THE HYPERNETWORK ARCHITECTURE: A HIERARCHICAL MOLECULAR INTERACTION MODEL OF BIOLOGICAL INFORMATION PROCESSING

by

José L. Segovia-Juárez

DISSERTATION

Submitted to the Graduate School

of Wayne State University,

Detroit, Michigan

in partial fulfillment of the requirements

for the degree of

DOCTOR OF PHILOSOPHY

2001

MAJOR: **COMPUTER SCIENCE**

Approved by:

Václav Rajlich 09/07/01

Advisor Date

Dr Robert G Reynolds

Mark Van Buskirk

DEDICATION

To the memory of Michael Conrad

To my parents

ACKNOWLEDGMENTS

I would like to express my gratitude to Professor Michael Conrad, who gave me support, advice, and friendship during the years he was my advisor from 1994 until his passing in December 2000.

I am particularly thankful to Deborah Conrad, Silvano Colombano, and Vaclav Rajlich for their friendship and support to finish this dissertation.

Many thanks also go to my colleagues from the Biocomputing Laboratory, the faculty and staff of the Department of Computer Science, the Graduate School of Wayne State University, and the Fulbright Commission of Lima, Peru.

Finally, I would like to thank my parents, Jose and Ernestina; my brother, Carlos; my sisters, Cristina, Teresa and Tania; my grandmother, Soledad; and my girlfriend, Lakhana. Thanks to all for their love and encouragement.

CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

This chapter provides the following. Section 1.1 defines the concept of learning and its relationship to memory, section 1.2 is about biological organisms as information processing entities. Section 1.3 describes the goals and contributions of the dissertation, and section 1.4 gives an overview of the chapters.

1.1 Learning and memory

According to Simon (1983), *learning* "denotes changes in the system that are adaptive in the sense that they enable the system to do the same task or tasks drawn from the same population more efficiently and more effectively the next time." This is a very broad definition; on the other hand, Carbonell et al. (1983) differentiate two forms of learning: 1) "knowledge acquisition" as the process of acquiring new symbolic information together with the ability to apply that information effectively, and 2) "skill refinement" as a process of refining learned skills by repeated practice.

Machine learning finds methods for computing machines to become intelligent. Such methods cover encoding knowledge; learning strategies, such as strategies for learning from instruction or from examples; and strategies for learning by deduction, by analogy, or by observation (Michalski, 1987). Machine learning models for knowledge acquisition comprise decision trees, instance based learning, artificial neural nets, and genetic algorithms, among others (Mitchell, 1997). Artificial neural nets and genetic algorithms are inspired by biological systems, where the complex dynamics of biological systems are usually very simplified.

Memory is measured by the observation of behavioral changes in an animal [or

system] after learning. *Memory* is a process that includes acquisition, consolidation, retention, and retrieval of information (Abel and Lattal, 2001).

In biological organisms, there is more evidence that memory is based at the molecular level (Abel and Lattal, 2001; Squire and Kandel, 2000; Vianna et al., 2001). The intra and inter-cellular molecule-based dynamics are responsible for learning and memory processes. The present study is an attempt to build a machine learning architecture based on a model of the dynamics of molecular interactions.

1.2 Biological organisms as information processing entities

Nature is organized in hierarchies. Biological system hierarchies run through subatomic particles, atoms, molecules, cells, organs, individuals, communities, and ecosystem levels. Each level of organization has its own characteristics. For example atomic characteristics can not be directly applied to organisms.

In the context of biological systems Gatlin (1972) says that “life can be defined operationally as an information processing system – a structural hierarchy of functioning units – that has acquired through evolution the ability to store and process information necessary of its own accurate reproduction”. Then Gatlin defines information as “the capacity to store and transmit meaning or knowledge, [but] not the meaning or knowledge itself “. In other words, the author finds conceptual differences between the knowledge itself, and the capacity to store and transmit knowledge which she calls *information*.

Biological systems can be viewed as information processing entities since they use, create, and process information in order to survive. Four and a half billion years of evolution has generated a huge variety of ways of finding solutions to various problems. A subset of these biological strategies can be applied to solve technical problems. There

are several approaches that use evolutionary techniques, such as genetic programming (Koza, 1992), evolutionary programming (Fogel et al., 1966), evolutionary strategies (Schwefel, 1995), genetic algorithms (Holland, 1975), and cultural algorithms (Reynolds, 1994; Reynolds and Zannoni, 1992). But none of these explicitly use the hierarchical properties found in the architecture of life.

This work is motivated by the evidence that the hierarchical organization of biological systems plays an important role in information processing, learning, and problem solving (Conrad, 1983; Simon, 1962, 1973). Currently most scientific activities are focused on the study of one particular level of organization. For example, biology studies cells and organisms, chemistry studies atoms and molecules, ecology studies ecosystems, and so on. Research integrating mechanisms across levels is necessary. The results of this research should lead to a coherent view of complex systems.

1.3 Goals and contributions

In this study, I designed and implemented a novel architecture, called the hypernetwork architecture, based on the hierarchical organization and principles of biological information processing. It integrates information flow from the molecular, cellular, and organismic levels.

The hypernetwork architecture is a theoretical model of a biological system, capturing elements of its organization, dynamics and evolution. The model is based on molecular evolution and the formation of networks of molecular interactions.

This is an architecture for machine learning, a tool for understanding how hierarchies work, a tool for studying evolutionary strategies, and a model for building molecular computers.

1.4 Overview

Chapter 2 is a broad overview of hierarchy theory, complex systems, evolution and adaptive surfaces, and discusses the problem of scale.

Chapter 3 describes some conceptual antecedents of the hypernetwork architecture, such as evolutionary algorithms, neural networks models, and Conrad's enzymatic neuron and percolation network models. The hypernetwork model has some elements of each of these, but it is based solely on particle interactions. This interactional point of view will be stressed in the following chapters.

Chapter 4 is a detailed description of the hypernetwork architecture. First, a description of the components of the model is presented: molecules, cells, and the organism. Secondly, the variation-selection algorithm based on molecular mutations is described. Some sections of the chapter were previously published by myself and M. Conrad (Segovia-Juarez and Conrad, 1999, 2001), excerpts reprinted with permission.

Chapter 5 shows the learning capabilities of the hypernetwork model for four classification problems: the N-input parity task, the tic-tac-toe endgame problem, the two x two-bit multiplier, and the double spiral data set.

Chapter 6 shows the effect of inhibition and negative feedback regulation on hypernetwork learning.

Chapter 7 is a study of the mutation-buffering capabilities of the hypernetwork organisms and discusses their importance. Sections of the chapter appear in (Segovia-Juarez and Colombano, 2001), excerpts reprinted with permission.

Chapter 8 shows the evolvability properties of the hypernetwork model. The relationship between the size of the model organisms and learning is explored with the

N-input parity and two x two parity problems.

Chapter 9 contains the conclusions, a summary of the experimental results, and pointers to future work.

CHAPTER 2

BACKGROUND

Biological systems are complex, multi-dimensional, non-linear systems, structurally organized into hierarchical levels. This chapter is a brief review of previous work on complex systems (section 2.1), hierarchical systems (section 2.2), evolution and adaptive surfaces (section 2.3), emergent properties (section 2.4), and discusses of the problem of scale (section 2.5).

2.1 Complex systems

Feedback models, studied by Wiener (1948), Ashby (1956) and others, recognized that living systems are complex feedback regulatory systems. These authors frequently use differential equations to describe the system. The problem with this approach is that it is not a good approximation to the study of hierarchies, since continuous representations are not suitable for modeling the complex discrete structures found in hierarchies (van Zandt, 1995).

Chemical reaction graphs were introduced by Kauffman (1993) to study molecular dynamics at the cellular level. The reactions are modeled by bipartite graphs (see Figure 2.1).

Green (1994) states that digraphs (directed graphs) are inherent in the relationships between elements, and in transitions between different states in biological systems. He argue that “the pattern of interactions ... influences biological processes”.

Beurle (1962) attempted to show that a random connected network of neuron cells could perform a useful function. Later Kauffman (1969) introduced Boolean networks to simulate complex dynamics. Boolean networks are made up of binary variables where

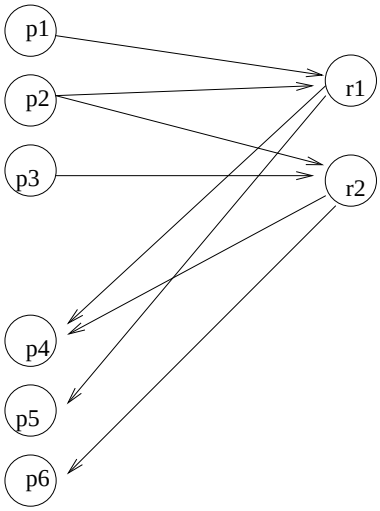


Figure 2.1: Bipartite Graph.p1, p2, p3, .., p5 are the *products* or *reactants* and r1, r2 are the *reactions* (Kauffman, 1993, 1996).

"each variable is regulated by some other variables in the network, which serve as its inputs. The *dynamical behavior* of each variable ... is governed by a logical switching rule or *Boolean function*" (Kauffman, 1993). Models using boolean networks are found in Wuensche (1994) , Capra (1996), and Somogyi and Fuhrman (1998).

Biological systems are complex self-regulatory systems. Clearly they are not random, but rather ordered systems. Their description can be verbal, logical, or with equations. Kinetic logic is an attempt to capture feedback regulatory mechanisms using logic (Thomas, 1990). This method includes asynchronic feedback loops that are intended to represent cellular dynamics. Martinet-Edelist (1994) applied this method to the study of virus dynamics.

A relational model proposed by Rosen (1972, 1985), named Metabolic-Repair (M-R)-systems makes a distinction between the repair machinery and metabolic activities.

Interactive systems, according to Wegner (1997, 1998), express a rich behavior and cannot be modeled by Turing machines. The behavior of interaction machines is

characterized by their *interaction histories* which are *non-enumerable*. Open systems like living systems are interactive systems, since the boundaries of the interaction histories is unknown; on the other hand Turing machines are completely closed systems. Interactions bring information into the system and improve computational success. Wegner, following a holistic point of view, proposes an "irreducibility" thesis "observable behavior of components is not expressible by inner behavior specifications; interaction semantics is not expressible by state-transition semantics and vice versa".

2.2 Hierarchical systems

There are many ways to define a hierarchical system, each depending on the context and discipline. Hierarchical social systems can be described as groupings of closely related entities, forming other higher scale entities with different functions. Here the hierarchical levels run from persons to cities, regions, countries, and so on.

The definition of hierarchies given by Simon (1962, 1973, 1996) is rather circular. By hierarchical system, he understands "a system that is composed of interrelated subsystems, each of the latter being in turn hierarchic in structure until I reach some lowest level of elementary subsystem".

A hierarchy can be defined as a partial ordering of a sets, as a universe of set within sets. The relationship between two sets of different hierarchical level is antisymmetric and transitive (Webster, 1979). However, in biological systems the effects across scale are not transitive (Salthe, 1985). O'Neill et al. (1986) define "constraints" to the asymetric relationship between hierarchical levels.

I define a hierarchy H as a 3-tuple H = <E, I, P>, where E is the set of elements E = { e1, e2, e3, .. }, I is the set of interactions or relationships among the elements,

including reflexive interactions, forming a network I = {i1, i2, i3, ...}, P is the set of properties (physical or logical) of that hierarchy P = { p1, p2, p3 ..}. From an atomistic point of view, elements of one hierarchy can contain other hierarchies of lower scale, until an atom-like structure is reached. This atomic-like structure is called the *hierarchy unit* H_0, defined by an element, a reflexive interaction, and a given set of properties (Figure 2.2.) Every hierarchical unit H_0 differs from others in at least one property (i.e., spatial location, charge, state, etc). Therefore it is not possible to abstract a global entity from a set of such entities without losing some information.

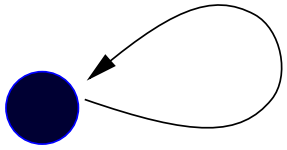


Figure 2.2: The Hierarchical Unit H_0. It is an atomic-like structure with its reflexive interaction.

The nested hierarchies form the hierarchical system (Figure 2.3) with the lowest level the elements of the hierarchy unit (H_0), the medium level (H1), and the higher level (H2). A set of elements of a given level form the next hierarchical level. For example, when a set of elements of level H1, form an element of level H2, the set of properties of H2 are different than those at H1, and the interactions are also different (even when they involve elements of level H1).

One property of any hierarchy is ***connectivity***. Connectivity indicates how strong the connections between the elements of one set of a given level are.

Connectivity of elements within a hierarchical level is stronger than connectivity between entities of the same hierarchical level. For example in Figure 2.3, connectivity among elements of level H_0 is stronger than connectivity among elements of level H1.

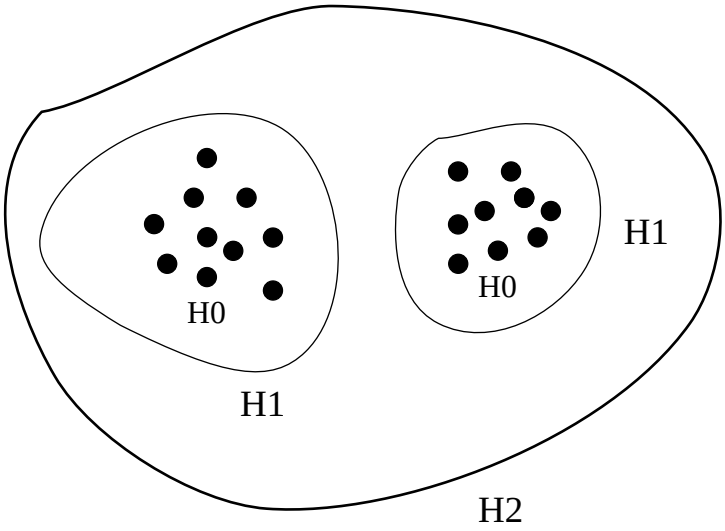


Figure 2.3: A nested hierarchical system

According to Simon (1996), there are two types of interactions in hierarchies: interactions among subsystems and interactions within subsystems. The interactions among subsystems are considered “weak” (they have low energy bonds, low frequency dynamics, based on the time that takes to finish a process relative to a given level) as compared to the relatively stronger interactions within subsystems (they have high energy bonds, and high frequency dynamics). He describes such subsystems as ”nearly decomposable” since ”interactions among the subsystems are weak but not negligible” (Simon, 1996).

The importance of hierarchies in understanding the origin of operation and the origin of life is addressed by Pattee (1970). He considers “hierarchical organization” to be a coherent collection of entities, such as found in living matter, and “hierarchical control” to be the main feature of this type of organization. Pattee states that the structure of hierarchies can be distinguished by forces, numbers, and time scales (Pattee, 1972, 1973). The strongest forces are responsible for smaller or lower-level structures. The time scale

is related to the forces: shortest time is related to stronger forces (occur at lower levels), and longest time is related to weaker forces and larger structures, that occur at higher levels.

Pattee (1972) recognized the following properties of control hierarchies:

- **A control hierarchy constrains the behavior of the elements of a collection so that they perform some coherent activity**. Thus the elements of the collection do not behave randomly but their behavior is under hierarchical control.

- **The coherent activity of hierarchical control systems is simpler than the detailed activities of its elements**. Thus some detail is lost in the control operation and the hierarchical control selects which features of the elements are most important for the behavior of the collective system.

- **New hierarchical constraints can continue to appear at higher levels without destroying the existing constraints at the lower levels**, suggesting that "if we could discover how any new functional organization or new classification is created spontaneously from a set of more or less disordered elements, we could generalize this discovery into a theory of hierarchical origins" (Pattee, 1972).

- **There are many physical structures that execute the same function and there are many descriptions of the same physical structure**. Pattee considered this an essential property of life, raising problems about limitations of language for describing structures, and limitations on defining functions.

Koestler (1976), proposed *holons* as hierarchical "quasi-autonomous" entities that

maintain individuality, but at the same time are part of a larger whole. This duality ”is inherent in the concept of hierarchical order”, and is therefore a universal characteristic of hierarchical elements of living systems. In short, the holon is an integrated body of its parts that will form, another holon, with others of their class. For example, holons can be cells, tissues, organs, individuals, populations, species, etc.

Allen and Hoekstra (1992), proposed criteria to order higher levels over lower levels as described below:

- **Bond strength**. The higher the level, the weaker the strength of the bonds that hold entities at that level together.

- **Relative frequency**. Higher levels have a longer return time in their processes, that is, they behave at a lower frequency.

- **Containment**. Configuring nested systems where the upper level is composed of lower levels. In such systems, the whole changes more slowly than its parts, and the whole is clearly the context of its parts.

- **Constraint**. Where “upper levels constrain lower levels by behaving at lower frequency, by doing nothing or even by refusing to act”(Ahl and Allen, 1996). There exist filters between levels that allow the passing of only certain information.

Conrad (1972, 1983) described hierarchical organizations of living systems with compartments. The compartments of the biota are cells, organs, organisms, and populations. The environment is decomposed into regions. The compartments are nested, and in order to avoid redundancy in the descriptions, the concept of partial state is defined

for each compartment. Thus, it is possible to specify the state of the biota in terms of the state of each of its compartments. The compartments are defined by the density of the connections among the components and by the scale in which they act (Conrad, 1995b).

Auger (1989) studied hierarchies in the framework of mathematics. Auger's hierarchies are groupings of strongly connected elements into more loose connections at the superior level.

2.3 Evolution and adaptive surfaces

Evolution is the collective effect of changes of gene frequencies of individuals in a population. *Natural selection* is the main agent of evolution. Natural selection results from interaction between organisms and their environment, and operates by "differential reproductive success" (Pianka, 1994) (i.e., success in transferring genetic information into the population). The mechanism of *variation* and *selection* increases the survival of populations in their environment ("fitness").

Types of natural selection.

According to Pianka (1994), there are several types of natural selection. Populations can have stabilizing selection (if the fitness actually remains the same), directional selection (when the population shifts its phenotype in order to reach a new fitness), or disruptive selection (when the population splits their phenotype).

Other types of natural selection are: frequency-dependent selection, which "occurs when the fitness of a particular phenotypic trait varies with its frequency in the population" (Pianka, 1994); age-specific selection; density-dependent selection; density-independent selection; kin selection; and sexual selection. A comparison of evolutionary

approaches to computer science is found in Bäck (1994), and Bäck et al. (1997).

The steps of evolutionary systems by natural selection are given in the algorithm of Figure 2.4. There are two main steps: **variation**, and **selection** of sub-populations that have the best fitness value.

Require: Sub-populations and the environment
Ensure: Increase the fitness of sub-populations
1: **loop** $\{$Forever$\}$
2: Sub-populations reproduce introducing mutated individuals (**Variation**)
3: **for** Each sub-population P_i **do**
4: Check the fitness for each sub-population P_i (**Selection**)
5: Some sub-population P_i has more reproductive success than the others.
6: **end for**
7: The population and external agents may modify the environment.
8: **end loop**

Figure 2.4: A generic variation-selection algorithm to evolve subpopulations towards their highest fitness value.

Adaptive surfaces.

Evolution may be viewed as a "hill-climbing" process on adaptive surfaces (Wright, 1932), leading to some global or local optima for each population. The key is to find the global optima, but the problem is that sometimes populations can get stuck in local optima.

In order to simulate an evolutionary system in a computer, it is convenient to have an idea of the "fitness", perhaps plotting the phenotype or genotype against some performance measure called fitness. This plot is also referred as an *adaptive landscape*. Many times the fitness is predefined; in other cases the fitness is an emergent property in the model, as for example in the evolutionary ecosystem models of Conrad and Pattee

(1970), and Conrad and Rizki (1980).

It is possible to increase the dimensionality of the adaptive landscape in such way that local peaks are converted into saddle points, making it possible to find a pathway to escape a given local optimum (Conrad, 1990a). This “extradimensional bypass” improves the possibility of finding the optimal point.

According to Simon (1973) hierarchical systems allow a speed up of evolution: “hierarchies will evolve much more rapidly from elementary constituents than will do non-hierarchic systems containing the same number of elements. Hence, almost all very large systems will have hierarchic organization” (Simon, 1973). Complexity of hierarchical systems improves the search for optimal points in the adaptive landscape.

2.4 Emergent properties

According to Simon (1996), emergence in the strong sense ”postulates new system properties and relations among subsystems that had no place in the system components”. The weaker interpretation of emergence states: ”emergence simply means that the parts of a complex system have mutual relations that do not exist for the parts in isolation”. In that sense a molecule has a function when it is in a given medium, but not in isolation.

I will define emergent properties as those collective properties that cannot be found inside a hierarchical level. For example, the behavior of each cell of an organ does not define all the properties of the organ; the behavior of every neuron can not express the behavior of the brain as a whole. This anti-reductionistic point of view, states that there are collective properties that cannot be deduced from the inside or from the entities in isolation, but can be observed from higher hierarchical levels, or when the relationships among the entities are established.

Another way of expressing this concept of emergence is with Figure 2.3. The properties and relationships of hierarchical level H2 cannot be deducted from the properties of every element of level H1. Thus I say that H2 has emergent properties over H1.

2.5 The problem of scale

Scale is a very important feature in nature. Among the properties where scale is observed are: dissipation, mass, time, number of objects, object lifespan (Feekes, 1986), and energy (Auger, 1989; Feekes, 1986; Pahl-Wostl, 1995). Moreover there are features unique at each level. For example, an atom has quantum properties that are not evident at the macro level (Conrad, 1994; Rosen, 1974).

The problem of scale and its importance was recognized by Levin (1992),and Holling (1992). Other references to the problem are in Allen and Hoekstra (1992); Allen and Starr (1982); Conrad (1979c, 1983); O'Neill (1989); O'Neill et al. (1986); Pahl-Wostl (1995); Pianka (1994); Webster (1979).

Levin (1992) gives importance to scale and the multi-scale problem in the following terms: [there is] "no single natural scale at which ecological phenomena should be studied; systems generally show characteristic variability on a range of spatial, temporal, and organizational scales. ... The key to prediction and understanding lies in the elucidation of mechanisms underlying observed patterns. ... Those mechanisms operate at different scales than those on which the patterns are observed."

Holling (1992) concludes that "the landscape is structured hierarchically by a small number of structuring processes into a small number of levels, each characterized by a distinct scale of "architectural" texture and of temporal speed variables". He also

states the need to develop a “meta-model” that would “focus explicitly on cross-scale interactions”.

An important fact to consider are properties across hierarchical levels. For example, Bonner (1965) found that there exist a linear log-log relationship between the length of an organism and its generation time. Rosen (1967), called this log-log relationship in biological systems an allometric law. Moreover, Holling (1992) found that there exists a log-log relationship among different hierarchical levels with respect to space and time of development (in the case of a forest) or decision time (in the case of carnivorous mammals). Similar relationships are described by Holling (1992), Pahl-Wostl (1995), Allen and Starr (1982), Peters (1983), and others. Models of hierarchies including scale should be able to capture at least some of the scale features bounded by limitations of memory and speed.

2.6 Concluding remarks

Biological organisms are complex, self-regulatory, hierarchical organized systems. This structure, based on levels of hierarchical organization, is appropiate for evolutionary search of optimal points in the fitness landscape, thereby speeding up evolution.

CHAPTER 3

ANTECEDENTS OF THE HYPERNETWORK MODEL

This chapter describes some of the conceptual antecedents of the hypernetwork model and related architectures. Section 3.1 briefly reviews evolutionary algorithms, section 3.2 describes neural network models, and section 3.3 describes Conrad's enzymatic neuron model. Finally, section 3.4 describes the percolation network model that addresses information flow across scales.

3.1 Evolutionary algorithms

The view of evolution as a metaphor for problem solving is the origin of evolutionary computation methods. Many computational problems involve finding solutions in a huge search space, like biological species do when they adapt to their niches in evolutionary time. Variation-selection mechanisms are central to the evolutionary computing approach to problem solving.

Evolutionary algorithms cover four main closely related approaches: genetic algorithms, evolutionary programming, evolution strategies, and cultural algorithms.

Genetic algorithms introduced by Holland (1975), uses a genetic representation to represent the solution of a problem. For a given problem, there is a population of possible solutions encoded in "chromosomes" (frequently using binary strings). A fitness value is evaluated for each "chromosome", reflecting the quality of the solution. Genetic operators are applied to create a new population: "Selection" of the best individuals to be reproduced; "crossover", where two parents recombine their genes, and random mutation are applied to produce an offspring.

An extension of genetic algorithms into programs is called **genetic programming**

(Koza, 1992). Programs usually expressed in trees, after variation (due to mutation or crossover) are selected until the best one is selected to satisfy a fitness function.

Evolutionary programming was originally intended to create artificial intelligence (Fogel et al., 1966; Fogel, 1999). The representation of the problem may be tailored to the problem, and during the mutation and creation of new individuals, no recombination operators are performed.

Evolution strategies were developed by Rechenberg and Schwefel (1995). They use a real value vector representation for optimization problems. A population of vectors is randomly created. The offspring vector is created by modifying each value according to a given distribution. Then, individuals with the least error are the parents for the next loop.

Cultural algorithms, introduced by Reynolds (1994), have two levels of evolution: cultural (macro level) and population (micro level). The cultural level is a model of human culture, a kind of knowledge repository that accelerates the problem solving process. The population level may be implemented by any evolutionary system, such as evolutionary programming, genetic algorithms, or evolution strategies. The knowledge, accumulated and processed in the ”belief space”, directs future individual actions. Evolution takes place at the population and at the belief space. There are communication channels between the macro and micro levels, and a scheme of updating the knowledge by selected individuals.

3.2 Neural network models

The fact that memory, the thought process, and in general intelligence are in relationship with neurons and their connectivity is the origin of the neural hypothesis for building neural network models.

There are between $7x10^{10}$ and $8x10^{10}$ neurons in humans (Schüz, 1995) and about thousands of contacts in the neurons (i.e., 100,000 synaptic connections in human Purkinje cells (Smith, 1994)).

According to the model, the brain is considered to be "a densely connected electrical switching network conditioned largely by the biochemical processes" (Zurada, 1992).

The neuron proposed by McCulloch and Pitts (1943) is a simplified unit, where the output is evaluated according to a non-linear function (usually a sigmoid function) of the inputs to the neuron. The inputs are a representation of dentritic connections from other neurons or from the environment, and the output is a binary signal to the axon (see Figure 3.1).

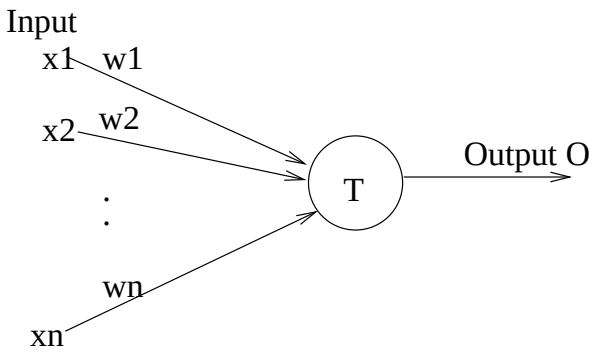


Figure 3.1: A McCulloch-Pitts neuron. The neuron has a thresholding element T that fires as a function of the weights w_i of the activated dentritic inputs x_i.

The first learning algorithm was proposed by Hebb on the basis of the formation of 'cell assemblies' with the reinforcement of synaptic connectivities (Hebb, 1949). In the model information is stored in the connections.

Single layer network of neurons called **perceptrons** were introduced in 1958 by Frank Rosenblatt. Perceptron neurons have limitations with respect to solving non-

linearly separable problems (Minsky and Papert, 1969). Multilayer learning methods were designed by Werbos in 1974 (Werbos, 1994) and independently by McClelland and Rumelhart (1986) solving non-linearly separable problems. Later, other supervised and unsupervised learning methods, feedback networks, associative memories, and many other variations were created. The description of these methods is beyond the scope of this work, but I should state that all of these, more or less, are based on the McCulloch-Pitts neuron model.

The neural networks have the following features according to Gurney (1997): Information is stored in the weights and distributed across the network; computation is relatively robust to noise and hardware failure; neural networks algorithms have good generalization properties.

3.3 Molecule-based models of information processing

The **enzymatic neuron model** proposed by Conrad (1974) is a theoretical model involving molecules as decision making elements. The abstraction of the neuron involves enzymatic dynamics that will form the output of the neuron. Moreover, he proposed the formation of networks with layers of enzymatic neurons, and proved that any McCulloch-Pitts network can be simulated by some enzymatic network. Networks of enzymatic neurons are amenable to evolution by "trial and error selection of the excitases [enzymes]" (Conrad, 1974).

A computer implementation of the enzymatic neuron is described in Chen (1993), where the system consists on "cytoskeletally controlled enzymatic neurons" (Chen and Conrad, 1997b) and layers of memory access neurons for indexing, named reference neurons (Conrad, 1977). This model was used to solve some navigational tasks and

categorization of Chinese characters (Chen and Conrad, 1997a).

The cytomatrix neuron model (Ugur and Conrad, 1999) is a neuron with multiple molecular dynamics, trained with an evolutionary learning algorithm. This architecture solved the 2-bit and the 4-bit parity problems, but it has difficulties in solving the 8-bit parity problem.

3.4 The percolation network model

The percolation network model is a multi-scale model that stresses the flow of information among different hierarchical levels (Conrad, 1979b, 1984, 1993, 1995a,b, 1997). Information percolates across scale if it is neither ignorable nor eliminable for the purposes of calculating the time development of the system as viewed at other scales. These influences go both bottom-up and top-down. Influences from the environment impinge on the top level of the system. The influences then filter down into lower levels (organs, cells) through “integrative dynamics”, until they reach lower levels (meso and microscale). Specific influences percolate to upper levels by “selective amplification” until they again reach the environment at the top level (see Figure 3.2).

