Diego Federici
Complex Adaptive Organically-inspired Systems group (CAOS)
Norwegian University of Science and Technology
Department of computer and information science
N-7491 Trondheim, Norway

February 1, 2005

Abstract

Evolutionary Computation based on indirect encoding strategies, aims to higher evolvability by reducing the dimensionality of the search space.

In this paper, we introduce an evolutionary friendly discrete-time leaky integrate and fire spiking neural model and evolve its topology, local connectivity and learning rules with both a multi-cellular Artificial Embryogeny system and direct encoding. The spiking networks are selected for their efficiency in a navigation task for simulated Khepera robots.

In previous work, to demonstrate its increased performance in the evolution of larger phenotypes, the embryogeny system was used to produce 2D patterns. In this paper, we use the same system to produce networks of increasing size showing that similar conclusions can be drawn in the evolution of neuro-controllers for embodied agents.

By focusing on the evolution of the topology of the network, the introduced neural model offers a route towards the automated design of very large networks.

1 Introduction

Biological systems are not generated directly form a blueprint, with each phenotypic trait expressed by a section of the genotype. On the contrary, DNA bases display multiple levels of interactions with the intracellular environment. These interactions, almost as a byproduct, are responsible for the development of the phenotype.

As a consequence, specific groups of genes can influence several distinct phenotypic traits, so that the dimension of the genotype can be quite independent of the phenotypic size. For example, just by taking in consideration the number of cells in a human body, there are $50T$ (Tera) cells circa, while it is estimated that our genotype contains only $30K$ genes ($45M$ DNA bases).

For Evolutionary Computation (EC) a compression of the genotype has the obvious advantage to restrict the search to an exponentially smaller space. On the other hand, an arbitrary restriction could exclude some of the best solutions from the search space. Additionally, the pleiotropy introduced by this indirect map from genotype to phenotype, reduces the correlation between the search and the solution spaces, with a small change in the genotype capable of catastrophic effects on the phenotype. Reduced correlation makes the incremental refinement of the phenotype more difficult, often reducing the evolvability for complex problems.

Still, development allows the reuse of genetic material which might lead to the emergence of modules in the phenotype. The bilateral symmetry found in most vertebrates is an example. The nervous system also presents many repeated substructures: for example the neocortex shows two levels of organization with repeated local micro-circuits.

In this paper we present a novel leaky integrate and fire spiking neural model particularly suited for evolution and development. The model is used to evolve developing neuro-controllers for a simulated Khepera robot equipped with a variable number of infrared sensors (IR). Development is based on a multicellular embryogeny system which has be shown to be scalable, general and capable of self-repair [9, 25, 11].

Results show that, in a collision avoidance task, the model evolves very rapidly a proper growth program for a spiking neural network. Also Compared with a direct encoding GA, results show that the model is well suited for evolution and artificial embryogeny systems.

1.1 related models

It is possible to partition the development of artificial neural networks (ANNs) into two major evolutionary approaches. The first one, is aimed at the evolution of a grammar which is repeatedly applied to the phenotype. Examples include the Matrix Rewriting scheme [19], the Cellular Encoding [14], Edge Encoding [20] and the GenRe system [15].

A second approach evolves the regulatory system of a cell with its metabolism and ability to duplicate. Cells are usually capable of releasing chemicals which diffuse in a simulated 2D environment, grow selective connections to other cells, move following some chemical gradient and sense the presence of neighboring cells [23, 6, 4].

There are different ways to model the cell's growth program. Inspired upon the eukaryotic gene regulatory systems, artificial gene regulatory networks (aGRN) can be found in [7, 16, 3, 22]. In these systems, genes' activation regulate (and is regulated by) the expression of other genes. Activated genes are then used to define the morphology of the evolved organisms. Albeit capable of generating very complex regulatory dynamics, these systems are often computationally expensive, making them unsuited for the development of large multi-cellular organisms.

To avoid this problem, other approaches adopt lighter models with comparable functionality. For example, inspired by the work of Kauffman [18] recursive boolean networks have been used in [23, 6]. In the model used in this paper, recursive neural networks (RNN) are used. Not only RNN are widely accepted as a model of gene regulatory systems[2, 8], but also, when compared to discrete boolean functions, the space of continuous functions representable by RNNs allows finer tuning and richer neutral space.

In previous work, a similar developmental model was used to produce an integrate and fire neural network for agents immersed in a discrete world [9]. The presented new leaky integrate and fire spiking neural network is used to control simulated Khepera robots and presents several advantages in terms of: richer behavior, evolvability, scalability and learning rules. As a result larger and more complex networks are produced.

2 Embryogenesis: development with embryonal stages

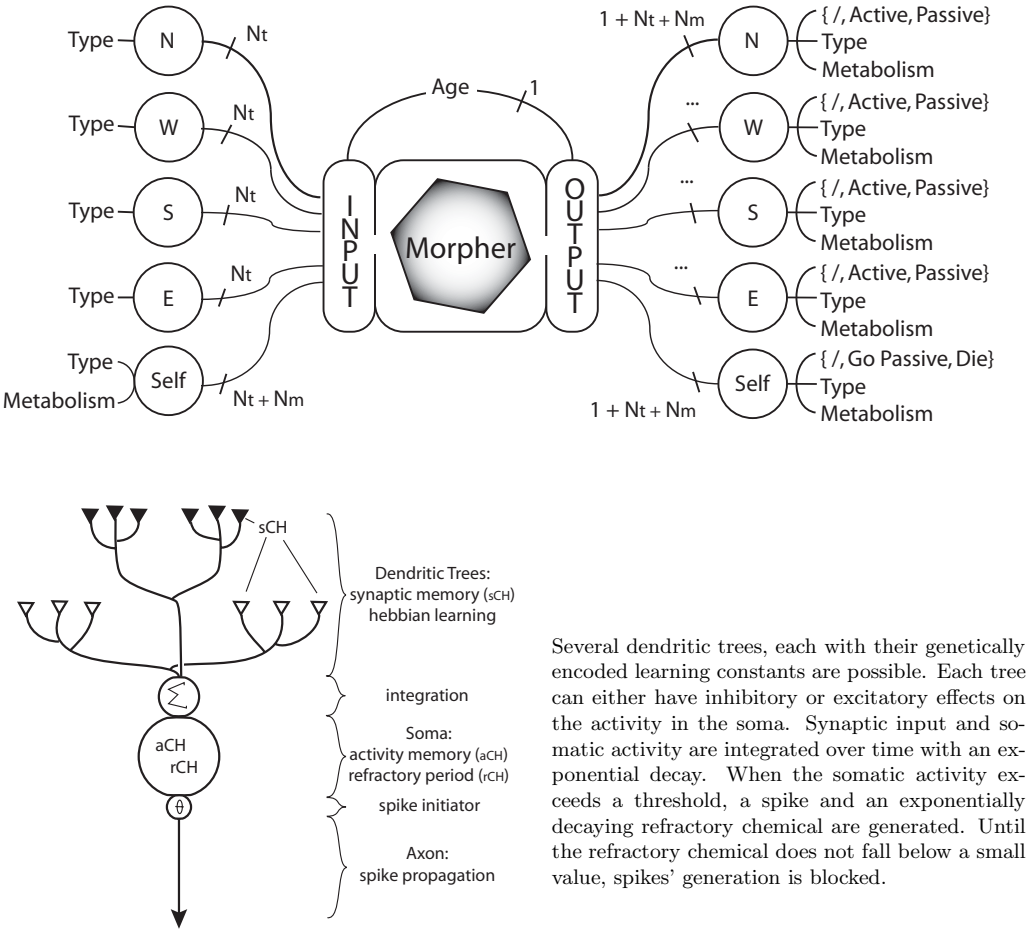
Organisms develop starting from a single replicating cell to reach the mature multi-cellular organisms in a precise number of developmental steps. At each developmental step, existing active cells can change their own type, alter their internal chemical concentrations and produce new cells. An active cell can also die or become passive.

The cell behavior is governed by identical RNNs (Morphers) directly encoded in the genotype with a floating point number used for each synaptic weight. The Morpher (Figure 1) is a single layer network, receiving as input the current cell type (discrete values), metabolism (real values) and age (real value), and the cell types of the neighboring cells in the four cardinal directions. Its output determines its new type and metabolism and, in case of replication, the types and metabolism of the newly generated cells. The cell age is initialized to 1 and decays exponentially.

During the course of evolution, similar to a process of gene duplication [24], new Morphers can be added to extend the growth program. With each one controlling a different embryonal stage, this mechanism of explicit neutral complexification allows richer and more timely responses of the growth program, overall increasing the evolvability of development. Please refer to [9, 11] for all the details concerning development with embryonal stages.

3 The neural model

Evolution searches the space of growth programs producing multi-cellular organisms. Each cell of these organisms is interpreted as a spiking neuron and used to control the behavior of a Khepera robot.



Several dendritic trees, each with their genetically encoded learning constants are possible. Each tree can either have inhibitory or excitatory effects on the activity in the soma. Synaptic input and somatic activity are integrated over time with an exponential decay. When the somatic activity exceeds a threshold, a spike and an exponentially decaying refractory chemical are generated. Until the refractory chemical does not fall below a small value, spikes' generation is blocked.

Figure 2: Depiction of the leaky integrate and fire spiking neuron.

The spiking neuron is an implementation of leaky integrate and fire neuron, similar to the one defined in [13]. The behavior of the spiking neuron j is computed following discrete time the equations (see also Figure 2):

$$\begin{aligned}
\text{synapse } i: \quad & \mathrm{sCH}_i(t) &&= \begin{cases} w_i + D_s\,\mathrm{sCH}_i(t-1) & \text{with an incoming spike} \\ D_s\,\mathrm{sCH}_i(t-1) & \text{else ways} \end{cases} \\
\text{activity of } j: \quad & \mathrm{aCH}_j(t) &&= D_a\,\mathrm{aCH}_j(t-1) + \sum_{i \in Synapses} \mathrm{sCH}_i(t) \\
\text{refract. of } j: \quad & \mathrm{rCH}_j(t) &&= D_r\,\mathrm{rCH}_j(t-1) + S_j(t) \\
\text{spike from } j: \quad & S_j(t) &&= \begin{cases} 1 & \text{if } \mathrm{aCH}_j(t) > \theta \ \wedge \ \mathrm{rCH}_j(t) < .1 \\ 0 & \text{else ways} \end{cases}
\end{aligned}$$

with $\{D_a,\ D_r,\ D_s,\ \Theta\} \in [0, 1]$ constants controlling the dynamic of the spiking neuron. In our implementation the synaptic efficacy w is either a fixed value $\in$ {-1, 0, 1} or can vary freely within $\pm[0, 1]$ subjected to pre/post synaptic correlation plasticity (e.g. Hebbian learning).

In biological neurons plasticity is typically associated to the presence of post synaptic activity, either derived from afferent neuro-transmitters or reverberating post-synaptic potentials, in conjunction with second messengers (e.g. cAMP) or local chemicals (e.g. Ca^{++}) [17, 5, 21]. As a result a wide spectrum of learning behaviors are possible [1], including, but only limited to, Hebbian and

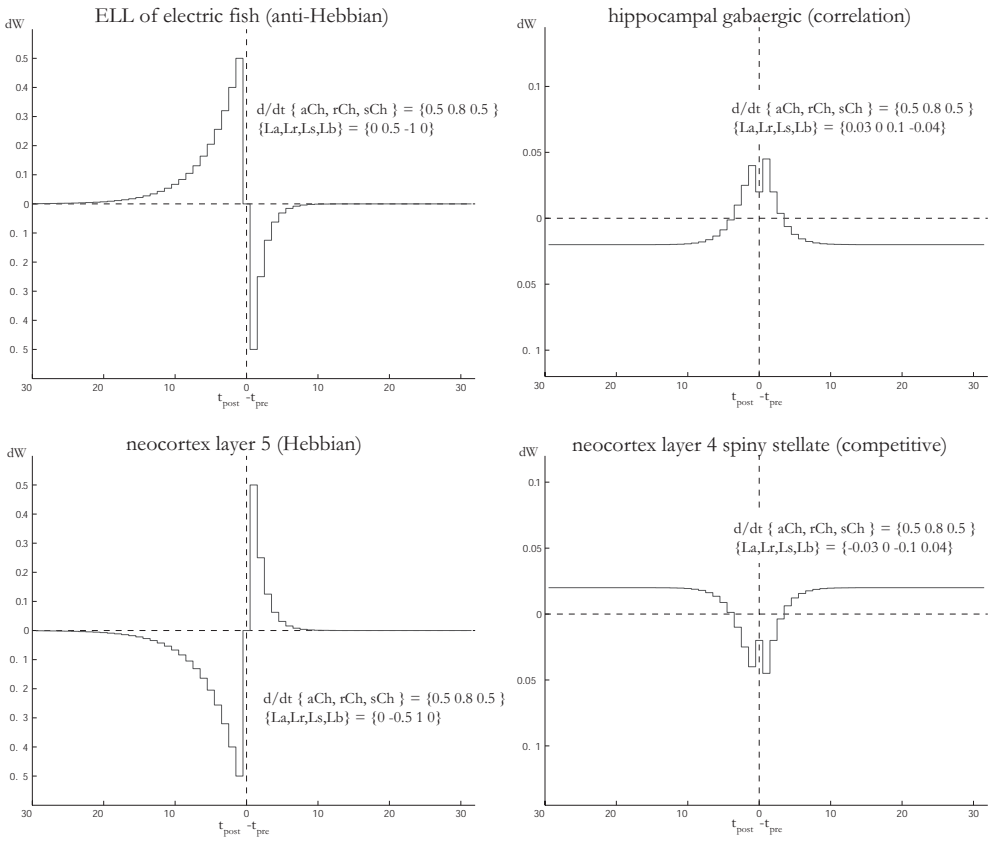


Figure 3: Some of the possible dW curves achievable by the proposed learning model, named after their biological counterparts. Learning is triggered by electrical activity at the post-synaptic level. The efficacy change depends on the local concentration of activity, refractory and synaptic chemicals and on the learning constants of the synaptic tree.

anti-Hebbian learning.

To model a wide range of learning rules still keeping the number of parameters to a minimum we propose a correlation ruled based on the local concentrations of synaptic, activity, and refractory chemicals. Whenever a synapse experiences electric activity, either because it was activated by neuro-transmitters or because a spike was generated in the post-synaptic neuron, the synaptic efficacy w is updated:

$$dw \quad = \quad L_a \text{ aCH} + L_r \text{ rCH} + L_s \text{ sCH} + L_b$$

with $\{L_a, L_r, L_s, L_b\} \in (-1, 1)$. Figure 3 shows some of the possible learning curves, named after their biological counterparts [1]. Compared to other implementation of Hebbian rules this method method is (1) activated only by post-synaptic electrical activity and (2) only driven by information available in the post-synaptic dendrite.

