Mutation buffering capabilities of the hypernetwork model

José L. Segovia-Juárez
Department of Computer Science
Wayne State University
407 State Hall. Detroit, MI 48202
(313)-831-7623
jls@cs.wayne.edu

Silvano Colombano
NASA-Ames Research Center
Computational Sciences Division
scolombano@mail.arc.nasa.gov

Abstract

The hypernetwork is a molecular interaction-based model that has learning capabilities. The adaptive algorithm randomly changes the molecular structures and selects the best individual. Experiments with the hypernetwork show the importance for evolution of the mutation buffering capabilities of the system. Mutation buffering allows the system to improve its search for peaks in the fitness landscape.

1. Introduction

The bootstrap principle of evolutionary adaptability proposed by M. Conrad [2] states "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 [3]).

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 [3, 5, 10], facilitating evolution. 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.

In this paper we show the mutation buffering capabilities of the hypernetwork model, a molecular interaction-based model that has learning capabilities, and we discuss its importance in searching the adaptive landscape [20] (see Figure 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.

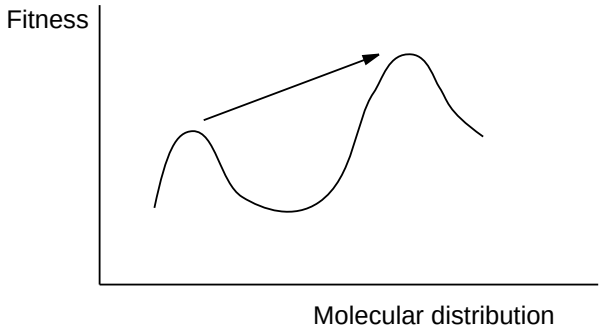

Figure 1. Fitness landscape.

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

We argue that evolvable hardware should incorporate mutation buffering capabilities. The system should allow some internal changes due to mutation, deterioration, or failure. The system should be able to maintain some of its original functionality under such circumstances, and at the same time, these molecular changes should allow a system to evolve to other configurations. These capabilities are of great importance for survival and evolution.

First we describe the hypernetwork model and two of the classification tasks it was given to learn. We then test the organisms that learned the tasks for their ability to maintain functionality in the presence of random molecular mutation or denaturation.

2. The hypernetwork model

The hypernetwork model [15, 16] is a biologically inspired model in which molecular interactions play an information processing role. It is a hierarchical model that illustrates the flow of influences from the macro to the mi-

cro level, and vice-versa. Its conceptual basis is Conrad's percolation network model [4, 7, 8, 9], and the enzymatic neuron model [1]. Following the classification scheme of Zebulim et al. [21], the hypernetwork architecture may be considered as a molecular interaction-based evolving platform.

The basic elements of the hypernetwork model are molecular interactions based on shape complementarity. The dynamic formation of networks of interactions represents biochemical reactions in the cell. From top to bottom, the organism is an organized set of cells, where each cell is represented by a cellular automaton. A molecule is placed in each location of the cell grid (see Figure 2).

The molecules are models of proteins. They are represented by binary strings, and have three sites: an *excitatory* site and an *inhibitory* site that set the molecule into the active state or the inactive state, respectively, and a *catalytic* site that can match the excitatory and inhibitory sites of neighbor molecules. If the excitatory matching is above a threshold, then the target molecule is activated. Inhibitory matching will inhibit the target molecule regardless of any other activation matching of the molecule.

The excitatory and inhibitory receptors of the molecule allow for the possibility of feedback regulatory networks with neighboring molecules.

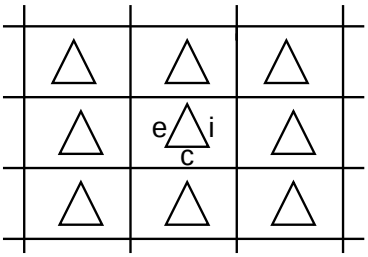

Figure 2. Partial view of a cell. The molecule in the center interacts with its eight neighbors. Its excitatory (e), inhibitory (i), and catalytic (c) sites are labelled.

The adaptive algorithm, described in detail in [15], is designed to learn to classify a set of binary input vectors. The input vector activates the receptor molecules of the input cells. Influences are propagated through each cell until they reach effector molecules. Effector molecules behave the way neurotransmitters do in neural systems when they reach receptors in other cells. Potential relationships of such cell to cell interactions are predefined, and are shown as dotted lines in Figure 3. The actual cell to cell interactions are dynamically formed by every input vector.