The cross-scale interactions can be exemplified with any action that the organism takes. When an organism expresses a behavioral action, it cells change state, and inside the cells molecules also undergo dynamic and structural changes. The influences of higher levels are filtered into the molecular level. In turn, molecular changes also are percolated up to affect the behavior of the organism.

The cyclic AMP system of neurons provides an example of cross-scale interactions and of the transduction-amplification process. Neurotransmitter molecules (e.g., epinephrine, vasopressin) bind excitatory receptors that activate G-proteins; these are first

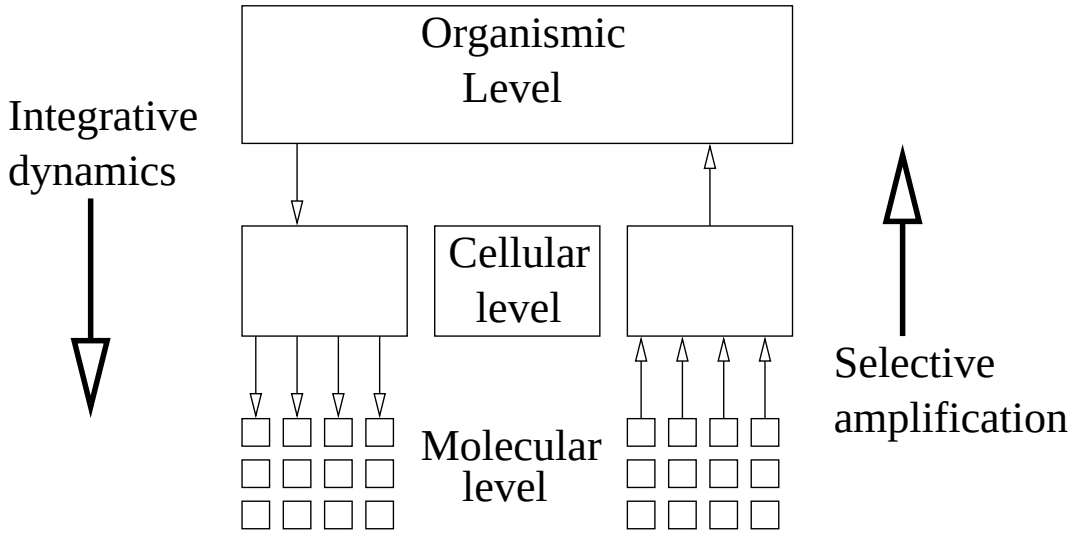


Figure 3.2: Schematic representation of a percolation network.

stage amplifying components that activate adenylate cyclase, the enzyme that produces the second messenger cAMP from ATP. The cAMP in turn activates target proteins (protein kinases) that activate effector proteins. These might be associated with DNA, with cytoskeleton, with synaptic vesicles, or may be ion channel proteins that control nerve impulse activity (Shepherd, 1994). The net result is a change in the internal network of the cell that can percolate up to higher levels of organization.

The cyclic nucleotide system is ubiquitous in the cells of higher organisms. Its role in controlling the firing behavior of central neurons suggests that the percolation network principle is operative in the brain (Liberman et al., 1985). The immune and developmental systems can also be viewed in this manner. The general feature is this: external influences (e.g., photons, messenger molecules) filter down to alter internal networks of molecular interactions within the cell. The influences are combined in space and time through these interactions, leading eventually to activation of cellular effector molecules. The

interactions percolate up from the molecular to the cellular level through interactions between effector and receptor molecules of different cells. Integration and selection of information occurs at all stages. Multiple scales contribute synergistically to this process. The key underlying mechanism is molecular conformation. The structure and function in biological systems is most essentially controlled by shape interactions among macromolecules.

3.5 Concluding remarks

Traditional neural network models view neurons as a switching units, where the internal dynamics is simplified down to a threshold function. However, neurons are complex information processing cells. A model that incorporates neuron enzymatic dynamics called “enzymatic neuron model” was proposed by Conrad (1974).

Evolutionary algorithms find solutions by searching with variation-selection mechanisms. These models typically do not have an explicit representation of hierarchies, or of the information flow across levels.

The hypernetwork architecture described in the next chapter is based on the enzymatic neuron model including a representation of hierarchical levels and molecular evolution.

CHAPTER 4

THE HYPERNETWORK ARCHITECTURE

This chapter provides the following. Section 4.1 contains the main features of the model description at each hierarchical level, the molecular, cellular, and organismic. Section 4.2 describes the levels of evolution in the hypernetwork architecture. Section 4.3 describes the molecular evolution algorithm, and section 4.4 is a discussion of the hypernetwork architecture.

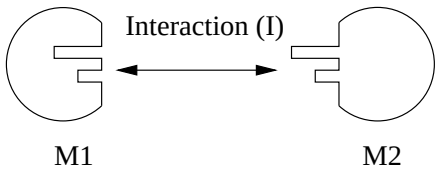
4.1 Model description

The hypernetwork architecture is inspired by a biological system (i.e. neuronal tissue), and attempts to capture the following properties: cross-scale flow of information, learning, evolvability, and mutation-buffering.

The design of the model is bottom up, from the molecular to the organismic levels. A first view is shown in Figure 4.1. The system has an environment with which it interacts via input and outputs.

4.1.1 Molecular level

This level is modeled by molecules reminiscent of proteins represented by a binary string. The *self-assembly* of proteins and other macromolecules, “a self-organizing process driven by free energy minimization” (Conrad, 1992), is modeled by *interactions* based on shape complementarity. Two molecules react if their recognition sites are complementary to each other as shown here:



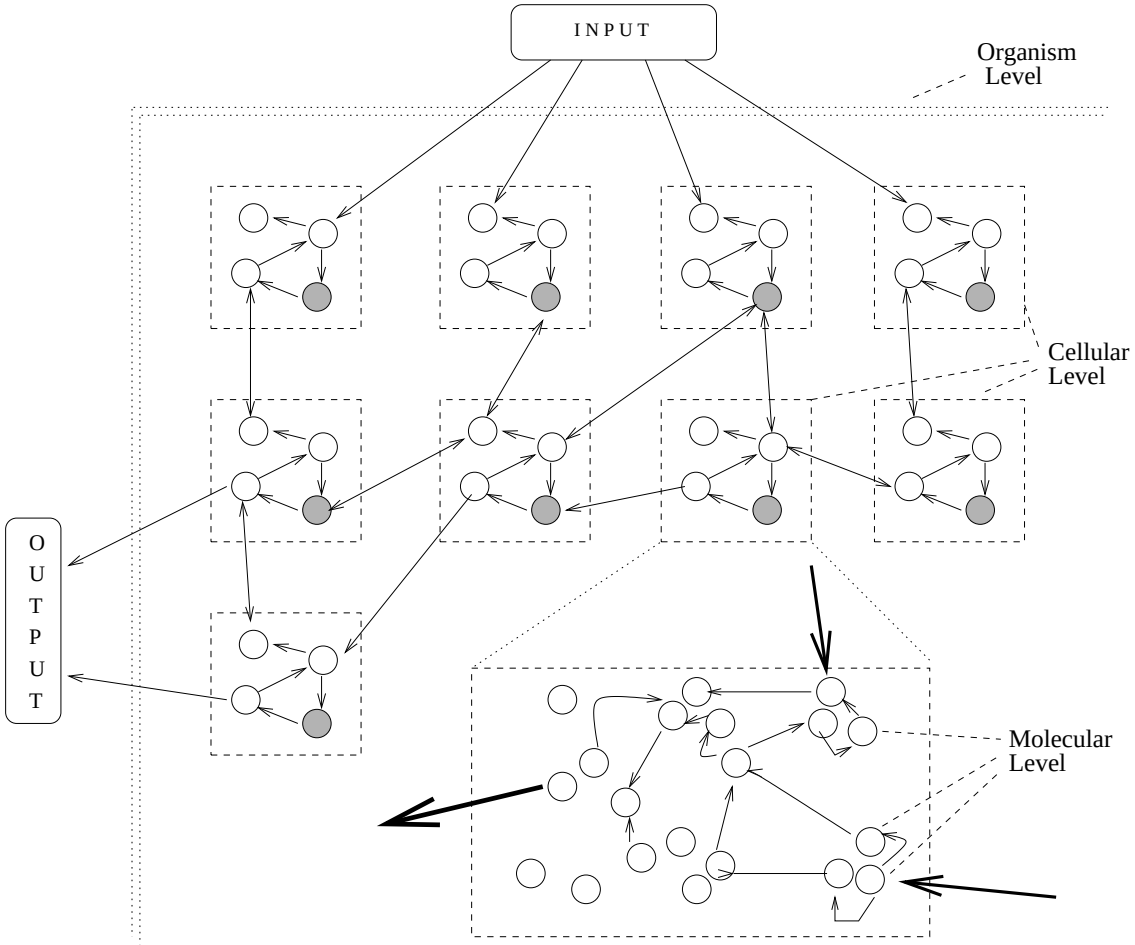


Figure 4.1: Three hierarchical levels of the hypernetwork model: molecular, cellular, and organismic. The arrows show interactions.

In a previous design (Segovia-Juárez and Conrad, 1999) the molecule had just one site with excitatory and catalytic properties. Currently the molecule has three parts of up to 14 bits each (see Figure 4.2). There are two receptor sites, *excitatory* and *inhibitory*, that put the molecule into the active state or the inactive state, respectively. The third site is called the *catalytic* site, by means of which the molecule will activate neighbor molecules if there is complementarity matching (i.e., matching above a threshold) to the receptor sites of the target molecules (Segovia-Juarez and Conrad, 2001).

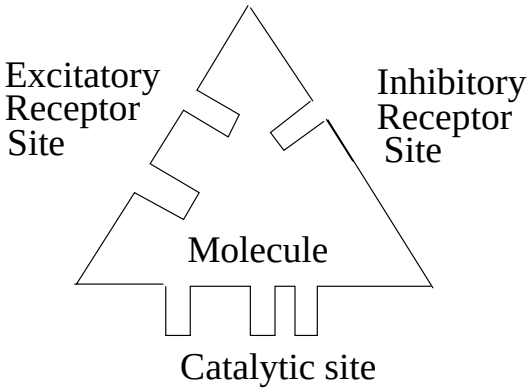


Figure 4.2: Representation of a molecule with inhibitory, excitatory and catalytic sites.

The existence of inhibitory and excitatory sites allows the creation of *positive* and *negative feedback regulatory networks* with neighbor molecules, that are described in more detail in Chapter 6.

Molecular types The model has four types of molecules: receptors, effectors, internals, and readouts.

- Receptors are molecules that are sensitive to external influences (from the environment or from molecules of other cells).
- Effectors are molecules that transfer information out of a cell into other cells via their receptors. They behave like neurotransmitters in neural tissues.
- Internals are molecules that, influenced by receptors, will form networks of interactions inside the cell.
- Readouts are molecules that obtain information from the effector molecules of output cells and send information to the external world.

Molecular States Molecules have the following states:

- Ready: When a molecule is waiting to be activated by one of its neighbors.
- Active: When a molecule is activated by one of its neighbors, then for one time step it can activate its neighbors if there is complementarity matching.
- Inactive: After being activated, the molecule will be inactive for one time step before going back to the *ready* state.
- Delayed: Occurring just in receptor molecules, this is the time delay until the molecule is activated. This is to simulate the timing of the interactions among cells.

The state diagram of the internal and effector molecules is shown in Figure 4.3, and that of receptor molecules is shown in Figure 4.4.

Output cells have readout structures that read the state of a particular molecule. If that molecule was activated, then the readout will be in an active state, otherwise it will be in the inactive state. A readout structure reads the state of just one molecule, but this could vary in future implementations.

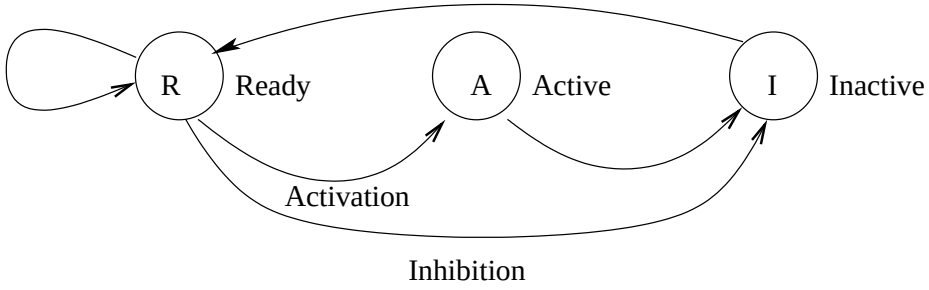


Figure 4.3: The state diagram of the *internal* and *effector* molecules.

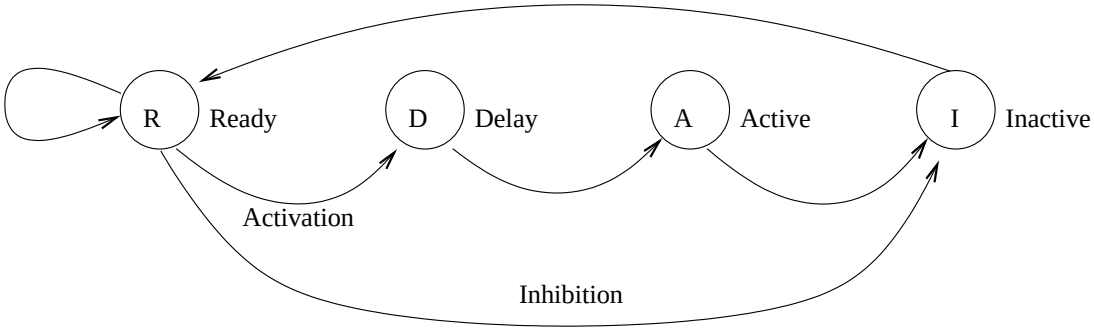


Figure 4.4: The state diagram of the *receptor* molecules.

4.1.2 Cellular level

The cell is modeled by a two-dimensional cellular automaton with wrap-around. The cascades of molecular reactions are modeled with *networks of molecular interactions* inside the cell.

Every cell has *receptor*, *internal*, and *effector* molecules. The output cells also have *readout* molecules. Molecules are placed randomly in the locations of the grid, and the relationship of neighborhood is shown in Figure 4.5 where molecule M may have interactions with eight neighbors.

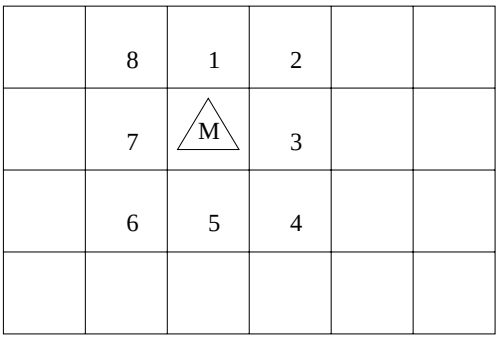


Figure 4.5: A molecule M, in the cell, interacts with eight neighbors.

Interactions start when the molecule is active. Using its catalytic site, it searches every one of its neighboring molecules for complementarity matching at their excitatory and inhibitory sites. The target molecule will be activated in the next time step if the matching is above a threshold and there is not any inhibitory matching to it. In case there is an inhibitory match, the target molecule will go to the inactive state in the next time step, regardless of any other activation.

The receptor molecules of the input cells are activated by external influences as described below.

Types of cells

There are three cell types:

- Input cells: have receptor molecules that gather influences from the environment.
- Internal cells: do not interact with the external world.
- Output cells: have readout molecules that communicate the state of the system to the environment.

The number of cells and the number of molecules in each cell are initial constants in this version of the model, but this can evolve in future enhancements. Experimental hypernetwork organisms usually have 25 to 49 molecules for each cell, and from 15 to 36 cells for each organism. In the cell, the number of receptor and effector molecules may vary between 4 and 10. Output cells have readout structures randomly located.

4.1.3 Organismic level

A spatially organized group of cells constitutes an organism (an organism is shown in Figure 4.6). The arrangement is given primarily by the cellular function. Input cells gather influences from the environment, and output cells deliver the global state of the organism to the environment, through the states of their readout molecules. In the experiments the organisms have two layers of internal cells.

In Figure 4.6 the potential cell to cell interactions are shown with dotted lines. The actual interactions are between the effector and receptor molecules of the respective cells (Figure 4.7). The cell-cell interactions can change depending on the influences of a particular input.

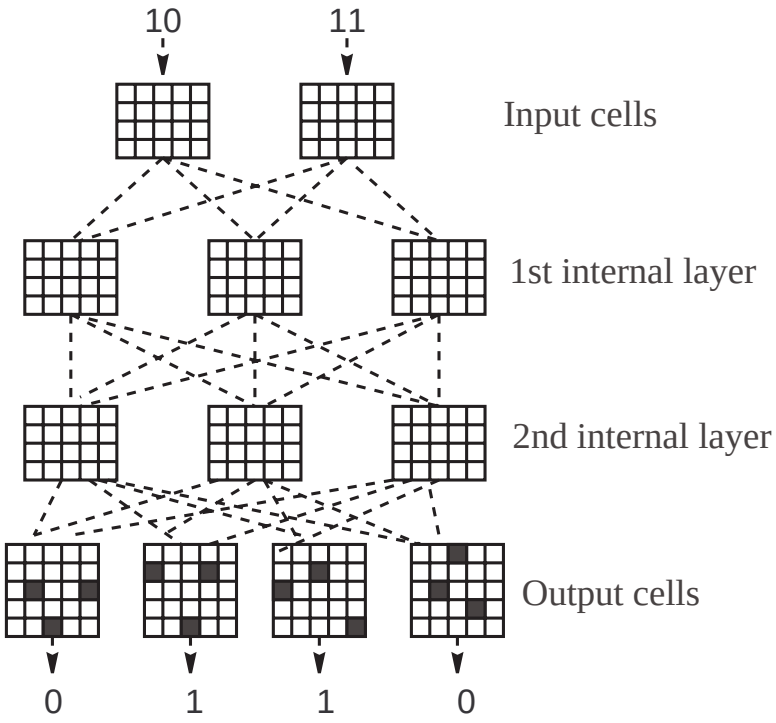


Figure 4.6: A hypernetwork with two input cells, two layers of internal cells, and an output layer with four cells.

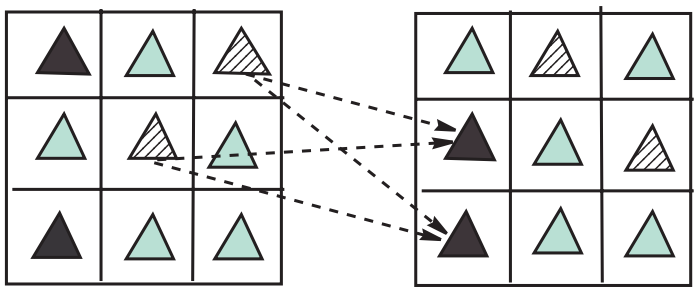


Figure 4.7: Effector-receptor inter cellular interactions. The Figure shows two cells with their molecules. Receptors are black, effectors are dashed, internal molecules are gray. Dotted lines show the potential effector-receptor interactions.

4.1.4 Population level

This level is not yet implemented, but involves two or more evolving populations with different organisms. Future implementations of the architecture can include this level. Several populations could compete in the search for global optimum.

4.2 Levels of evolution in the hypernetwork architecture

The hypernetwork architecture supports evolution acting at different hierarchical levels. Molecules can change their structure, cells their function, organisms their behavior, and at the population level there is speciation. The hypernetwork architecture allows the study of evolution at those different hierarchical levels. Every evolutionary level includes the next lower one, as a nested hierarchy:

Molecular level evolution.

At this level molecular structures are mutated during reproduction. Every molecule has a probability of mutation, independent of any other factor. The molecule to be mutated can change a fraction of its structure randomly. The rate of mutation and the fraction of molecular change are experimental parameters.

Cellular level evolution.

At this level the organism can change, add or delete molecules within a cell, set the molecular types (effector, receptor, internal), and modify the location and number of readout structures. It includes the molecular level evolution.

Organismic level evolution.

This level includes changes in the number and types of cells, as well as relationships

among cells. It includes the cellular level evolution. The theory of neuronal group selection of Edelman and Finkel (1984) can be included at this level.

Population level evolution.

This level is based on reproduction with mutation and selection of the best individuals. It includes evolution at the organismic level.

Ecosystem level evolution.

At this level several populations evolve to learn different tasks. They may compete or cooperate in the process. It includes evolution at the level of populations.

4.3 The variation-selection algorithm

At the present state of development the hypernetwork architecture has implemented the molecular level coupled with the population level of evolution. An overview of the variation-selection algorithm is shown in Figure 4.8. An organism is reproduced with molecular mutation (evolution at molecular level), then its performance or fitness is evaluated. The performance of the child with the parent is compared, and choose the best one as the organism to be reproduced in the next loop.

The hypernetwork gathers influences from the environment using receptor molecules in the input cells (see Figure 4.6). Influences flow through the organism by dynamical formation of networks of interactions of molecular structures. The interactions, as mentioned before, are based on the shape complementarity of molecules. Finally, the influences arrive at the effector molecules of output cells, where the information is gathered from the outside by means of the readout structures. The state of each cell is "ON" if any readout molecule is active, "OFF" otherwise.

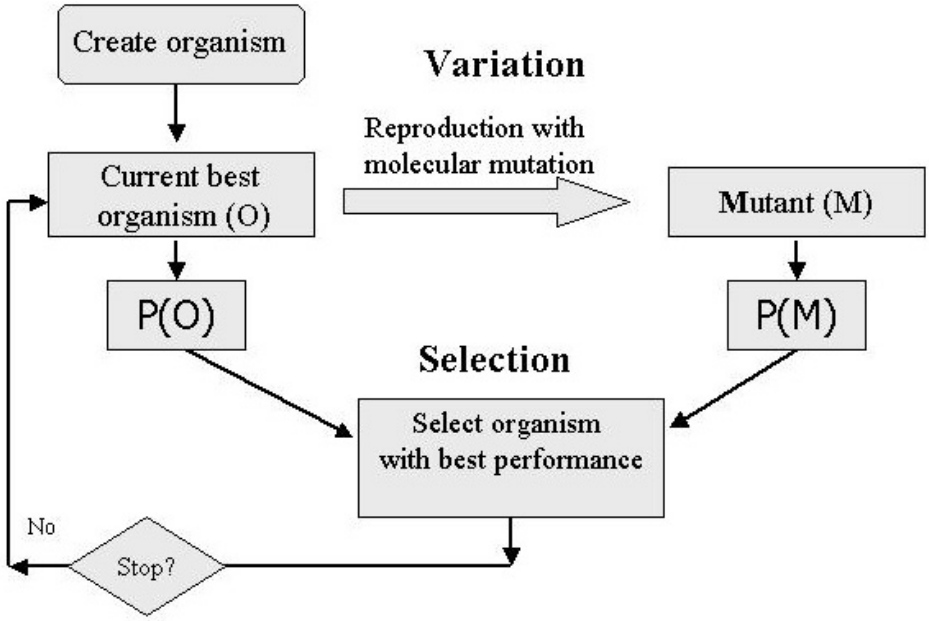


Figure 4.8: An overview of the variation-selection algorithm

Influences from the environment come in the form of a binary string that will activate the receptor molecules of input cells. and the output of the organism is another binary string formed by the concatenation of states from the output cells.

The output string is compared with a desired one to measure the performance of the organism for this particular input. Then, the process is applied to every input vector of the set to be learned by the algorithm. This iteration is called an *epoch*.

After the algorithm has processed all the input vectors from the set, its global performance is evaluated. A detailed description follows in section 4.3.2.

4.3.1 Initialization of the organisms

The initial organism is created with a previously established molecular size, cellular size, possible cell to cell interactions, and number of cells. The molecular structures are initialized randomly, as are the locations of the readout molecules in the output cells.

4.3.2 Testing the performance of an organism

The next three steps are performed for every input vector.

- Input of influences into the hypernetwork

  The input vector is split into small two bit fractions. Each of these will activate an input cell (via activation of its receptors). Molecular receptors have two-bit molecules at their receptor site. Every input cell has at least four different receptor molecules, thereby assuring that at least one of them will be activated by each fraction of the input vector.

- Spread of influences

  Once the receptor molecules of the input cells are activated, they will activate their neighbors. In this way the influences travel through the cell until they reach effector molecules. Once the effector molecules are active, they will search for receptor molecules of other cells, according to the cell to cell topology. The cell-cell interactions are formed dynamically, based on which effectors were activated at that time.

- Generation of output vectors

  Eventually, influences will activate the effector molecules of the output cells. When this happens the output of the cell is evaluated according to the state of the readout molecules. The global state of each output cell is “1” if there is at least one readout molecule activated, otherwise its global state will be ”0”. The output vector is formed by the concatenation of all the cellular binary states. Then the Hamming distance from the output vector to the desired output vector is measured.

The performance of the organism was obtained from the sum of distances evaluated previously for each input vector.

4.3.3 Reproduction with molecular mutation

Once the performance or fitness of the organism is obtained and if it is below 100% learning or a termination condition is not fulfilled, the organism is reproduced with mutation. Every molecule has a small probability of mutation, and it changes just a fraction of its structure.

The performance of the offspring is evaluated and compared to its parents as described above. The algorithm for hypernetwork learning is described in Figure 4.9.

Require: Two organisms, Input Vectors (I), Desired output vectors (D)

1: Initialize organism A
2: Reproduce organism A with mutation to generate organism B
3: **repeat**
4: **for** Organism A and B **do**
5: **for** Every input vector I_i, desired output vector D_i **do**
6: Read input vector I_i into **input cells**
7: **repeat**
8: Propagate interactions through cells
9: Form cell to cell interactions
10: **until** Effectors of output cells are activated
11: Read output vector O_i from the **output cells**
12: **end for**
13: Evaluate the global error for each organism $E_o = \sum_1^n (O_i - D_i)/n$
14: **end for**
15: *Selection.* Obtain the organism with best performance (smaller E_o) to become organism A
16: **if** $(E_A > \epsilon)$ **then**
17: *Variation.* Reproduce organism A with mutation to generate organism B
18: **end if**
19: **until** $(E_A \leq \epsilon)$ or termination condition

Figure 4.9: The variation-selection algorithm for hypernetwork learning.

4.4 Discussion

Features of scale are inherently difficult to model in computers. For example the actual number of molecules of a cell could not even be represented. However, the representation of hierarchies in the hypernetwork model contribute to understanding the complex system and to building one with more features amenable to evolution.

Scale features are modeled with the relative proportionality of entities at each level. In the experiments the number of molecules are kept at about ten times more than the number of cells, in each organism. To represent the frequency of changes and the relationships of strength of interactions at different levels, the architecture has a smaller number of cell to cell interactions than of molecular interactions, and there is a delay of one time step after the receptor-effector molecules of different cells interact.

There are major differences between neural nets and the hypernetwork architecture. Neural nets do not represent scale features, and the cell dynamics is represented by a threshold function. Neural nets store their memory in the synaptic weights. In contrast the hypernetwork architecture incorporates a representation of scale from the molecular level potentially up to the ecosystem level. Moreover, the hypernetwork stores its memory in its molecular structures distributed across the organism, and the synaptic connections are formed dynamically by external influences.

4.5 Conclusions

The hypernetwork architecture has two components: the structure and the dynamics. The structure consists of molecular, cellular, and organismic levels. The dynamics is based on the molecular interactions modeled after protein self-assembly. External influences on receptor molecules of input cells percolate into the cells to trigger cascades

of molecular reactions, expressed in the form of networks of molecular interactions. Intracellular dynamics are filtered up to higher levels to the organismic level. The output of the organism is obtained from the readout molecules of the output cells.

The adaptive algorithm for learning is based on molecular evolution. An organism is reproduced with molecular mutation, and the best organism is selected for the next iteration.

CHAPTER 5

SOLVING PROBLEMS WITH THE HYPERNETWORK MODEL

The current implementation of the hypernetwork was evaluated with four sets of problems:

- The N-input parity problem
- The tic-tac-toe endgame problem
- The two x two bit multiplier
- The double spiral problem

In the next section I will show the results of learning for these problems.

5.1 Solving the N-input parity problem

The N-input parity problem is to compute the odd parity of binary strings of length N (2^N vectors, it is one of 2^{2^N} different boolean functions). The network must output a ”one” if the input has an odd number of ”one” bits, and ”zero” otherwise. The two-input parity problem (XOR function) is known to be impossible to solve for first order perceptrons (Minsky and Papert, 1969). Tesauro and Janssens (1988) found that a neural network with back-propagation could learn up to N=8, but it did not converge for N=10. The *NOVEL* method, a global optimization method in a neural network, can learn up to 99.8% in the case of N=10 (Shang and Wah, 1996). This problem was found difficult for genetic algorithms (Langdon and Poli, 1998).

Table 5.1 shows the average learning for the (4-10)-input parity task. The hypernetwork achieved learning up 100% in the cases of N=4 and N=6. In the case

of N=8, the hypernetwork obtained 100% learning in four of the ten runs, with an average of 96.80 % for all 10 runs. The 10-input parity task is exponentially more complex, but in spite of its complexity I achieve 100% learning in one organism, and a correct classification average of 93.13%. Learning the tasks for N=12 or greater is left for future experiments.

N	No. of training patterns	Average %	95% Conf.Level	Minimum %	Maximum %	n runs
4	16	100	0	100	100(10)	10
6	64	100	0	100	100(10)	10
8	256	96.80	2.66	90.63	100(4)	10
10	1024	93.13	8.85	81.25	100(1)	5

Table 5.1: Results of learning the (4-10)-input parity task. The number of organisms that obtain the maximum value of 100% correct classification is shown in parenthesis.