4 Simulations: wall avoidance

Here we present results from evolutionary runs, each consisting of 100 generations, 100 individuals and 25% elitism. The fitness is computed taking the average performance from the 4 worst of 6 independent runs, consisting of 200 100ms activation phases in a 50x50cm box. The fitness rewards

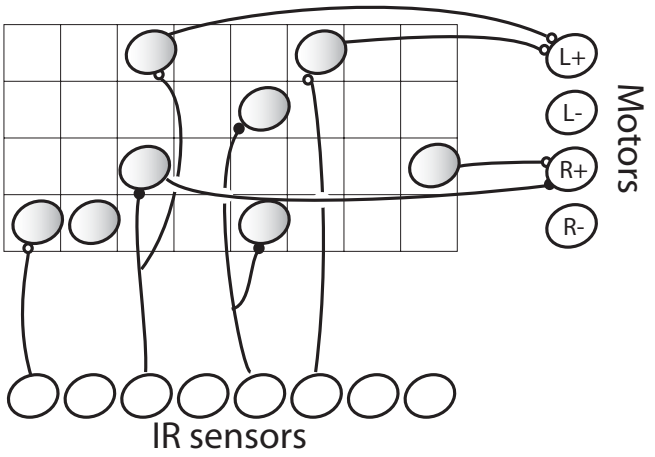


Figure 4: Structure of a layer of bipolar spiking neurons receiving input from 8 IR sensors and connecting to the actuators of Khepera's motors. The input (w_i) and output weights (w_o) are encoded by the type of the cell, while θ from the cell metabolism. All other parameters are fixed.

fast and straight movement and is computed at each step as:

$$Fit = \sum_i (Wheel_l^+ + Wheel_r^+)\,(.5\,(1 - abs(Wheel_l - Wheel_r)))^{\frac{1}{4}}$$

where $Wheel_i^+$ is set to zero if the wheel i spins backward, $(Wheel_l^+ + Wheel_r^+)$ measures the forward speed of the Khepera, and $1 - abs(Wheel_l - Wheel_r)$ is maximal when the robot proceeds straight.

The offspring is generated applying Gaussian noise with .1 variance to each Morpher weight with .05 probability. Crossover is applied to 10% of the population and is produced by shuffling the parents' output subnets. Finally, to reduce convergence, individuals are grouped in families based on their phenotypic similarity (see [9] for details), with the effect of rewarding less common phenotypes.

With development, the initial zygote cell replicates and differentiate to grow the mature multi-cellular phenotype. Each cell is specified by its discrete type and real-valued metabolic chemical concentrations. The cell type is itself a 3-valued vector (base 3), whose size is determined beforehand, depending on the necessities of the spiking neural model.

4.1 bipolar cells with IR sensors

Each cell of the multi-cellular organism represents a bipolar neuron connecting an IR sensor to a motor neuron. The organism is a layer of neurons N_{IR} wide and 4 high, where N_{IR} is the number of infrared sensors (8 in this case). As Figure 4 shows, the neurons are connected to the input (X) and output (Y) layers based on their 2D position. The normalized activity of the motor neurons calculates the speed of each wheel of the Khepera robot:

$$\text{Wheel}_l = .9\,L_+ + .1\,L_-; \qquad \text{Wheel}_r = .9\,R_+ + .1\,R_-;$$

where $L_\pm$ and $R_\pm$ is the concentration of the activity chemical (aCH) in the non-spiking motor neurons. Notice that aCH can vary between -1 and 1.

To encode a bipolar neuron, each cell is specified by a 2D vector of ternary values and a single metabolic concentration. The type specifies the efficacy of the afferent (w_i) and output (w_o) synapses $w \in \{+1, 0, -1\}$; the chemical concentration encodes the single threshold value $\theta \in [-1, 1]$. The other neural parameters are fixed (see Figure 6).

Figure 6 shows the best evolved controller and its behavior (top), and the fitness scores of the best individuals of each of 20 independent populations. Results show that in average, it takes less

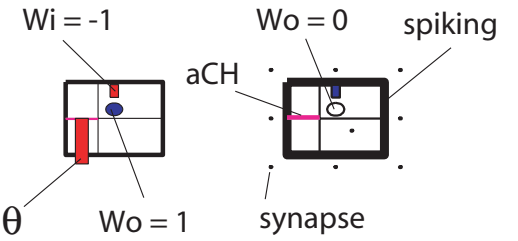


Figure 5: Explanation of the symbols used in the depiction of evolved neuro-controllers.

then 10 generations to produce controllers capable of wall-avoidance (fitness $\geq$ .40ca). After that, fitness can be increased only by achieving faster and straighter movements. Even if the task is relatively simple, the evolutionary speed is still very high when compared to related work [12].

4.2 tripolar cells with IR sensors: adaptive weights

In this case we add a plastic dendritic tree to the bipolar neuron. The tree can form horizontal local connections to the the nine neurons around its position (itself and the eight neighbors). All the synaptic efficacies are initialized to a small value (.1) and will be allowed to change in $[0, 1]$ following the learning rule specified by each cell. In this case, each cell has to specify 1 additional ternary discrete value for the sign of the adaptive synapses $w_h \in \{+, not\, active, -\}$, and 4 real values for the learning constants of $L_a \in \{-.1, 0, .1\}$, $L_r \in \{-.5, 0, .5\}$, $L_s \in \{-1, 0, 1\}$ and $L_b \in \{-.1, 0, .1\}$. Therefore, in order for the mature phenotype to encode tripolar neurons, each cell must contain a 3-dimensional 3-valued type vector and 5 real valued metabolic chemicals: more than double the requirements of bipolar neurons.

Even though, results show a higher performance, see Figure 7. Horizontal connections are not necessary in this type of task, nevertheless adaptive connections are exploited to reduce the network compexity.

For example, the adaptive weights of the two neurons connected to the right wheel in the controller of Figure 7 usually converge to 0 quite early. At that point the two neurons are activated and depressed always at the same time to produce fast clockwise turns. But depending on the initial starting position, still active horizontal connections produce a decoupling of the two neurons to produce earlier and smother turns (arrow in figure). As a result, fitter trajectories can be produced.

4.3 Comparison with direct encoding: performance

The modularity of the presented spiking neuron model makes it suitable for developmental multicellular systems. Still, it is interesting to investigate whether with a direct encoding of the phenotype similar results are achieved.

For this purpose, in this section we evolve directly the neuro-controller phenotypes bypassing the ontogenetic process.

The genotype is one-to-one representation of the phenotype: a N_{IR}x4 matrix with each position containing a cell. Akin to development, cells contain both discrete types and metabolic chemical concentrations. Mutation operates with a direct change of each independent cell attribute with a probability (m_p), affecting in average 10 attributes per mutation (binomial distribution). When affected, discrete attributes are randomized to a new value, continuous ones are perturbed with Gaussian noise with .001 variance (m_σ). The value of the variance has been found optimal among the following:

m_σ	0.05	0.005	0.001	0.0005	0.0001
mean (max) % fitness	27 (32)	62 (77)	70 (78)	70 (77)	68 (81)

All the other evolutionary details remain unchanged.

Compared to the results obtained with development, direct encoding shows a slower fitness increase, even if both systems converge to the same performance values. This suggests that both

systems search in comparable solution spaces, with development not being restricted to only a few viable solutions. In fact, direct analysis of the best neuro-controllers from all populations show always different individuals. On the other hand, the best networks produced with development contain statistically fewer neurons ($p < 10^{-3}$): in average 8.22 against 15.2.

4.4 Scalability

The 8 IR sensors of a Khepera robot offer a really minimalistic platform for the evolution of embodied agents. Sensors in animals are more typically counted by the millions.

Development systems have been proposed as a solution to the scalability problem of high dimensional solution spaces. In the evolution of specific 2D patterns, the development system used in this paper has been shown to scale well to bigger phenotypes, with evolutionary performance showing high correlation with the complexity of the targets [25, 11]. In this section, we test scalability for larger neuro-controllers.

The number of IR sensors can be arbitrarily increased in simulation, at the same time increasing the width of the neuro-controller (N_{IR}). When 8 IR are used, they are placed according to the standard Khepera specifications [**?**]. When more sensors are used, they are equally distributed over the robot circumference.

The difficulty of the task remains unchanged since, simply by ignoring the additional sensors, the previous controllers would still work. On the other side, evolution must cope with bigger networks, whose refinement could impose a liability for overall performance.

Figure 9 shows that, while with direct encoding, performance degrades dramatically with the size of the phenotype, with development, this effect is more contained.

Development is capable of exploiting the intrinsic regularity of good controllers to accelerate the discovery of proper growth programs.

4.5 Development: Regularity and Performance

In general, development friendly targets are those that present some exploitable regularity [25, 11].

In the presented simulations, IR sensors are arranged in the input vector in an clockwise fashion, with the first position of the input vector taken by the leftmost rear sensor. In this section, the evolvability of development is tested when this orderly arrangement is replaced by a random one.

For every evolutionary run an independent random permutation of the sensor's positions is generated. The performance computed over 20 runs with bipolar spiking neural networks with 64 and 128 randomly arranged sensors is shown below:

IR sensors:	64		128	
	max	average	max	average
clockwise order	.84	.63	.81	.60
random order	.63	.36	.72	.33

With random sensors' arrangement, the performance of the developmental system is greatly reduced. Both with 64 and 128 sensors, results are statistically significant, with averages differing for more than 2 standard deviations.

Taken from the perspective of the robot's behavior, with an ordered disposition two neighboring sensors carry a similar information. In fact, they point at angles that differ by only $2\pi/N_{IR}$: 5.6^o for 64 and 2.8^o for 128 sensors.

The role of a neuron, in terms of the input it receives and the behavior it triggers, will be often similar to the role of its neighbors. Since neuron's functionality is provided by the type and metabolism of the corresponding cell, neighboring cells will share similarities with the consequence that fit organisms will have more regular shapes.

Quite opposite is the case of randomly placed sensors. Here evolution must find a development program capable of producing organisms for which neighboring cells encode neurons of probably unrelated functionality. Fit organisms appear less regular with clear results in terms of performance.

5 Conclusions

In this paper we have introduced a novel evolutionary friendly adaptive spiking network model. The discrete time neural model is based on the leaky integrate and fire spiking neural network [13]. Its adaptive rules are based on the correlation between post-synaptic electric activity and the local concentrations of synaptic, activity and refractory chemicals. Compared to standard Hebbian and anti-Hebbian learning rules, the model presented in this paper can produce more types of learning curves comprising, but not being limited to, those found in various animals [1].

The adaptive mechanism is also computationally efficient and requires only four additional parameters per learning rule. This is particularly interesting since efficient rules are fundamental for evolutionary applications.

The spiking neural model has been used to control IR based navigation for a simulated Khepera robot. Neuro-controllers have been evolved both with multi-cellular development with embryonal stages [10, 11] and direct encoding. While both strategies can produce good controllers, development appears more parsimonious and shows a steeper performance increase.

Also, when applied to the evolution of Khepera robots equipped with extra IR sensors, the developmental system displays a sustained high performance, in accordance with what presented in the evolution of 2D patterns [25, 11]. This is true as long as sensors are presented to the network in an orderly fashion. With neighboring cells having a similar role on overall behavior, evolution takes advantage of the regularity of fit organisms.

Adaptivity is not necessary for the navigation task, still the introduction of horizontal learning synaptic trees has proved beneficial to the evolutionary speed of the development system, despite the fact that the number of parameters defining a neuro-controller is more than doubled. An opposite picture emerges with direct encoding, where the larger search space prevails against the increased versatility of the adaptive networks.

As a conclusion, we believe, that efficient multi-cellular development offers a great evolutionary advantage in the discovery of large networks. It must be noted that the approach adopted in the presented model, differs form the traditional 'evolution of weight matrices' (e.g. the matrix rewriting scheme [19]). In our case, weights are encoded indirectly as excitatory / inhibitory connections, learning rules and the topological properties of mature phenotypes. The result is that neuro-controllers can be produced exploiting the intrinsic regularity of good solutions.

References

[1] Abbott, L. and S. Nelson: 2000, 'Synaptic Plasticity: Taming the Beast'. *Nature Neuroscience* pp. 3:1178–1183.

[2] Bar-Yam, Y.: 1997, *Dynamics of Complex Systems*. Perseuas Books, Reading (Ma).

[3] Bentley, P.: 2003, 'Evolving Fractal Gene Regulatory Networks for Robot Control'. *Proceeding of the European Congress of Artificila Life, ECAL 2003* pp. 753–762.

[4] Cangelosi, A., S. Nolfi, and D. Parisi: 1994, 'Cell division and migration in a 'genotype' for neural networks'. *Network* pp. 5:497–515.

[5] Carew, T.: 2000, *Behavioral Neurobilogy*. Sinauer Associated Inc.

[6] Dellaert, F. and R. Beer: 1994, 'Toward an Evolvable Model of Development for Autonomous Agent Synthesis'. *Proceedings of Artificial Life IV* pp. 246–257.

[7] Dellaert, F. and R. Beer: 1996, 'A developmental model for the evolution of complete autonomous agents'. *Proceeding of Simulated Adaptive Behaviour IV, SAB 1996)* pp. 393–401.

[8] E. Mjolsness, D.H. Sharp, J. R.: 1991, 'A connectionist model of development'. *Journal of Theoretical Biology* pp. 152(4):429–453.

[9] Federici, D.: 2004a, 'Evolving a neurocontroller through a process of embryogeny'. *in proceedings of Simulated Adaptive Behaviour, SAB 2004*.

[10] Federici, D.: 2004b, 'Increasing evolvability for developmental programs'. *proceeding of WORLDS 2004*.

[11] Federici, D.: forthcoming 2005, 'Evolution and Development of a Multi-Cellular Organism: Scalability, Resilience and Neutral Complexification'. *Submitted to Artificial Life Journal*.

[12] Floreano, D. and F. Mondada: 1994, 'Automatic creation of an autonomous agent: Genetic evolution of a neural-network driven robot'. *Proceedings of the Third International Conference on Simulation of Adaptive Behavior, SAB 94*.

[13] Gerstner, W. and W. Kistler: 2002, *Spiking Neuron Models, Single Neurons, Populations, Plasticity*. Cambridge University Press.

[14] Gruau, F.: 1994, *Neural Network Synthesis using Cellular Encoding and the Genetic Algorithm*. PhD Thesis, Ecole Normale Superieure de Lyon.

[15] Hornby, G. S. and J. B. Pollack: 2001, 'The Advantages of Generative Grammatical Encodings for Physical Design'. In: *Proceedings of the 2001 Congress on Evolutionary Computation, CEC 2001*. pp. 600–607.