Once the influences arrive at the effector molecules of output cells the output vector is obtained by reading the

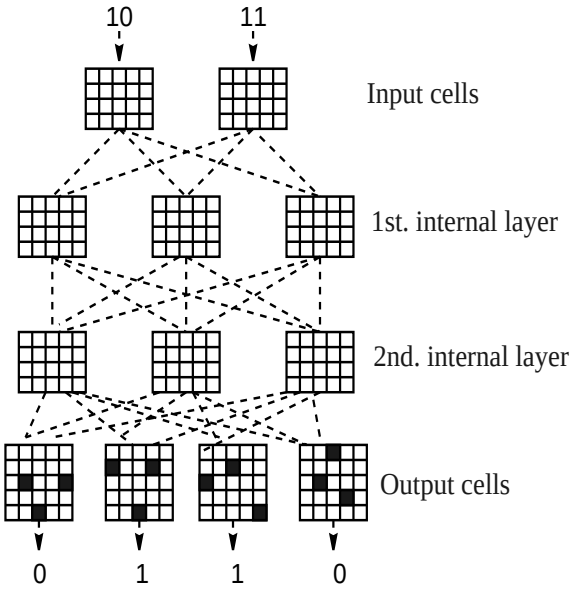

Figure 3. A hypernetwork with 2 input cells, 2 layers of internal cells, and an output layer with 4 cells.

state of the readout structures. If any molecule close to a readout molecule was activated then the output is "1", otherwise it is "0". We concatenate the bits to form the output vector. We compare the output vector with the desired one to find the error for this particular case.

We apply this procedure for each one of the input vectors, and we find the global error based on all of them. If the organism does not achieve 100% learning, it is reproduced with mutation.

At the moment of reproduction each molecule is selected for mutation with a probability of about 0.8%, and, if chosen, it randomly flips a fraction of its bits. In current experiments usually 30% of the bits are to be randomly changed.

Once a new organism is generated this way we test it with the training vectors. If it has better performance than its parent, we keep it as the better individual, otherwise we use the parent to make another mutant and go through the cycle for a determined number of epochs, or until the organism achieves 100% learning.

3. Learning tasks

For the purpose of this work, we found organisms to solve the two x two-bit multiplier, and the N-input parity tasks. In both cases the goal was to learn the complete table. The trained organisms were used to test their robustness as explained in the next sections.

3.1. Two x Two-bit multiplier

The topology used to train organisms for the two x two-bit multiplier task is shown in Figure 3. We obtain 99.22% average learning for ten organisms, with 100% learning in five cases. It takes less than 10 minutes to learn the task with a Pentium III 800. Each organism has a total of 304 molecules distributed in 12 cells.

3.2. N-input parity test

The N-input parity problem is to find the odd parity of 2^N binary input vectors. The organism must answer a "1" if the input vector contains an odd number of "1" bits, and "0" otherwise. The simple case of N=2 (the exclusive or, or XOR function) cannot be solved with single layer perceptrons [13]. Solving the generic N-input parity problem with genetic algorithms is difficult [11]. Multiple layer neural networks can solve the problem [18, 19], even in the case of N=10 [17].

With the hypernetwork model we obtained the results shown in Table 1. We got up to 100% learning for N=6, N=8, and N=10. The number of epochs was up to 150,000.

N	Average %	Std. Dev.	Minimum %	Maximum %	n runs
6	100	0	100	100(10)	10
8	96.80	3.725	90.63	100(4)	10
10	93.13	7.127	81.25	100(1)	5

Table 1. Results of learning the N-input classification task, N from 6 to 10. The number of organisms that obtain the maximum value of 100% correct classification is shown in parentheses. From [14].

The number of molecules and cells in the organism are shown in Table 2. Organisms that learn the 8-input parity and the 10-input parity task have 236 and 256 molecules respectively.

N	No. I,M,O Cells	No. of Molecules
6	3,6,1	216
8	4,6,1	236
10	5,6,1	256

Table 2. Organism characteristics for different N-input classification tasks. Number of: I=input cells, M=internal cells, O=output cells.

4. Testing the mutation buffering capabilities of the organisms

We wanted to test the robustness of the hypernetwork to structural changes. The changes could be at the molecular, cellular, or organismic levels. In this paper we 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 we 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, we 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 4 to 11 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 we explain the details for each experiment.