5.1.1 Learning the 4-input parity task

The organism used to learn the 4-input parity task is shown in Figure 5.1. It has two input cells, 6 internal cells, and one large output cell. The size of the cells is shown with the cell grid. All the ten organisms learn the task by 20,000 epochs (see Table 5.2 and Figure 5.2). Details of the experimental parameters are found in the Appendix.

5.1.2 Learning the 6-input parity task

The organism used to learn the 6-input parity task is shown in Figure 5.3. It has 3 input cells, 6 internal cells, and one large output cell. The size of the cells is shown with the cell grid. All organisms learn the tasks by 40,000 epochs (see Table 5.3 and Figure 5.4). Details of the experimental parameters are found in the Appendix.

Exp. No.	No. of epochs	Correct % classification
1	20,062	100
2	2,637	100
3	7,037	100
4	7,215	100
5	4,160	100
6	5,046	100
7	10,253	100
8	14,862	100
9	3,516	100
10	12,931	100
Mean	8,772	100
95% CL	4,060	0

Table 5.2: Learning the 4-input parity task. Details of ten runs. CL is the confidence level.

5.1.3 Learning the 8-input parity task

The organisms used for training the 8 and 10-input parity tasks are not more complex than those used for the (4-6)-input parity tasks. I added just one input cell to read the larger input vector (see Figure 5.5). I obtained 100% learning in 4 of the 10 runs. Results are shown in Table 5.4 and the learning curves are shown in Figure 5.6.

5.1.4 Learning the 10-input parity task

The organism trained to learn the 10-input parity task is shown in Figure 5.7. It has 5 input cells, internal cells, and one output cell. Learning the 1024 input vectors was achieved in only one of the five runs (see Table 5.5 and Figure 5.8).

Exp. No.	No. of epochs	Correct % classification
1	21,280	100
2	9,873	100
3	21,523	100
4	38,297	100
5	9,852	100
6	38,192	100
7	11,002	100
8	12,483	100
9	18,914	100
10	17,325	100
Mean	19,874	100
95% CL	7,618	0

Table 5.3: Learning the 6-input parity task. Details of ten runs. CL is the confidence level.

Exp. No.	No. of epochs	Correct % classification
1	53,599	100.00
2	150,000	90.63
3	106,120	100.00
4	150,000	96.88
5	150,000	98.44
6	150,000	96.88
7	116,447	100.00
8	74,435	100.00
9	97,500	92.19
10	150,000	94.54
Mean	119,810	96.80
95% CL	25,789	2.66

Table 5.4: Learning the 8-input parity task. Details of ten runs with up to 150,000 epochs each. CL is the confidence level.

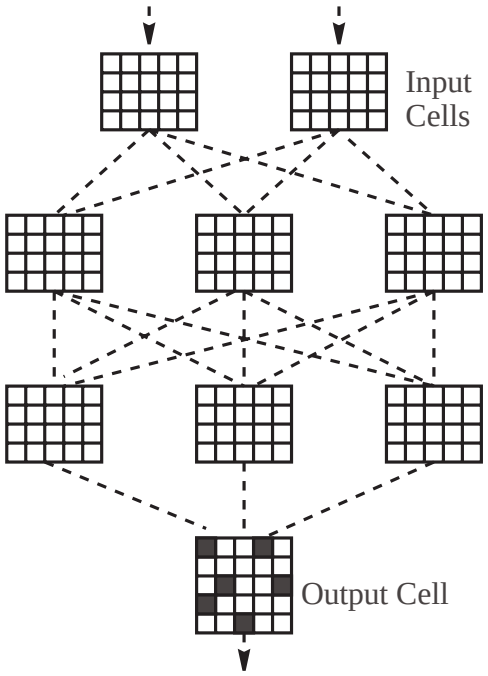


Figure 5.1: Structure of an organism used to train the 4-input parity task.

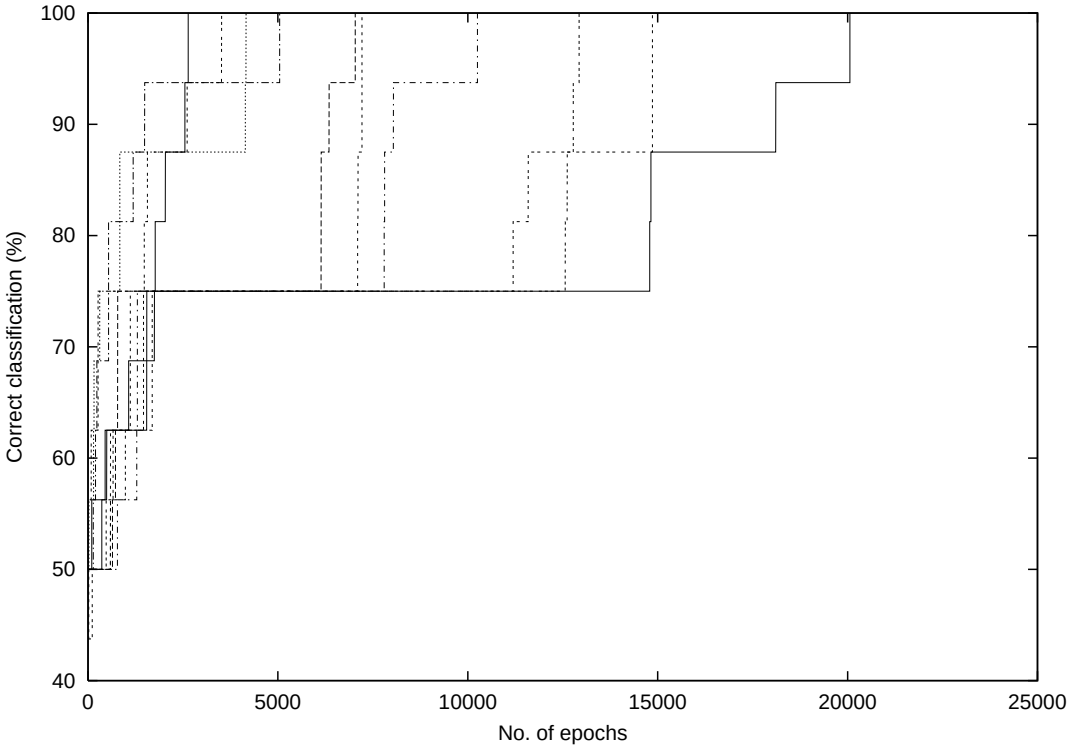


Figure 5.2: Learning curves for 10 organisms training the 4-input parity task. Correct classification (%) is on the vertical axis.

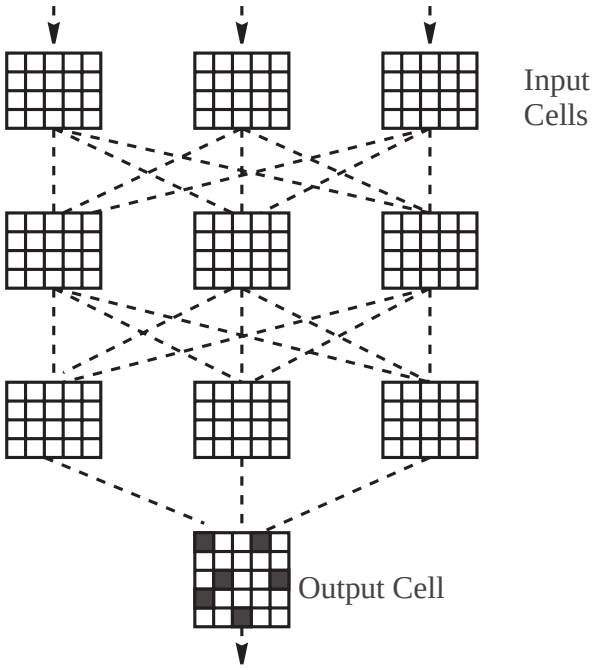


Figure 5.3: Structure of an organism used to train the 6-input parity task.

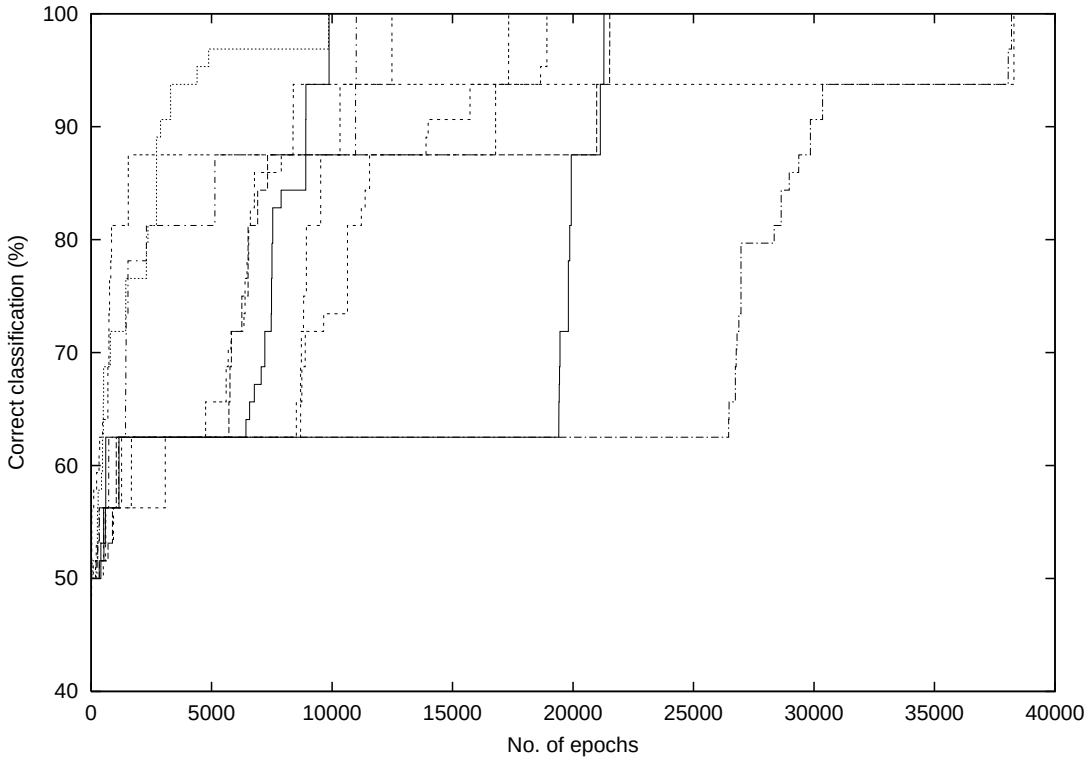


Figure 5.4: Learning curves for 10 organisms training the 6-input parity task. Correct classification (%) is on the vertical axis.

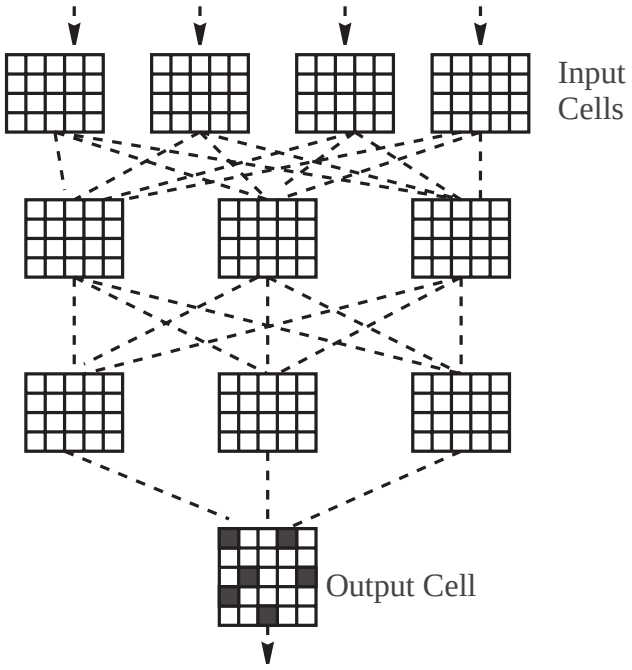


Figure 5.5: An organism used to train the 8-input parity task.

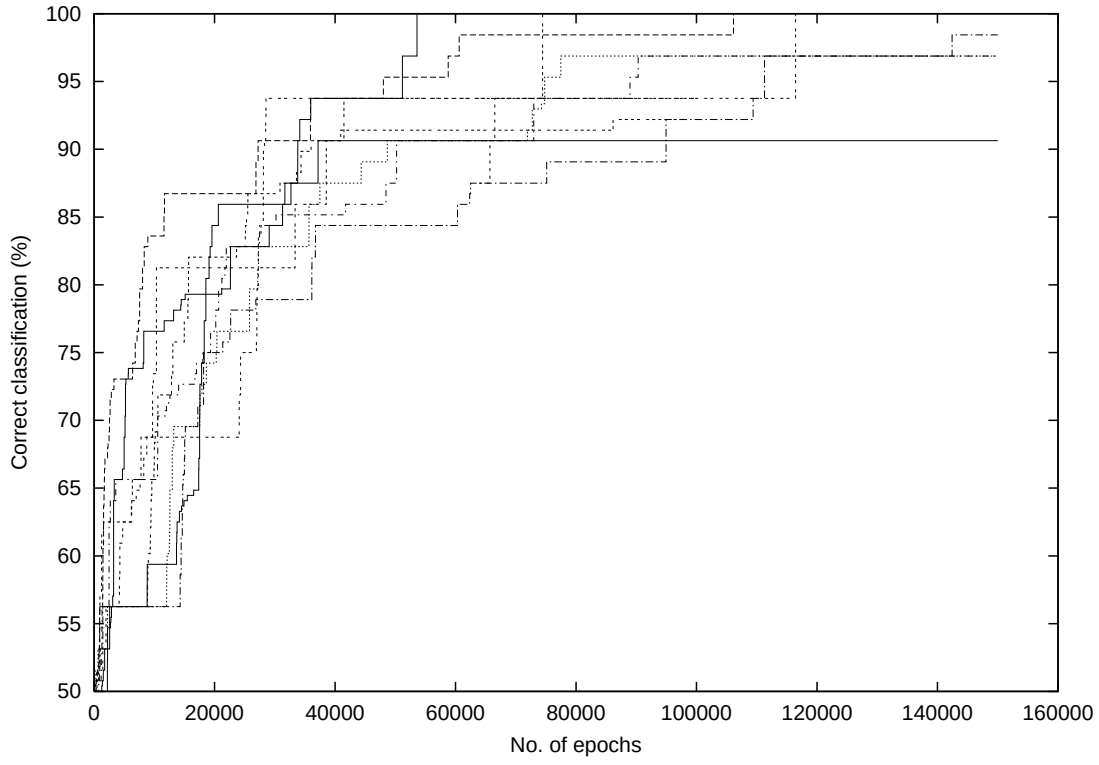


Figure 5.6: Learning curves for 10 organisms training the 8-input parity task. Correct classification (%) is on the vertical axis.

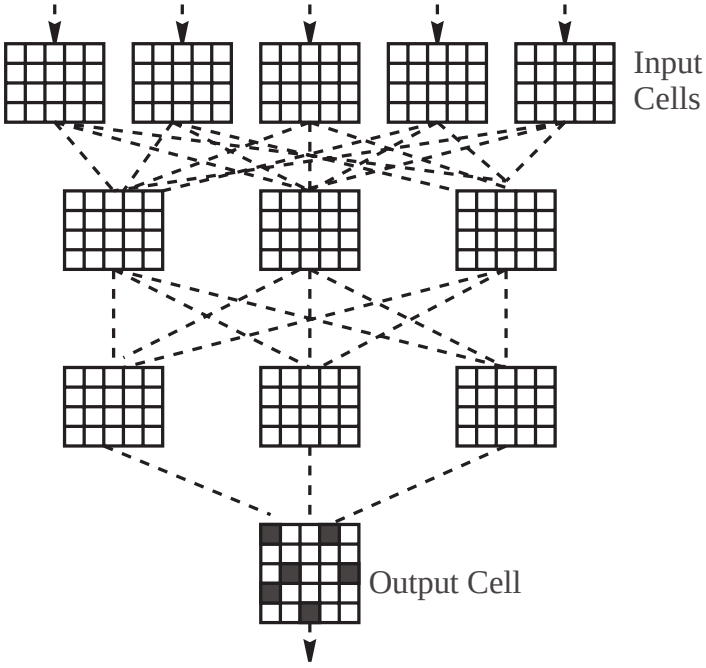


Figure 5.7: An organism used to train the 10-input parity task.

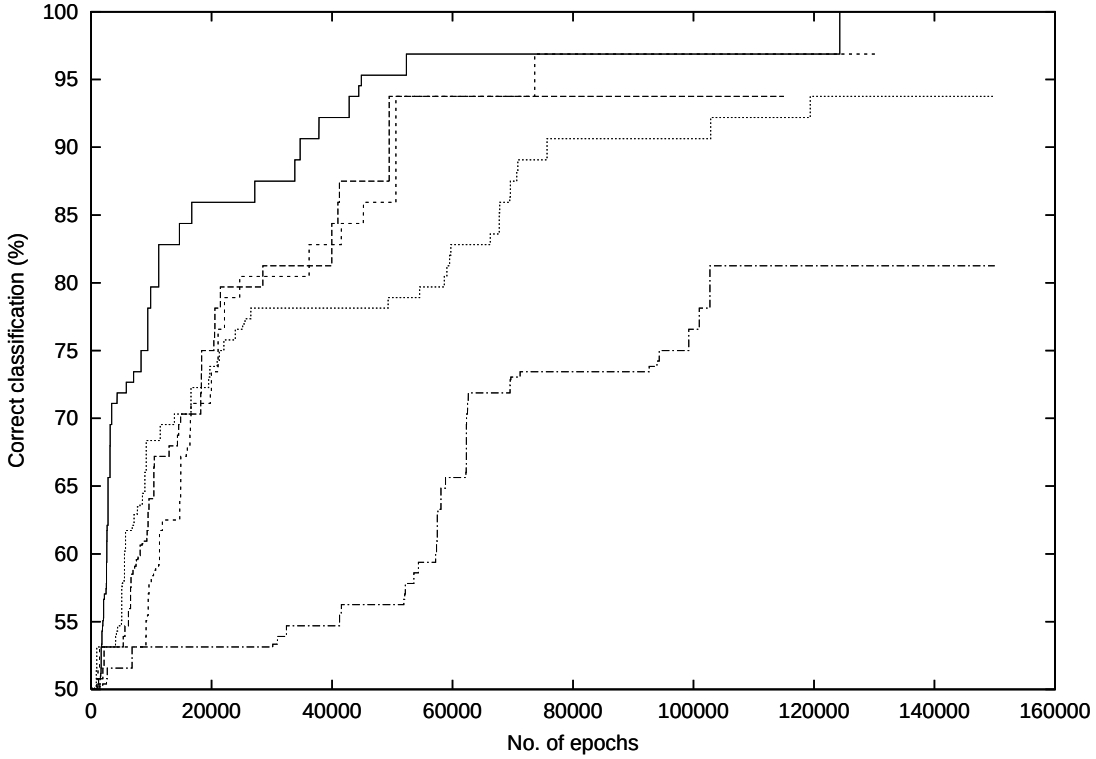


Figure 5.8: Learning curves for 5 organisms training the 10-input parity task. Correct classification (%) is in the vertical axis.

Experiment number	No. of epochs	Correct % classification
1	124,270	100.00
2	115,000	93.75
3	130,600	96.88
4	150,000	93.75
5	150,000	81.25
Mean	133,974	93.13
95% CL	19,428	8.85

Table 5.5: Learning the 10-input parity task. Results of of five runs, with up to 150,000 epochs each. CL is the confidence level.

5.2 Solving the tic-tac-toe endgame problem

The tic-tac-toe endgame problem contains 958 possible legal endgame boards (Aha, 1991; Matheus, 1990; Matheus and Rendell, 1989). About 65.3% of these instances are positive (i.e., winners for a player ”x”, assumed to have played first). The task is to learn to classify the endgame board configuration into winners or losers for player “x”.

The original data, obtained from “www.ics.uci.edu/pub/mlearn/databases/tic-tac-toe/” was given with three values for each cell of the board (player ”x”, player ”o”, and blank ”b”), and the output as positive or negative. I transform the value ”x” into ”01”, ”o” into ”10”, and ”b” into ”00”. In order to obtain the input vector I concatenate the board configuration into a vector of 18 binary digits. I construct the training set with a randomly chosen 70% (671 vectors) of the initial 958 vectors, using 30% (287 vectors) as the test set.

Aha (1991), with variations of an instance-based learning algorithm, obtained performance between 82% and 99%.

Results of seven experiments, run with the organism of Figure 5.9, are shown in Table 5.6. The best organism learned 94.99% of the training set, and had 89.61% of the answers correct in the test set. On average I obtained 90.55% of correct classification for the training set and 84.74% for the test set. The learning curves are shown in Figure 5.10.

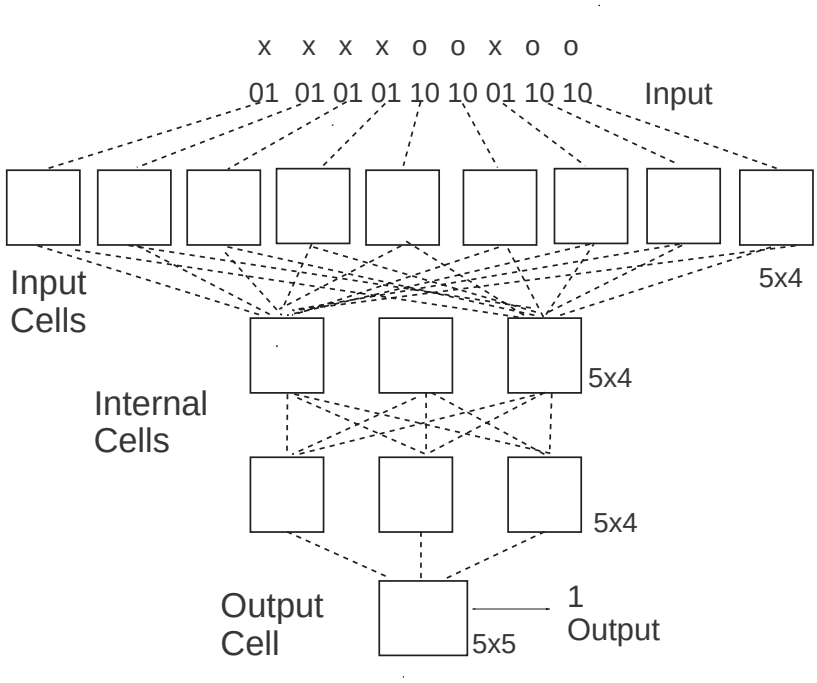


Figure 5.9: Organism used to learn the tic-tac-toe endgame problem. The organism has 9 input cells, 3 cells in each of two internal layers, and one output cell. The size of the cells is shown. Only part of the potential cell to cell interactions is shown.

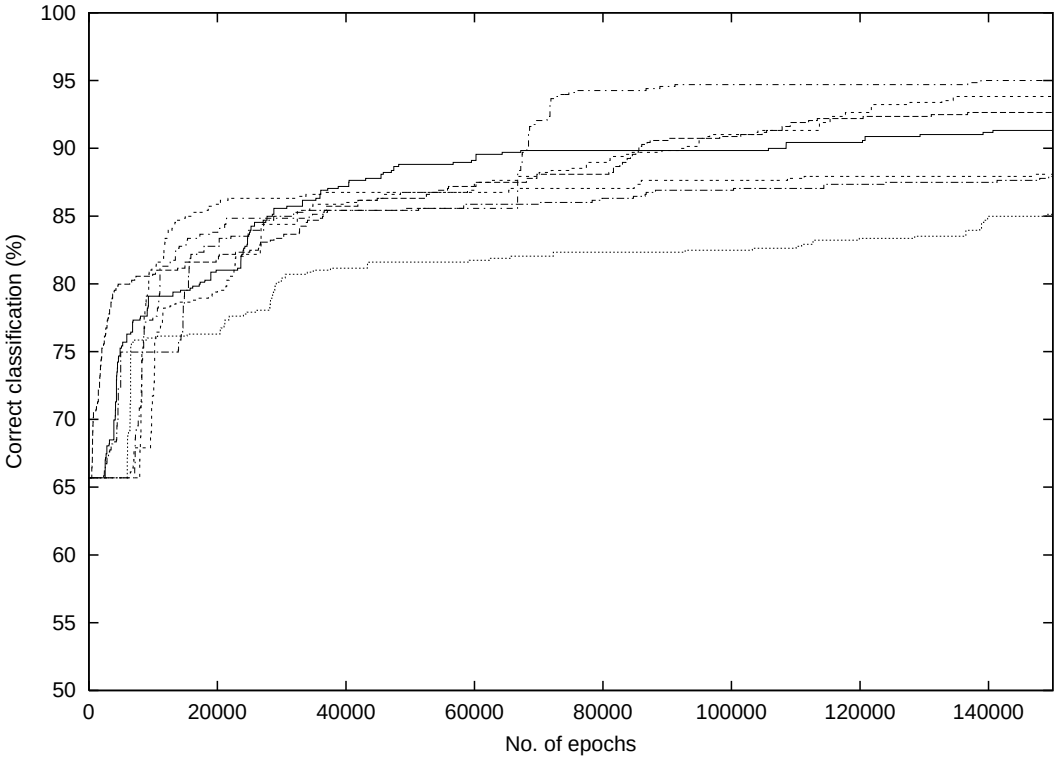


Figure 5.10: Learning curves for 7 organisms for the tic-tac-toe endboard problem.

Experiment number	Correct % training set	Correct % test set
1	91.32	87.45
2	92.64	84.94
3	93.82	85.30
4	85.13	82.08
5	87.93	83.15
6	94.99	89.61
7	88.07	80.65
Mean	90.55	84.74
95% CL	3.33	2.87

Table 5.6: Results for 7 organisms learning the tic-tac-toe endboard problem. The number of epochs is 150,000, and every run takes about one day on a Pentium III 550 MHz. CL is the confidence level.

5.3 Learning the two x two-bit multiplier table

The two x two bit multiplier is a circuit in which the logic operation is two by two-bit multiplication. The task for the hypernetwork is to learn its truth table, achieving the functionality of the multiplier.

The topology used to train is shown in Figure 5.11. Each organism has a total of 304 molecules distributed in 12 cells. I obtained 99.22% average learning for 10 organisms, with 100% learning in five of the ten cases (see Table 5.7). It takes less than 10 minutes to learn the task with a Pentium III 550 MHz (see Figure 5.12).

Experiment Number	No. of epochs	Correct % learning
1	200,000	98.44
2	69,776	100.00
3	200,000	98.44
4	82,805	100.00
5	86,569	100.00
6	200,000	98.44
7	200,000	98.44
8	160,589	100.00
9	62,223	100.00
10	200,000	98.44
Mean	146,196	99.22
95% CL	44,704	0.59

Table 5.7: Results for 10 organisms learning the two x two bit multiplier table. CL is the confidence level.

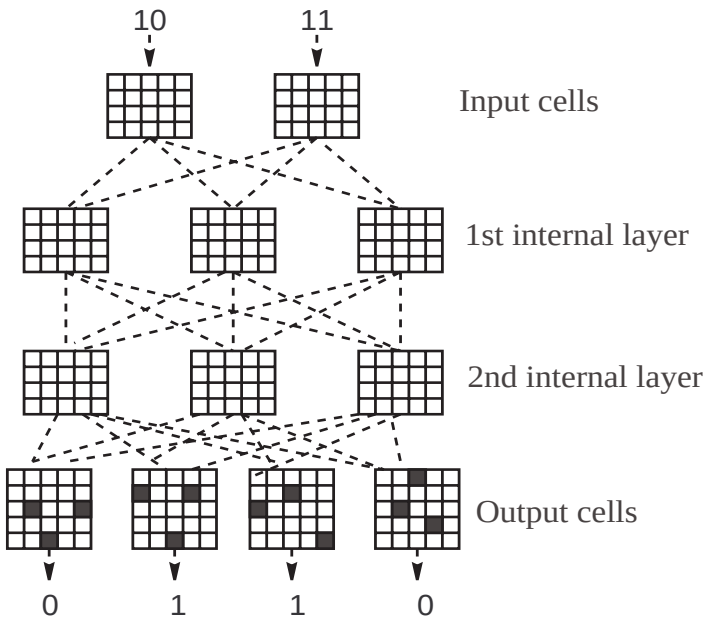


Figure 5.11: Organism used to learn the two x two bit multiplier truth table. The organism has two input cells, two layers of internal cells, and an output layer with 4 cells.