[16] Jacobi, N.: 1995, 'Harnessing morphogenesis'. *Proceeding of the International Conference on Information Processing in Cells and Tissues* pp. 29–41.

[17] Kandel, E., J. Schwartz, and T. Jessell: 2000, *Principles of Neural Science*. McGraw-Hill.

[18] Kauffman, S.: 1993, *The origins of order*. New York: Oxford University Press.

[19] Kitano, H.: 1990, 'Designing neural networks using genetic algorithms with graph generation system'. *Complex Systems* pp. 4(4):461–476.

[20] Luke, S. and L. Spector: 1996, 'Evolving Graphs and Networks with Edge Encoding: Preliminary Report'. In: J. R. Koza (ed.): *Late Breaking Papers at the Genetic Programming 1996 Conference*. pp. 117–124.

[21] Markram, H., J. Lübke, M. Frotscher, and B. Sakmann: 1997, 'Regulation of Synaptic Efficacy by Coincidence of Post-synaptic APs and EPSPs'. *Science* pp. 275:213–215.

[22] Mattiussi, C. and D. Floreano: 2004, 'Connecting Transistors and Proteins'. *ALife9: Proceedings of the Ninth International Conference on Artificial Life*.

[23] Miller, J.: 2003, 'Evolving developmental programs for adaptation, morphogenesys, and self-repair'. *Proceeding of the European Congress of Artificial Life, ECAL 2003* pp. 256–265.

[24] Ohno, S.: 1970, *Evolution by Gene Duplication*. Springer.

[25] Roggen, D. and D. Federici: 2004, 'Multi-cellular development: is there scalability and robustness to gain?'. *proceedings of Parallel Problem Solving from Nature, PPSN VIII 2004*.

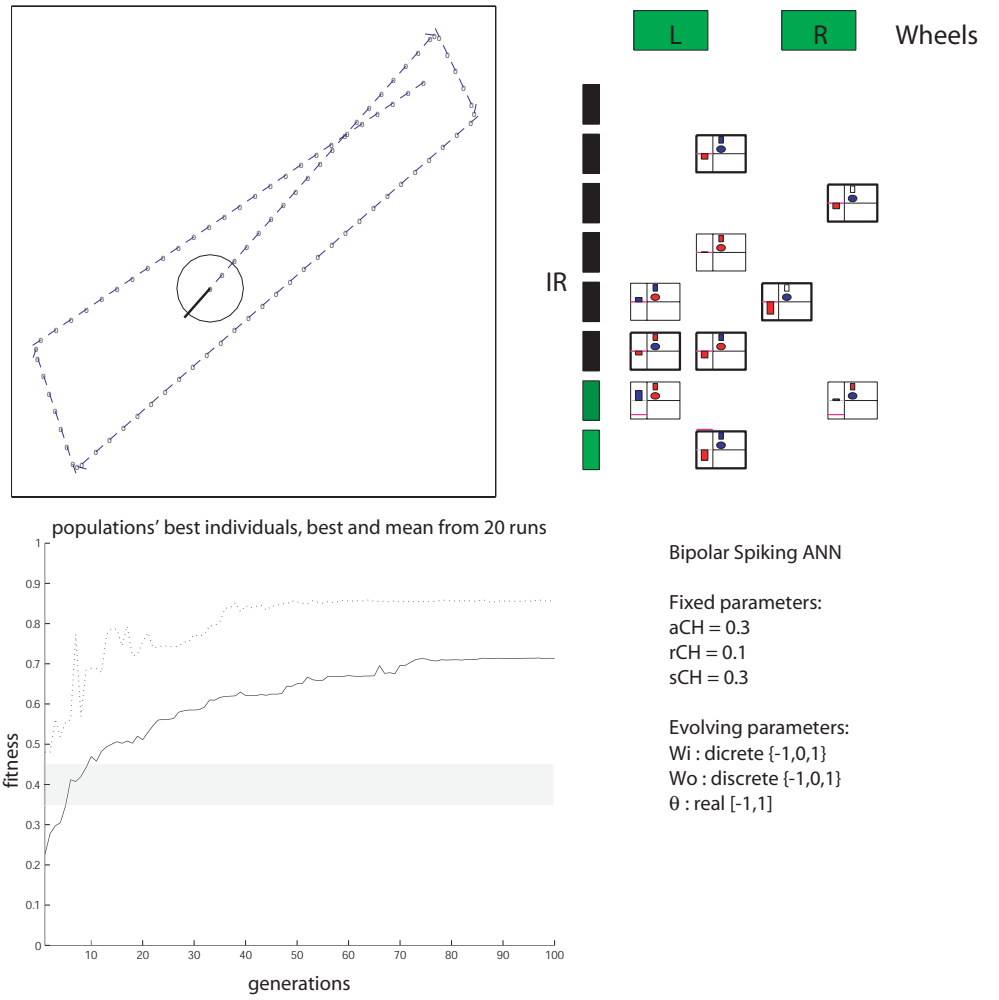


Figure 6: Evolving a development program for Kheperas spiking neuro-controllers: bipolar neurons. Evolution selects for straight and fast navigation. Fitness scores above .40ca require the ability to detect and avoid walls. In Figure: the best scoring bipolar controller (top right) and the behavior it generates (top left).

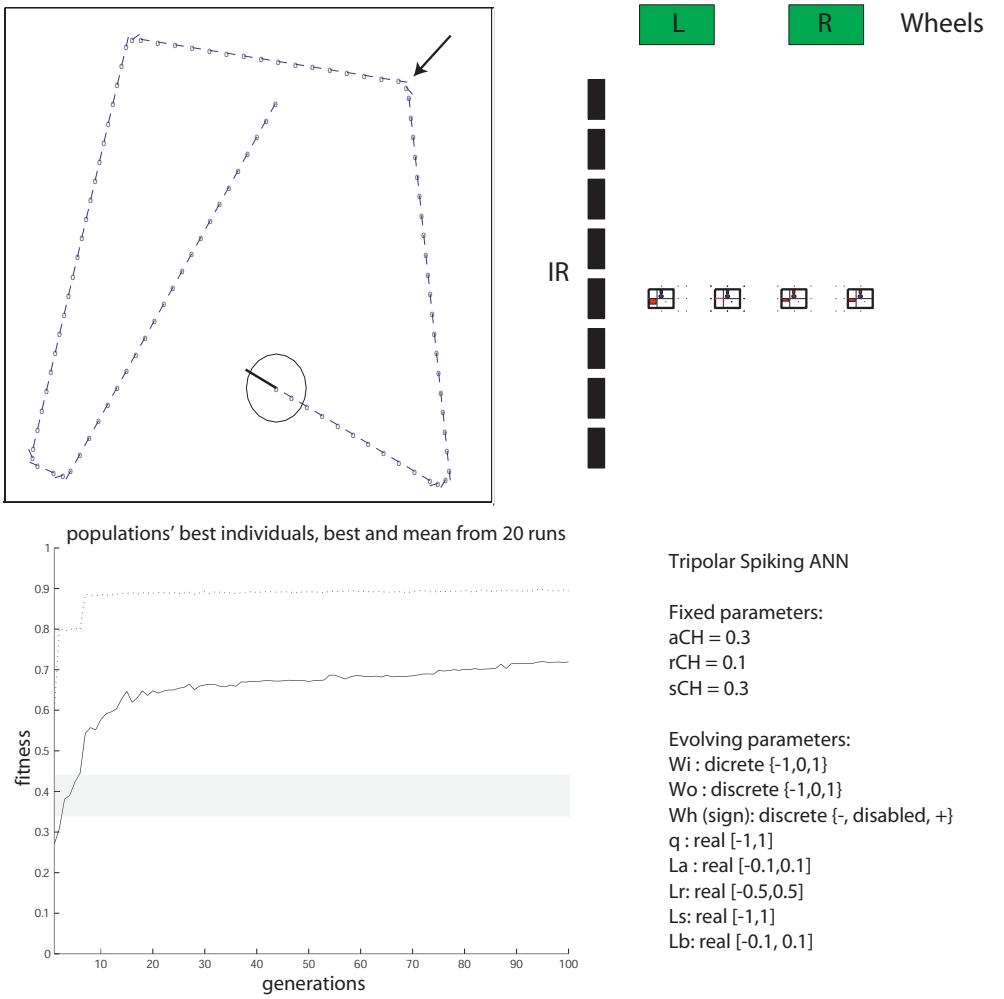


Figure 7: Evolving a development program for Kheperas spiking neuro-controllers: tripolar neurons. Evolution selects for straight and fast navigation. Fitness scores above .40ca require the ability to detect and avoid walls. In Figure: the best scoring tripolar controller (top right) and the behavior it generates (top left). The arrow highlights the only apparent use of the horizontal connections.

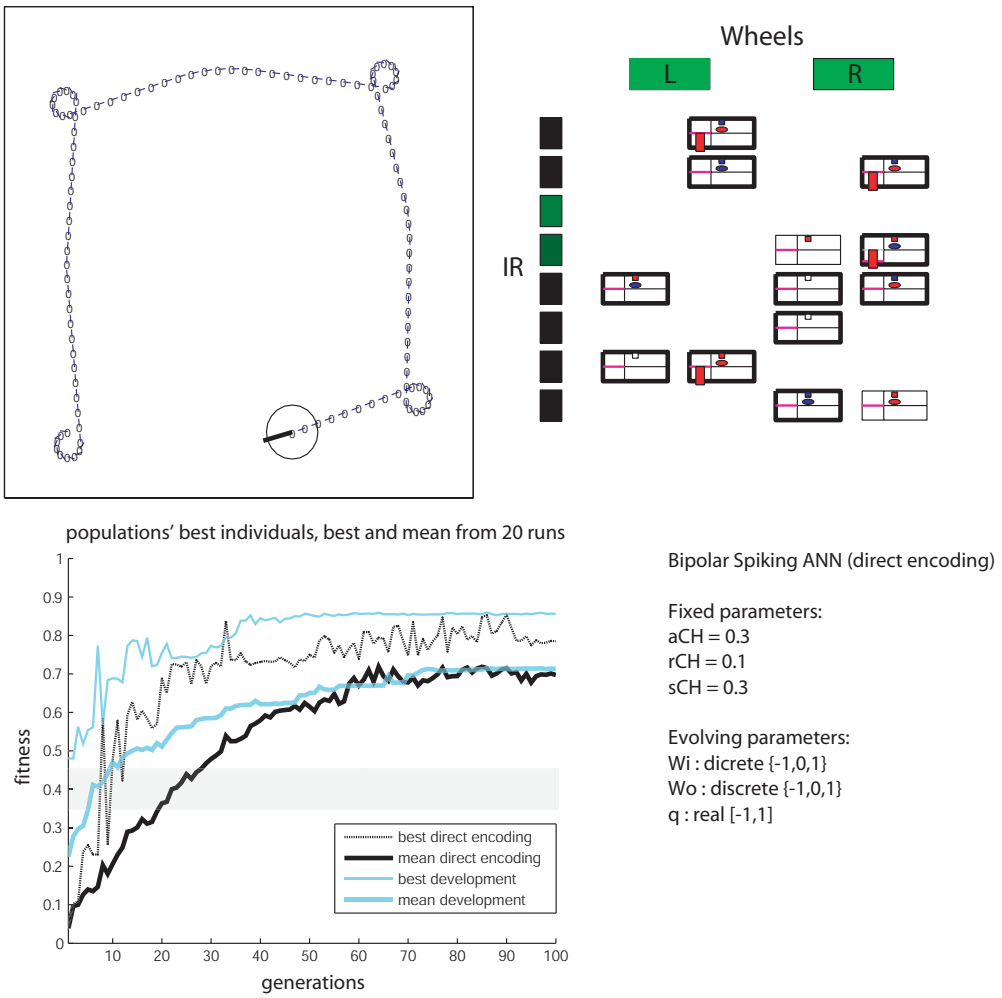


Figure 8: Evolving direct encoded spiking neuro-controllers: bipolar neurons. Evolution selects for straight and fast navigation. Fitness scores above .40ca require the ability to detect and avoid walls. In Figure: the best scoring bipolar controller (top right) and the behavior it generates (top left). Results are also compared with those obtained with development.

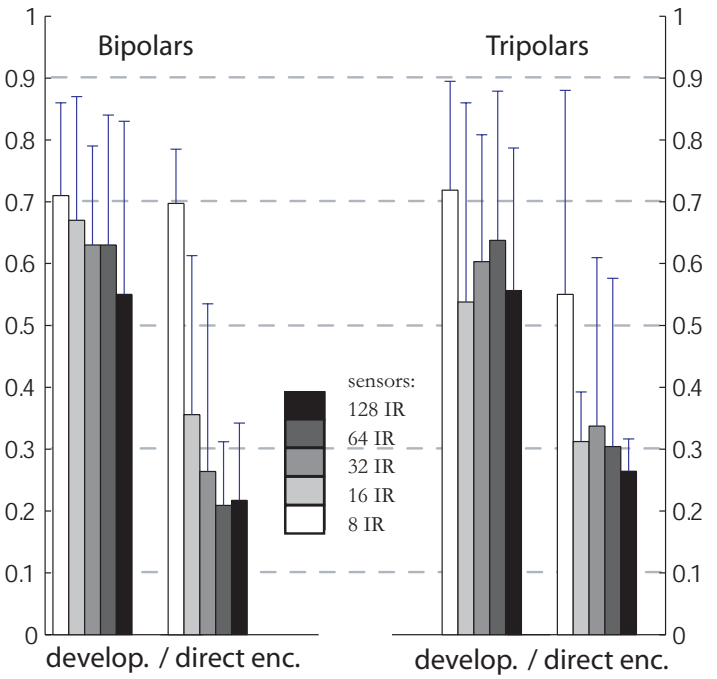


Figure 9: Average and maximum performance from 20 simulations for each parameters setting with bipolar and tripolar neurons. Comparison between the performance of evolution by direct encoding and development for varying neuro-controller sizes.

J Fault-tolerance by Regeneration: Using Development to Achieve Robust Self-Healing Neural Networks

Author: Diego Federici. technical report: submitted to the International Joint Conference on Neural Networks, IJCNN 2005

Abstract:

Opposed to the standard paradigm of 'fault-tolerance by redundancy', ontogeny offers the possibility to engineer artificial organisms which can re-grow faulty components. Similar to what happens in nature, organisms display self-healing: an homeostatic process which allows proper operation while suffering faults.
In this paper we present a system which evolves developing spiking neural networks capable of controlling simulated Khepera robots in a wall avoidance task. Development is controlled by a decentralized process executed by each cell identical growth program.
To test the system's self-healing capability, networks are (1) subjected to random faults during development and (2) mutilated during operation.
Results demonstrate how development can (i) rapidly produce proper neuro-controllers and (ii) re-grow neurons to recover normal operation.
These results show that development, originally proposed to increase the evolvability of large phenotypes, also allows the production of artifacts with sustained fault-tolerance. These artifacts would be especially well-suited for tasks that require long periods of operation in absence of external maintenance.