4.1. Buffering capabilities of the two x two-bit multiplier organisms

From each of five organisms we generated another twenty, resulting in sub-populations of 100 mutants. Figure 4 shows that in the sub-population where three molecules were mutated randomly, 40 mutants show 100% correct classification. Also, in the in 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 show that about 10% of the mutants perform as well as their parents (100 % correct learning). In contrast, in the sub-population with six randomly denatured molecules we observed that 30% have the same performance as their parents (see Figure 5).

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

4.2. Buffering capabilities of the 6-input parity and 8-input parity organisms

Figures 6 and 7 show the performance of subpopulations obtained from 6-input parity organisms. In the figures we observed 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 of 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 8 and 9).

4.3. Buffering capabilities of the 10-input parity test organism

We tested the buffering capabilities in the organism that was able to learn the complete 10-input parity table. In Figure 10 we see that after a one-molecule mutation, 10 individuals (50%) have the same performance as their ancestor. However, if five molecules or more were mutated, the organisms were very damaged, not showing more than 80% learning.

On the other hand, in the case of molecular denaturation there was a more gradual decay of the performance. It was possible to find 10% of the mutants with at most four denatured molecules (0.78%) behaving like their parents.

5. Discussion

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

- Evolutive potential allows exploration of different paths in the fitness landscape, when environmental conditions change, or internal changes force the organism to have a different behavior.
- Buffering ability allows the organism and its components to survive when some elements of it, like cells, or molecules, are damaged.

5.1. Buffering at the molecular level

At the molecular level mutation buffering is modelled 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 [12].

5.2. Buffering at the organismic level

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

5.3. Evolvable hardware and 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” [6].

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. Evolvable hardware could benefit from mutation buffering capabilities. We argue that, at least in some cases, it is important for a system to perform a 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. While we have not yet attempted to test this proposition, FPGAs seem suited as a hardware platform for the hypernetwork architecture, promising online system evolvability.

6. 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 them to explore efficiently on the fitness landscape. The hypernetwork model of biological information processing could serve as a model for the physical realization of evolvable hardware.

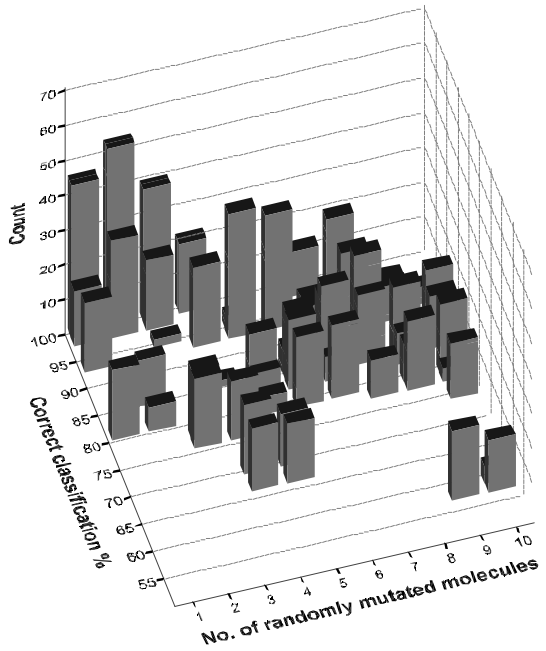

Figure 4. Performance and mutant count of sub-populations generated by random mutation from five two x two-bit multiplier organisms. There are 100 mutants in each sub-population.

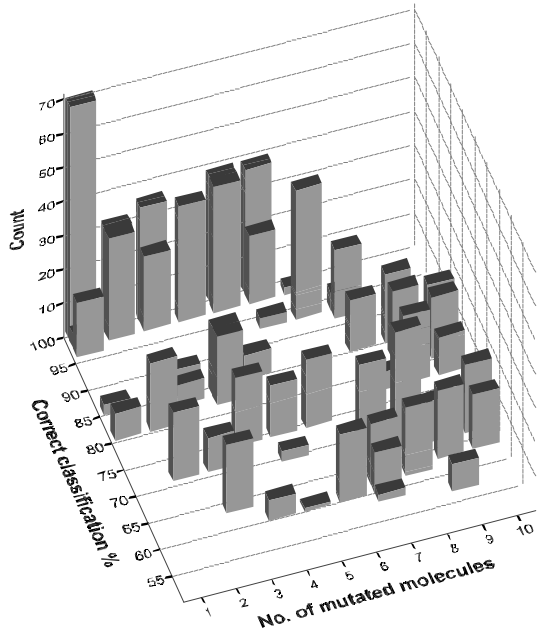

Figure 5. Performance and mutant count of sub-populations generated by molecular denaturation from five two x two-bit multiplier organisms. There are 100 mutants in each sub-population.