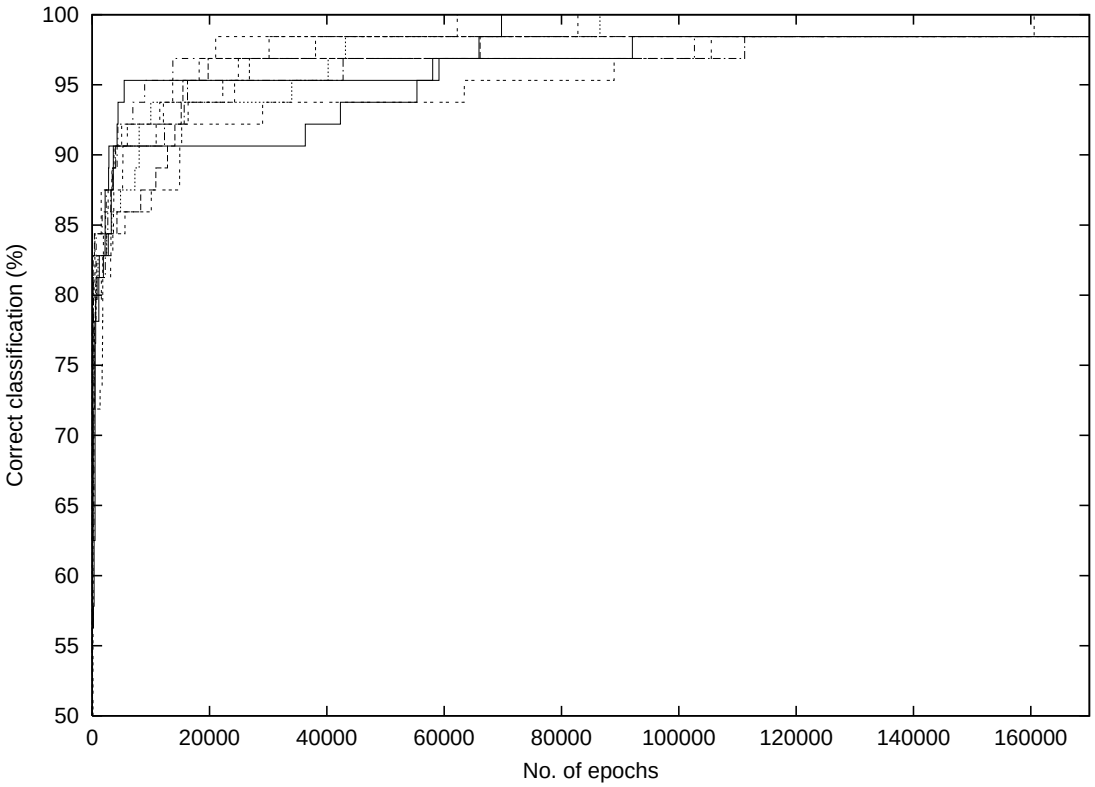


Figure 5.12: Learning curve for two x two bit multiplier truth table. I ran up to 200,000 epochs in each run, which takes about 10 minutes in a Pentium III 550 MHz. CL is the confidence level.

5.4 Learning the double spiral data set

5.4.1 Double spiral with 225 points

The organism must learn to discriminate the spiral of ”1s” from the spiral of ”0s” from the double spiral data set of 225 vectors shown in Figure 5.13. This data set is slightly different than another double spiral data set proposed by Lang and Witbrock (1989) that has 194 points in a x-y plane. This is a highly non-linear problem, solvable for example with back-propagation (Lang and Witbrock, 1989), fuzzy logic and adaptive resonance theory neural networks (Carpenter et al., 1992), genetic programming (Koza, 1992), among others.

The 8-bit input vectors were formed by concatenation of the x and y coordinates shown in Figure 5.13. The input vectors were trained with the organism in Figure 5.11. Results of eight runs show an average of 94.03% correct classification (an average of 211.57 out of 225 vectors were classified correctly), in 150,000 epochs (see Table 5.8). The learning curves are shown in Figure 5.15.

Experiment number	Correct % classification	No. of correct vectors
1	92.01	207
2	95.56	215
3	93.33	210
4	93.33	210
5	96.00	216
6	96.89	218
7	88.89	200
8	94.23	212
Mean	94.03	211
95% CL	2.47	4.79

Table 5.8: Results for learning the double spiral data set with 225 vectors. The number of epochs was 150,000. CL is the confidence level.

0001	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0010	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1
0011	1	0	1	1	1	1	1	1	1	1	1	1	1	0	1
0100	1	0	1	0	0	0	0	0	0	0	0	0	1	0	1
0101	1	0	1	0	1	1	1	1	1	1	1	0	1	0	1
0110	1	0	1	0	1	0	0	0	0	0	1	0	1	0	1
0111	1	0	1	0	1	0	1	1	1	0	1	0	1	0	1
1000	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1
1001	1	0	1	0	1	0	0	0	1	0	1	0	1	0	1
1010	1	0	1	0	1	1	1	1	1	0	1	0	1	0	1
1011	1	0	1	0	0	0	0	0	0	0	1	0	1	0	1
1100	1	0	1	1	1	1	1	1	1	1	1	0	1	0	1
1101	1	0	0	0	0	0	0	0	0	0	0	0	1	0	1
1110	1	1	1	1	1	1	1	1	1	1	1	1	1	0	1
1111	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0
	1	1	1	1	0	0	0	0	1	1	1	1	0	0	0
	1	1	0	0	1	1	0	0	1	1	0	0	1	1	0
	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1

Figure 5.13: The double spiral data set with 225 points. See text for details.

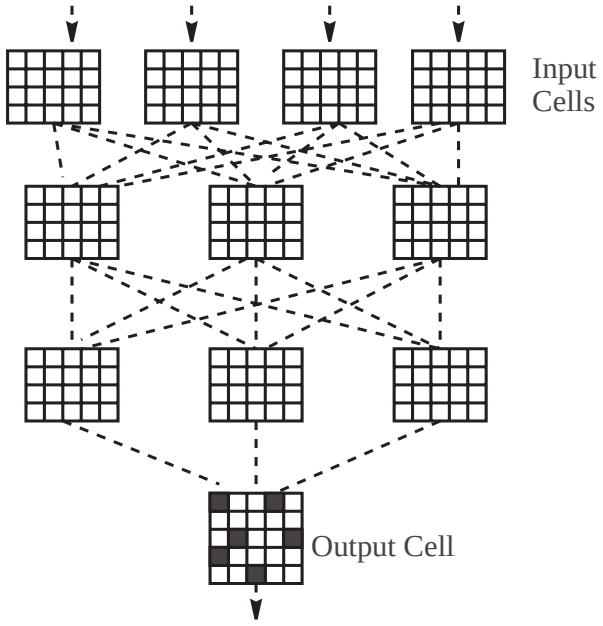


Figure 5.14: Organism used to learn the double spiral data set. The organism has 4 input cells, two layers of internal cells, and an output layer with 4 cells.

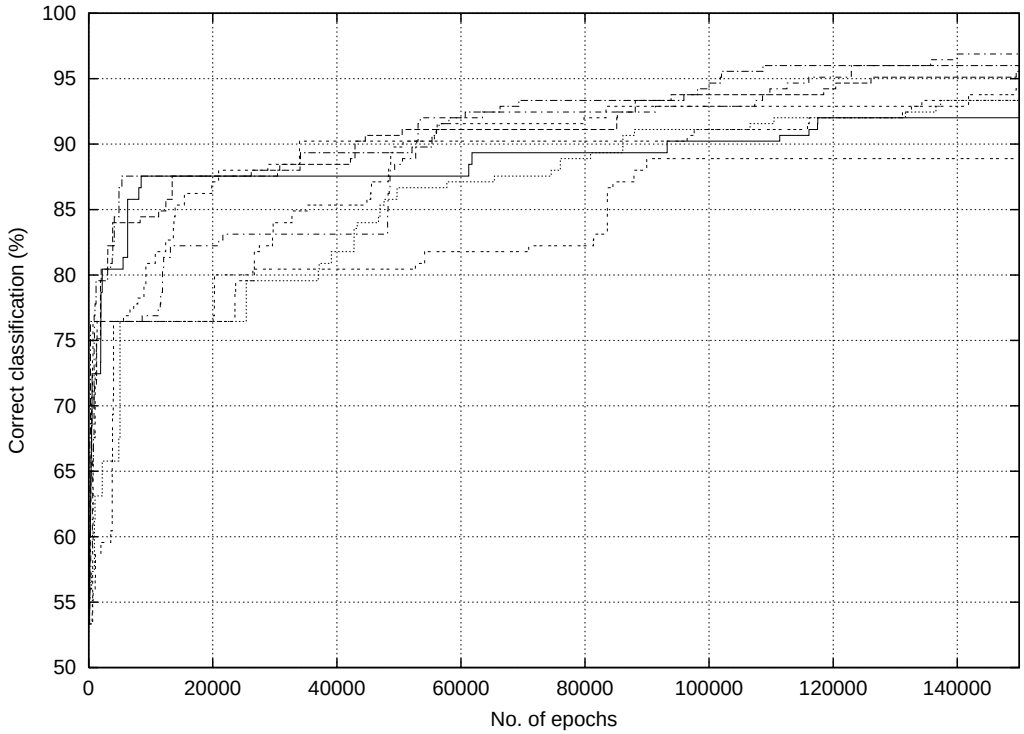


Figure 5.15: Learning the double-spiral data set with 225 points. Every experiment takes about seven hours on a Pentium III 550 MHz.

5.4.2 Double spiral with 56 points

From the 225-point data set I took the first 56 vectors, starting from the center of the spiral (see Figure 5.16), in order to test the performance of the hypernetwork with a smaller data set. I used the same organism shown in Figure 5.11. Results of the 10 runs are shown in Table 5.9. I obtained an average of 97.5% learning, with 100% in three cases. The learning curve of the organisms is in Figure 5.17.

Experiment number	No. of epochs	Correct % classification	No. of correct vectors
1	150,000	98.21	55
2	133,354	100.00	56
3	85,536	100.00	56
4	150,000	98.21	55
5	150,000	96.43	54
6	150,000	98.21	55
7	150,000	98.21	55
8	150,000	91.07	51
9	69,917	100.00	56
10	150,000	94.64	53
Mean	133,881	97.50	54.6
95% CL	21,655	2.01	1.13

Table 5.9: Results of learning the double spiral data set with 56 vectors. The number of epochs was up to 150,000. CL is the confidence level.

0101	1	1	1	1	1	1	1
0110	1	0	0	0	0	0	1
0111	1	0	1	1	1	0	1
1000	1	0	1	0	1	0	1
1001	1	0	0	0	1	0	1
1010	1	1	1	1	1	0	1
1011	0	0	0	0	0	0	1
1100	1	1	1	1	1	1	1

1	1	1	1	0	0	0
0	0	0	0	1	1	1
1	1	0	0	1	1	0
1	0	1	0	1	0	1

Figure 5.16: The double spiral data set with 56 points

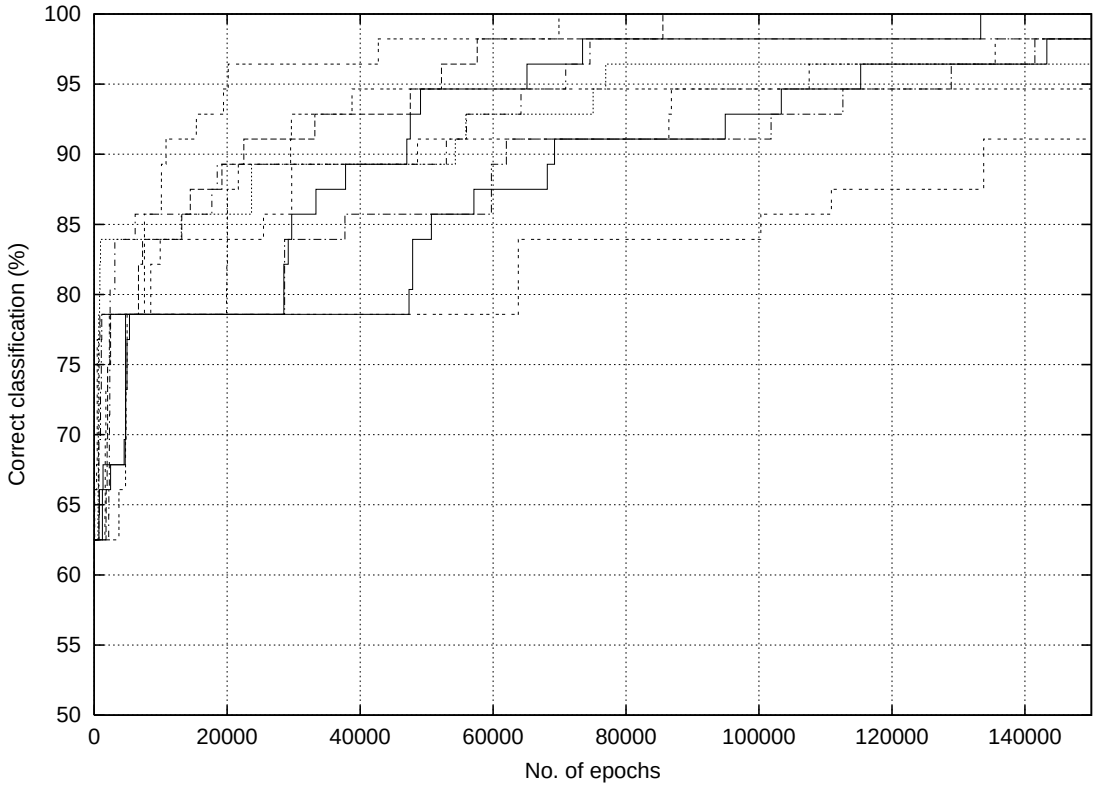


Figure 5.17: Learning the double-spiral data set with 56 points. Ten runs with up to 150,000 epochs.

5.5 Conclusions

This chapter explores the learning capabilities of the hypernetwork architecture. I obtained up 100% learning for the (4-10)-input parity tasks, up to 94.99% learning 671 points of the tic-tac-toe endboard problem, and 100% learning the two x two bit multiplier truth table. For the double spiral data set I obtained 96.89% learning in the case of 225 points, and 100% in the case of 56 points. Generalization properties of the hypernetwork architecture are shown only with the tic-tac-toe endboard problem. I obtained up to 89.61% on a test set of 287 vectors. More experiments are needed to show generalization in different problems.

The problems explored are of different nature, synthetic and from real problems. The organisms that solve the problems have a similar structure and parameter setting (see appendix).

These results show the learning capabilities of the hypernetwork architecture for solving these fairly complex tasks with not much different organisms and parameter settings.

CHAPTER 6

THE EFFECT OF INHIBITION AND FEEDBACK REGULATION ON LEARNING

6.1 Introduction

Feedback control mechanisms in biological systems (Ashby, 1956; von Bertalanffy, 1968; Wiener, 1948) operate from the molecular to the ecosystem levels (Aon and Cortassa, 1997; DeAngelis, 1995). There are regulatory feedback mechanisms of gene expression (Jacob and Monod, 1961). At the cellular level, the complex intra-cellular dynamics has many positive and negative feedback regulatory mechanisms (Alberts et al., 1994). At the tissue level, the endocrine system is an example of inter-cellular dynamics.

Regulation at the molecular level seems important for understanding memory (Bhalla and Iyengar, 1999; Squire and Kandel, 2000). An example of negative feedback inside neurons is in glutamate induced molecular cascades. Glutamate is a major excitatory neurotransmitter that plays an important role in memory and learning in the Central Nervous System (CNS) (McDonald and Johnston, 1990). Excessive amounts of glutamate can result in pathological damage in the CNS (Danysz et al., 1995). There are two types of glutamate receptors, the ionotropic and metabotropic receptors. The ionotropic receptors are for fast, and metabotropic for slower and lasting generation of molecular cascades (Francesconi and Duvoisin, 2000). There are many types of metabotropic glutamate receptors (mGluR1-8), and they have functions in regulating membrane excitability, synaptic transmission, and neurotransmitter release, among others. The mGluR1α can initiate both the $InsP_3/Ca^{2+}$ cascade where protein kinase C (PKC) is activated, and the cAMP cascade where cAMP-dependent protein kinase (PKA) is activated. Francesconi and Duvoisin (2000) found that there is *negative feedback regulation* of mGluR by

inhibition of mGluR1α by PKC, via the $InsP_3$ pathway. Such regulatory mechanisms may play an important role in learning.

Other examples of negative regulation are during axon growth, and the development of connectivity (Dodd and Schuchardt, 1995), molecular inhibitory mechanisms play an important role in synaptic plasticity (Abel et al., 1998), and removal of inhibitory constraints is required for memory formation in *Aplysia* (Abel et al., 1998).

Several approaches to the study of complex regulatory networks exist, for example Thieffry and Romero (1999) have a modulatory approach with boolean networks, and Sakamoto et al. (1998) analyze feedback inhibition with Michaelis-Menten-type reactions.

My approach to the study of feedback regulation is to model it on the hypernetwork architecture.

Experimental results on the hypernetwork architecture show that learning is improved when molecules have structures that allow them to became inactive. These inhibitory features regulate the formation of cascades of molecular interactions in the cell. This effect is observed inside neural cells.

Using this framework, this chapter stresses the role of inhibition and the formation of intra-cellular negative feedback regulation in hypernetwork learning.

6.2 Formation of regulatory networks in the hypernetwork architecture

The molecular structure and molecular relationships in a cell allow for the formation of negative and positive feedback regulation. An example of activation and inhibition of molecules is shown in Figure 6.1, where molecule M1 activates molecule M3 and inhibits M4 at time “t+1”. The molecule is in an active or inactive state for one time step, then

when it is back in a ready state it can be activated by another molecule.

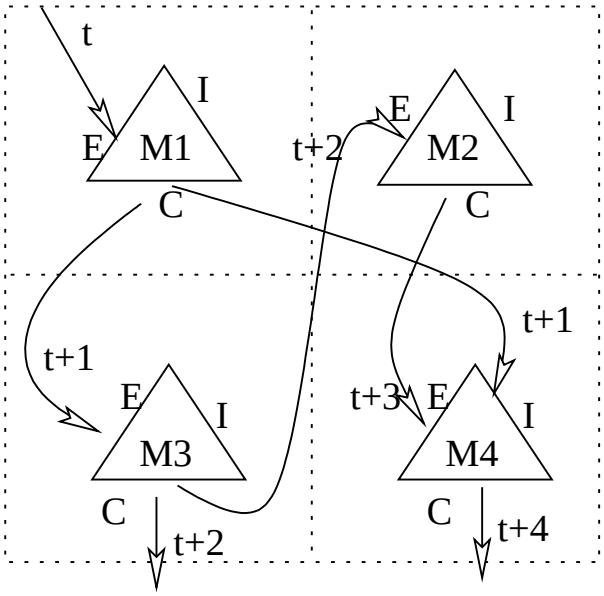


Figure 6.1: Molecule M1 inhibits M4 at time “t+1”, later after time “t+3” M4 is ready to be activated by M2. The excitatory (E), inhibitory (I), and catalytic (C) sites of each molecule are shown.

The formation of *positive feedback regulatory networks* in the hypernetwork architecture is possible when a target molecule could activate one of the molecules that cause its activation in the first place. For example in Figure 6.2 molecule M1 activates molecule M3 at time “t+1”, then molecule M3 activates molecule M4 at time “t+2”. Later molecule M4 will activate molecule M1 at time “t+3”, allowing the possibility of more activations of molecules M3 and M4 in future time steps.

Negative feedback regulatory networks are formed by means of the inhibitory site of the molecular structures. For example, in Figure 6.3 activated molecule M1 activates molecule M3 at time “t+1”, and molecule M3 activates M4 at time “t+2”. Then molecule M4 inhibits molecule M1 at time “t+3”. There are other possibilities of interactions depending on the structures and states of the molecules.

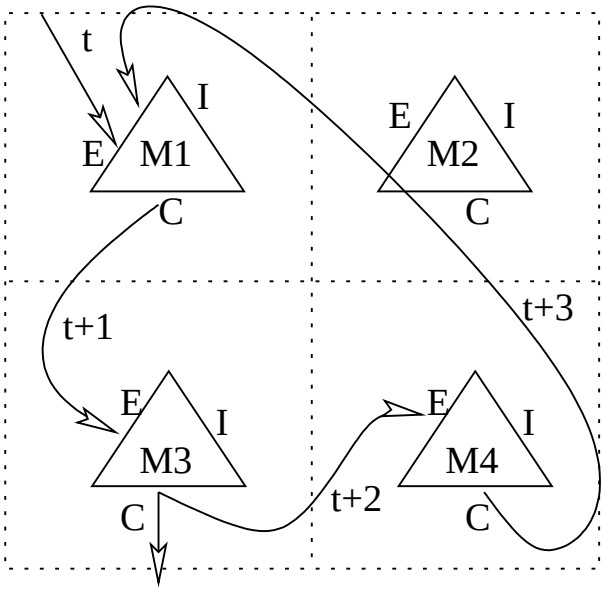


Figure 6.2: Example of a positive feedback regulatory network. Molecule M1 is activated at time “t”, then it activates M3 at time “t+1”, which in turn it activates M4 at “t+2”. Molecule M4 in turn can activate M1 at time “t+3”, producing more activations of molecules M1, M3 and M4. The excitatory (E), inhibitory (I), and catalytic (C) sites of each molecule are shown.

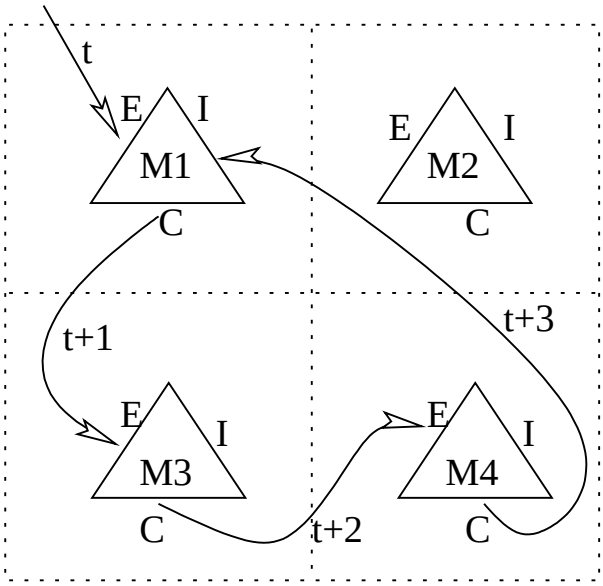


Figure 6.3: Example of a negative feedback regulatory network. Molecule M1 is activated at time “t” and one of its cascade products, molecule M4, will inhibit the activation of M1 at time “t+3”. The excitatory (E), inhibitory (I), and catalytic (C) sites of each molecule are shown.

6.3 Effect of inhibition on learning

The experiments are divided in two parts. First, I observe the effect of the inhibition threshold and the percentage of inhibitory molecules on learning the 4-input parity task. Secondly, I study the effect of inhibition on learning the N input parity tasks, with N=4 to N=8.

6.3.1 Effect of threshold on inhibition and percentage of inhibitors on learning the 4-input parity task.

Hypernetwork organisms were trained to learn the 4-input parity tasks, varying the threshold for molecular inhibition from 10% to 90% of the string matching. Each cell had 20% of its molecules with inhibitory sites. Figure 6.4 shows the average number of epochs required for learning the task. Learning took a greater number of epochs when the threshold was low (10%), or high (90%). This corresponds to over-inhibition in the first case, and over-excitation of the molecular cascades in the second case. Figure 6.4 shows that an inhibition threshold between 30% and 70% gave almost similar results in the number of epochs required for learning. For the following experiments I set the value of inhibition threshold to 70%.

The relation of the number of molecules with inhibitory ability in each cell and learning is shown in Figure 6.5. I observe a decreasing curve in the number of epochs needed to learn the 4-input parity task, as a function of the percentage of molecules with inhibitory sites.

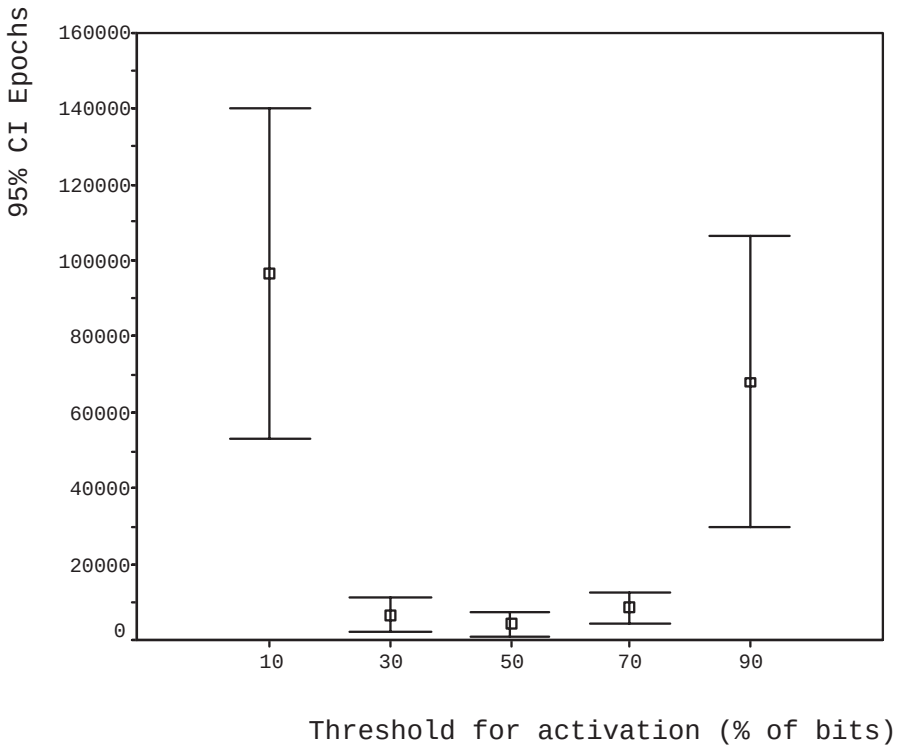


Figure 6.4: Average number of epochs for learning the 4-input parity task as a function of the threshold to inhibit a molecule. There is 20% of molecules with inhibitory sites in each cell of the organism. Average of 10 runs, up to 150,000 epochs. Molecular activation threshold is 60%. Molecule string length is 14 bits/site. CI is confidence interval.

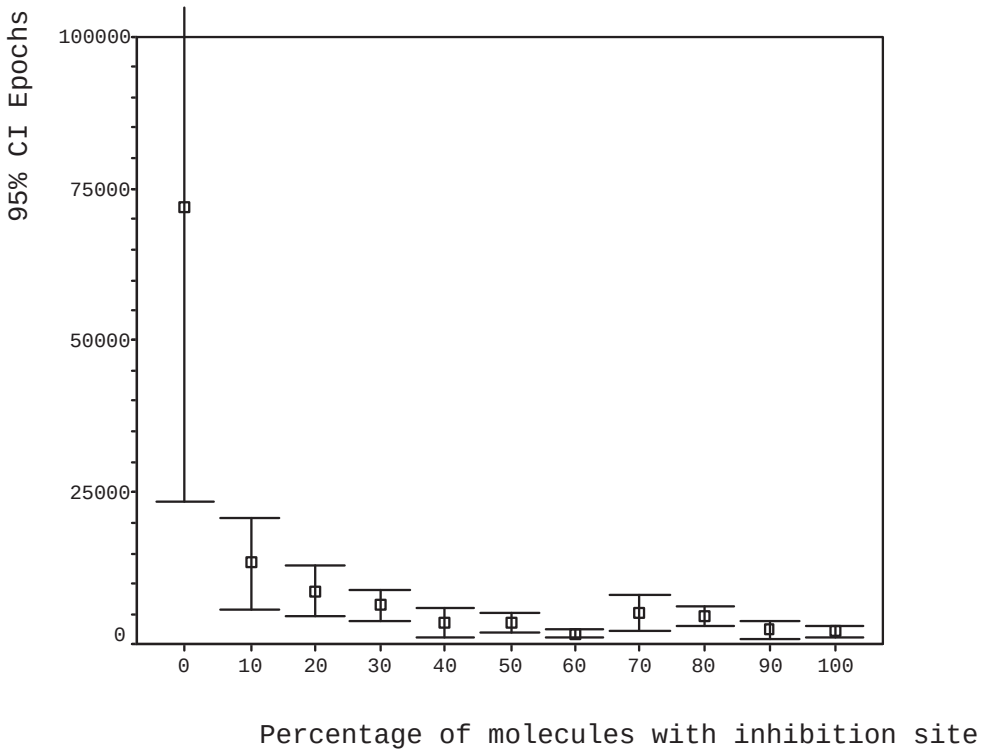


Figure 6.5: 4-Input parity learning average as a function of the percentage of molecules in the cell that exhibit inhibitory sites. Molecular inhibition threshold is set to 70%, and activation threshold to 60%. Average of 10 runs, up to 150,000 epochs. Molecule string length is 14 bits/site. CI is the confidence interval.

6.3.2 Effect of inhibition on learning the 4-8 input parity tasks

I ran experiments for learning the N-input parity task, with and without inhibition. Values of N where N=4, N=6, and N=8.

Figure 6.6 shows the average number of epochs required for learning the 4-8 input parity tasks. The number of epochs needed to solve the problem is larger when the system has no inhibition. I ran up to 150,000 epochs, and in the case of N=8, Figure 6.6 shows this bias.