Objective:

Test the self-healing properties of development on a functional neural network organism.

Conclusions:

Self-healing is achieved by a functional homeostasis: similar to what happens in biological organisms, evolved developmental programs control the appearances of faults with a continuous cellular genesis.

Fault-tolerance by Regeneration: Using Development to Achieve Robust Self-Healing Neural Networks

Diego Federici
Complex Adaptive Organically-inspired Systems group (CAOS)
Norwegian University of Science and Technology
Department of computer and information science
N-7491 Trondheim, Norway

***Abstract*— Opposed to the standard paradigm of 'fault-tolerance by redundancy', ontogeny offers the possibility to engineer artificial organisms which can re-grow faulty components. Similar to what happens in nature, organisms display self-healing: an homeostatic process which allows proper operation while suffering faults.**

In this paper we present a system which evolves developing spiking neural networks capable of controlling simulated Khepera robots in a wall avoidance task. Development is controlled by a decentralized process executed by each cell's identical growth program.

To test the system's self-healing capability, networks are (1) subjected to random faults during development and (2) mutilated during operation.

Results demonstrate how development can (i) rapidly produce proper neuro-controllers and (ii) re-grow neurons to recover normal operation.

These results show that development, originally proposed to increase the evolvability of large phenotypes, also allows the production of artifacts with sustained fault-tolerance. These artifacts would be especially well-suited for tasks that require long periods of operation in absence of external maintenance.

I. Introduction

Due to the fact that their functionality is distributed, it is well known that neural networks tend to display a good resistance to damage: with the loss of a few nodes or connections the system operation is often preserved to an acceptable degree.

Still, living organisms present an additional source of robustness which is derived by their self-healing capabilities. Common are the examples of the lizard's tail and the limb of a salamander, which can re-grow after being severed. In truth, most living organisms present a continuous regeneration in order to withstand the interaction with the physical world.

Having to operate for a long time, there is a clear evolutionary advantage for organisms capable of self-healing. In this case, self-healing is de facto an adaptation to frequent sources of faults typical of hazardous environments.

Recent results have shown that the central nervous system is not an exception. Stem cells are constantly produced in the Substantia Nigra and its malfunctioning is being associated to Parkinson's disease [18]. It has also been shown that damaged light receptors are regenerated [17].

Opposed to the more classical engineering principle of fault-tolerance by redundancy, the picture that emerges from living organisms is of a fault-tolerance by regeneration. The advantage is that such systems can in principle operate without limits of time and be deployed in hazardous environments without the need of external technical assistance.

Work presented in [5], [16] has shown how the evolution of large phenotypes of specific shapes can benefit from the use of an indirect genetic encoding scheme based on development. With a gene required for each phenotypic trait, the more traditional direct encoding shows an exponential increase of search spaces as larger phenotypes are being evolved. On the contrary, by evolving a growth program, the size of genotype can be highly independent from the size of the phenotype. In fact, with development, genes can be recursively used multiple times.

Additionally, as a side effect, developing organisms also display emergent self-repair: dead (faulty) cells appear to be regenerated by the original growth program [14], [5], [16]. Opposed to the view of fault-tolerance achieved by redundancy, this emergent feature of development would allow the design of hardware and software devices which would re-grow damaged components.

Previous analysis has focused on static 2D patterns, leaving open the question about the applicability of the method in the evolution of functional organisms. In this paper, the Artificial Embryogeny model of [5], [16] is used to create Spiking Neural Networks (SNN) to control simulated Khepera robots. The introduced plastic SNN model has been designed to offer high evolvability under development.

Results show that not only the method is capable of producing very effectively proper neuro-controllers but also, these controllers are capable of recovering from damage with a regenerative process reminding the one seen in natural organisms. This is valid for networks with both plastic and non-plastic synapses.

A. related work

Methods directed towards the evolution of ANNs have always been confronted with a dimensionality problem. The

number of inter-connections typically grows quadratically with the size of the neural network layers, while the possible connections among layers grows combinatorially.

Instead of evolving a genotype containing a direct representation of each connection, it is possible to evolve the parameters specifying a generative process which will determine each connection's topology and efficacy. In such a way, the combinatorial explosion of the search space can be prevented and evolution can proceed within tractable limits.

For example, it is possible to evolve the rules of a grammar, which, recursively applied to the phenotype, will produce the mature network. Examples are provided by the Matrix Rewriting scheme [10], the Cellular Encoding [7], Edge Encoding [12] and the GenRe system [8].

Some models include additional contextual information in each rule definition [8], so that phenotypic traits variations can be generated.

Another approach is inspired upon Cellular Automata (CA). In this case, the target of evolution is the regulatory program of a cell. Through the interaction with a simulated environment, cells accordingly duplicate and differentiate to produce mature organisms. Cells are usually capable of sensing neighboring cells [4], releasing chemicals which either diffuse in the simulated environment [14] or are localized in the cytoplasm of the cell [5], grow selective connections to neighboring cells, and migrate [2].

Concerning fault-tolerance, the CA-based models presented in [14], [16] have shown emergent (neutral) self-repair properties. In [5] multi-cellular organisms showed increased resilience when additionally selected for that feature.

Still, searching in a restricted space, it is possible for good solutions to lay outside the reach of the development program. In fact, compression is generally higher for regular targets and development is de facto a decompression of the genotype. A serious question then is how much these methods are viable for the evolution of irregular targets.

Hints in this direction, also come from a study [11] on the Matrix Rewriting scheme [10], showing how the genotype-phenotype correlation decreases with the complexity of the phenotype: lower correlation is usually associated to less efficient evolutionary search.

Similar results have been presented in [5], showing that more complex (irregular) patterns were more difficult to evolve. Still, as the targets increased in size, development produced better results than direct encoding in all tested cases.

II. The Model

Spiking neuro-controllers are produced in 2 steps. The first recursive one uses the growth program encoded in the genotype to develop a multi-cellular phenotype. The latter, translates each cell into a spiking neuron.

A. Multi-Cellular Development

Organisms develop starting from a single cell (zygote) to reach maturation in a precise number of developmental steps. Cells replicate and can release simulated chemicals in intracellular space (metabolism).

At each developmental step, existing active cells can change their own type, alter their metabolism and produce up to four new cells in any of the cardinal directions North, East, South and West. An active cell can also choose to die or become passive. Once passive, cells can no longer change their state or produce new cells.

The mother cell specifies each new cell type and metabolism, and whether they are active or passive. If necessary, existing cells are pushed sideways to create space for the new cells. When a cell is pushed outside the boundaries of the grid it is permanently lost.

Each cell's behaviour is governed by a unique growth program based on local variables, and implemented by a simple recursive neural network (Morpher) with linear transfer function and 4 hidden nodes. Recursive connections are provided by each cell type and metabolism.

The Morpher is directly specified by the genotype, which contains a floating point number for each synaptic weight. Its inputs are the current cell type and metabolism, and the cell types of the neighboring cells in the four cardinal directions. Its output determines its new type and metabolism and, in case of replication, each type and metabolism of the newly generated cells. An additional local variable, the cell age, is set to 1 at birth and decays exponentially.

To increase the evolvability of development, a method based on Embryonal Stages is also used. Development with Embryonal Stages (DES) allows the Neutral Complexification of the genotype. Similar to gene duplication in biological organisms [15], DES increases the sophistication of evolved phenotypes by adding new chromosomes to the genotype.

Each chromosome encodes a complete Morpher and controls a predefined development phase, i.e. a range of development steps. By increasing the time-specificity of development (heterochrony) DES increases the sophistication of evolved phenotypes [5]. DES operates as follows:

Set Up: all initial genotypes contain a single chromosome. The corresponding Morpher controls development from the first step to the last.

Gene Duplication: during evolution, a new chromosome can be inserted. Each new chromosome is an exact duplicate of the latest added one. The corresponding duplicated Morpher is responsible for development of a share of the development steps of its duplicate. For example, imagine that Morpher M_i controls development from step k to j. When the duplicated Morpher M_{i+1} is added, M_i will control the development steps from k to $(k+j)/2$, while M_{i+1} will control steps from $(k+j)/2+1$ to j. The total number of development steps always remains constant.

New chromosomes are always associated to the final steps of development. Being exact copies, at first new chromosomes do not alter development, and are therefore neutral. But eventual successive mutations can independently affect each independent chromosomes.

In this paper, only the latest added chromosome is subjected

to evolutionary search. All others remain fixed. A more extensive description of the developmental model, and in particular of Development with Embryonal Stages, can be found in [5].

B. from cells to neurons

Each cell of an organism is a neuron of a spiking neuro-controller. The type and metabolic concentrations of a cell are used to specify the internal dynamics and synaptic properties of its corresponding neuron. The position of the cell within the organism is used to produce the topological properties of neuron: its connections to inputs, outputs and other neurons.

The cell type is a discrete ternary vector $\{-1, 0, 1\}^{N_S}$ and is used to determine the sign of the N_S synaptic connections. The metabolism is a real valued vector $[-1, 1]^{N_P}$ and is used to specify the neuron threshold and its learning parameters.

The actual values of N_S and N_P can vary depending on the needs of the specif spiking neuro-controller adopted. In this paper two neuro-controller models are used.

Networks of **bipolar** cells have a single input synaptic tree with discrete efficacy ($\{-1, 0, 1\}$), a single axonal output with discrete efficacy ($\{-1, 0, 1\}$, and a threshold value ($[-1, 1]$). For a bipolar cell $N_S = 2$ and $N_P = 1$. To produce bipolar cells, the morpher requires 12 inputs and 20 outputs.

Networks of **tripolar** cells are similar to the bipolar ones, but an additional horizontal dendritic tree is present. The plastic tree connects to the 8 most proximal neurons only and is specified by its learning rule (4 real variables) and sign ($\{-1, 0, 1\}$). For tripolar cells $N_S = 3$ and $N_P = 5$. To produce bipolar cells, the morpher requires 16 inputs and 45 outputs.

Finally, the 2D position of a cell in the mature organism specifies the neuron's input and output connections. A neuron placed in (x, y) is connected to input x and output y. For tripolar neurons, horizontal connections spread only to the 8 closest neighbors, see Figure 1.

The normalized activity of the motor neurons calculates the speed of each wheel of the Khepera robot:

$$W_l = .9\, L_+ + .1\, L_-; \qquad W_r = .9\, R_+ + .1\, R_-;$$

where $L_\pm$ and $R_\pm$ is the concentration of the activity chemical (aCH, see below and Figure 2) in the non-spiking motor neurons multiplied by their output weights.

C. spiking neuro-controller

The presented discrete-time spiking neuron model is a leaky integrate and fire neuron implementation, similar to the one introduced in [6]. The behavior of the spiking neuron j is computed following equations in Figure 2.

In this implementation the synaptic efficacy w is either a fixed value $\in$ {-1, 0, 1} or can vary freely within $\pm[0, 1]$ subjected to pre/post synaptic correlation plasticity (e.g. Hebbian learning).

In biological neurons plasticity is typically associated to the presence of post synaptic activity, either derived from afferent neuro-transmitters or reverberating post-synaptic potentials, in conjunction with second messengers (e.g. cAMP) or local

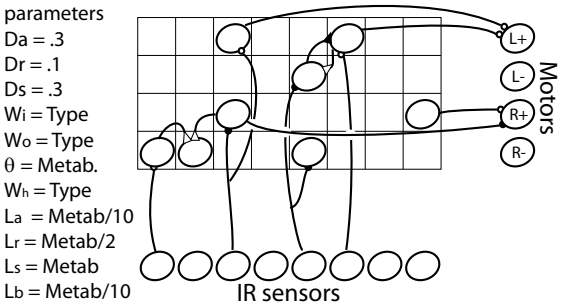


Fig. 1. Structure of a layer of tripolar spiking neurons receiving input from 8 IR sensors and connecting to the 4 actuators of Khepera's motors. The (x,y) position of the cell specifies the input (x) and output (y) connectivity. The input (w_i), hidden (w_h) and output weights (w_o) are encoded by the type of the cell and are fixed discrete values {-1,0,1}, the neuron's threshold $\theta \in [-1, 1]$ is encoded by the cell metabolism. For the plastic horizontal connections, the learning constants $\{L_a,\, L_r,\, L_s,\, L_b\}$ are also given by the cell metabolism. All other parameters are fixed. Bipolars are similar to tripolar spiking neurons but they lack the plastic horizontal connections.

chemicals (e.g. Ca^{++}) [9], [3], [13]. As a result a wide spectrum of learning behaviors are possible [1], including, but only limited to, Hebbian and anti-Hebbian learning.

To model a wide range of learning rules still keeping the number of parameters to a minimum we propose a correlation ruled based on the local concentrations of synaptic, activity, and refractory chemicals. Whenever a synapse experiences electric activity, either because it was activated by neuro-transmitters or because a spike was generated in the post-synaptic neuron, the synaptic efficacy w is updated:

$$dw \;=\; L_a\ \mathrm{aCH} + L_r\ \mathrm{rCH} + L_s\ \mathrm{sCH} + L_b$$

with $\{L_a,\, L_r,\, L_s,\, L_b\} \in (-1, 1)$. This rule can produce, without being limited to, all the learning curves shown in [1].

D. evolution

Here we present results from evolutionary runs, each consisting of 200 generations, 100 individuals and 25% elitism. Mutation normally perturbs each weight with a .05 probability and .005 variance. 10% of the offspring is produced by multi-point crossover.

The fitness is computed taking the average performance from the 8 worst of 10 independent runs, consisting of 200 100ms activation steps in a 50x50cm box. The fitness rewards fast and straight movement and is computed at each step as:

$$Fit = \sum_i (W_l^+ + W_r^+)\,(.5\,(1 - abs(W_l - W_r)))^{\frac{1}{4}}$$

where W_i^+ is set to zero if the wheel i spins backward, $(W_l^+ + W_r^+)$ measures the forward speed of the Khepera, and $1 - abs(W_l - W_r)$ is maximal when the robot proceeds straight.

All the values read from the IR sensors of the simulated Khepera are perturbed with Gaussian noise with 0.05 variance.