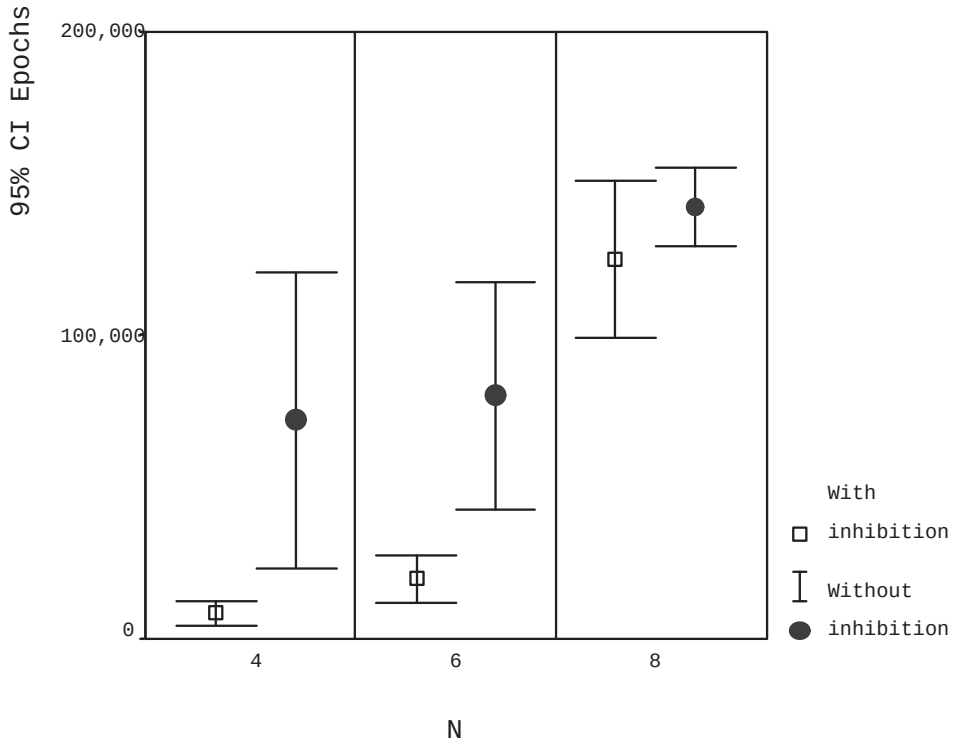


Figure 6.6: Average of ten runs in the number of epochs for the N-input parity task N=4, N=6, and N=8, with and without inhibition. Runs where terminated when learning was achieved or 150,000 epochs were reached. Threshold for molecular inhibition = 70%, threshold for molecular activation = 60%, percentage of molecules with inhibitory sites = 20%.

Figure 6.7 shows the average learning for the same tasks. The figure shows that organisms that have molecular inhibition show faster learning than the ones that did not

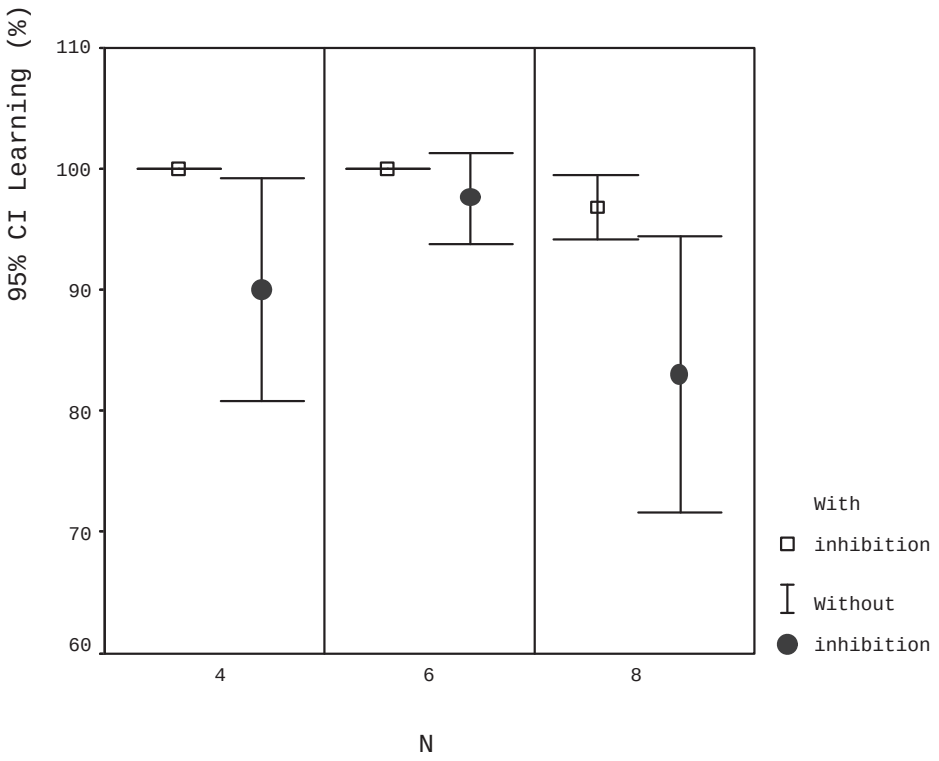


Figure 6.7: Learning average of ten runs for the N-Input parity task, N=4, N=6, and N=8 with and without inhibition. Runs where terminated when learning was achieved or 150,000 epochs were reached. Threshold for molecular inhibition = 70%, threshold for molecular activation = 60%, percentage of molecules with inhibitory sites = 20%.

have molecular inhibition. I obtain 100% learning with inhibition in N=4 and N=6, but without inhibition organisms have more difficulty learning the parity tasks.

The learning curve for the 4-input parity task with no inhibition is shown in Figure 6.8. Of the ten runs, four did not converge (see Table 6.1). In the case of learning the 6-input parity task two cases did not converge (see Figure 6.9 and Table 6.2). I got 100% learning in every run of both tasks when the organisms had molecular inhibition (see Figures 5.2 and 5.4).

Learning the 8-input parity task with no molecular inhibition is shown in Figure 6.10. Of the ten runs, only two reached 100 % learning, as shown in Table 6.3.

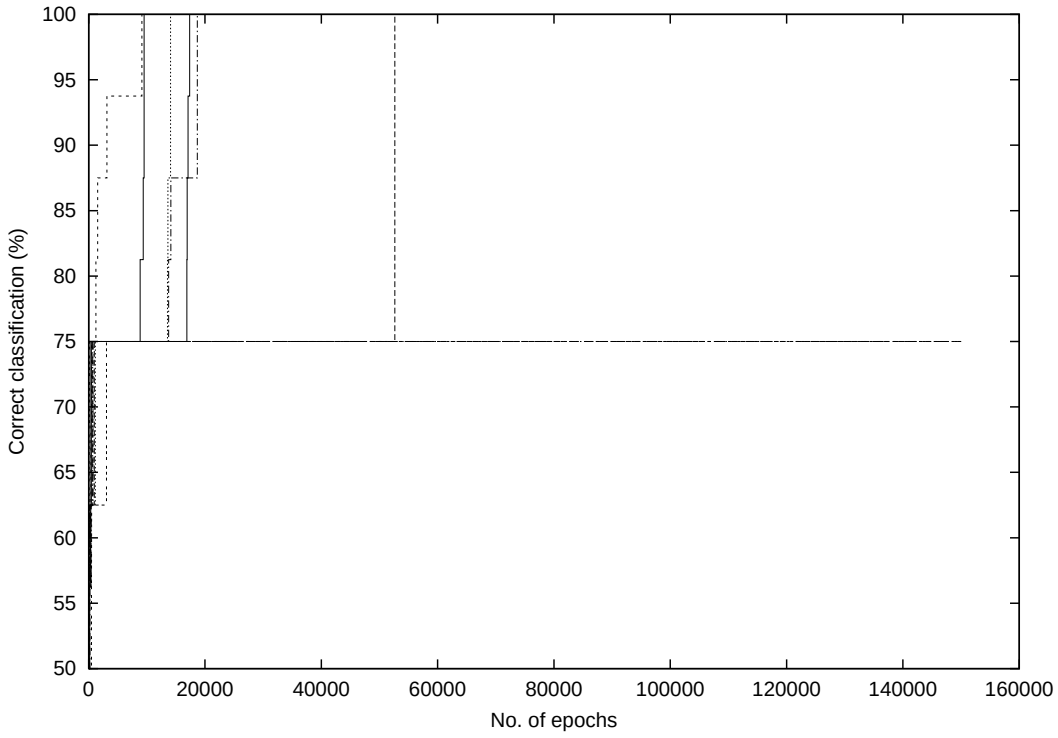


Figure 6.8: Learning curves for the 4-input parity task with no inhibition. Of the ten runs, four did not improve over 75 %.

Exp. No.	No. of epochs	Correct % classification
1	9,534	100
2	52,627	100
3	9,141	100
4	14,086	100
5	18,684	100
6	150,000	75
7	150,000	75
8	150,000	75
9	150,000	75
10	17,357	100
Mean	72,143	90
95% CL	48,720	9.23

Table 6.1: Results of learning the 4-input parity task with no molecular inhibition. CL is the confidence level.

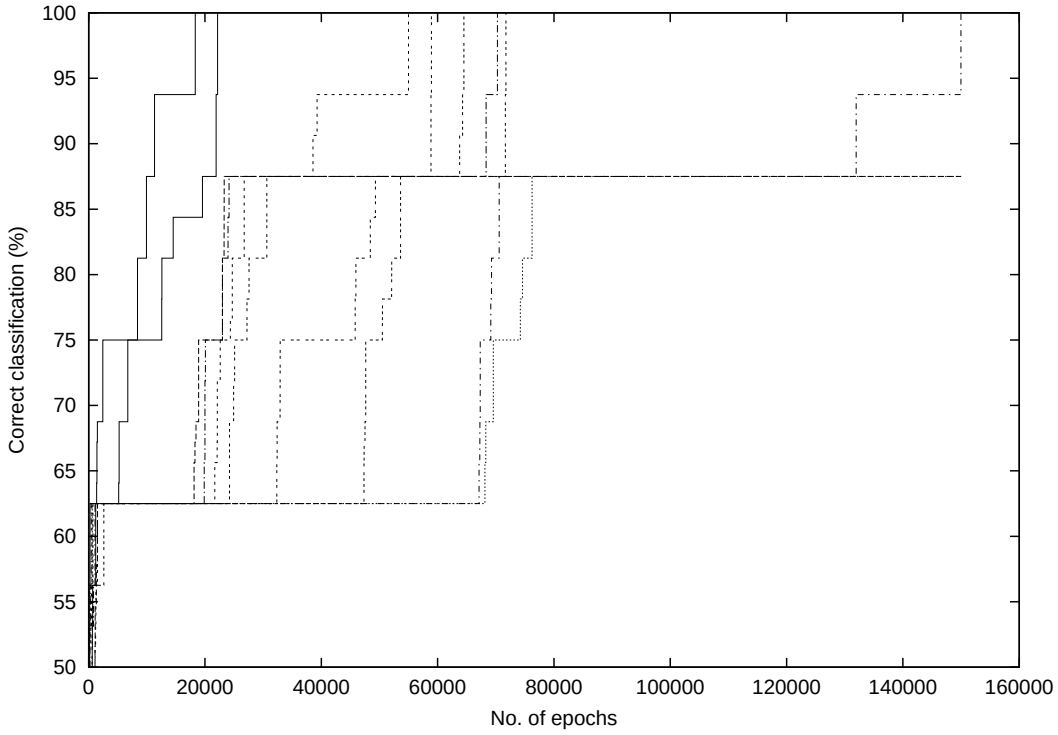


Figure 6.9: Learning curves for 6-input parity task without inhibition. Of ten runs, two did not improve over 87.5%.

Exp. No.	No. of epochs	Correct % classification
1	18,309	100
2	22,168	100
3	150,000	87.5
4	54,978	100
5	150,000	87.5
6	70,279	100
7	149,957	100
8	58,919	100
9	71,739	100
10	64,499	100
Mean	81,085	97.5
95% CL	36,402	3.77

Table 6.2: Results of learning the 6-input parity task with no molecular inhibition. CL is the confidence level.

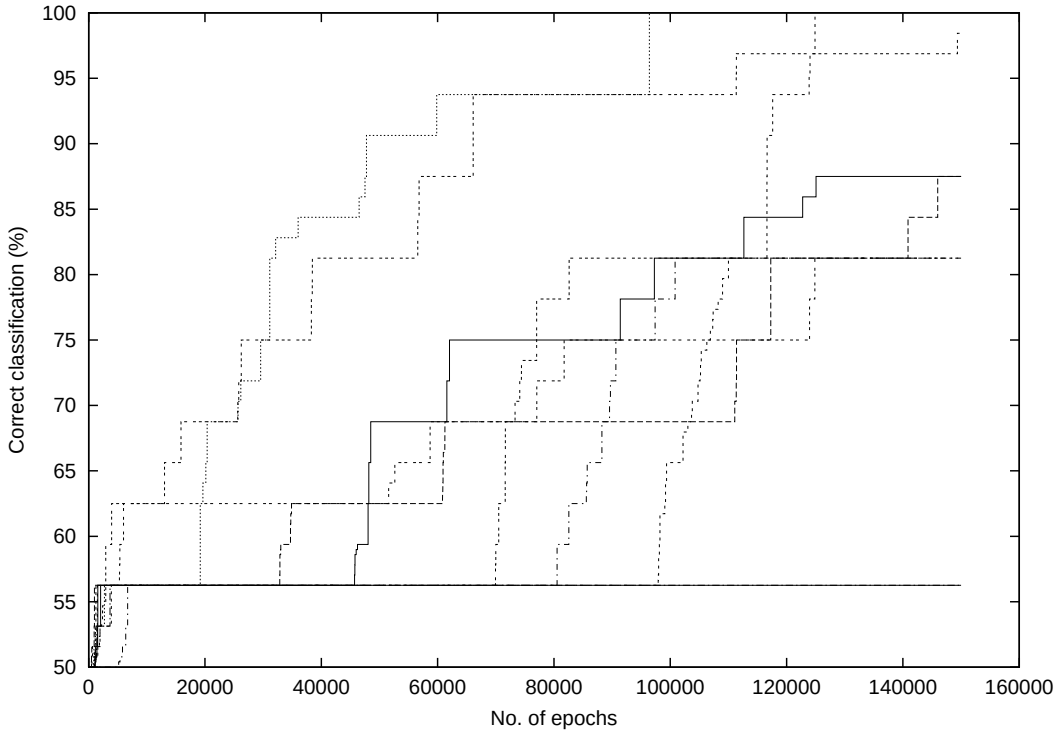


Figure 6.10: Learning curves for the 8-input parity task without inhibition.

Exp. No.	No. of epochs	Correct % classification
1	150,000	87.50
2	150,000	56.25
3	150,000	87.50
4	124,867	100.00
5	96,407	100.00
6	150,000	56.25
7	150,000	81.25
8	150,000	81.25
9	150,000	98.44
10	150,000	81.25
Mean	142,127	82.87
95% CL	12,805	11.43

Table 6.3: Results of learning the 8-input parity task with no molecular inhibition. CL is the confidence level.

6.4 Conclusions

In general, negative feedback mechanisms stabilize a system, controlling the positive feedback systems of over-reactions. Biological systems exhibit feedback regulation at several hierarchical levels. Feedback regulatory networks formed in the cells of hypernetwork organisms are important for information processing. Those inhibitory features are important for keeping the homeostatic balance in the organism, thus improving learning ability.

Experimental results show that, in the hypernetwork architecture, learning improves when the molecules have functional inhibition sites allowing inhibition and the formation of molecular negative feedback regulation inside cells. Moreover, the number of molecules with inhibitory capacity could be large in proportion to the total number of molecules. This suggest a likelihood that more molecular inhibitory mechanisms will be discovered in biological systems, and that the importance of these will be even greater in the context of learning and development.

Currently the hypernetwork architecture has evolution at the molecular level only. Future implementations will have evolution and regulation at the cellular level, where cells and their interactions could be evolved. Experiments on these models could give more information about the evolution of regulatory mechanisms involving cells and tissues.

CHAPTER 7

MUTATION BUFFERING CAPABILITIES OF THE HYPERNETWORK MODEL

7.1 Introduction

The bootstrap principle of evolutionary adaptability proposed by Conrad (1979a) states that ”the degree of gradualism with which protein function changes ... is both a condition for and a consequence of evolution by variation and natural selection” (Conrad, 1983).

This principle is based on the mutation buffering concept, where at different hierarchical levels, structural redundancies serve to buffer the effect of changes in the components, facilitating evolution (Conrad, 1983, 1985). For instance, at molecular level two versions of a protein may have the same function, but one of them may give the organism more evolutionary advantages. There are several examples of such variants of proteins such as allozymes and isoenzymes.

The system should allow some internal changes due to mutation, deterioration, or failure. These capabilities are of great importance for the survival and evolution of such systems. Other complex systems show error tolerance (Albert et al., 2000).

In this chapter I show the mutation buffering capabilities of the hypernetwork model (Segovia-Juarez and Colombano, 2001), and I discuss its importance in searching the adaptive landscape (Wright, 1932) (see Figure 7.1). The fitness landscape in this case is a multi-dimensional space of molecular structure distributions.

Molecular buffering percolates up to the organismic level and allows organisms to behave differently in previously unknown environments or to produce similar behavior, to the same input, even when they do not have identical low level molecular structures.

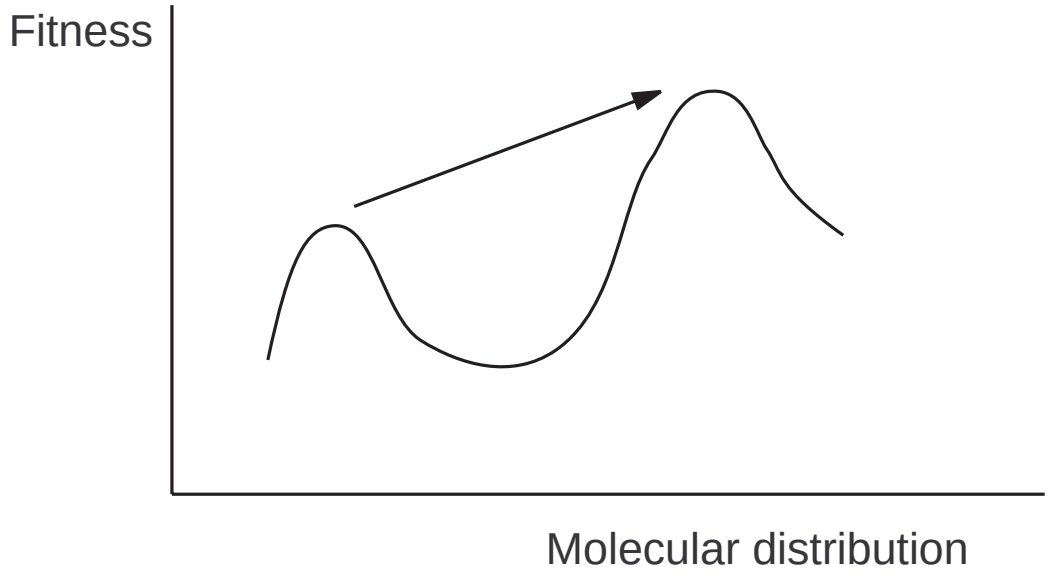


Figure 7.1: Fitness landscape.

These capabilities are a driving force in evolution. They allow biological organisms to search for other peaks in the fitness landscape without losing functionality.

Using the organisms that learned the two x two-bit multiplier and (6-10) input parity tasks, some experiments were conducted to test the ability to maintain functionality in the presence of random molecular mutation or denaturation.

7.2 Testing the mutation buffering capabilities of the organisms

I investigate the robustness of the hypernetwork to structural changes. The changes could be at the molecular, cellular, or organismic levels. In this part I concentrate on changes at the level of molecular structures.

The organisms that achieved 100% learning in the previous tasks underwent molecular mutations during reproduction. There are two types of substitution: random mutation, and molecular denaturation.

- Random substitution: a molecule is substituted for another molecule with random structure.

- Molecular ”denaturation”: all the “1” bits of the molecule are replaced by ”0”. This is a representation of a molecule that has less complexity than others.

From each trained organism I generated a sub-population of twenty organisms with mutations in one molecule, another sub-population of twenty individuals with mutations in two molecules, and so on. After the sub-populations were obtained, I tested their performance against the training set to find out how the mutations affected the performance of each subpopulation. This procedure was done for every experiment and for each type of mutation. Receptor molecules from input cells were not considered for mutation in order to assure that all the input vectors were read, and that the mutation affected only the dynamics of the networks of interactions.

Figures 7.2 to 7.9 show the performance of the different sub-populations. The boxes represent the number of mutants that have a given performance or percentage of correct classifications, for each sub-population. Below I explain the details for each experiment.

7.2.1 Buffering capabilities of the two x two-bit multiplier organisms

From each of five organisms I generated another twenty, resulting in sub-populations of 100 mutants. Figure 7.2 shows that in the sub-population where three molecules were mutated randomly, 40 mutants show 100% correct classification. Also, in the sub-population where two molecules were mutated, close to 70 mutants were able to obtain 100% correct classification. The results for the sub-population with six mutated molecules are that about 10% of the mutants performed as well as their parents (100

% correct learning). In contrast, in the sub-population with six randomly denatured molecules I observed that 30% have the same performance as their parents (see Figure 7.3).

The total number of molecules in each organism is 304, and six molecules represent 1.97% of them.

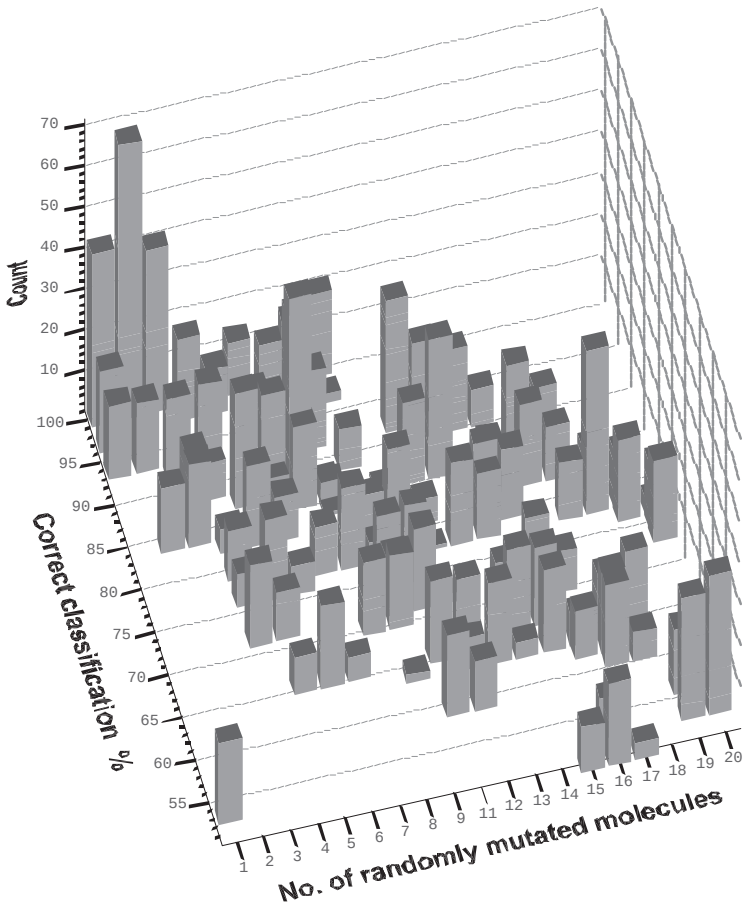


Figure 7.2: Performance and mutant count of sub-populations generated by random mutation from five two x two-bit multiplier organisms. There are 100 mutants in each sub-population.

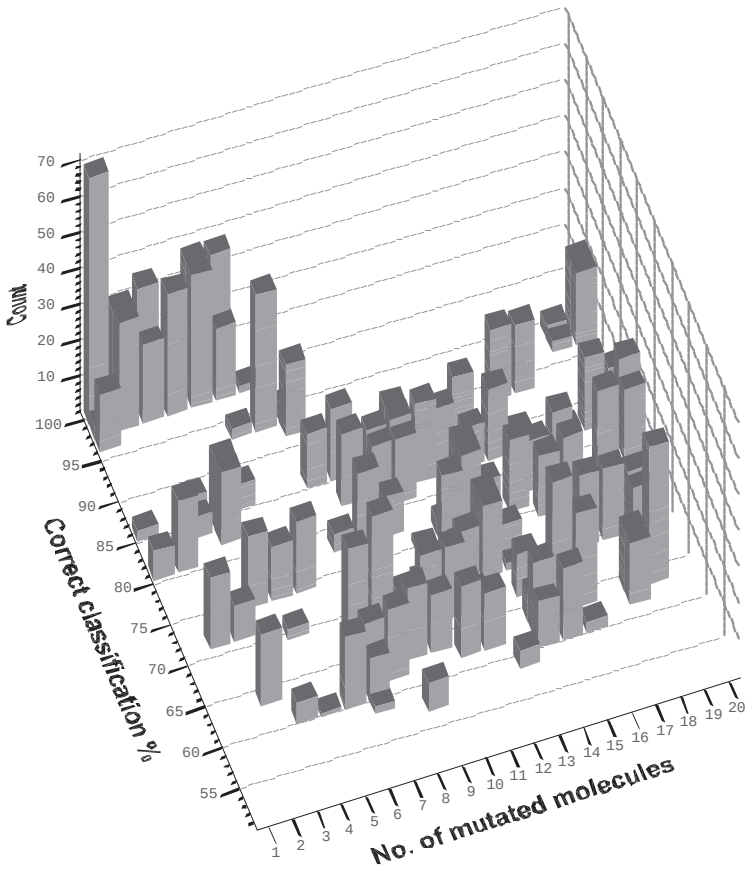


Figure 7.3: Performance and mutant count of sub-populations generated by molecular denaturation from five two x two-bit multiplier organisms. There are 100 mutants in each sub-population.

7.2.2 Buffering capabilities of the 6-input parity and 8-input parity organisms

Figures 7.4 and 7.5 show the performance of sub-populations obtained from 6-input parity organisms. In the figures I note that, as expected, the performance of the organisms in the sub-populations decreases as the number of mutated molecules increases, but there are subpopulations from 4 to 6 mutated or denaturized molecules where about 10% of the individuals obtain 100% learning. The same behavior is observed in sub-populations from 8-input parity organisms (see Figures 7.6 and 7.7.)

7.2.3 Buffering capabilities of the 10-bit parity test organism

I tested the buffer capabilities in the organism that was able to learn the complete 10-input parity table. In Figure 7.8 is observed that after a one-molecule mutation, 50% of individuals have the same performance as their ancestor. However, if the mutation were five molecules or more, the organisms were very damaged, not showing more than 70% learning.

On the other hand, in the case of molecular denaturation there was a more gradual decay of the performance, but a relative high percent of correct answers persists for up to four denatured molecules (0.78%) behaving like their parents.

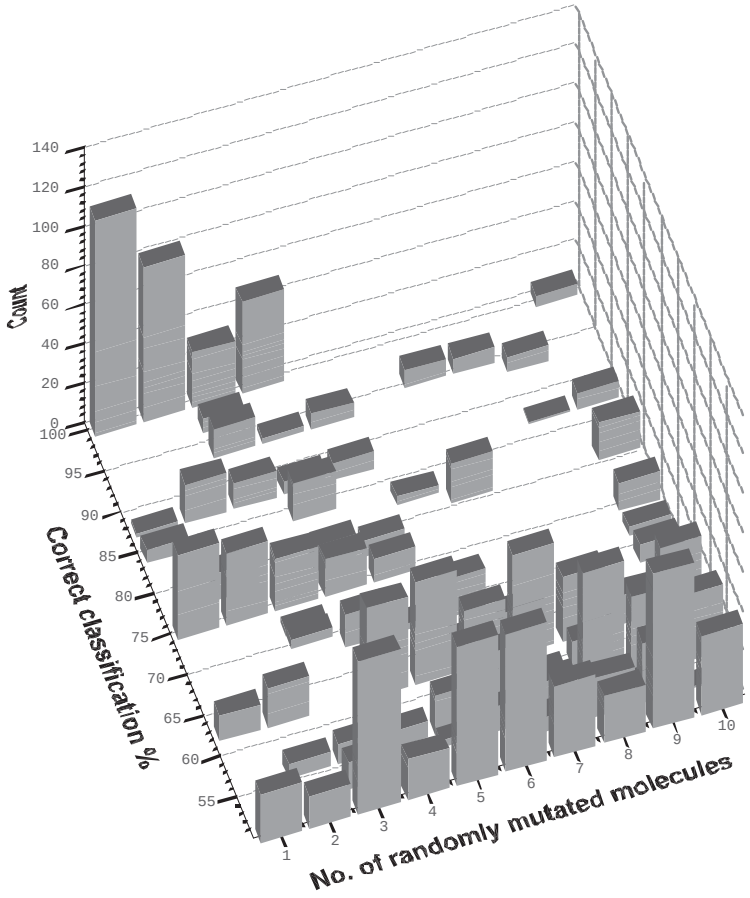


Figure 7.4: Performance and mutant count of sub-populations generated by random mutation from ten 6-input parity organisms. There are 200 mutants in each sub-population.

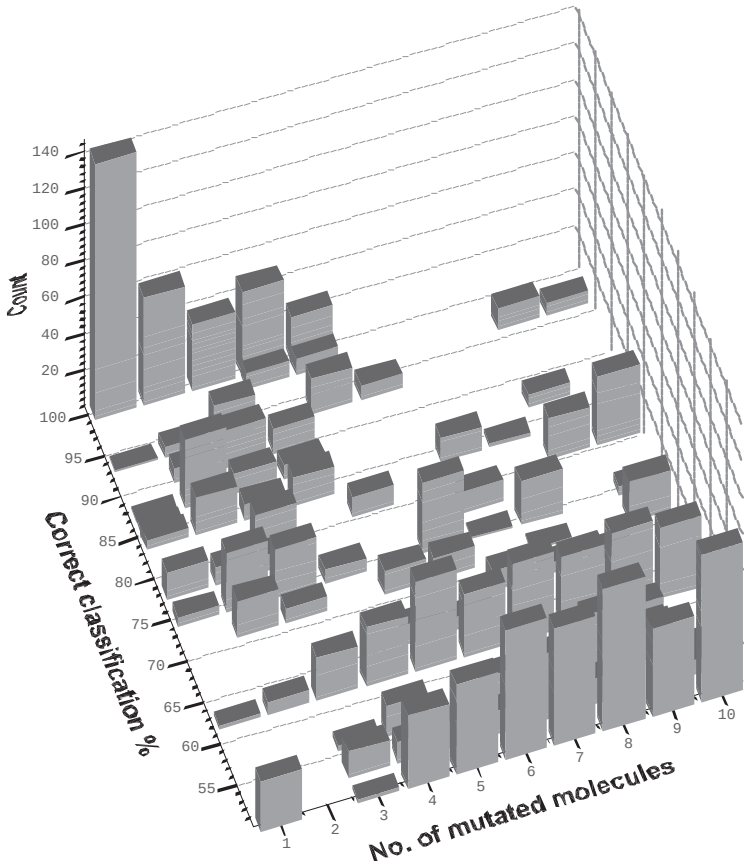


Figure 7.5: Performance and mutant count of sub-populations generated by molecular denaturation from ten 6-input parity organisms. There are 200 mutants in each sub-population.