To simulate external faults, at each developmental step every cell has a 1/20 probability to be removed (die). Also fitness

$$
\begin{aligned}
\text{synapse } i: \quad & \text{sCH}_i(t) && = \begin{cases} w_i + D_s\,\text{sCH}_i(t-1) & \text{with an incoming spike} \\ D_s\,\text{sCH}_i(t-1) & \text{else ways} \end{cases} \\
\text{activity of } j: \quad & \text{aCH}_j(t) && = D_a\,\text{aCH}_j(t-1) + \sum_{i \in Synapses} \text{sCH}_i(t) \\
\text{refract. of } j: \quad & \text{rCH}_j(t) && = D_r\,\text{rCH}_j(t-1) + S_j(t) \\
\text{spike from } j: \quad & S_j(t) && = \begin{cases} 1 & \text{if aCH}_j(t) > \theta \ \wedge \ \text{rCH}_j(t) < .1 \\ 0 & \text{else ways} \end{cases}
\end{aligned}
$$

Fig. 2. Equations determining the operation of the discrete-time leaky integrate and fire neuron. $\{D_a,\ D_r,\ D_s,\ \Theta\} \in [0, 1]$ are constants controlling the dynamic of the spiking neuron. In the simulations presented $\{D_a,\ D_r,\ D_s\}$ are set to {.3, .1, .3}, while Θ is evolved.

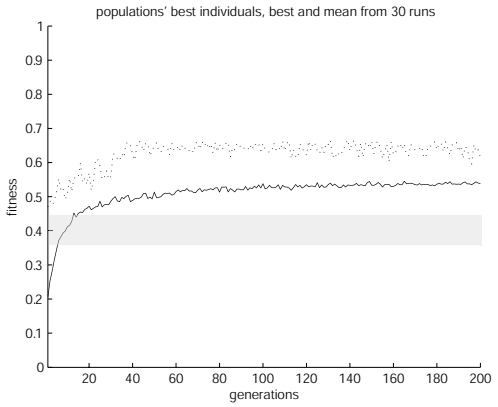


Fig. 3. Average an maximum fitness plot from 30 runs of bipolar neuro-controllers. Fitness scores above circa .40 require proper wall detection and avoidance.

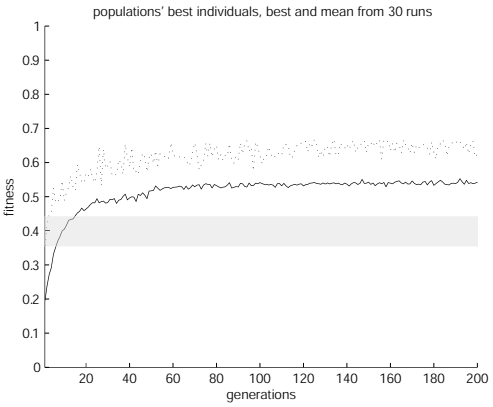


Fig. 4. Average an maximum fitness plot from 30 runs of tripolar neuro-controllers. Fitness scores above circa .40 require proper wall detection and avoidance.

tests are partitioned in two phases: after 100 activation steps controllers are mutilated: each cell is removed with a .25 probability[1]. Ontogeny is then resumed for three additional (healing) development steps before executing the last 100 activation steps.

A maximum of 3 chromosomes are possible, see Section II-A. New chromosomes are introduced at generations 66 and 133. When all present, the first controls development from step 1 to 5, the second from 6 to 7, and the third controls the healing phase (steps 8 to 10).

High fitness values result from the combination of fast straight movement, wall avoidance and recovery from neural damage both during and after development.

III. Results

Figures 3 and 4 show the average fitness plots from 30 evolutionary runs with populations of bipolar and tripolar controllers.

Of particular interest is the average number of generations needed to reach a fitness score of circa .40 (15-20 in both cases), since that level of performance requires proper wall avoidance. Also, both populations converge to similar fitness values: .62/.54 (max/mean) for bipolar networks and .63/.54 (max/mean) for tripolar ones.

[1]with the exception that at least 1 cell is always removed

A. Comparison to Populations Evolved without Faults

To test how difficult is to evolve the self-healing feature, bipolar and tripolar networks populations have been evolved without faults being applied (non fault-tolerant populations).

In this case, average performance reaches a .40 value in less than 10 generations. It appears then, that the additional regenerative capability is obtained at the expense of just a few more generations.

When testing the best evolved individuals in the absence of faults, we can compare the best-case performance of fault-tolerant (FT) and non fault-tolerant ($\overline{FT}$) populations. The average results from 30 evolved populations for each group are displayed in the following table (250 fitness tests):

network type	max/mean fitness	
	FT	$\overline{FT}$
bipolar	.86 / .70	.86 / .72
tripolar	.89 / .69	.89 / .72

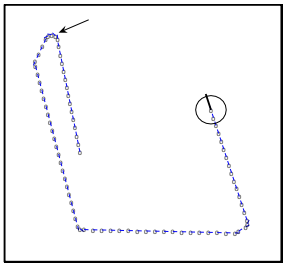


Fig. 5. Trace of the best evolved bot behaviour (tripolar network). Network without sustained faults. The arrow points at the only apparent use of horizontal plastic connections: an increased sensitivity to frontal wall detection. With the first smoother turn a fitter trajectory is obtained.

These results highlight that without faults, both FT and $\overline{FT}$ populations perform a comparably fit behaviour[2] consisting of fast straight movement with quick turns (see Figure 5). Tripolar networks sometimes manage to evolve plastic synaptic connections which are used to produce, on necessity, a single smooth turn (the arrow in Figure 5). With that, they can score a subtle extra .03 fitness.

B. Fault-Tolerance

There are two sources of faults: one takes place during each development step (cell mortality), the second occurs in the middle of the fitness evaluation (mutilation).

While cell mortality is intended to model faults randomly occurring during the configuration phase, mutilation tests the organism capability to recover from an accident during operation.

With varying sources of faults, the following table displays the fitness values from 250 independent tests of the best individual of each one of the 30 FT populations:

network type	average fitness $\pm$ std			
	no faults	only mutilation	only mortality	mortality + mutilation
bipolar	.70 $\pm$.10	.57 $\pm$.09	.60 $\pm$.08	.54 $\pm$.08
tripolar	.69 $\pm$.09	.57 $\pm$.11	.61 $\pm$.08	.54 $\pm$.09

Notice that, during the 10 developmental steps (7 to maturation plus 3 for the healing phase), mortality has a cumulated .40 chance to strike any alive cell, higher than the .25 death probability due to mutilation. Still, cell mortality strikes at random times, while mutilation takes place in a single shot. As a result, the effects of mutilation have somewhat heavier consequences on performance.

While there is statistical difference ($p < 0.01$) between the group tested without faults and the others, this is not true among the different sources of faults. In all cases, damaged individuals perform in average well above the .40 minimal requirement.

[2]Notice that fitness cannot reach the value of 1 because of the necessity to turn in the proximity of walls

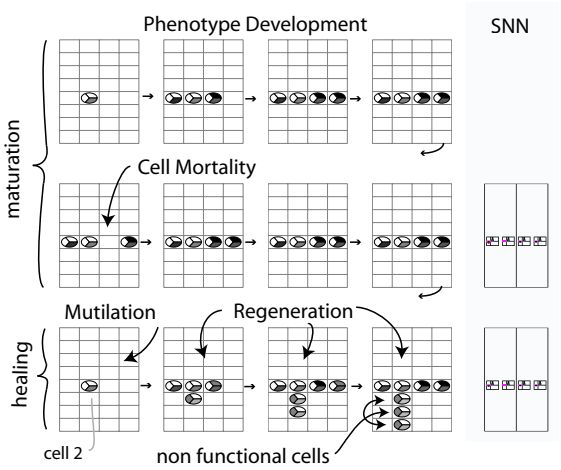


Fig. 6. Example of development of a bipolar fault tolerant organism. On the left the development steps (maturation and healing phases): cells variables (2 discrete types and 1 metabolic chemical) are represented by three shades of gray, ranging from black (-1) to white (+1). On the right the corresponding spiking neural network. Healthy organisms contain 4 functional cells. More cells are present after a mutilation event, still without affecting behaviour.

Finally, in all tested cases, bipolar and tripolar networks do not have any significant difference, suggesting that the tripolar horizontal plastic connections have no influence on the self-healing process.

C. Analysis of a Regenerating Organism

In this section, the mechanism that drives the neural regeneration process is analyzed. To perform an extensive analysis, the simplest bipolar controller among the best performing ones has been chosen, Figure 6.

In the chosen example, an healthy organism contains 4 cells and scores .86 fitness. The following table shows the fitness score achieved after various mutilation events:

recovered fitness	event probability	lost cells
0%	4.7%	(1 and 2)
1-10%	6.5%	(2 but not 1)
51-99%	22.0%	2 and (2 and 4)
100%	66.8%	any but not 2

88% of the times, mutilations are recovered to produce neuro-controllers which are either healthy (66.8%) or still capable of wall avoidance (22%).

These tests also highlight the centrality of cell 2 (see Figure 6), which is responsible for both the initial morphogenesis and the homeostatic process. Cell 2 can be sometimes regenerated during development but not during the healing phase.

The maturation and healing phases share similarities but are quite different. During maturation, fault-tolerance is achieved with a continuous cell production performed by cell 2. If lost, cell 2 is regenerated by cell 1 (to its left).

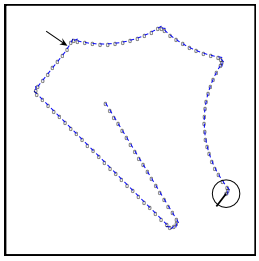


Fig. 7. Trace of the behaviour from the controller of Figure 6 after a mutilation event occurring with a .185 probability (arrow in figure), representing the most common source of unrecoverable mutilation. Being lost is the second cell, which is responsible for the self-healing process, its loss prevents further regeneration. The bot is still capable of avoiding walls, displaying a sub-optimal navigation strategy (.44 fitness)

During the healing phase, cell 2 produces cells with slightly lower metabolism (neuron thresholds). The reason behind this is that at the start of the fitness evaluation, robots are placed in random positions. Therefore controllers must be capable of a robust wall avoidance. On the contrary, after mutilation the robot is already navigating and a strong wall aversion is traded for longer and fitter trajectories. As a side effect, cell 1 looses the ability to regenerate cell 2, and an entirely new group of non-functional cells is generated (their output weight is 0).

It turns out that, because of how fitness is computed, after mutilation longer trajectories are more valuable than a more robust regeneration.

Quite interestingly, the loss of cell 2 is associated to the production of two sub-optimal neuro-controllers still capable of wall avoidance: the first proceeds at slower speed (.83 fitness), the second producing fast but curved trajectories (.44 fitness, also shown in Figure 7).

IV. Conclusions

We have presented an ontogeny-based approach to the design of fault-tolerant neural networks. By evolving a growth program for Leaky Integrate and Fire Spiking Neural Networks, it has been possible to produce resilient neuro-controllers for simulated Khepera robots.

Aimed at the production of neuro-controllers subjected to externally caused faults, evolution exploits the developmental process to produce individuals capable of self-healing. Fault-tolerance is achieved by regenerating lost cells, with a process which reminds of biological organisms.

Fault handling is achieved by functional homeostasis: while the phenotype continuously grows new cells, bot behaviour is stable. This is similar to what happens in biological organisms in the case of the epidermis and partially also in the central nervous system.

When compared to the evolution of non fault-tolerant controllers: (i) individuals require only 50-100% more generations to evolve robust wall-avoidance, (ii) similar behavioral strategies are evolved, (iii) while related work argued for morpher simplicity (i.e. no hidden layers) [5], hidden nodes are required to achieve fault-tolerant neuro-controllers.

The presented results confirm the hypothesis formulated in related work [14], [5], [16], suggesting that development based on local information has an intrinsic robustness. With a distributed growth process, ontogeny can be exploited to produce self-healing devices. Similarly to biological organisms, such devices would display an increased tolerance to external hazards by regenerating faulty components.

References

[1] L.F. Abbott and S.B. Nelson. Synaptic plasticity: Taming the beast. *Nature Neuroscience*, 3:1178–1183, 2000.

[2] A. Cangelosi, S. Nolfi, and D. Parisi. Cell division and migration in a 'genotype' for neural networks. *Network: Computation in Neural Systems*, 5:497–515, 1994.

[3] T.J. Carew. *Behavioral Neurobiology: The Cellular Organization of Natural Behavior*. Sinauer Associated Inc., 2000.

[4] F. Dellaert and R.D. Beer. Toward an evolvable model of development for autonomous agent synthesis. In R.Brooks and P. Maes, editors, *Proceedings of Artificial Life IV*, pages 246–257. MIT Press Cambridge, 1994.

[5] D. Federici and K.Downing. Evolution and development of a multi-cellular organism: Scalability, resilience and neutral complexification. *Submitted to Artificial Life Journal*, forthcoming 2005.

[6] W. Gerstner and W.M. Kistler. *Spiking Neuron Models, Single Neurons, Populations, Plasticity*. Cambridge University Press, 2002.

[7] F. Gruau. *Neural Network Synthesis using Cellular Encoding and the Genetic Algorithm*. PhD thesis, Ecole Normale Superieure de Lyon, 1994.

[8] Gregory S. Hornby and Jordan B. Pollack. The advantages of generative grammatical encodings for physical design. In *Proceedings of the 2001 Congress on Evolutionary Computation, CEC 2001*, pages 600–607. IEEE Press, 27-30 2001.

[9] E.R. Kandel, J.H. Schwartz, and T.M. Jessell. *Principles of Neural Science*. McGraw-Hill, 2000.

[10] H. Kitano. Designing neural networks using genetic algorithms with graph generation system. *Complex Systems*, 4:4:461–476, 1990.

[11] P.K. Lehre and P. C Haddow. Developmental mappings and phenotypic complexity. In Ruhul Sarker, Robert Reynolds, Hussein Abbass, Kay Chen Tan, Bob McKay, Daryl Essam, and Tom Gedeon, editors, *Proceeding of the Congress on Evolutionary Computation, CEC 2003*, 2003.

[12] Sean Luke and Lee Spector. Evolving graphs and networks with edge encoding: Preliminary report. In John R. Koza, editor, *Late Breaking Papers at the Genetic Programming 1996 Conference*, pages 117–124, 1996.

[13] H. Markram, J. Lübke, M. Frotscher, and B. Sakmann. Regulation of synaptic efficacy by coincidence of post-synaptic aps and epsps. *Science*, 275:213–215, 1997.

[14] J.F. Miller. Evolving developmental programs for adaptation, morphogenesys, and self-repair. In Wolfgang Banzhaf, Jens Ziegler, and Thomas Christaller, editors, *Proceeding of the European Congress of Artificial Life, ECAL 2003*, pages 256–265, 2003.

[15] S. Ohno. *Evolution by Gene Duplication*. Springer, 1970.

[16] D. Roggen and D. Federici. Multi-cellular development: is there scalability and robustness to gain? In Xin Yao, E. Burke, J.A. Lozano, and al., editors, *proceedings of Parallel Problem Solving from Nature 8, PPSN 2004*, pages 391–400, 2004.