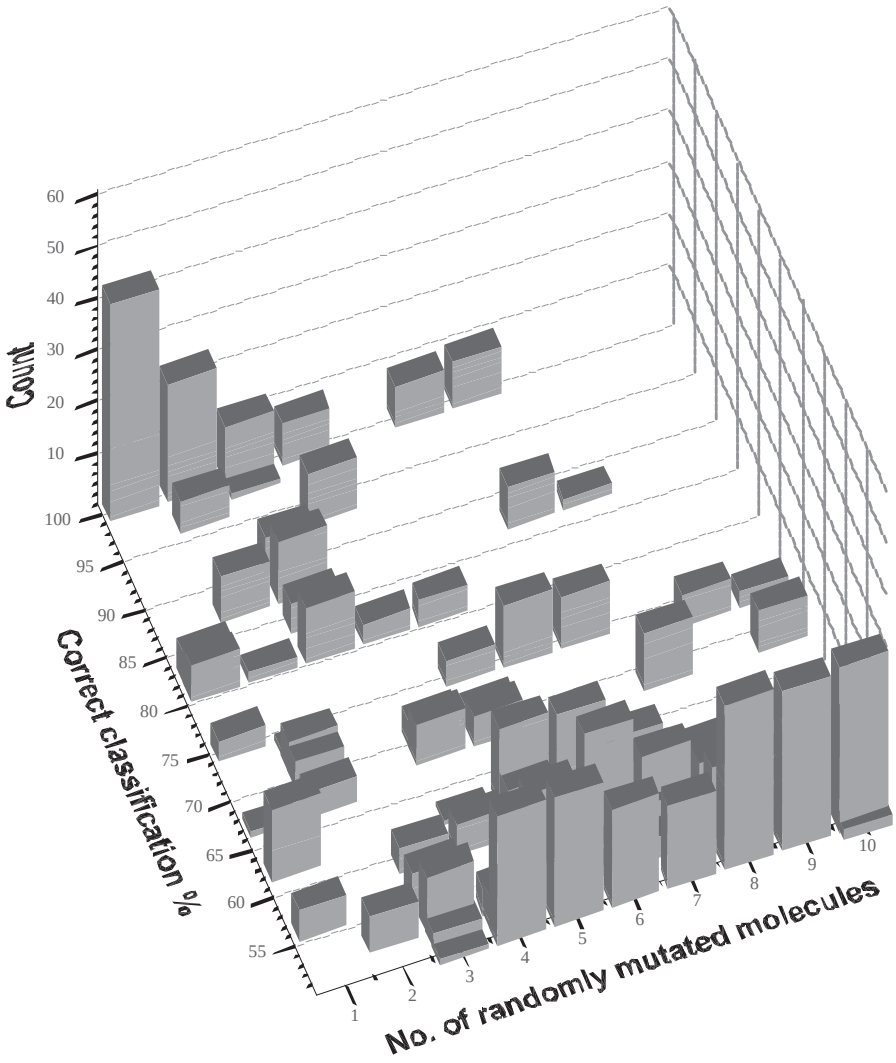


Figure 7.6: Performance and mutant count of sub-populations generated by random mutation from five 8-input parity organisms. There are 100 mutants in each sub-population.

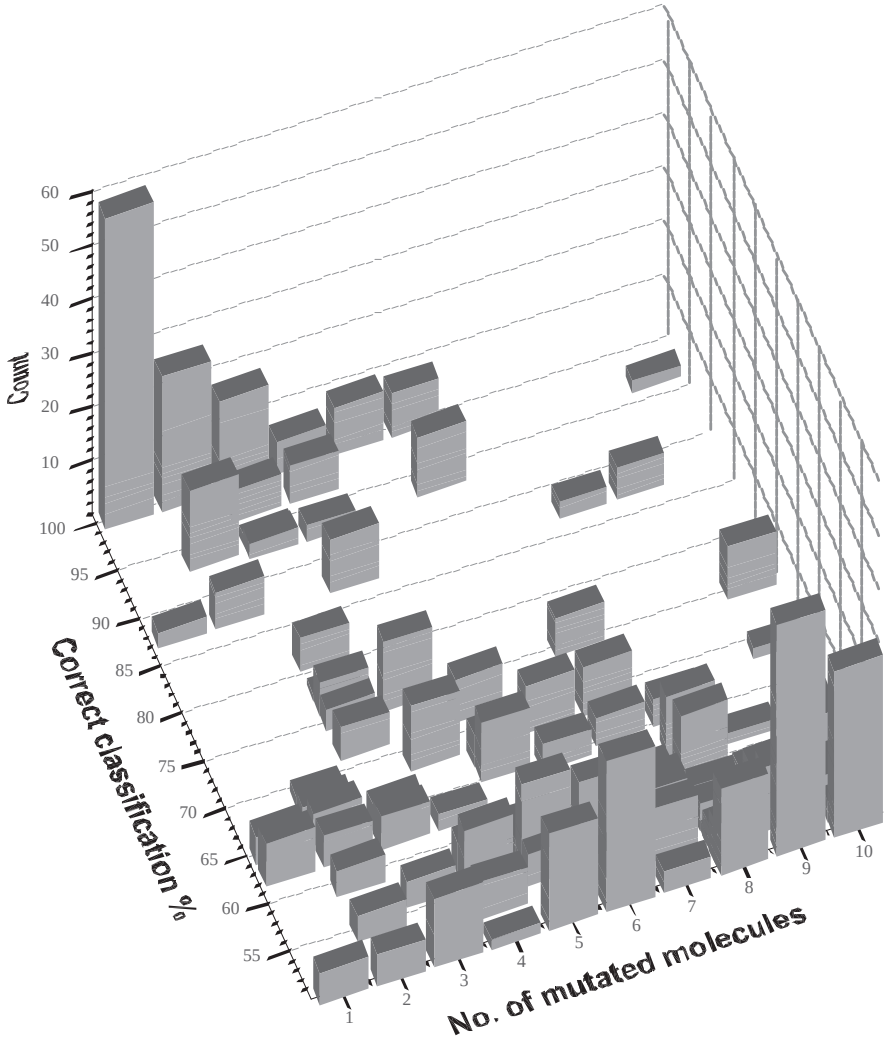


Figure 7.7: Performance and mutant count of sub-populations generated by molecular denaturation from five 8-input parity organisms. There are 100 mutants in each sub-population.

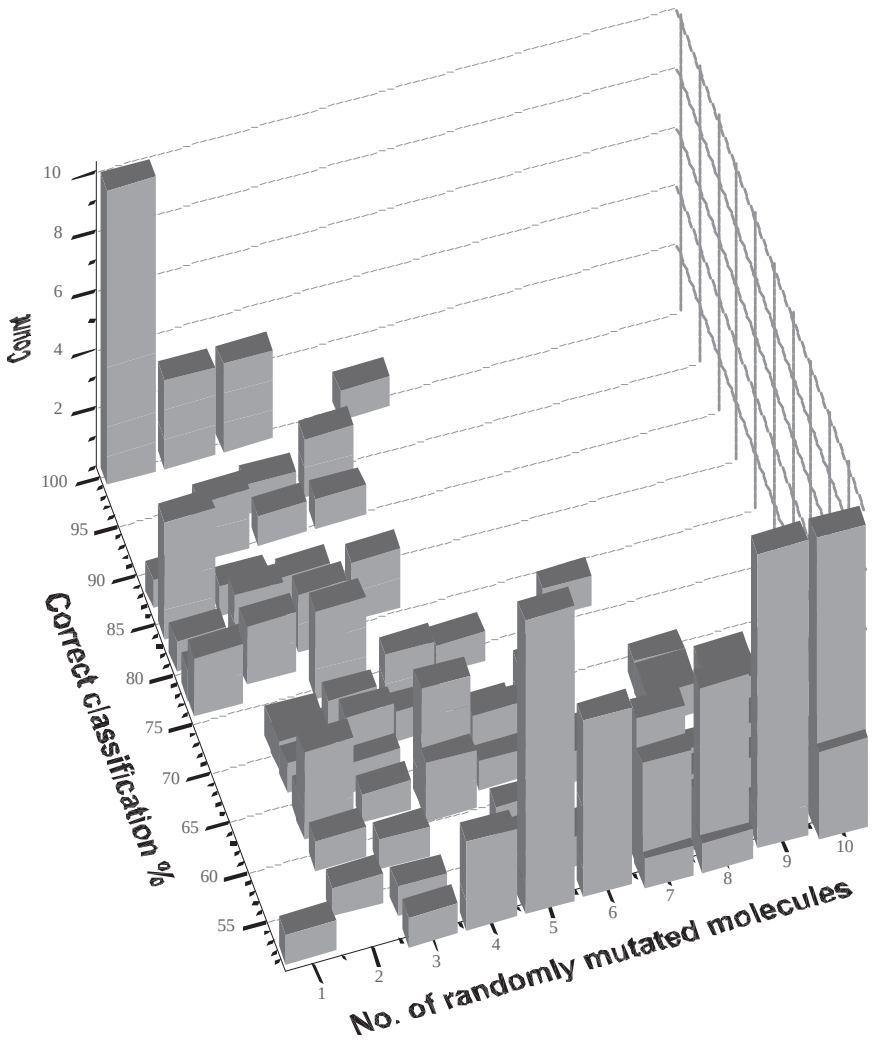


Figure 7.8: Performance and mutant count of sub-populations generated by random mutation from one 10-input parity organism. There are 20 mutants in each sub-population.

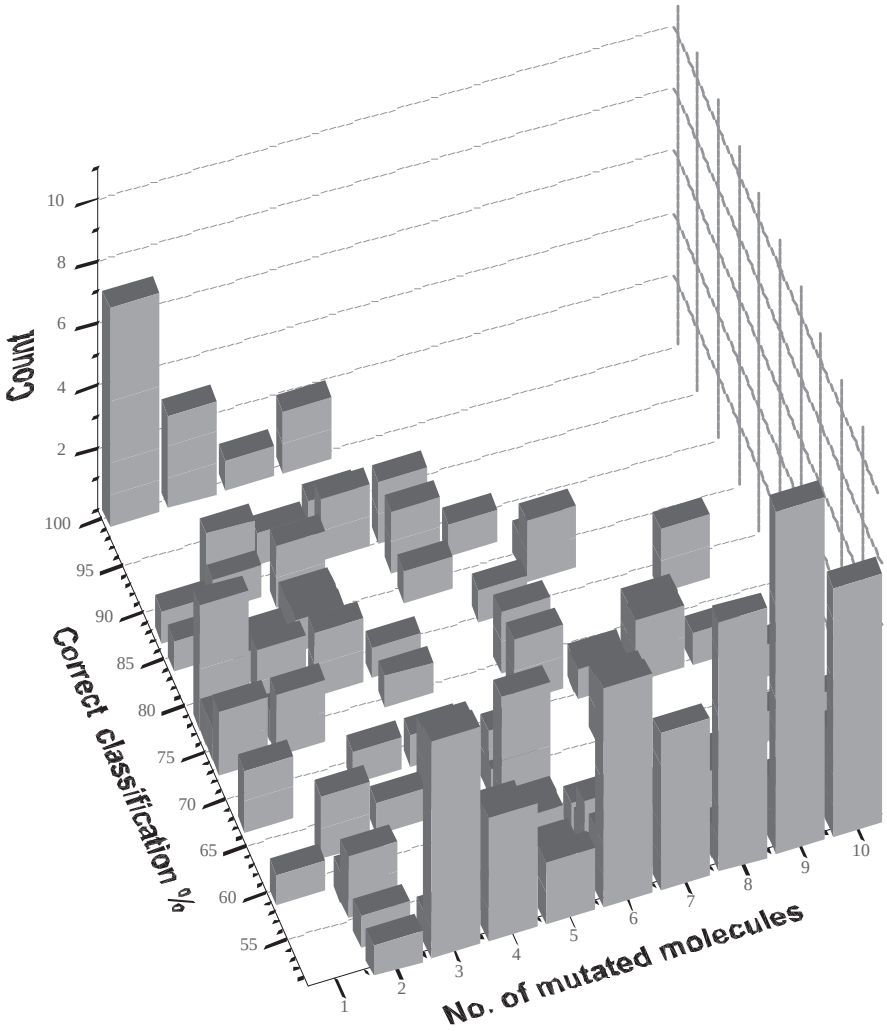


Figure 7.9: Performance and mutant count of sub-populations generated by molecular denaturation from one 10-input parity organism. There are 20 mutants in each sub-population.

7.3 Discussion

Biological systems have mutation buffering capabilities at the molecular, cellular, and organismic levels. These capabilities are important in two aspects:

- Potential for evolution allows exploration of different paths in the fitness landscape, when environmental condition changes, or internal changes force the organism to have a different behavior.

- Buffering ability allows to the organism and its components to survive when some elements of it, like cells or molecules, are damaged.

7.3.1 Buffering at the molecular level

At the molecular level mutation buffering is modeled in the learning procedure by allowing a molecule to flip just 15% of its bits. In this way the hypernetwork may generate molecules with different structures but with similar function (isozymes). The isozymes can become a valuable asset when other neighbor molecules are mutated, allowing the formation of a new dynamics previously unlikely if that molecule had kept its original structure.

Formation of new molecular network dynamics allows the evolving organism to search for other peaks in the fitness landscape. This buffering capacity has been observed, for instance, in catalytic RNA (Lehman et al., 2000).

7.3.2 Buffering at the organismic level

I study the buffering capabilities of organisms by generating a subpopulation of mutants. From the experiments I found that about ten to twenty percent of mutants with up to five mutations (or 1% - 2% of their molecules), show the same function as their

parents. These mutants explore the fitness landscape simultaneously, making the search more efficient. Also, the organisms are able to perform well, even when they suffer some molecular damage.

The experiments show that in some cases it could be better for a trained organism to denature some of its molecules, instead of mutating them to some other functional structures. New structures could give the organisms other unwanted features that would undermine the original function. This is part of a necessary compromise between acquiring new functions and maintaining all the necessary ones.

7.3.3 The trade-off principle

The trade-off principle of adaptability says: ”a computing system can not at the same time have high programmability, high computational efficiency, and high evolutionary adaptability” (Conrad, 1988).

Current implementations of computers have very high programmability, but very low evolvability. Some implementations of systems could have a high degree of computational efficiency but poor programmability, such as molecular pattern recognition systems. I argue that, at least in some cases, for instance in autonomous exploration, it is important for a system to perform the task and, at the same time, to be able to adapt to changes in its environment. The hypernetwork model shows the feasibility of building a physical realization of such characteristics.

7.4 Conclusions

Mutation-buffering properties are observed in hypernetwork organisms. The generation of mutants in the population that have the same or even better fitness value allows the system to explore efficiently on the fitness landscape. The hypernetwork

model of biological information processing could serve as model for the study of buffering capabilities of biological inspired computing systems.

CHAPTER 8

EVOLVABILITY PROPERTIES OF THE HYPERNETWORK

8.1 Introduction

Evolvability in biological systems is understood as the ability of a system to a self-organize through a variation-selection process (Conrad, 1990b), “the capacity to evolve” (Turney, 1999), and the “dispositional capacity to produce exhibited evolvability” (Nehaniv, 2000). According to Conrad (1998), this ability or capacity of organisms is based on the self-organization dynamics and mutation-buffering capabilities of their components.

In an organism the self-organization dynamics are based on the interaction and emergent properties of its components organized hierarchically (i.e., atomic interactions, molecules, cells, organisms). Moreover, the components are capable of having redundant features that allow for the same functionality when they have structural variation (mutations or damage). Examples are: pleiotropy at the genetic level, isozymes at the macro-molecular level, multiple reaction networks at the cellular level, and similar behavior observed in different organisms. Mutation-buffering capabilities allow the system to look for optimal structures in the adaptive fitness landscape, without losing previously required necessary functions.

The exploration of a complex fitness landscape by organisms is realized in the hypernetwork architecture. Learning is interpreted as traveling through a complex fitness landscape in search of a molecular distribution with required properties (not necessarily optimal).

Elements considered to play a role in evolvability are the following:

- At the molecular level, the complexity of the molecules, molecular redundancy, the ways that molecules interact to form positive and negative feedback regulatory networks.

- At the cellular level, the number and types of molecules in the cell (effectors, receptors, etc.).

- At the organismic level, the relationships among cells (cell to cell interactions), and the formation of networks of cells.

The goal of this chapter is to show how the complexity of molecular and organismic structures affects their evolvability, expressed in terms of their success for learning. The study covers the effects of molecular size and total organismic cell number on the evolvability of organisms.

Molecular complexity plays a role in biological information processing. Simple molecules would form only a limited network of interactions, thus negatively affecting performance of the organism. The number of molecules and cells also plays a role in learning. This relationship is explored in the context of learning two tasks: the N-input parity task, and the two x two bit multiplier truth table. Both tasks are described in Chapter 5. The experiments described below attempt to answer the question as to how the complexity of the components affect the evolvability of organisms in the hypernetwork architecture.

The plan of the chapter is as follows: First the experiments performed are described, followed by a discussion of the results and conclusions.

8.2 Evolvability experiments with the hypernetwork architecture

To explore the effect of molecular and organismic complexity on evolvability I train organisms to solve the following tasks: to learn the two x two-bit multiplier, and the (6-8)-input parity tasks. In each case two organisms with different numbers of molecules and cells were used, and experiments were run with 2-bit, 6-bit, 10-bit, and 14-bit length at each site of the molecule.

I find that in general the complexity of larger organisms would allow them to solve the tasks more quickly and efficiently than smaller ones. The complexity of the molecular structures expressed by their length would have a similar function.

8.2.1 Task description

The organisms used to solve the two x two-bit multiplier truth table are shown in Figure 8.1. The smaller and larger organisms have 96 and 304 molecules, respectively (see Table 8.1). The experimental parameters are shown in Table 8.2.

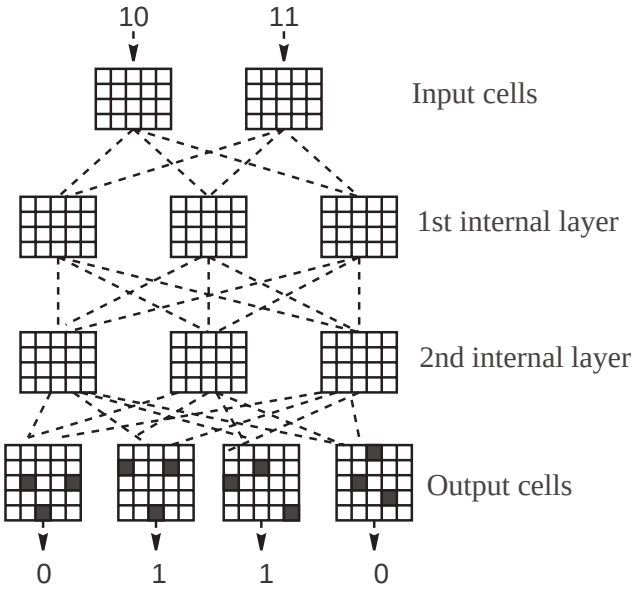
Task	Organism size	No. of cells	Total no. molecules	No. of cell-cell interactions
2 x 2 Multiplier	Large	12	304	27
	Small	8	96	7
6-input parity	Large	10	216	21
	Small	6	60	5
8-input parity	Large	11	236	24
	Small	7	84	6

Table 8.1: Some characteristics of the experimental organisms

The organisms train to solve the 6-input parity task are shown in Figure 8.2. The smaller organism has six cells with a total of 60 molecules, and the larger organism has ten cells with 216 molecules. The organisms for training the 8-input parity task are shown in Figure 8.3. The small organism has 7 cells with a total of 84 molecules, and the larger one 11 cells with a 236 molecules (see Table 8.1).

Parameter	2x2 Multiplier Large	2x2 Multiplier Small	6-inp. parity Large	6-inp. parity Small	8-inp. parity Large	8-inp. parity Small
No. of cells in the organism	12	8	10	6	11	7
Threshold of mol. activation	60%	60%	60%	60%	60%	60%
Threshold of mol. inhibition	70%	70%	70%	70%	70%	70%
Probability of mol. mutation	0.6	0.6	0.9	0.9	0.8	0.8
Percentage of mol. change	30%	30%	30%	30%	30%	30 %
No. of receptors for cell	4	4	4	4	4	4
No. of effectors for cell	4	4	4	4	4	4
No. of internal molec./ cell	12	4	12	4	12	4
No. readouts in output cell	3	4	6	3	6	3
% of inhibitors / cell	20%	30%	20%	20%	20%	20%

Table 8.2: Parameters of the experimental organisms.



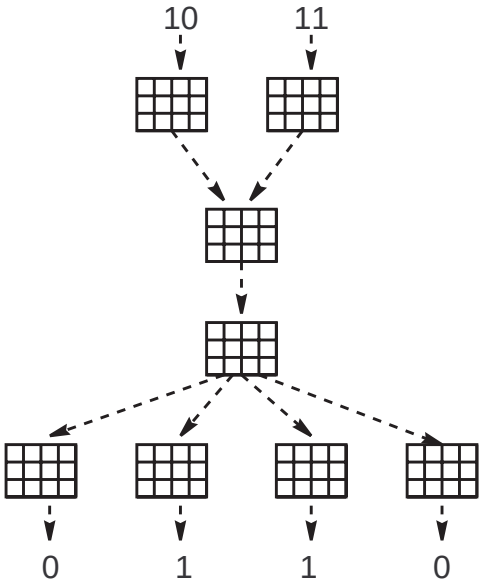


Figure 8.1: Large and small organisms trained to solve the two x two bit multiplier

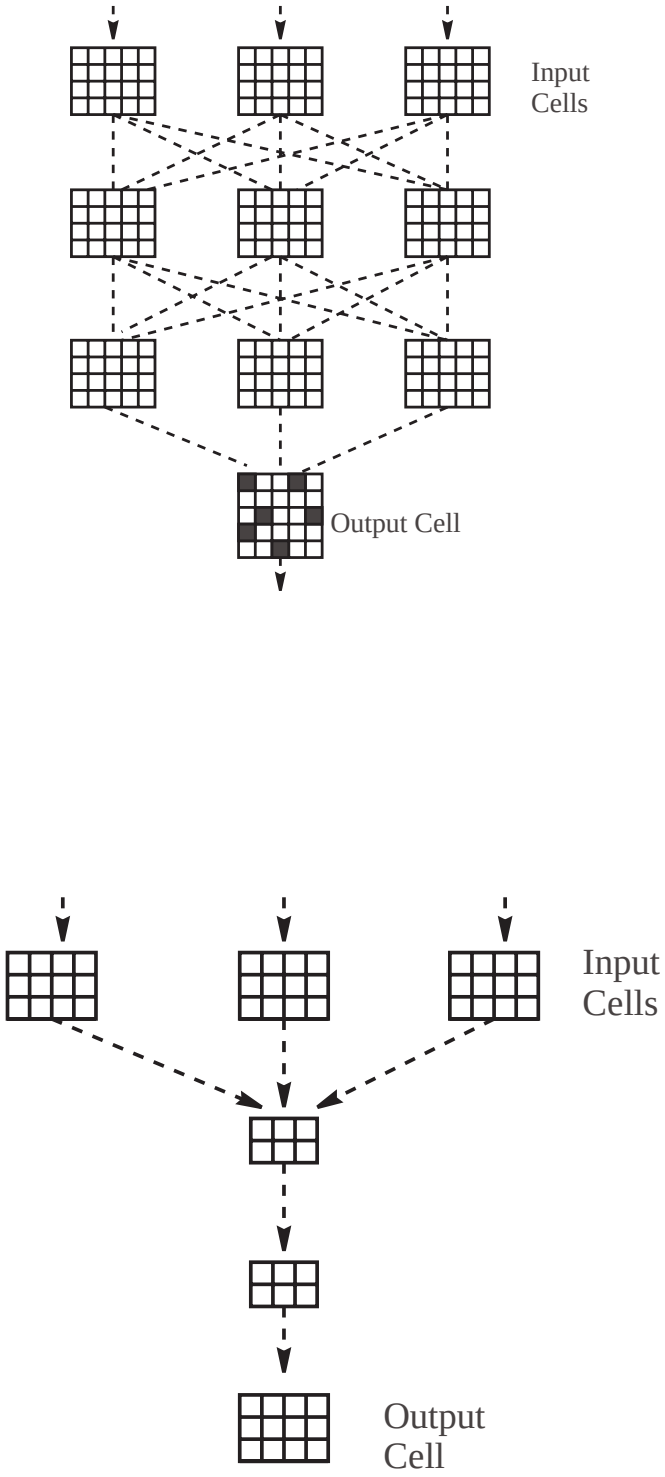


Figure 8.2: Large and small organisms trained to solve the 6-input parity task

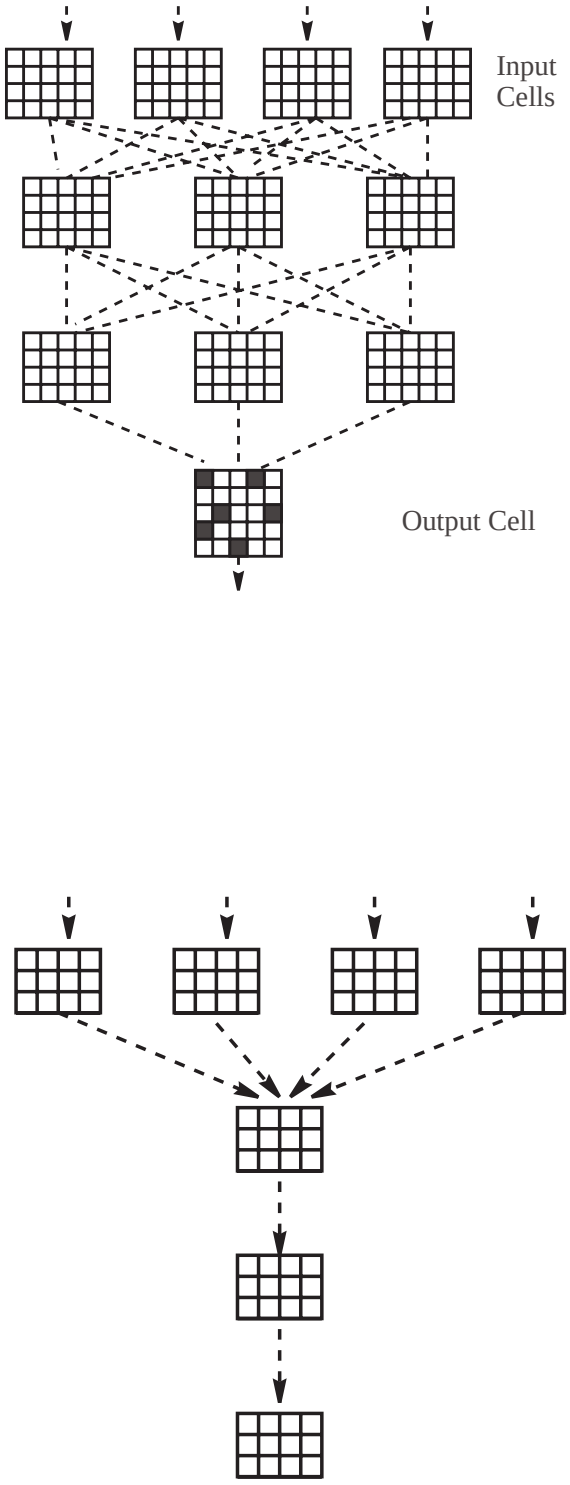


Figure 8.3: Large and small organisms trained to solve the 8-input parity task

8.2.2 Effect of molecular complexity

There is a tendency for learning to be improved by larger molecules. This effect is observed in learning the two x two-bit multiplier by the small and large organisms (see Figure 8.4).

The effect of molecular complexity has interesting results in the case of the (6-8)-input parity task. The tendency for improved learning performance is observed clearly in small organisms, but in large organisms the performance was better when each site of a molecule had 2 and 14 bits than when they had 6 and 10 bits (see Figures 8.5, and 8.6). When organisms have medium sized molecules, there exists a greater probability of finding neighboring molecules with which to form interactions, thereby causing over-excitation in the hypernetwork, and setting the output cells to "ON" most of the time. This problem may be avoided with smaller molecules, with very large molecules, restricting the size of the cells, or increasing the threshold of molecular activation.

The fact that I did not observe the problem of non-specific molecules in solving the two x two-bit multiplier may be due to the nature of the task. The N-Input parity problem has a "needle in a haystack" fitness landscape (Langdon and Poli, 1998).

8.2.3 Effect of the size of the organism

Organisms with more cells and more molecules per cell learn better in solving the two x two multiplier problem (see Figure 8.4). This was also observed in learning the (6-8)-input parity tasks (see Figures 8.5 and 8.6) , but only in the cases where each site of the molecule had 2 and 14-bits. The excessive number of networks of interactions that are generated in larger organisms with molecular site size of 6 and 10 bits cause the poor performance. This could be adjusted by changing the threshold for activating

molecules from 70% to a large number. Further experiments are needed to show the role of organismic size in learning.