[17] D.M. Wu, T. Schneiderman, J. Burgett, P. Gokhale, L. Barthel, and P.A.Raymond. Cones regenerate from retinal stem cells sequestered in the inner nuclear layer of adult goldfish retina. *Invest Ophthalmol Vis Sci.*, 42(9):2115–24, 2001.

[18] M. Zhao, S. Momma, K. Delfani, M. Calren, R.M. Cassidy, C.B.Johansson, H. Brismar, O. Shupliankov, J. Frisen, and A.M. Janson. Evidence for neurogenesis in the adult mammalian substantia nigra. *Proc Natl Acad Sci USA*, 100(13):7925–30, 2003.

Bibliography

Abbott, L.F., and S.B. Nelson. 2000. "Synaptic Plasticity: Taming the Beast." *Nature Neuroscience* 3:1178–1183.

Arbib, M. A. 2001. In *The Mirror System, Imitation, and the Evolution of Language*, edited by Chrystopher Nehaniv and Kerstin Dautenhahn. The MIT Press.

Beer, R.D. 1995. "On the dynamics of small continuous-time recurrent neural networks." *Adaptive Behavior* 3 (4): 469–510.

———. 2003. "The Dynamics of Active Categorical Perception in an Evolved Model Agent." *Adaptive Behavior* 11 (4): 209–243.

Bentley, P.J., and S. Kumar. 1999, 13-17. "Three Ways to Grow Designs: A Comparison of Embryogenies for an Evolutionary Design Problem." Edited by Wolfgang Banzhaf, Jason Daida, Agoston E. Eiben, Max H. Garzon, Vasant Honavar, Mark Jakiela, and Robert E. Smith, *Proceedings of the Genetic and Evolutionary Computation Conference*, Volume 1. Orlando, Florida, USA: Morgan Kaufmann, 35–43.

Bonabeau, E., M. Dorigo, and G. Theraulaz. 1999. *Swarm intelligence: from natural to artificial systems*. Oxford University Press, Inc.

Boyd, R., and P. Richerson. 2000. Pages 143–162 in *Memes: Universal Acid or Better Mouse Trap*, edited by R. Aunger. Oxford University Press.

Brooks, R. A. 1991. "Intelligence without representation." *Artificial Intelligence* 47:139–159.

Calvin, W.H., and D. Bickerton. 2000. *Lingua ex Machina: Reconciling Darwin and Chomsky with the human brain*. MIT Press,.

Cangelosi, A. 2004, July. "The sensorimotor bases of linguistic structure: Experiments with grounded adaptive agents." Edited by S. Schaal et al., *SAB04*. Los Angeles: Cambridge MA, MIT Press, 487–496. Proceedings of the Eighth International Conference on the Simulation of Adaptive Behaviour: From Animals to Animats 8.

Cangelosi, A., S. Nolfi, and D. Parisi. 1994. "Cell division and migration in a 'genotype' for neural networks." *Network: Computation in Neural Systems* 5:497–515.

Carew, T.J. 2000. *Behavioral Neurobiology: The Cellular Organization of Natural Behavior*. Sinauer Associated Inc.

Carr, C.E., and M.Konishi. 1990. "A circuit detection of interneural time differences in the brain stem of the barn owl." *Journal of Neuroscience* 10:3227–3246.

Cavalli-Sforza, L.L., and M.W. Feldman. 1981. *Cultural Transmission and Evolution: A Quantitative Approach*. Priceton university press.

Chalmers, D.J. 1990. "Syntactic transformations on distributed representations." *Connection Science* 2:53–62.

Chrisman, L. 1991. "Learning recursive distributed representations for holistic computation." *Connection Science*, vol. 3.

Clark, A., and R. Grush. 1999. "Towards a Cognitive Robotics." *Journal of Adaptive Behavior* 7:1:5–16.

Dawkins, R. 1976. *The selfish gene*. Oxford university press.

Delcomyn, F. 1998. *Foundations of Neurobilogy*. W.H. Freeman and Co.

Dennett, D.C. 1978. *Brainstorms*. MIT Press.

———. 1995. *Darwin's dangerous idea*. Little, Brown and Co.

Downing, K. 2005. "The Predictive Basis of Situated and Embodied Artificial Intelligence." *submitted to GECCO-2005*.

Elman, J.L. 1990. "Finding structure in time." *Cognitive Science* 14:179–211.

Floreano, D., and F. Mondada. 1996. "Evolution of Plastic Neurocontrollers for Situated Agents." Edited by P. Maes, M. Mataric, J-A. Meyer, J. Pollack, and S. Wilson., *From Animals To Animats 4, SAB 96*. MIT Press.

Fog, A. 1999. *Cultural Selection*. Kluwer Academic Publishers.

Gazzaniga, M., R. Ivry, and G. Mangun. 2002. *Cognitive neuroscience: The biology of the mind (2nd Ed.)*. Norton and Co.

Gerstner, W., and W.M. Kistler. 2002. *Spiking Neuron Models, Single Neurons, Populations, Plasticity*. Cambridge University Press.

Gil-White, F.J. 2004. "Common misunderstandings of memes (and genes): The promise and the limits of the genetic analogy to cultural transmission processes." Edited by S. Hurley and N. Chater, *Perspectives on Imitation: From Mirror Neurons to Memes*. MIT Press.

Goldberg, D. 1989. *Genetic Algorithms in Search, Optimization, and Machine Learning*. Reading, MA: Addison-Wesley.

Gould, S.J. 1991. *Bully for Brontosaurus*. New York: Norton.

Gruau, F. 1994. "Neural Network Synthesis using Cellular Encoding and the Genetic Algorithm." Ph.D. diss., Ecole Normale Superieure de Lyon.

Harnad, S. 1990. "The symbol grounding problem." *Physica D* 42:335–346.

Haykin, S. 1999. *Neural Networks. A Comprehensive Foundation. 2nd ed.* New Jersey: Prentice-hall.

———. 2001. *Adaptive Filter Theory . 4th ed.* New Jersey: Prentice-hall.

Hesslow, G. 2002. "Conscious thought as simulation of behaviour and perception." *Trends in Cognitive Sciences* 6:242–247.

Holland, J.H. 1992. *Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control and Artificial Intelligence.* MIT Press.

Hull, D.L. 1982. "The naked meme." Edited by H.C. Plotkin, *Learning development and culture, essays in evolutionary epistemology.* John Wiley and Sons, 273–327.

Kandel, E.R., J.H. Schwartz, and T.M. Jessell. 2000. *Principles of Neural Science.* McGraw-Hill.

Kitano, H. 1990. "Designing neural networks using genetic algorithms with graph generation system." *Complex Systems* 4:4:461–476.

Kohonen, T., ed. 2001. *Springer Series in Information Sciences: Self-Organizing Maps, 3rd ed.* Springer.

Kwansy, S.C., and B.L. Kalman. 1994. "Tail-recursive representations and simple recurrent networks." *Connection Science* 7:1:61–80.

Lehre, P.K., and P. C Haddow. 2003. "Developmental Mappings and Phenotypic Complexity." Edited by Ruhul Sarker, Robert Reynolds, Hussein Abbass, Kay Chen Tan, Bob McKay, Daryl Essam, and Tom Gedeon, *Proceeding of the Congress on Evolutionary Computation, CEC 2003.*

Maass, W. 1995. "On the Computational Complexity of Networks of Spiking Neurons." *Advances in Neural Information Processing Systems* 7:183–190.

Maass, W., and C.M. Bishop, eds. 1998. *Pulsed Neural Networks.* Cambridge, MA: MIT Press.

Markram, H., J. Lübke, M. Frotscher, and B. Sakmann. 1997. "Regulation of Synaptic Efficacy by Coincidence of Post-synaptic APs and EPSPs." *Science* 275:213–215.

McCarthy, J., and P.J. Hayes. 1969. "Some Philosophical Problems from the Standpoint of Artificial Intelligence." Edited by D.Michie and B.Meltzer, *Machine Intelligence 4.* 463–502.

Meeden, L., G. McGraw, and D. Blank. 1993. "Emergent control and planning in an autonomous vehicle." *Proceedings to the 15th annual conference of the cognitive science society.* 735–740.

Miller, J.F. 2003. "Evolving developmental programs for adaptation, morphogenesys, and self-repair." Edited by Wolfgang Banzhaf, Jens Ziegler, and Thomas Christaller, *Proceeding of the European Congress of Artificial Life, ECAL 2003.* 256–265.

———. 2004. "Evolving a self-repairing, self-regulating, french flag organism." Edited by Kalyanmoy Deb, Riccardo Poli, Wolfgang Banzhaf, Hans-Georg Beyer, Edmund K. Burke, Paul J. Darwen, Dipankar Dasgupta, Dario Floreano, James A. Foster, Mark Harman, Owen Holland, Pier Luca Lanzi, Lee

Spector, Andrea Tettamanzi, Dirk Thierens, and Andrew M. Tyrrell, *Proceeding of Genetic and Evolutionary Compuation, GECCO 2004*. 129–139.

Mitchell, T.M. 1997. *Machine Learning*. McGraw-Hill.

Ohno, S. 1970. *Evolution by Gene Duplication*. Springer.

Pfeifer, R., and C. Scheier. 1999. *Understanding intelligence*. MIT Press.

Pollack, J.B. 1990. "Recursive Distributed Representations." *Artificial Intelligence* 46:77–105.

Rieke, F., D. Warland, R. de R. van Steveninck, and W. Bialek. 1997. *Spikes: Exploring The Neural Code*. London, England: The MIT Press.

Rizzolatti, G., L. Fadiga, V. Gallese, and L. Fogassi. 1996. "Premotor cortex and the recognition of motor actions." *Cognitive Brain Research* 3:131–141.

Schacter, D.L. 1996. "Illusory memories: A cognitive neuroscience analysis." *Proceedings of the National Accademy of Science of the USA* 93:13527–13533.

Smith, T., P. Husbands, A. Philippides, and M. O'Shea. 2002. "Temporally Adaptive Networks: Analysis of GasNet Robot Control Networks." Edited by Standish, Abbass, and Bedau, *Artificial Life VIII*. 274–282.

Steels, L. 2003. "Intelligence with representation." *Philosophical Transactions: Mathematical, Physical and Engineering Sciences* 361 (1811): 2381–2395.

Stoianov, I. 2000. "Recurrent Autoassociative Networks and Holistic Computations." Edited by Shun-Ichi Amari, C. Lee Giles, Marco Gori, and Vincenzo Piuri, *International Joint Conference on Neural Networks, IJCNN 2000*. 5605–5610.

Suddendorf, T., and M. Corballis. 1997. "Mental time travel and the evolution of the human mind." *IGenetic, Social, and General Psychology Monographs* 132(2):133–167.

Sugita, Yuuya, and Jun Tani. 2005. "Learning Semantic Combinatoriality from the Interaction between Linguistic and Behavioral Processes." *Adaptive Behavior*.

Sutton, Richard S., and Andrew G. Barto. 1998. *Reinforcement Learning: An Introduction*. Cambridge, MA: MIT Press.

Tani, J. 1996. "Model-based learning for mobile robot navigation from the dynamical systems perspective." *IEEE Trans. on Systems, Man, and Cybernetics Part B: Cybernetics* 26(3):421–436.

———. 2003. "Symbols and Dynamics in Embodied Cognition: Revisiting a Robot Experiment." Edited by M. V. Butz, O. Sigaud O., and P. Gerard, *Anticipatory Behavior in Adaptive Learning Systems*. Springer-Verlag, 167–178.

Teichmann, S.A., and M.M. Babu. 2004. "Gene regulatory network growth by duplication." *Nature Genetics* 36(5):492–6.

Tesauro, Gerald. 1995. "Temporal difference learning and TD-Gammon." *Communications of the ACM* 38(3):58–68.

Webb, B. 2002. "Robots in invertebrate neuroscience." *Nature* 417:359–363.

Werbos, P.J. 1989. "Backpropagation and Neurocontrol: a review and prospectus." *Proceedings of the International Joint Conference on Neural Networks, IJCNN 1989.* 209–216.

———. 1990. "Backpropagation through time: what it does and how to do it." *Proceedings of the IEEE*, Volume 78(10). 1550–1560.

Wolfram, S., ed. 2002. *A new kind of science.* Wolfram Media.

Wu, D.M., T. Schneiderman, J. Burgett, P. Gokhale, L. Barthel, and P.A.Raymond. 2001. "Cones regenerate from retinal stem cells sequestered in the inner nuclear layer of adult goldfish retina." *Invest Ophthalmol Vis Sci.* 42(9):2115–24.

Wu, L., D. Wells, J. Tay, J. Mendis nad M.A. Abbot, A. Barnitt, E. Quinlan, A. Heynen, J.R. Fallon, and J.D Richter. 1998. "CPEB-mediated cytoplasmic polyadenylation and the regulation of experience-dependent translation of a-CaMKII mRNA at synapses." *Neuron* 21:1129–1139.

Zhao, M., S. Momma, K. Delfani, M. Calren, R.M. Cassidy, C.B.Johansson, H. Brismar, O. Shupliankov, J. Frisen, and A.M. Janson. 2003. "Evidence for neurogenesis in the adult mammalian substantia nigra." *Proc Natl Acad Sci USA* 100(13):7925–30.

Ziemke, T. 2001. "The Construction of 'Reality' in the Robot: Constructivist Perspectives on Situated AI and Adaptive Robotics." *Foundations of Science* 6(1):163–233.

———. 2005. "Cybernetics and Embodied Cognition: On the Construction of Realities in Organisms and Robots." *Kybernetes.*

Ziemke, T., D.A Jirenhed, and G. Hesslow. 2005. "Internal Simulation of Perception: a Minimal Neuro-Robotic model."

Bibliography of Article A

J.L. Elman, 'Finding structure in time,' *Cognitive Science*, 14:179-211, 1990.

I. Stoianov, 'Recurrent Autoassociative Networks and Holistic Computations,' *International Joint Conference on Neural Networks*, Como (Italy), 2000.

S. Haykin, *Neural Networks. A Comprehensive Foundation*, Prentice-hall, New Jersey, second edition 1999.

J.B. Pollack, 'Recursive Distributed Representations,' *Artificial Intelligence*, 46:77-105, 1990.

D.J. Chalmers, 'Syntactic transformations on distributed representations,' *Connection Science*, 2:53-62, 1990.

L. Chrisman, 'Learning recursive distributed representations for holistic computation,' *Connection Science*, , 3:345-365, 1991.

J. Tani, 'Model-based learning for mobile robot navigation from the dynamical systems perspective,' *IEEE Trans. system, man, and cybernetics Part B*,26(3):421-436, 1996.

L. Meeden, G. McGraw, D. Blank, 'Emergent control and planning in an autonomous vehicle,' *Proc. 15th annual conference of the cognitive science society*, Lawrence Erlbraun, Hillsdale NJ, 1993.