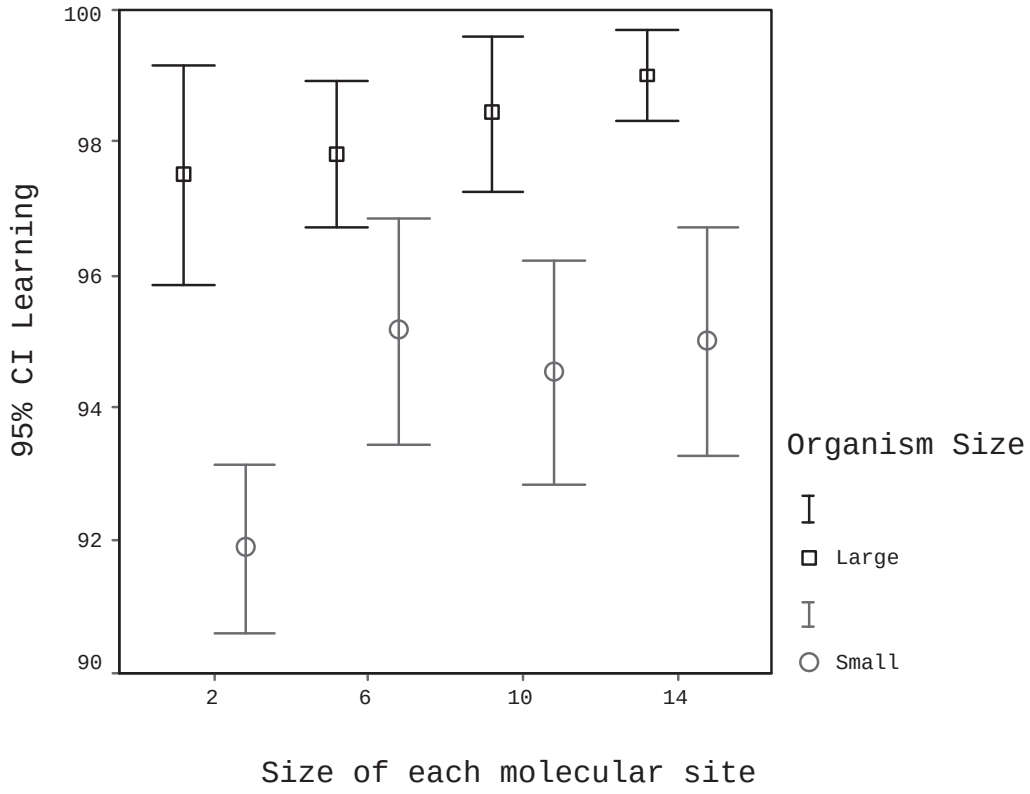


Figure 8.4: Effect of molecular size on learning the two x two-bit multiplier. It is shown the 95% Confidence Interval of learning of ten runs, with 200,000 epochs each.

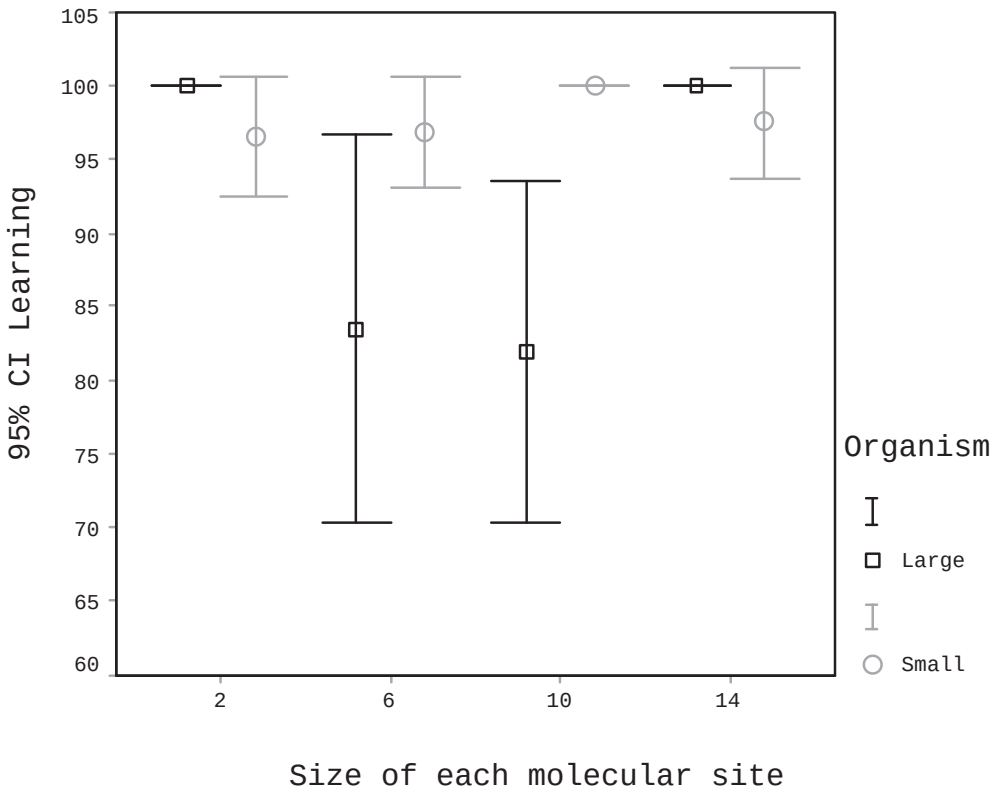


Figure 8.5: Effect of molecular size on large and small organisms learning the 6-input parity task. It is shown the 95% Confidence Interval of learning of ten runs, with up to 150,000 epochs each.

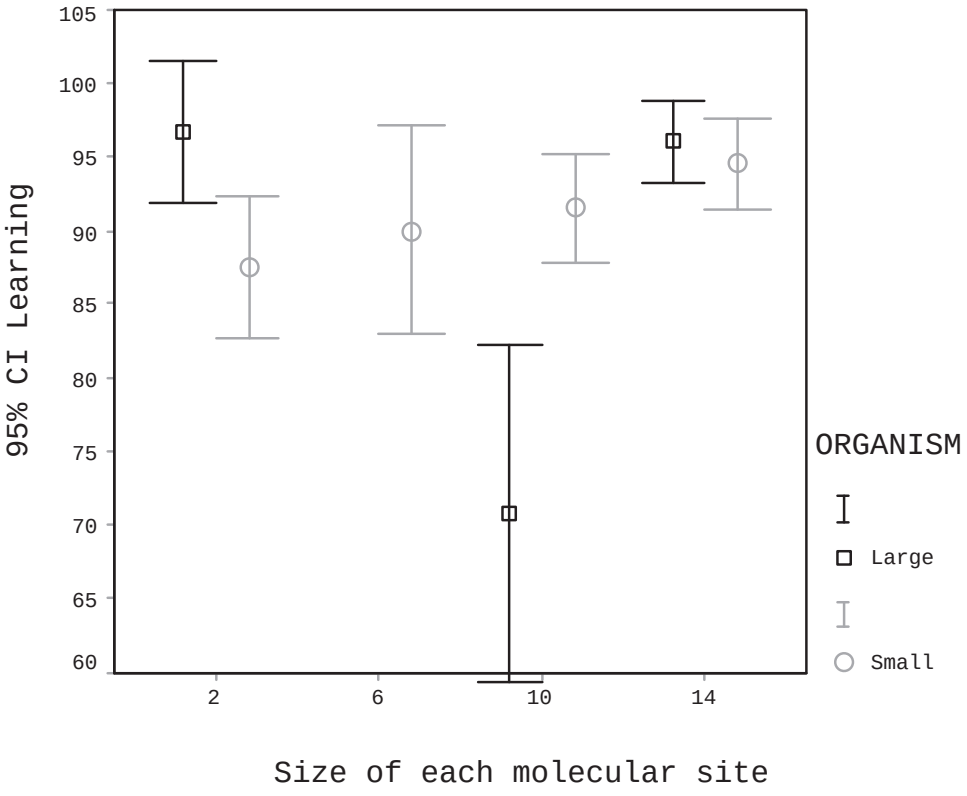


Figure 8.6: Effect of molecular size on large and small organisms learning the 8-input parity task. Average of ten runs, with up to 150,000 epochs each.

8.3 Concluding remarks

Experimental results how that, as noted by Conrad (1990a), evolvability is linked to an increase in component redundancy in the context of a hierarchical structure. While these results are preliminary, I have shown how the hypernetwork architecture could be a useful tool for experimenting on evolvability features such as compartamentalization, component redundancy and mutation-buffering. These properties of organisms facilitate the search for better solutions on the fitness landscape, and allow generation of novelty in the variation-selection algorithm. The relationship between evolvability and structural complexity is still in debate (Wagner, 1999), but it is possible that different strategies could coexist in different populations.

CHAPTER 9

CONCLUSIONS AND FUTURE DIRECTIONS

9.1 Conclusion

A novel computational model of biological information processing, the hypernetwork architecture, has been designed and implemented. The architecture is a theoretical model of a biological system, such as a neural tissue. It is a multi-level vertical model that includes representations of scale, information flow, feedback regulation control, and learning. All hierarchical features are formulated in terms of interactions of elementary macromolecular subunits. Molecules, represented by binary strings, form networks of interactions. The molecular interactions are based on shape complementarity.

A cell comprises a set of molecules of four types. Receptors gather information from other cells or the environment; effectors send influences to the receptors of other cells; internals interact with effectors, receptors, and with other internals; readout structures send signals to the environment reflecting the current cellular state.

The hypernetwork interacts with the environment by means of molecules in its input and output cells. Influences impinging on the receptor molecules of input cells dynamically form networks of molecular interactions. There are positive and negative feedback regulatory networks at the molecular level. Cell to cell interactions are formed by effector-receptor molecules of the interacting cells. Once the influences reach the effector molecules of output cells, the output state of a cell is obtained from its readout molecules.

The system is molded to perform classification tasks through a variation-selection algorithm acting on the structure of the molecular subunits. This molecular evolutionary

algorithm allows the organism to search for peaks in the fitness landscape.

The hypernetwork architecture will facilitate the study of hierarchical control and evolution at the different levels. It should also provide insights into how future bio-computers might be constructed.

9.2 Summary of experimental results

From the large number of simulations performed with the hypernetwork architecture, the major results are summarized as follows:

1. Learning capabilities.

   The hypernetwork architecture exhibits capabilities for addressing fairly complex classification problems. Hypernetwork organisms solved the following benchmark tasks:

   - The N-Input parity problem from N = 4 to N=10.
   - A type of double spiral problem with 56 points with up to 100%, and with 225 points with up to 96.89% learning.
   - The two x two multiplier truth table.
   - The tic-tac-toe endgame problem with 95% learning of a training set of 658 vectors, and up to 89.61% of a test set of 287 vectors.

2. Mutation-buffering capabilities

   Hypernetwork organisms exhibit mutation-buffering capabilities in the context of evolving populations. From experiments I found that as many as twenty percent of mutants with up to 2% molecular mutations retain the same function as their parents.

3. Evolvability properties.

   Results from experiments with hypernetwork organisms confirm that evolvability is linked to an increase in component complexity and redundancy, as expressed in terms of molecular complexity and organismic size.

4. Molecular regulation properties

   Inhibition and negative feedback regulation at the molecular level play a very important role in learning.

9.3 Future work

- Evolution in hierarchical systems

  The current implementation of the hypernetwork architecture has molecular level evolution coupled with selection of individuals. Complexity at the molecular level could be increased with the introduction of complex approaches to generating artificial molecules, such as Molnets (Colombano et al., 2000).

  Future implementations could include evolution at the level of cell interactions in order to better understand how cells change function in evolutionary time or in the course of development, and the role of regulation involving specialized cells. Moreover, evolving populations could be introduced to find solutions in parallel.

- Experimental test-bed

  The hypernetwork architecture can be used as an experimental test-bed in the areas of artificial life, evolution and adaptability theories, and models of molecular regulation.

- Exploring the hypernetwork evolutionary learning algorithm

  The hypernetwork architecture exhibits generalization properties. However, more experimentation is needed to find the strengths and weaknesses of the hypernetwork evolutionary learning algorithm, when compared with other learning algorithms such as back-propagation and decision trees. Evolution across scales is also a potential unique feature that requires further exploration.

- Towards the physical realization of the hypernetwork

  The molecular interaction basis of the hypernetwork, a representation of biological entities, could facilitate the design of future “bio-computers” based on networks of molecular interactions and hierarchical control. This architecture complements Conrad’s ideas of conformation-driving computing (Conrad, 1995c).

APPENDIX A. DESCRIPTION OF THE EXPERIMENTAL PARAMETERS.

Parameter	Description
create_organism #1	Create an organism with #1 cells
threshold_activation #1	Threshold of matching #1 to activate a molecule
threshold_inhibition #1	Threshold to inhibit a molecule (matching over #1 percentage of the string bits)
mutation_rate_mol_cell #1	Mutation rate #1 of the molecule
mutation_rate_atom_mol #1	Percentage of bits #1 that randomly flip when a molecule is mutating
molecular_shape_size #1	Size of the molecules in the organism (#1 bits)
create_cell #1 #2 rect x y #3 #4	Create a cell with parameters: #1 = Cell number #2 = no. of molecules/cell rect = rectangular topology x = side of the rectangle y = side of the rectangle #3 = cell type #4 = no. of readouts/cell
n_receptors_cell #1	no. of receptors in the cell #1
n_effectors_cell #1	no. of effectors in the cell #1
n_internal_cell #1	no. of internal molecules in the cell #1
p_inhibitors_cell #1	Percentage of molecules in the cell #1 with active inhibitory site
copy_structure_cell #1 #2	Copy cell properties from cell #1 to cell #2
cell_cell #1 #2	To set potential cell to cell relationships from cell #1 to cell #2
fraction_input_vector #1	Number of bits #1 to activate an input cell
minimal_global_error #1	#1 is the desired error to stop training
seeds clock clock	The initial seeds of the random number generator are set to a function of the clock

APPENDIX B. PARAMETER FILES OF THE EXPERIMENTS IN CHAPTER FIVE.

Parameter file for the organism trained to solve the 4-input parity task.

```
create_organism          9
threshold_activation    60
threshold_inhibition    70
mutation_rate_mol_cell  0.2
mutation_rate_atom_mol   30
molecular_shape_size     14

create_cell       1  20 rect 5 4  input 0
n_receptors_cell  1   4
n_effectors_cell  1   4
n_internal_cell   1  12
p_inhibitors_cell 1  20

create_cell       2  20 rect 5 4 input  0
n_receptors_cell  2   4
n_effectors_cell  2   4
n_internal_cell   2  12
p_inhibitors_cell 2   20

create_cell       3  20 rect 5 4 internal1 0
n_receptors_cell  3   4
n_effectors_cell  3   4
n_internal_cell   3  12
p_inhibitors_cell 3   15

create_cell       4  20 rect 5 4 internal1 0
n_receptors_cell  4  4
n_effectors_cell  4  4
n_internal_cell   4  12
p_inhibitors_cell 4  20

create_cell        5  20 rect 4 5 internal1  0
copy_structure_cell 4 5

create_cell        6  20 rect 5 4 internal2 0
n_receptors_cell   6   4
n_effectors_cell   6   4
n_internal_cell    6  12
p_inhibitors_cell  6   20

create_cell        7  20 rect 5 4 internal2 0
create_cell        8  20 rect 5 4 internal2 0
```

```
copy_structure_cell 6  7
copy_structure_cell 6  8


create_cell        9  25 rect 5 5 output 6
n_receptors_cell   9   5
n_effectors_cell   9   5
n_internal_cell    9  15
p_inhibitors_cell  9  20

cell_cell 1  3
cell_cell 1  4
cell_cell 1  5
cell_cell 2  3
cell_cell 2  4
cell_cell 2  5
cell_cell 3  6
cell_cell 3  7
cell_cell 3  8
cell_cell 4  6
cell_cell 4  7
cell_cell 4  8
cell_cell 5  6
cell_cell 5  7
cell_cell 5  8
cell_cell 6  9
cell_cell 7  9
cell_cell 8  9

fraction_input_vector  2
minimal_global_error   0.001
seeds clock clock

end.
```

Parameter file for the organism trained to solve the 6-input parity task

```
create_organism        10
threshold_activation   60
threshold_inhibition   70
mutation_rate_mol_cell 0.9
mutation_rate_atom_mol 30
molecular_shape_size   14

create_cell       1  20 rect 5 4  input 0
n_receptors_cell  1   4
n_effectors_cell  1   4
n_internal_cell   1  12
p_inhibitors_cell 1  20

create_cell       2  20 rect 5 4 input  0
n_receptors_cell  2   4
n_effectors_cell  2   4
n_internal_cell   2  12
p_inhibitors_cell 2   20

create_cell       3  20 rect 5 4 input  0
n_receptors_cell  3   4
n_effectors_cell  3   4
n_internal_cell   3  12
p_inhibitors_cell 3  20

create_cell       4  20 rect 5 4 internal1 0
n_receptors_cell  4  4
n_effectors_cell  4  4
n_internal_cell   4  12
p_inhibitors_cell 4  20

create_cell        5  20 rect 4 5 internal1 0
create_cell        6  20 rect 4 5 internal1 0
copy_structure_cell 4 5
copy_structure_cell 4 6

create_cell        7  20 rect 5 4 internal2 0
n_receptors_cell   7   4
n_effectors_cell   7   4
n_internal_cell    7  12
p_inhibitors_cell  7   20

create_cell        8  20 rect 5 4 internal2 0
create_cell         9  20 rect 5 4 internal2 0
```

```
copy_structure_cell 7  8
copy_structure_cell 7  9

create_cell        10  25 rect 5 5 output 6
n_receptors_cell   10   5
n_effectors_cell   10   5
n_internal_cell    10  15
p_inhibitors_cell  10  20


cell_cell 1  4
cell_cell 1  5
cell_cell 1  6

cell_cell 2  4
cell_cell 2  5
cell_cell 2  6

cell_cell 3  4
cell_cell 3  5
cell_cell 3  6

cell_cell 4  7
cell_cell 4  8
cell_cell 4  9

cell_cell 5  7
cell_cell 5  8
cell_cell 5  9

cell_cell  6  7
cell_cell  6  8
cell_cell  6  9

cell_cell 7   10
cell_cell 8   10
cell_cell 9   10

fraction_input_vector  2

minimal_global_error   0.001
seeds clock clock

end.
```

Parameter file for the organism trained to solve the 8-input parity task

```
create_organism        11
threshold_activation   60
threshold_inhibition   70
mutation_rate_mol_cell  0.8
mutation_rate_atom_mol 30
molecular_shape_size 14

create_cell       1  20 rect 5 4  input 0
n_receptors_cell  1   4
n_effectors_cell  1   4
n_internal_cell   1  12
p_inhibitors_cell 1  20

create_cell       2  20 rect 5 4 input  0
n_receptors_cell  2   4
n_effectors_cell  2   4
n_internal_cell   2  12
p_inhibitors_cell 2   20

create_cell       3  20 rect 5 4 input  0
n_receptors_cell  3   4
n_effectors_cell  3   4
n_internal_cell   3  12
p_inhibitors_cell 3  20

create_cell       4  20 rect 5 4 input  0
n_receptors_cell  4   4
n_effectors_cell  4   4
n_internal_cell   4  12
p_inhibitors_cell 4  20

create_cell       5  20 rect 5 4 internal1 0
n_receptors_cell  5  4
n_effectors_cell  5  4
n_internal_cell   5  12
p_inhibitors_cell 5  20

create_cell        6  20 rect 4 5 internal1 0
create_cell        7  20 rect 4 5 internal1 0
copy_structure_cell 5 6
copy_structure_cell 5 7

create_cell        8  20 rect 5 4 internal2 0
```

```
n_receptors_cell   8   4
n_effectors_cell   8   4
n_internal_cell    8  12
p_inhibitors_cell  8   20

create_cell        9   20 rect 5 4 internal2 0
create_cell        10  20 rect 5 4 internal2 0
copy_structure_cell 8  9
copy_structure_cell 8 10

create_cell        11  25 rect 5 5 output 6
n_receptors_cell   11   5
n_effectors_cell   11   5
n_internal_cell    11  15
p_inhibitors_cell  11  20


cell_cell 1  5
cell_cell 1  6
cell_cell 1  7

cell_cell 2  5
cell_cell 2  6
cell_cell 2  7

cell_cell 3  5
cell_cell 3  6
cell_cell 3  7

cell_cell 4  5
cell_cell 4  6
cell_cell 4  7

cell_cell 5  10
cell_cell 5  8
cell_cell 5  9

cell_cell 6  10
cell_cell 6  8
cell_cell 6  9

cell_cell  7  8
cell_cell  7  9
cell_cell  7  10
```

```
cell_cell 8   11
cell_cell 9   11
cell_cell 10  11

fraction_input_vector  2

minimal_global_error   0.001
seeds clock clock

end.
```

Parameter file for the organism trained to solve the 10-input parity task

```
create_organism        12
threshold_activation   60
threshold_inhibition   70
mutation_rate_mol_cell  0.6
mutation_rate_atom_mol 30
molecular_shape_size 14

create_cell       1  20 rect 5 4  input 0
n_receptors_cell  1   4
n_effectors_cell  1   4
n_internal_cell   1  12
p_inhibitors_cell 1  20

create_cell       2  20 rect 5 4 input  0
n_receptors_cell  2   4
n_effectors_cell  2   4
n_internal_cell   2  12
p_inhibitors_cell 2   20

create_cell       3  20 rect 5 4 input  0
n_receptors_cell  3   4
n_effectors_cell  3   4
n_internal_cell   3  12
p_inhibitors_cell 3  20

create_cell       4  20 rect 5 4 input  0
n_receptors_cell  4   4
n_effectors_cell  4   4
n_internal_cell   4  12
p_inhibitors_cell 4  20

create_cell       5  20 rect 5 4 input  0
n_receptors_cell  5   4
n_effectors_cell  5    4
n_internal_cell   5 12
p_inhibitors_cell 5  20


create_cell       6  20 rect 5 4 internal1 0
n_receptors_cell  6  4
n_effectors_cell  6  4
n_internal_cell   6  12
p_inhibitors_cell 6  20
```

```
create_cell        7  20 rect 4 5 internal1 0
create_cell        8  20 rect 4 5 internal1 0
copy_structure_cell 6  7
copy_structure_cell 6  8

create_cell        9  20 rect 5 4 internal2 0
n_receptors_cell   9   4
n_effectors_cell   9   4
n_internal_cell    9  12
p_inhibitors_cell  9   20

create_cell        10  20 rect 5 4 internal2 0
create_cell        11  20 rect 5 4 internal2 0
copy_structure_cell 9  10
copy_structure_cell 10 11

create_cell        12  25 rect 5 5 output 6
n_receptors_cell   12   5
n_effectors_cell   12   5
n_internal_cell    12  15
p_inhibitors_cell  12  20


cell_cell 1  6
cell_cell 1  7
cell_cell 1  8

cell_cell 2  6
cell_cell 2  7
cell_cell 2  8

cell_cell 3  6
cell_cell 3  7
cell_cell 3  8

cell_cell 4  6
cell_cell 4  7
cell_cell 4  8

cell_cell 5  6
cell_cell 5  7
cell_cell 5  8

cell_cell 6  9
cell_cell 6  10
```

```
cell_cell 6  11

cell_cell  7  9
cell_cell  7  10
cell_cell  7  11

cell_cell  8  9
cell_cell  8  10
cell_cell  8  11


cell_cell 9    12
cell_cell 10   12
cell_cell 11  12

fraction_input_vector  2

minimal_global_error   0.001
seeds clock clock

end.
```

Parameter file for the organism trained to solve the double spiral data set

```
create_organism        11
threshold_activation   60
threshold_inhibition   70
mutation_rate_mol_cell  0.8
mutation_rate_atom_mol 30
molecular_shape_size 14

create_cell       1  20 rect 5 4  input 0
n_receptors_cell  1   4
n_effectors_cell  1   4
n_internal_cell   1  12
p_inhibitors_cell 1  20

create_cell       2  20 rect 5 4 input  0
n_receptors_cell  2   4
n_effectors_cell  2   4
n_internal_cell   2  12
p_inhibitors_cell 2   20

create_cell       3  20 rect 5 4 input  0
n_receptors_cell  3   4
n_effectors_cell  3   4
n_internal_cell   3  12
p_inhibitors_cell 3  20

create_cell       4  20 rect 5 4 input  0
n_receptors_cell  4   4
n_effectors_cell  4   4
n_internal_cell   4  12
p_inhibitors_cell 4  20

create_cell       5  20 rect 5 4 internal1 0
n_receptors_cell  5  4
n_effectors_cell  5  4
n_internal_cell   5  12
p_inhibitors_cell 5  20

create_cell        6  20 rect 4 5 internal1 0
create_cell        7  20 rect 4 5 internal1 0
copy_structure_cell 5 6
copy_structure_cell 5 7

create_cell        8  20 rect 5 4 internal2 0
n_receptors_cell   8   4
```

```
n_effectors_cell   8   4
n_internal_cell    8  12
p_inhibitors_cell  8   20

create_cell        9   20 rect 5 4 internal2 0
create_cell        10  20 rect 5 4 internal2 0
copy_structure_cell 8  9
copy_structure_cell 8 10

create_cell        11  25 rect 5 5 output 6
n_receptors_cell   11   5
n_effectors_cell   11   5
n_internal_cell    11  15
p_inhibitors_cell  11  20


cell_cell 1  5
cell_cell 1  6
cell_cell 1  7

cell_cell 2  5
cell_cell 2  6
cell_cell 2  7

cell_cell 3  5
cell_cell 3  6
cell_cell 3  7

cell_cell 4  5
cell_cell 4  6
cell_cell 4  7

cell_cell 5  10
cell_cell 5  8
cell_cell 5  9

cell_cell 6  10
cell_cell 6  8
cell_cell 6  9

cell_cell  7  8
cell_cell  7  9
cell_cell  7  10

cell_cell 8   11
```

```
cell_cell 9   11
cell_cell 10  11

fraction_input_vector  2

minimal_global_error   0.001
seeds clock clock

end.
```

Parameter file for the organism trained to solve the two x twx multiplier

```
create_organism         12
threshold_activation    60
threshold_inhibition    70
mutation_rate_mol_cell  0.6
mutation_rate_atom_mol   30
molecular_shape_size     14

create_cell       1  20 rect 5 4  input 0
n_receptors_cell  1   4
n_effectors_cell  1   4
n_internal_cell   1  12
p_inhibitors_cell 1  20

create_cell       2  20 rect 5 4 input  0
n_receptors_cell  2   4
n_effectors_cell  2   4
n_internal_cell   2  12
p_inhibitors_cell 2   20

create_cell       3  20 rect 5 4 internal1 0
n_receptors_cell  3   4
n_effectors_cell  3   4
n_internal_cell   3  12
p_inhibitors_cell 3   20

create_cell       4  20 rect 5 4 internal1 0
n_receptors_cell  4  4
n_effectors_cell  4  4
n_internal_cell   4  12
p_inhibitors_cell 4  30

create_cell        5  20 rect 5 4 internal1 0
n_receptors_cell   5   4
n_effectors_cell   5   4
n_internal_cell    5  12
p_inhibitors_cell  5   30

create_cell        6  20 rect 5 4 internal2 0
n_receptors_cell   6   4
n_effectors_cell   6   4
n_internal_cell    6  12
p_inhibitors_cell  6   30
```

```
create_cell        7  20 rect 5 4 internal2 0
copy_structure_cell 6 7

create_cell         8  20 rect 5 4 internal2 0
copy_structure_cell 6 8


create_cell        9  25 rect 5 5 output 3
n_receptors_cell   9   5
n_effectors_cell   9   5
n_internal_cell    9  15
p_inhibitors_cell  9  20

create_cell        10  25 rect 5 5 output 3
n_receptors_cell   10   5
n_effectors_cell   10   5
n_internal_cell    10  15
p_inhibitors_cell  10  20

create_cell        11  25 rect 5 5 output 3
n_receptors_cell   11  5
n_effectors_cell   11   5
n_internal_cell    11  15
p_inhibitors_cell  11  20

create_cell        12  25 rect 5 5 output 3
n_receptors_cell   12   5
n_effectors_cell   12   5
n_internal_cell    12  15
p_inhibitors_cell  12  20


cell_cell 1  3
cell_cell 1  4
cell_cell 1  5

cell_cell 2  3
cell_cell 2  4
cell_cell 2  5

cell_cell 3  6
cell_cell 3  7
cell_cell 3  8

cell_cell 4  6
```

```
cell_cell 4  7
cell_cell 4  8

cell_cell 5  6
cell_cell 5  7
cell_cell 5  8

cell_cell 6  9
cell_cell 6  10
cell_cell 6  11
cell_cell 6  12

cell_cell 7  9
cell_cell 7  10
cell_cell 7  11
cell_cell 7  12

cell_cell 8  9
cell_cell 8  10
cell_cell 8  11
cell_cell 8  12


fraction_input_vector  2

minimal_global_error   0.001
seeds clock clock

end.
```

Parameter file for the organism trained to solve the tic-tac-toe endgame problem

```
create_organism        16
threshold_activation   60
threshold_inhibition   70
mutation_rate_mol_cell  0.8
mutation_rate_atom_mol 30
molecular_shape_size 14

create_cell       1  20 rect 5 4  input 0
n_receptors_cell  1   4
n_effectors_cell  1   4
n_internal_cell   1  12
p_inhibitors_cell 1  20

create_cell       2  20 rect 5 4 input  0
n_receptors_cell  2   4
n_effectors_cell  2   4
n_internal_cell   2  12
p_inhibitors_cell 2   20

create_cell       3  20 rect 5 4 input  0
n_receptors_cell  3   4
n_effectors_cell  3   4
n_internal_cell   3  12
p_inhibitors_cell 3  20

create_cell       4  20 rect 5 4 input  0
n_receptors_cell  4   4
n_effectors_cell  4   4
n_internal_cell   4  12
p_inhibitors_cell 4  20

create_cell       5 20 rect 5 4 input  0
n_receptors_cell  5   4
n_effectors_cell  5   4
n_internal_cell   5  12
p_inhibitors_cell 5  20

create_cell       6  20 rect 5 4 input  0
n_receptors_cell  6   4
n_effectors_cell  6   4
n_internal_cell   6  12
p_inhibitors_cell 6  20
```

```
create_cell       7  20 rect 5 4 input  0
n_receptors_cell  7   4
n_effectors_cell  7   4
n_internal_cell   7  12
p_inhibitors_cell 7  20

create_cell       8  20 rect 5 4 input  0
n_receptors_cell  8   4
n_effectors_cell  8   4
n_internal_cell   8  12
p_inhibitors_cell 8  20

create_cell       9  20 rect 5 4 input  0
n_receptors_cell  9   4
n_effectors_cell  9   4
n_internal_cell   9  12
p_inhibitors_cell 9  20

create_cell       10  20 rect 5 4 internal1 0
n_receptors_cell  10 4
n_effectors_cell  10 4
n_internal_cell   10  12
p_inhibitors_cell 10  20

create_cell       11  20 rect 4 5 internal1 0
create_cell        12  20 rect 4 5 internal1 0
copy_structure_cell 10 11
copy_structure_cell 10 12

create_cell        13   20 rect 5 4 internal2 0
n_receptors_cell   13    4
n_effectors_cell   13    4
n_internal_cell    13   12
p_inhibitors_cell  13   20

create_cell         14   20 rect 5 4 internal2 0
create_cell         15   20 rect 5 4 internal2 0
copy_structure_cell  13  14
copy_structure_cell  13  15

create_cell        16  25 rect 5 5 output 6
n_receptors_cell   16   5
n_effectors_cell   16   5
n_internal_cell    16  15
p_inhibitors_cell  16  20
```

```
cell_cell 1  10
cell_cell 1  11
cell_cell 1  12

cell_cell 2  10
cell_cell 2  11
cell_cell 2  12

cell_cell 3  10
cell_cell 3  11
cell_cell 3  12

cell_cell 4  10
cell_cell 4  11
cell_cell 4  12

cell_cell 5  10
cell_cell 5  11
cell_cell 5  12

cell_cell 6  10
cell_cell 6  11
cell_cell 6  12

cell_cell 7  10
cell_cell 7  11
cell_cell 7  12

cell_cell 8  10
cell_cell 8  11
cell_cell 8  12

cell_cell 9  10
cell_cell 9  11
cell_cell 9  12


cell_cell 10  13
cell_cell 10  14
cell_cell 10  15

cell_cell 11  13
cell_cell 11  14
```

```
cell_cell 11  15

cell_cell 12  13
cell_cell 12  14
cell_cell 12  15

cell_cell 13  16
cell_cell 14  16
cell_cell 15  16

fraction_input_vector  2

minimal_global_error   0.001
seeds clock clock

end.
```

BIBLIOGRAPHY

Abel, T. and Lattal, K. M. (2001). Molecular mechanisms of memory acquisition, consolidation and retrieval. *Current Opinion in Neurobiology*, 11:180–187.

Abel, T., Martin, K. C., Bartsch, D., and Kandel, E. R. (1998). Memory supressor genes: Inhibitory constraints on the storage of long-term memory. *Science*, 279:338–341.

Aha, D. H. (1991). Incremental constructive induction: An instance-based approach. In *Proceedings of the Eighth International Workshop on Machine Learning*, pages 117–121. Morgan Kaufmann, Evanston, ILL.

Ahl, V. and Allen, T. (1996). *Hierarchy Theory: A Vision, Vocabulary, and Epistemology*. Columbia University Press, New York.

Albert, R., Jeong, H., and Barabási, A.-L. (2000). Error and attack tolerance of complex networks. *Nature*, 406:378–382.

Alberts, B., Bray, D., Lewis, J., Raff, M., and Watson, J. D. (1994). *Molecular Biology of the Cell*. Garland Pub.

Allen, T. and Hoekstra, T. W. (1992). *Toward a Unified Ecology*. Columbia University Press, New York.

Allen, T. and Starr, T. B. (1982). *Hierarchy: Perspectives for Ecological Complexity*. The University of Chicago Press, Chicago and London.

Aon, M. and Cortassa, S. (1997). *Dyamic Biological Organization*. Chapman & Hall.

Ashby, R. (1956). *Introduction to Cybernetics*. John Wiley, New York.

Auger, P. (1989). *Dynamics and Thermodynamics in Hierarchically Organized Systems*. Oxford, Pergamon Press.

Bäck, T. (1994). Evolutionary algorithms: Comparision of approaches. In Paton (1994), pages 227–243.

Bäck, T., Hammel, U., and Schwefel, H.-P. (1997). Evolutionary computation: Comments on the history and current state. *IEEE Transactions on Evolutionary Computation*, 1(1):3–17.

Beurle, R. L. (1962). Functional organization in random networks. In Von Foerster, H. and Zopf, G. W., editors, *Principles of Self-Organization*, pages 291–311. Pergamon Press, New York.

Bhalla, U. S. and Iyengar, R. (1999). Emergent properties of networks of biological systems. *Science*, 283:381–387.

Bonner, J. T. (1965). *Size and Cycle: an essay on the structure of biology*. Princeton University Press, Princeton, New Jersey.

Capra, F. (1996). *The Web of Life*. Anchor Books, Doubleday, New York.

Carbonell, J. G., Michalski, R. S., and Mitchell, T. M. (1983). An overview of machine learning. In Michalski, R. S., Carbonell, J. G., and Mitchell, T. M., editors, *Machine Learning, An Artificial Intelligence Approach*, pages 1–23. Morgan Kauffmann Publishers, Inc.

Carpenter, G. A., Grossber, S., Markuzon, N., John, R., and Rosen, D. B. (1992). Fuzzy

ARTMAP: A neural network architecture for incremental supervised learning of analog multidimensional maps. *IEEE Transactions on Neural Networks*, 3(5):698–712.

Chen, J.-C. (1993). *Computer Experiments on Evolutionary Learning in a Multilevel Neuromolecular Architecture*. PhD thesis, Wayne State University.

Chen, J.-C. and Conrad, M. (1997a). Evolutionary learning with a neuromolecular architecture: a biologically motivated approach to computational adaptability. *Soft Computing*, 1:19–34.

Chen, J.-C. and Conrad, M. (1997b). Pattern categorization and generalization with a virtual neuromolecular architecture. *Neural Networks*, 10(1):111–123.

Colombano, S., New, M. H., Pohorille, A., Scargle, J., Stassinopoulos, D., Pearson, M., and Warren, J. (2000). Evolutionary cell computing: From protocells to self-organized computing. In *Proceedings of the Computational Aerosciences Workshop*, Moffett Field, CA. NASA Ames Research Center.

Conrad, M. (1972). Statistical and hierarchical aspects of biological organization. In Waddington, C., editor, *Towards a Theoretical Biology*, pages 189–220. Edinburgh University Press, Edinburgh.

Conrad, M. (1974). Molecular information processing in the central nervous system. Part I: Selection circuits in the brain. In Conrad, M., Güttinger, and Dal Cin, M., editors, *Physics and Mathematics onf the Nervous System*, pages 82–107, Springer-Verlag. Berlin-Heildelberg-New York.

Conrad, M. (1977). Principle of superposition-free memory. *J. Theoretical Biology*, 67:213–219.

Conrad, M. (1979a). Bootstrapping on the adaptive landscape. *BioSystems*, (11):167–182.

Conrad, M. (1979b). Hierarchical adaptability theory and its cross-correlation with dynamical ecological models. In Halfon, E., editor, *Theoretical Systems Ecology*, pages 132–150. Academic Press, New York.

Conrad, M. (1979c). *Theoretical Systems Ecology*, chapter Hierarchical Adaptability Theory and Its Cross-Correlation with Dynamical Ecological Models, pages 132–150. Academic Press, New York.

Conrad, M. (1983). *Adaptability: The Significance of Variability from Molecule to Ecosystem*. Plenum Press, New York and London.

Conrad, M. (1984). Microscopic-macroscopic interface in biological information processing. *Biosystems*, 16:345–363.

Conrad, M. (1985). The mutation buffering concept of biomolecular structure. *Proc. Int. Symp. Biomol. Struct. Interactions, Suppl. J. Biosci.*, 8(3-4):669–679.

Conrad, M. (1988). The price of programmability. In Herken, R., editor, *The Universal Turing Machine. A Half-Century Survey*, pages 285–307. Verlag Kammerer & Unverzagt, Hamburg-Berlin.

Conrad, M. (1990a). The geometry of evolution. *BioSystems*, 24:61–81.

Conrad, M. (1990b). Molecular computing. In Yovits, M. C., editor, *Advances in*

Computers, volume 31, pages 235–324. Academic Press, Inc., Boston, San Diego, New York.

Conrad, M. (1992). Molecular computing: The lock-key paradigm. *Computer*, 25(11):11–20.

Conrad, M. (1993). Emergent computation through self-assembly. *Nanobiology*, 2:5–30.

Conrad, M. (1994). Speedup of self-organization through quantum mechanical parallelism. In Mishra, R., Maaß, D., and Zwierlein, E., editors, *On Self-Organization: An Interdisciplinary Search for a Unifying Principle*, pages 92–108. Springer-Verlag, Berlin, Heidelberg, New York.

Conrad, M. (1995a). Cross-scale interactions in biomolecular information processing. *BioSystems*, 35:157–160.

Conrad, M. (1995b). Multiscale synergy in biological information processing. *Optical Memory and Neural Networks*, 4(2):89–98.

Conrad, M. (1995c). Surpassing computational limits with bioelectronic and molecular electronic technologies: Towards the multiscale computational architecture. *Future Electronic Devices Journal*, 6(Suppl. 2):46–60.

Conrad, M. (1997). From Mach's principle to human health: a multiscale model of homeostasis. *ADVANCES: The Journal of Mind-Body Health*, 13(4):7–11.

Conrad, M. (1998). Towards high evolvability dynamics. In Van de Vijver, G. et al., editors, *Evolutionary Systems: Biological and Epistemological Perspectives*

on Selection and Self-organization, pages 33–43. Kluwer Academic Publishers, Dordrecht.

Conrad, M. and Pattee, H. (1970). Evolution experiments with an artificial ecosystem. *J. Theoret. Biol.*, 28:393–409.

Conrad, M. and Rizki, M. (1980). The artificial worlds approach to emergent evolution. *BioSystems*, 23:247–260.

Danysz, W., C.G., P., Bresink, I., and G., Q. (1995). Glutamate in CNS disorders-A revived target for drug development. *Drugs News Perspectives*, (8):261–277.

DeAngelis, D. L. (1995). The nature and significance of feedback in ecosystems. In Patten, B., Jörgensen, S. E., and Auerbach, S., editors, *The Part-Whole Relation in Ecosystems*, pages 450–467. Prentice Hall PTR, Englewood Cliffs, New Jersey 07632.

Dodd, J. and Schuchardt, A. (1995). Axon guidance: a compelling case for repelling growth cones. *Cell*, 81:471–474.

Edelman, G. M. and Finkel, L. H. (1984). Neuronal group selection in the cerebral cortex. In Edelman, G. M., Gall, W. E., and Cowan, W. M., editors, *Dynamic Aspects of Neocortical Function*, pages 653–695. Wiley-Interscience.

Feekes, Gerrit, B. (1986). *The Hierarchy of Energy Systems, From Atom to Society*. Pergamon Press, Oxford.

Fogel, L., Owens, A., and Walsh, M. (1966). *Artificial Intelligence Through Simulated Evolution*. Wiley, New York.

Fogel, L. J. (1999). *Intelligence Thorugh Simulated Evolution: forty years of evolutionary programming*. John Wiley & Sons, Inc.

Francesconi, A. and Duvoisin, R. M. (2000). Opposing effects of protein kinase C and protein kinase A on metabotropic glutamate receptor signaling: Selective desensitation on the inositol triphosphate/Ca^{2+} pathway by phosphorylation of the receptor-G protein-couple domain. *Proceedings of the National Academy of Sciences*, 97(11):6185–6190.

Gatlin, L. L. (1972). *Information Theory and the Living System*. Columbia University Press, New York and London.

Green, D. G. (1994). Connectivity and the evolution of biological systems. *Journal of Biological Systems*, 2(1):91–103.

Gurney, K. (1997). *An Introduction to Neural Networks*. UCL Press.

Hebb, D. O. (1949). *The Organization of Behavior*. John Wiley & Sons.

Holland, J. (1975). *Adaptation in Natural and Artificial Systems*. University of Michigan Press, Ann Arbor.

Holling, C. (1992). Cross-scale morphology, geometry, and dynamics of ecosystems. *Ecological Monographs.*, 62(4):447–502.

Jacob, F. and Monod, J. (1961). Genetic regulatory mechanisms in the syntehsis of proteins. *Journal of Molecular Biology*, (3):318–356.

Kauffman, S. (1969). Metabolic stability and epigenesis in randomly connected nets. *J. Theoret. Biol.*, 22:437–467.

Kauffman, S. A. (1993). *The Origins of Order: Self-Organization and Selection in Evolution*. Oxford University Press, New York, Oxford.

Kauffman, S. A. (1996). Investigations: The nature of autonomous agents and the worlds they mutually create. Technical report, Santa Fe Institute. http://www.santafe.edu/sfi/People/kauffman/Investigations.html.

Koestler, A. (1976). *The Ghost in the Machine*. Hutchinson, London.

Koza, J. R. (1992). *Genetic Programming*. MIT Press, Cambridge, MA.

Lang, K. and Witbrock, M. J. (1989). Learning to tell two spirals apart. In *Proceedings of the 1988 Connectionist Models Summer School*, pages 52–59.

Langdon, W. B. and Poli, R. (1998). Why ”building blocks” don’t work on parity problems. Technical Report CSRP-98-17, University of Birmingham, School of Computer Science.

Langton, C. G., editor (1994). *Artificial Life III*, Redwood City, California. Santa Fe Institute, Studies in the Sciences of Complexity, Addison-Wesley Publishing Company.

Lehman, N., Delle Donne, M., West, M., and Dewey, T. G. (2000). The genotypic landscape during in vitro evolution of a catalytic RNA: Implications for phenotypic buffering. *Journal of Molecular Evolution*, 50(5):481–490.

Levin, S. (1992). The problem of pattern and scale in ecology. *Ecology*, 73(6):1943–1967.

Liberman, E., Minina, S., Mjakotina, O., Shklovsky-Kordy, N., and Conrad, M. (1985). Neuron generator potentials evoked by intracellular injection of cyclic nucleotides and mechanical distension. *Brain Research*, (338):33–44.

Martinet-Edelist, C. (1994). A logical description of the evolution of feedback loop systems and its application to rhabdovirus infection. *Journal of Biological Systems*, 2(1):55–72.

Matheus, C. J. (1990). Adding domain knowledge to sbl through feature construction. In *Proceedings of the Eighth National Conference on Artificial Intelligence*, pages 803–808, Boston, MA. AAAI Press.

Matheus, C. J. and Rendell, L. A. (1989). Constructive induction on decision trees. In *Proceedings of the Eleventh International Joint Conference on Artificial Intelligence*, pages 645–650, Detroit, MI. Morgan Kaufmann.

McClelland, T. L. and Rumelhart, D. E. (1986). *Parallel Distributed Processing*. MIT Press and PDP Group.

McCulloch, W. and Pitts, W. (1943). A logical calculus of the ideas immanent in nervous activity. *Bulletin of Mathematical Biophysics*, (5):115–133.

McDonald, J. and Johnston, W. (1990). Physiological and pathophysiological role of excitatory amino acids during central nervous system development. *Brain Research Reviews*, 15:41–70.

Michalski, R. (1987). Learning strategies and automated knowledge acquisition: an

overview. In Bolc, L., editor, *Computational Models of Learning*, pages 1–19. Springer-Verlag, Berlin, Heilderber, New York, London, Paris, Tokyo.

Minsky, M. and Papert, S. (1969). *Perceptrons: An Introduction to Computational Geometry*. MIT Press, Cambridge, MA.

Mitchell, T. (1997). *Machine Learning*. McGraw Hill.

Nehaniv, C. L. (2000). Measuring evolvability as the rate of complexity increase. In Nehaniv, C. L., editor, *Proceedings of the Evolvability Workshop at the the Seventh International Conference on the Simulation and Synthesis of Living Systems (Artificial Life 7) 1-2 August 2000, Reed College, Portland, Oregon, USA*, pages 66–68.

O'Neill, R. (1989). Perspectives in hierarchy and scale. In Roughgarden, J., May, R., and A., L. S., editors, *Perspectives in Ecological Theory*, pages 140–156. Princeton University Press.

O'Neill, R., DeAngelis, D., Waide, J., and Allen, T. (1986). *A Hierarchical Concept of Ecosystems*. Princeton University Press, Princeton, New Jersey.

Pahl-Wostl, C. (1995). *The Dynamic Nature of Ecosystems: Chaos and Order Entwined*. John Wiley & Sons, Chichester, England.

Paton, R., editor (1994). *Computing with Biological Metaphors*, London. Chapman & Hall.

Pattee, H. H. (1970). The problem of biological hierarchy. In Waddington, C., editor, *Towards a Theoretical Biology*, pages 117–136. Aldine Publishing Company, Chicago.

Pattee, H. H. (1972). The nature of hierarchical control in living matter. In Rosen, R., editor, *Foundations of Mathematical Biology*, volume 1, pages 1–22. Academic Press, New York and London.

Pattee, H. H. (1973). The physical basis and origin of hierarchical control. In Pattee, H. H., editor, *Hierarchy Theory, The Challenge of Complex Systems*, pages 71–107. George Braziller, New York.

Peters, R. H. (1983). *The ecological implications of body size*. Cambridge University Press, Cambridge.

Pianka, E. R. (1994). *Evolutionary Ecology, Fifth Edition*. HarperCollins College Publishers, New York.

Reynolds, R. G. (1994). An introduction to cultural algorithms. In Sebald, A. and Fogel, L., editors, *Proceedings of the Third Annual Conference on Evolutionary Programming*, pages 131–139, River Edge, NJ. World Scientific.

Reynolds, R. G. and Zannoni, E. (1992). Why does cultural evolution proceed at a faster rate than biological evolution? In Gilbert, N., editor, *Simulating Societies: A symposium on Approaches to Simulating Social Phenomena and Social Processes*, pages 81–91, Guilford, UK. University of Surrey.

Rosen, R. (1967). *Optimality principles in Biology*. Butteworths, London.

Rosen, R. (1972). Some relational cell models: the metabolic-repair system. In *Foundations of Mathematical Biology*, volume 2, pages 217–253. Academic Press, New York.

Rosen, R. (1974). The role of quantum theory in biology. *Int. J. Quantum Chem.: Quantum Biology Symp.*, 1:229–232.

Rosen, R. (1985). Organisms as causal systems which are not mechanisms: An essay into the nature of complexity. In Rosen, R., editor, *Theoretical Biology and Complexity: Three Essays on the Natural Philosophy of Complex Systems*, pages 165–203. Academic Press, Orlando.

Sakamoto, N., de Atauri, P., and Cascante, M. (1998). Effects of feedback inhibition on transit time in a linear pathway of michaelis-menten-type reactions. *BioSystems*, 45:221–235.

Salthe, S. N. (1985). *Evolving Hierarchical Systems*. Columbia University Press, New York.

Schüz, A. (1995). Neuroanatomy in a computational perspective. In Arbib, M. A., editor, *The Handbook of Brain Theory and Neural Networks*, pages 622–626. The MIT Press.

Schwefel, H.-P. (1995). *Evolution and optimum seeking*. Wiley, New York.

Segovia-Juarez, J. and Colombano, S. (2001). Mutation-buffering capabilities of the hypernetwork model. In Keymeulen, D., Stoica, A., Lohn, J., and Zebulum, R. S., editors, *Proceedings of the Third NASA/DoD Workshop on Evolvable Hardware*, pages 7–13, 12-14 July, Long Beach, California, USA. IEEE Computer Society.

Segovia-Juarez, J. and Conrad, M. (1999). Hypernetwork model of biological information processing. In Angeline, P. J., Michalewicz, Z., Schoenauer, M., Yao, X., and Zalzala,

A., editors, *Proceedings of the 1999 Congress on Evolutionary Computation*, volume 1, pages 511–516, 6-9 July, Mayflower Hotel, Washington D.C., USA. IEEE Press.

Segovia-Juarez, J. and Conrad, M. (2001). Learning with the molecular-based hypernetwork model. In Kim, J.-H., Byoung-Tak, Z., Fogel, G., and Kuscu, I., editors, *Proceedings of the 2001 Congress on Evolutionary Computation*, volume 2, pages 1177–1182, 27-30 May, COEX, Seoul, Korea. IEEE Press.

Segovia-Juárez, J. L. and Conrad, M. (1999). Hypernetwork model of biological information processing. In Angeline, P. J., Michalewicz, Z., Schoenauer, M., Yao, X., and Zalzala, A., editors, *Proceedings of the 1999 Congress on Evolutionary Computation*, volume 1, pages 511–516, Mayflower Hotel, Washington D.C., USA. IEEE Press.

Shang, Y. and Wah, B. W. (1996). Global optimization for neural network training. *IEEE Computer*, 3:45–54.

Shepherd, G. (1994). *Neurobiology*. Oxford University Press, New York.

Simon, H. A. (1962). The architecture of complexity. *Proceedings of the American Philosophical Society*, 106(6):467–482.

Simon, H. A. (1973). The organization of complex systems. In Pattee, H. H., editor, *Hierarchy Theory, The Challenge of Complex Systems*, pages 1–28. George Braziller, New York.

Simon, H. A. (1983). Why should machines learn? In Michalski, R. S., Carbonell, J. G.,

and Mitchell, T. M., editors, *Machine Learning, An Artificial Intelligence Approach*, pages 25–37. Morgan Kauffmann Publishers, Inc.

Simon, H. A. (1996). *The Sciences of the Artificial, Third Edition*. The MIT Press, Cambridge, Massachusetts.

Smith, C. U. M. (1994). The complexity of brains: A biologist's view. In Stonier, R. J. and Yu, X. H., editors, *Complex Systems: Mechanism of Adaptation*, pages 93–100. IOS Press and Ohmsha, Amsterdam and Tokyo.

Somogyi, R. and Fuhrman, S. (1998). Distributivity, a general information theoretic network measure, or why the whole is more than the sum of its parts. New York and London. Plenum Press.

Squire, L. and Kandel, E. (2000). *Memory: From Mind to Molecules*. Scientific American Library.

Tesauro, G. and Janssens, B. (1988). Scaling relationships in back-propagation learning. *Complex Systems*, 2:39–84.

Thieffry, D. and Romero, D. (1999). The modularity of biological regulatory networks. *BioSystems*, 50:49–59.

Thomas, R. (1990). *Biological Feedback*. CRC Press, Boca Raton.

Turney, P. D. (1999). Increasing evolvability considered as a large-scale trend in evolution. In Wu, A., editor, *Proceedings of the 1999 Genetic and Evolutionary Computation Conference Workshop Program*, pages 43–46.

Ugur, A. and Conrad, M. (1999). Building evolution friendliness into cellular automaton dynamics: The cytomatrix neuron model. In *Proceedings of the 1999 Congress on Evolutionary Computation*, pages 2071–2077. IEEE Press.

van Zandt, T. (1995). Continous approximations in the study of hierarchies. *RAND Journal of Economics*, 26(4):575–590.

Vianna, M. R. M., Szapiro, G., McGaugh, J. L., Medina, J., and Izquierdo, I. (2001). Retrieval of memory for fear-motivated training initiates extinction requiring protein synthesis in the rat hippocampus. *Proc. Natl. Acad. Sci. USA.*, 98(21):12251–12254.

von Bertalanffy, L. (1968). *General System Theory. Foundations, Development, Applications.* George Braziller, New York.

Wagner, G. P. (1999). The evolvability of complex organisms: The quantitative genetic perspective. In Wu, A., editor, *Proceedings of the 1999 Genetic and Evolutionary Computation Conference Workshop Program*, pages 47–50.

Webster, J. (1979). *Theoretical Systems Ecology*, chapter Hierarchical Organization of Ecosystems, pages 119–129. Academic Press, New York.

Wegner, P. (1997). Why interaction is more powerful than algorithms. *Communications of the ACM*, 40(5):80–91.

Wegner, P. (1998). Interactive foundations of computing. *Theoretical Computer Science*, 192(2):315–351.

Werbos, P. J. (1994). *The roots of backpropagation: from ordered derivatives to neural networks and political forecasting*. J. Wiley & Sons.

Wiener, N. (1948). *Cybernetics*. MIT Press (reprinted 1961), Cambridge, Mass.

Wright, S. (1932). The roles of mutation, inbreeding, crossbreeding and delection in evolution. In *Proceedings of the Sixth International Congress of Genetics*, pages 356–366.

Wuensche, A. (1994). The ghost in the machine: Basins of attraction of random boolean networks. In Langton (1994), pages 465–501.

Zurada, J. M. (1992). *Introduction to Artificial Neural Systems*. West Publishing Company, St. Paul, New York, Los Angeles.

ABSTRACT

THE HYPERNETWORK ARCHITECTURE:
A HIERARCHICAL MOLECULAR INTERACTION MODEL OF BIOLOGICAL INFORMATION PROCESSING

by

JOSE L. SEGOVIA-JUAREZ

DECEMBER 2001

Advisors: Vaclav Rajlich Ph.D., Silvano Colombano Ph.D.

Major: Computer Science

Degree: Doctor of Philosophy

A novel architecture for machine learning, the hypernetwork architecture, has been specially designed and implemented in this study. This is a multi level, vertical model of a biological information processing system that includes the flow of information and feedback regulation control inspired by biological systems. The levels considered are the molecular, cellular and organismic.

The molecular level consists of many molecules, i.e., binary string representations derived from enzyme-like structures. Each molecule has an excitatory and a catalytic site, but each has an optional inhibitory site. A molecular interaction, binary string matching, represents a biomolecular self-assembly process. Dynamic formation of networks of molecular interactions represents reaction cascades in biological cells.

Molecules are placed in cells modeled by cellular automata, and an organized group of cells forms an organism. Cell to cell interactions are produced by effector-receptor molecules of the cells.

External influences on the receptor molecules of input cells dynamically trigger cascades of molecular interactions inside the cells of the organism. Then the cascade activates readout molecules on the output cells to form the output of the cell.

Hypernetwork organisms learn classification tasks by means of a variation-selection algorithm based on molecular evolution. Each iteration consists of an organism being reproduced with random molecular mutation, and the better one being chosen to perform the task. The mutation-buffering capabilities of the hypernetwork allow to search for optimal peaks in the fitness landscape.

The hypernetwork effectively learns classification tasks such as the (4-10)-input parity problem, the tic-tac-toe endgame problem, the two x two bit multiplier truth table, and a type of double spiral data set. Experimental results show that, in the hypernetwork architecture, learning improves when molecules exhibit inhibitory sites. This improvement is the result of molecular inhibition and negative feedback regulation inside the cells.

AUTOBIOGRAPHICAL STATEMENT

JOSE L. SEGOVIA-JUAREZ

Education

- Ph.D. in Computer Science, 2001.
  Wayne State University, Detroit, MI.
  Dissertation: The hypernetwork architecture: a hierarchical molecular interaction model of biological information processing.
  Advisors: Vaclav Rajlich and Silvano Colombano.

- Master's degree in Computer Science [Magíster en Informática], 1991.
  Pontificia Universidad Católica del Peru, Lima-Peru.
  Master's Thesis: Un intérprete para modelos en Dinámica de Sistemas.

- Professional Title in Biology, 1988 and B.Sc. in Biology, 1987.
  Universidad Peruana Cayetano Heredia, Lima-Peru.
  B.Sc. Thesis: Cambios en la Estructura de la Vegetación en la Reserva Altoandina de Pampa Galeras. Rol de los Grandes Herbívoros.

Publications

Segovia-Juarez, J. and Conrad, M. Learning with the molecular-based hypernetwork model. Proceedings of the 2001 Congress on Evolutionary Computation, 27-30 May 2001, Seoul-Korea. pp: 1177-1182, May 2001.

Segovia-Juarez, J. and Conrad, M. Hypernetwork Model of Biological Information Processing. Proceedings of the 1999 Congress on Evolutionary Computation, 6-9 July, Washington D.C., USA. pp: 511-515, July 1999.

Sethi, I. K., Coman, I., Day B., Jiang, F., Li, D., Segovia-Juarez, J., Wei, G. and You, B. ColorWISE: A System for Image Similarity Retrieval Using Color. Storage and Retrieval for Image and Video Databases VI - SPIE Proceedings, 28 - 30 January 1998, San Jose, CA, USA. pp: 140 -149. 1998.

Segovia-Juarez, J. and Conrad, M. Modeling vertical information flow in biological systems. Quantum Approaches to Consciousness. Conference at Northern Arizona University, Flagstaff, Arizona, July 28 - August 1, 1999.

Segovia-Juarez, J. and Hoces-Roque, D. Un modelo dinámico de la población de vicuñas. Estudios de saca. Boletín de Lima. 14(80):23-28. 1992.

Fellowships

Thomas Rumble Graduate Fellowship. Wayne State University, 09/1995 - 05/1996.
Fulbright Fellowship. Fulbright Commission of Peru, 08/1993 - 08/1995.