D.A Jirenhed, G. Hesslow, T. Ziemke, 'Exploring internal simulation of perception in mobile robots,' *Fourth European Workshop on Advanced Mobile Robots*, Lund (Sweden), 2001.

L.M. Barker, *Learning and behavior*, Prentice Hall, New Jersey, 1996.

H.C. Blodgett, 'The effect of the introduction of reward upon maze performance of rats,' *University of California publications in psychology*, 4:113-134, 1929.

E.C. Tolman, 'Cognitive maps in rats and men,' *The psychological Review*, 55(4), 189-208, 1948.

P. Campolucci, A. Uncini, F. Piazza, 'A Unifying View of Gradient Calculations and Learning for Locally Recurrent Neural Networks,' *Proc. of WIRN97, Italian Workshop on Neural Networks*, Vietri sul Mare (Italy), 1997.

M. Chester, *Neural Networks*, Prentice-hall, New Jersey, 1993.

J.L. Elman, 'Origins of language: a conspiracy theory,' *The emergence of language*, Hillsdale NJ, Lawrence Erlabraun, 1999.

J. Shranger, M.H. Johnson, 'Dynamic plasticity influences the emergence of function in a simple cortical array,' *Neural Networks*, 8:1-11, 1996.

S.C. Kwansy, B.L. Kalman, 'Tail-recursive representations and simple recurrent networks,' *Connection Science*, 7(1):61-80, 1994.

Bibliography of Article B

T. Carew, *Behavioral Neurobiology: The Cellular Organization of Natural Behavior*. Sinauer Assoc, 2000.

E. Tolman, "Cognitive maps in rats and men," *The Psychological Review*, vol. 55, no. 4, pp. 189–208, 1948.

B. Kosko, "Bidirectional associative memories," *IEEE Transactions on Systems, Man and Cybernetics*, vol. 18, no. 1, pp. 49–60, 1988.

K. G. Haines and R. Hecht-Nielson, "A bam with increased information storage capacity," *Proceedings of the IEEE 1988 International Conference on Neural Networks*, vol. 1, pp. 181–190, 1988.

M. H. Hassoun and P. B. Watta, "The hamming associative memory and its relation to the exponential capacity dam," *IEEE International Conference on Neural Networks, ICNN*, 1996.

K. Demura, "One shot learning and adaptive similarity," *PhD Thesis*, 1996. [Online]. Available:
http://www.his.kanazawa-it.ac.jp/~demura/research/papers/demura_thesis.pdf

Y. Wu and S. N. Batalama, "Improved one-shot learning for feedforward associative memories with application to composite pattern association," *IEEE Transactions on Systems, Man and Cybernetics*, vol. 31, no. 1, pp. 119–125, 2001.

J. Austin and K. Lees, "A search engine based on neural correlation matrix memories," *Neurocomputing*, vol. 35, no. 1, pp. 55–72, november 2000.

G. D. Micheli, *Synthesis and Optimization of Digital Circuits*. McGraw-Hill, 1994.

Bibliography of Article C

J.M. Baldwin: A new factor in evolution. American Naturalist **30** 441–451 (1896)

K. Downing: Reinforced Genetic Programming. Genetic Programming & Evolvable Machines. Kluwer Publishers 259–288 (2001)

P. Turney: Myths and legends of the Baldwin Effect. Proceedings of the ICML-96 (13th International Conference on Machine Learning) 135–142 (1996)

P. Turney, D. Whitley, R. Anderson: Evolution, learning and instict: 100 years of the Baldwin effect. Evolutionary Computation **4:3** 206–212 (1996)

R.K. Belew, M. Mitchell: Adaptive individuals in evolving populations: models and algorithms. addison-wesley (1996)

G.E. Hinton, S.J. Nowlan: How learning can guide evolution. Complex Systems **1** 495–502 (1987)

L.M. Chalupa, B. L. Finlay: Development and organization of the retina : from molecules to function. Plenum Press, New York (1998)

R. French, A. Messinger: Genes, phenes and the Baldwin effect. In Proceedings of Artificial Life IV 277–282 (1994)

R. Dawkins: The selfish gene. Oxford university press (1976)

E.R. Kandel, J.S. Schwartz, T.M. Jessell: Principles of neural science. Elsevier Science Publishing Co., New York (2000) 4th edition

H. Markram, J. L'bke, M. Frotscher, B. Sakmann: Regulation of Synaptic Efficacy by Coincidence of Post-synaptic APs and EPSPs. Science, **275** 213–215 (1997)

T.J. Carew Behavioral Neurobiology: The Cellular Organization of Natural Behavior Sinauer Associates (2000)

T. Arita, R. Suzuki Interactions between learning and evolution: The outstanding strategy generated by the Baldwin effect. In Proceedings of Artificial Life VII, 196–205 (2000)

Bibliography of Article D

J.B. Lamarck: Philosophie zoologique. (1809)

J.M. Baldwin: A new factor in evolution. American Naturalist **30** 441–451 (1896)

C.R. Houck, J.A. Joines, M.G. Kay and J.R Wilson: Empirical Investigation of the Benefits of Partial Lamarckianism. Evolutionary Computation **5(1)**, 31-60 (1997)

P. Turney: Myths and legends of the Baldwin Effect. Proceedings of the ICML-96 (13th International Conference on Machine Learning) 135–142 (1996)

R.K. Belew, M. Mitchell: Adaptive individuals in evolving populations: models and algorithms. addison-wesley (1996)

G.E. Hinton, S.J. Nowlan: How learning can guide evolution. Complex Systems **1** 495–502 (1987)

R. French, A. Messinger: Genes, phenes and the Baldwin effect. In Proceedings of Artificial Life IV 277–282 (1994)

G. Mayley: Landscapes, Learning Costs and Genetic Assimilation, In Evolutionary Computation, Evolution, Learning and Instict: 100 years of the Baldwin Effect, (1996)

G. Mayley: Guiding or hiding. In the Proceedings of the Fourth European Conference on Artificial Life (ECAL'97) (1997)

L. Spector and S. Luke: Cultural Transmission of Information in Genetic Programming. In Genetic Programming 1996: Proceedings of the First Annual Conference, 209-214 (1996)

L. Spector, S. Luke: Culture Enhances the Evolvability of Cognition, in proceedings of Cognitive Science, (1996)

C. Wellock and B.J. Ross: An Examination of Lamarckian Genetic Algorithms. 2001 Genetic and Evolutionary Computation COnference, Late Breaking Papers, 9(11), 474-481 (2001)

S. Nolfi, J.L. Elman and D. Parisi: Learning and Evolution in Neural Networks. Adaptive Behavior, Vol. 3(1), 5-28 (1994)

D. Floreano and F. Mondada: Evolutionary Neurocontrollers for Autonomous Mobile Robots. Neural Networks, 11, 1461-1478 (1998)

J. R. Koza: Genetic Programming II. MIT Press, Cambridge (1994)

J.H. Holland: Adaptation in natural and artificial systems. MIT press, Cambridge (1975)

E.R. Kandel, J.S. Schwartz, T.M. Jessell: Principles of neural science. Elsevier Science Publishing Co., New York (2000) 4th edition

H. Markram, J. Lbke, M. Frotscher, B. Sakmann: Regulation of Synaptic Efficacy by Coincidence of Post-synaptic APs and EPSPs. Science, **275** 213–215 (1997)

J. Cassens, Z. Constantinescu Flp It's Magic: SourceMage GNU/Linux as HPC Cluster OS. To appear in Proceedings Linuxtag 2003, (2003)
http//clustis.idi.ntnu.no/

Bibliography of Article E

Stanley, K., Miikulainen, R.: A taxonomy for artificial embryogeny. Artificial Life 9(2):93–130 (2003)

Kitano, H.: Designing neural networks using genetic algorithms with graph generation system. Complex Systems, 4(4):461–476 (1990)

Lehre, P., Haddow, P.C.: Developmental mappings and phenotypic complexity. Proceeding of CEC 2003 (2003)

Tufte, G., Haddow, P.C.: Insertion of functionality into development on an sblock platform. Proceeding of CEC 2003 (2003)

Federici, D.: Evolving a neurocontroller through a process of embryogeny. submitted to SAB conference 2004 (2004)

Gruau, F., ed.: Neural Network Synthesis using Cellular Encoding and the Genetic Algorithm. PhD Thesis, Ecole Normale Sup'erieure de Lyon (1994)

Bentley, P., Kumar, S.: Three ways to grow designs: A comparison of embryogenies for an evolutionary design problem. Proceeding of GECCO 1999, 35-43 (1999)

Miller, J.: Evolving developmental programs for adaptation, morphogenesys, and self-repair. Proceeding of ECAL 2003, 256–265 (2003)

Kodjabachian, J., Meyer, J.: Evolution and development of control architectures in animats. Robotics and Autonomous Systems, 16(2-4) (1995)

Wikipedia: Ontogeny and phylogeny. http:// en2.wikipedia.org /wiki /Ontogeny_and_phylogeny (2003)

Cassens, J., Fülöp, Z.C.: It's magic: Sourcemage gnu/linux as hpc cluster os. in Proceedings Linuxtag 2003, Karlsruhe, Germany (2003)

Bibliography of Article F

Lehre, P., Haddow, P.C.: Developmental mappings and phenotypic complexity. Proceeding of CEC 2003, 62–68 (2003)

Tyrrell, A.M., Sanchez, E., Floreano, D., Tempesti, G., Mange, D., Moreno, J.M., Rosenberg, J., Villa, A.: POEtic Tissue: An Integrated Architecture for Bio-Inspired Hardware. In Tyrrell, A.M., et al., eds.: Proc. of the 5th Int. Conf. on Evolvable Systems (ICES 2003), Berlin, Springer (2003) 129–140

Roggen, D., Floreano, D., Mattiussi, C.: A Morphogenetic Evolutionary System: Phylogenesis of the POEtic Tissue. In Tyrrell, A.M., et al., eds.: Proc. of the 5th Int. Conf. on Evolvable Systems (ICES 2003), Berlin, Springer (2003) 153–164

Stanley, K., Miikulainen, R.: A taxonomy for artificial embryogeny. Artificial Life 9(2):93–130 (2003)

Federici, D.: Using embryonic stages to increase the evolvability of development. to appear in proceedings of WORLDS workshop at GECCO 2004 (2004)

Miller, J.: Evolving developmental programs for adaptation, morphogenesys, and self-repair. Proceeding of ECAL 2003, 256–265 (2003)

Coen, E.: The art of genes. Oxford University Press, New York (1999)

Haddow, P.C., Tufte, G., van Remortel, P.: Shrinking the Genotype: L-systems for EHW? In Liu, Y., et al., eds.: Proc. of the 4th Int. Conf. on Evolvable Systems (ICES 2001), Berlin, Springer (2001) 128–139

Eggenberger, P.: Cell interactions as a control tool of developmental processes for evolutionary robotics. In Maes, P., Mataric, M.J., Meyer, J.A., Pollack, J., Wilson, S.W., eds.: From Animals to Animats 4: Proceedings of the Fourth International Conference on Simulation of Adaptive Behavior, Cambridge, MA, MIT Press-Bradford Books (1996) 440–448

Gruau, F.: Automatic definition of modular neural networks. Adaptive Behavior **3** (1994) 151–183

Kumar, S., Bentley, P.J.: Biologically inspired evolutionary development. In Tyrrell, A.M., et al., eds.: Proc. of the 5th Int. Conf. on Evolvable Systems (ICES 2003), Berlin, Springer (2003) 57–68

Kitano, H.: Designing neural networks using genetic algorithms with graph generation system. Complex Systems, 4(4):461–476 (1990)

Luke, S., Spector, L.: Evolving graphs and networks with edge encoding: Preliminary report. In Koza, J.R., ed.: Late Breaking Papers at the Genetic Programming 1996 Conference. (1996) 117–124

Hornby, G.S., Pollack, J.B.: The advantages of generative grammatical encodings for physical design. In: Proceedings of CEC 2001, 600–607. (2001)

Bibliography of Article G

Banzhaf, W.: 1994, 'Genotype-Phenotype-Mapping and Neutral Variation — A case study in Genetic Programming'. *Proceedings Parallel Problem Solving From Nature III, PPSN'94* pp. 322 – 332.

Bar-Yam, Y. (ed.): 1997, *Dynamics of Complex Systems*. Perseuas Books, Reading (Ma).

Bentley, P.: 2003, 'Evolving Fractal Gene Regulatory Networks for Robot Control'. *Proceeding of the European Congress of Artificila Life, ECAL 2003* pp. 753–762.

Bentley, P. and S. Kumar: 1999, 'Three Ways to Grow Designs: A Comparison of Embryogenies for an Evolutionary Design Problem'. *Proceeding of GECCO 1999* pp. 35–43.

Bongard, J.: 2002, 'Evolving modular genetic regulatory networks'. In: *Proceedings of the 2002 Congress on Evolutionary Computation (CEC2002)*. pp. 1872–1877.

Cangelosi, A., S. Nolfi, and D. Parisi: 1994, 'Cell division and migration in a 'genotype' for neural networks'.

Cassens, J. and Z. C. Fülöp: 2003, 'It's Magic: SourceMage GNU/Linux as HPC Cluster OS'. *in Proceedings Linuxtag 2003*.

Dellaert, F. and R. Beer: 1994, *Toward an Evolvable Model of Development for Autonomous Agent Synthesis*, pp. 246–257. MIT Press Cambridge.

Dellaert, F. and R. Beer: 1996, 'A developmental model for the evolution of complete autonomous agents'.

E. Mjolsness, D.H. Sharp, J. R.: 1991, 'A connectionist model of development'. *Journal of Theoretical Biology* pp. 152(4):429–453.

Eggenbergen-Hotz, P.: 1997, 'Evolving Morphologies of Simulated 3d Organisms Based on Differential Gene Expression'. *Proceedings of the 4th European Conference on Artificial Life (ECAL97)*.

Eggenbergen-Hotz, P.: 2004, 'Comparing direct and developmental encoding schemes in artificial evolution: A case study in evolving lens shapes'. *Proceeding of the Congress on Evolutionary Computation, CEC 2004* pp. 752–757.

Federici, D.: 2004, 'Evolving a neurocontroller through a process of embryogeny'. *in proceedings to SAB conference 2004*.

Gruau, F. (ed.): 1994, *Neural Network Synthesis using Cellular Encoding and the Genetic Algorithm*. PhD Thesis, Ecole Normale Superieure de Lyon.

Harvey, I.: 2001, 'Artificial Evolution: A Continuing SAGA'. *Proceedings of 8th Intl. Symposium on Evolutionary Robotics, ER2001* pp. 94–109.

Hornby, G. S. and J. B. Pollack: 2001a, 'The Advantages of Generative Grammatical Encodings for Physical Design'. In: *Proceedings of the 2001 Congress on Evolutionary Computation, CEC 2001*. pp. 600–607.

Hornby, G. S. and J. B. Pollack: 2001b, 'Body-Brain Co-evolution Using L-systems as a Generative Encoding'. In: *Proceedings of the Genetic and Evolutionary Computation Conference, GECCO-2001*. pp. 868–875.

Jacobi, N.: 1995, 'Harnessing morphogenesis'. *Proceeding of the International Conference on Information Processing in Cells and Tissues* pp. 29–41.

Kauffman, S. (ed.): 1993, *The origins of order*. New York: Oxford University Press.

Kitano, H.: 1990, 'Designing neural networks using genetic algorithms with graph generation system'. *Complex Systems* pp. 4(4):461–476.

Lehre, P. and P. C. Haddow: 2003, 'Developmental Mappings and Phenotypic Complexity'. *Proceeding of the Congress of Evolutionary Computation, CEC 2003*.

Luke, S. and L. Spector: 1996, 'Evolving Graphs and Networks with Edge Encoding: Preliminary Report'. In: J. R. Koza (ed.): *Late Breaking Papers at the Genetic Programming 1996 Conference*. pp. 117–124.

Mattiussi, C. and D. Floreano: 2004, 'Connecting Transistors and Proteins'. *ALife9: Proceedings of the Ninth International Conference on Artificial Life*.

ARJ Software Inc. *http://www.arjsoftware.com*.

Miller, J.: 2003, 'Evolving developmental programs for adaptation, morphogenesys, and self-repair'. *Proceeding of the European Congress of Artificial Life, ECAL 2003* pp. 256–265.

Miller, J.: 2004, 'Evolving a self-repairing, self-regulating, french flag organism'. *Proceeding of Genetic and Evolutionary Compuation, GECCO 2004* pp. 129–139.

Ohno, S. (ed.): 1970, *Evolution by Gene Duplication*. Springer.

Roggen, D. and D. Federici: 2004, 'Multi-cellular development: is there scalability and robustness to gain?'. *proceedings of Parallel Problem Solving from Nature, PPSN VIII 2004*.

Stanley, K. and R. Miikulainen: 2003, 'A Taxonomy for Artificial Embryogeny'. *Artificial Life 9(2):93–130*.

Stanley, K. and R. Miikulainen: 2004, 'Competitive Coevolution Through Evolutionary Complexification'. *Journal of Artificial Intelligence Research 21: 63-100*.

Teichmann, S. and M. Babu: 2004, 'Gene regulatory network growth by duplication'. *Nature Genetics* pp. 36(5):492–6.

Tufte, G. and P. C. Haddow: 2003, 'Building Knowledge into Development Rules for Circuit Design'. *Proceedings of the International Conference on Evolvable Systems, ICES 2003* pp. 117–128.

Tyrrell, A., E. Sanchez, D. Floreano, G. Tempesti, D. Mange, J. Moreno, J. Rosenberg, and A. Villa: 2003, 'POEtic Tissue: An Integrated Architecture for Bio-inspired Hardware'. *Proceedings of the International Conference on Evolvable Systems, ICES 2003* pp. 129–140.

Wikipedia: 2003, 'Ontogeny and phylogeny'.
http://en2.wikipedia.org/wiki/Ontogeny_and_phylogeny.

Wolfram, S. (ed.): 2002, *A new kind of science*. Wolfram Media.

Zhang, J.: 2003, 'Evolution by gene duplication: an update'. *Trends in Ecology and Evolution 18:(6):192-198*.

Bibliography of Article H

Bentley, P. (2003). Evolving fractal gene regulatory networks for robot control. *Proceeding of ECAL 2003, 753–762.*

Bentley, P. and Kumar, S. (1999). Three ways to grow designs: A comparison of embryogenies for an evolutionary design problem. *Proceeding of the Genetic and Evolutionary Computation Conference 1999, 35-43.*

Dellaert, F. and Beer, R. (1996). A developmental model for the evolution of complete autonomous agents. In *proceedings the fourth international conference on Simulation of Adaptive Behavior*, pages 393–401.

Federici, D. (2004). Increasing evolvability for developmental programs. *Under submission to the Genetic and Evolutionary Computation Conference 2004.*

Gruau, F., (Ed.) (1994). *Neural Network Synthesis using Cellular Encoding and the Genetic Algorithm.* PhD Thesis, Ecole Normale Sup'erieure de Lyon.

Hornby, G. S. and Pollack, J. B. (2001). The advantages of generative grammatical encodings for physical design. In *Proceedings of the 2001 Congress on Evolutionary Computation*, pages 600–607. IEEE Press.

Jacobi, N. (1995). Harnessing morphogenesis. *Proceeding of information processing in cells and tissues, 29-41.*

Kauffman, S., (Ed.) (1993). *The origins of order.* New York: Oxford University Press.

Kitano, H. (1990). Designing neural networks using genetic algorithms with graph generation system. *Complex Systems, 4(4):461–476.*

Lehre, P. and Haddow, P. C. (2003). Developmental mappings and phenotypic complexity. *Proceedings of the 2003 Congress on Evolutionary Computation.*

Luke, S. and Spector, L. (1996). Evolving graphs and networks with edge encoding: Preliminary report. In Koza, J. R., (Ed.), *Late Breaking Papers at the Genetic Programming 1996 Conference*, pages 117–124.

Miller, J. (2003). Evolving developmental programs for adaptation, morphogenesys, and self-repair. *Proceeding of European Conference on Artificial Life 2003, 256–265.*

Stanley, K. and Miikulainen, R. (2003). A taxonomy for artificial embryogeny. *Artificial Life 9(2):93–130.*

Tufte, G. and Haddow, P. C. (2003). Insertion of functionality into development on an sblock platform. *Proceedings of the 2003 Congress on Evolutionary Computation.*

Wikipedia. Ontogeny and phylogeny. *http:// en2.wikipedia.org /wiki /Ontogeny_and_phylogeny.*

Bibliography of Article I

L. Abbott and S. Nelson. Synaptic plasticity: Taming the beast. *Nature Neuroscience*, 3:1178–1183, 2000.

P. Bentley. Evolving fractal gene regulatory networks for robot control. In W. Banzhaf, J. Ziegler, and T. Christaller, editors, *Proceeding of the European Congress of Artificial Life, ECAL 2003*, volume 2801/2003, pages 753–762. Springer-Verlag, 2003.

A. Cangelosi, S. Nolfi, and D. Parisi. Cell division and migration in a 'genotype' for neural networks. *Network: Computation in Neural Systems*, 5:497–515, 1994.

T. Carew. *Behavioral Neurobiology: The Cellular Organization of Natural Behavior*. Sinauer Associated Inc., 2000.

F. Dellaert and R. Beer. Toward an evolvable model of development for autonomous agent synthesis. In R.Brooks and P. Maes, editors, *Proceedings of Artificial Life IV*, pages 246–257. MIT Press Cambridge, 1994.

F. Dellaert and R. Beer. A developmental model for the evolution of complete autonomous agents. In P. Maes, M. J. Mataric, J.-A. Meyer, J. Pollack, and S. W. Wilson, editors, *From Animals To Animats 4: SAB 1996*, pages 393–401, 1996.

J. R. E. Mjolsness, D.H. Sharp. A connectionist model of development. *Journal of Theoretical Biology*, 152(4):429–453, 1991.

D. Federici. Evolving a neurocontroller through a process of embryogeny. In S. Schaal, A. Ijspeert, A. Billard, S. Vijayakumar, J. Hallam, and J. Meyer, editors, *From Animals To Animats 8: SAB 2004*, pages 373–384, 2004.

D. Federici and K.Downing. Evolution and development of a multi-cellular organism: Scalability, resilience and neutral complexification. *Submitted to Artificial Life Journal*, forthcoming 2005.

D. Floreano and F. Mondada. Automatic creation of an autonomous agent: Genetic evolution of a neural-network driven robot. In D. Cliff, P. Husbands, J. Meyer, and S. Wilson, editors, *From Animals To Animats 3, SAB 94*. MIT Press, 1994.

W. Gerstner and W. Kistler. *Spiking Neuron Models, Single Neurons, Populations, Plasticity*. Cambridge University Press, 2002.

F. Gruau. *Neural Network Synthesis using Cellular Encoding and the Genetic Algorithm*. PhD thesis, Ecole Normale Superieure de Lyon, 1994.

G. S. Hornby and J. B. Pollack. The advantages of generative grammatical encodings for physical design. In *Proceedings of the 2001 Congress on Evolutionary Computation, CEC 2001*, pages 600–607. IEEE Press, 27-30 2001.

G. S. Hornby and J. B. Pollack. Body-brain co-evolution using L-systems as a generative encoding. In L. Spector, E. D. Goodman, A. Wu, W. B. Langdon, H.-M. Voigt, M. Gen, S. Sen, M. Dorigo, S. Pezeshk, M. H. Garzon, and E. Burke, editors, *Proceedings of the Genetic and Evolutionary Computation Conference, GECCO-2001*, pages 868–875. Morgan Kaufmann, 7-11 2001.

N. Jacobi. Harnessing morphogenesis. In P. Bentley and S. Kumar, editors, *On Growth, Form and Computers*. Academic Press, 2003.

E. Kandel, J. Schwartz, and T. Jessell. *Principles of Neural Science*. McGraw-Hill, 2000.

S. Kauffman. *The origins of order*. Oxford University Press, New York, 1993.

H. Kitano. Designing neural networks using genetic algorithms with graph generation system. *Complex Systems*, 4:4:461–476, 1990.

S. Luke and L. Spector. Evolving graphs and networks with edge encoding: Preliminary report. In J. R. Koza, editor, *Late Breaking Papers at the Genetic Programming 1996 Conference*, pages 117–124, 1996.

H. Markram, J. Lübke, M. Frotscher, and B. Sakmann. Regulation of synaptic efficacy by coincidence of post-synaptic aps and epsps. *Science*, 275:213–215, 1997.

C. Mattiussi and D. Floreano. Connecting transistors and proteins. In J.Pollack, M. Bedau, P. Husbands, T. Ikegami, and R. Watson, editors, *ALife9: Proceedings of the Ninth International Conference on Artificial Life*, pages 9–20, 2004.

J. Miller. Evolving developmental programs for adaptation, morphogenesys, and self-repair. In W. Banzhaf, J. Ziegler, and T. Christaller, editors, *Proceeding of the European Congress of Artificial Life, ECAL 2003*, pages 256–265, 2003.

S. Ohno. *Evolution by Gene Duplication*. Springer, 1970.

D. Roggen and D. Federici. Multi-cellular development: is there scalability and robustness to gain? In X. Yao, E. Burke, J. Lozano, and al., editors, *proceedings of Parallel Problem Solving from Nature 8, PPSN 2004*, pages 391–400, 2004.

K. Stanley and R. Miikulainen. A taxonomy for artificial embryogeny. *Artificial Life*, 9(2):93–130, 2003.

Wikipedia. Ontogeny and phylogeny.
http://en2.wikipedia.org/wiki/Ontogeny_and_phylogeny, 2003.

Bibliography of Article J

L.F. Abbott and S.B. Nelson. Synaptic plasticity: Taming the beast. *Nature Neuroscience*, 3:1178–1183, 2000.

A. Cangelosi, S. Nolfi, and D. Parisi. Cell division and migration in a 'genotype' for neural networks. *Network: Computation in Neural Systems*, 5:497–515, 1994.

T.J. Carew. *Behavioral Neurobiology: The Cellular Organization of Natural Behavior*. Sinauer Associated Inc., 2000.

F. Dellaert and R.D. Beer. Toward an evolvable model of development for autonomous agent synthesis. In R.Brooks and P. Maes, editors, *Proceedings of Artificial Life IV*, pages 246–257. MIT Press Cambridge, 1994.

D. Federici and K.Downing. Evolution and development of a multi-cellular organism: Scalability, resilience and neutral complexification. *Submitted to Artificial Life Journal*, forthcoming 2005.

W. Gerstner and W.M. Kistler. *Spiking Neuron Models, Single Neurons, Populations, Plasticity*. Cambridge University Press, 2002.

F. Gruau. *Neural Network Synthesis using Cellular Encoding and the Genetic Algorithm*. PhD thesis, Ecole Normale Superieure de Lyon, 1994.

Gregory S. Hornby and Jordan B. Pollack. The advantages of generative grammatical encodings for physical design. In *Proceedings of the 2001 Congress on Evolutionary Computation, CEC 2001*, pages 600–607. IEEE Press, 27-30 2001.

E.R. Kandel, J.H. Schwartz, and T.M. Jessell. *Principles of Neural Science*. McGraw-Hill, 2000.

H. Kitano. Designing neural networks using genetic algorithms with graph generation system. *Complex Systems*, 4:4:461–476, 1990.

P.K. Lehre and P. C Haddow. Developmental mappings and phenotypic complexity. In Ruhul Sarker, Robert Reynolds, Hussein Abbass, Kay Chen Tan, Bob McKay, Daryl Essam, and Tom Gedeon, editors, *Proceeding of the Congress on Evolutionary Computation, CEC 2003*, 2003.

Sean Luke and Lee Spector. Evolving graphs and networks with edge encoding: Preliminary report. In John R. Koza, editor, *Late Breaking Papers at the Genetic Programming 1996 Conference*, pages 117–124, 1996.

H. Markram, J. Lübke, M. Frotscher, and B. Sakmann. Regulation of synaptic efficacy by coincidence of post-synaptic aps and epsps. *Science*, 275:213–215, 1997.

J.F. Miller. Evolving developmental programs for adaptation, morphogenesys, and self-repair. In Wolfgang Banzhaf, Jens Ziegler, and Thomas Christaller, editors, *Proceeding of the European Congress of Artificial Life, ECAL 2003*, pages 256–265, 2003.

S. Ohno. *Evolution by Gene Duplication*. Springer, 1970.

D. Roggen and D. Federici. Multi-cellular development: is there scalability and robustness to gain? In Xin Yao, E. Burke, J.A. Lozano, and al., editors, *proceedings of Parallel Problem Solving from Nature 8, PPSN 2004*, pages 391–400, 2004.

D.M. Wu, T. Schneiderman, J. Burgett, P. Gokhale, L. Barthel, and P.A.Raymond. Cones regenerate from retinal stem cells sequestered in the inner nuclear layer of adult goldfish retina. *Invest Ophthalmol Vis Sci.*, 42(9):2115–24, 2001.

M. Zhao, S. Momma, K. Delfani, M. Calren, R.M. Cassidy, C.B.Johansson, H. Brismar, O. Shupliankov, J. Frisen, and A.M. Janson. Evidence for neurogenesis in the adult mammalian substantia nigra. *Proc Natl Acad Sci USA*, 100(13):7925–30, 2003.