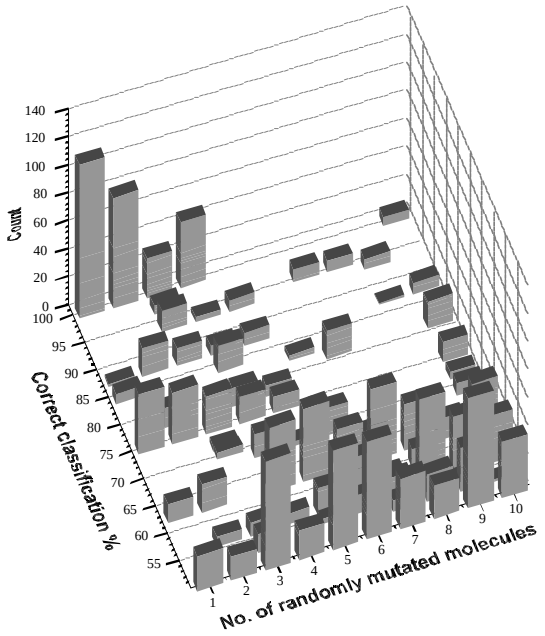

Figure 6. Performance and mutant count of sub-populations generated by random mutation from ten 6-input parity organisms. There are 200 mutants in each sub-population.

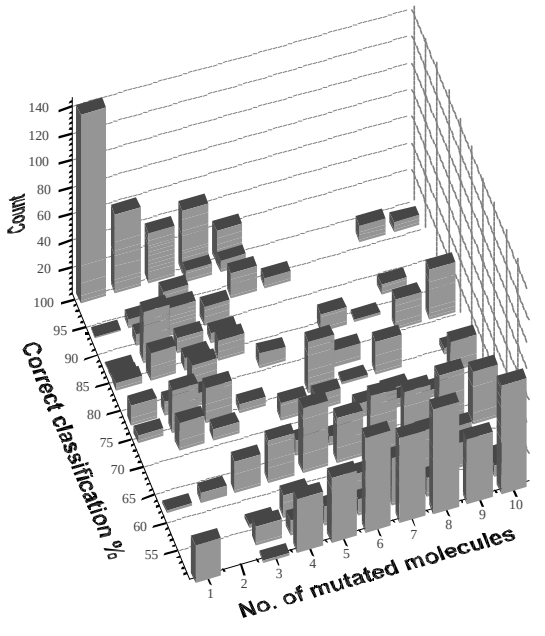

Figure 7. Performance and mutant count of sub-populations generated by molecular denaturation from ten 6-input parity organisms. There are 200 mutants in each sub-population.

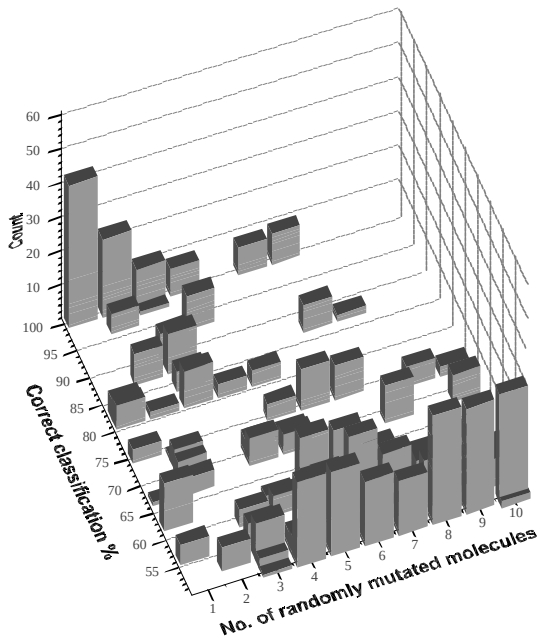

Figure 8. Performance and mutant count of sub-populations generated by random mutation from five 8-input parity organisms. There are 100 mutants in each sub-population.

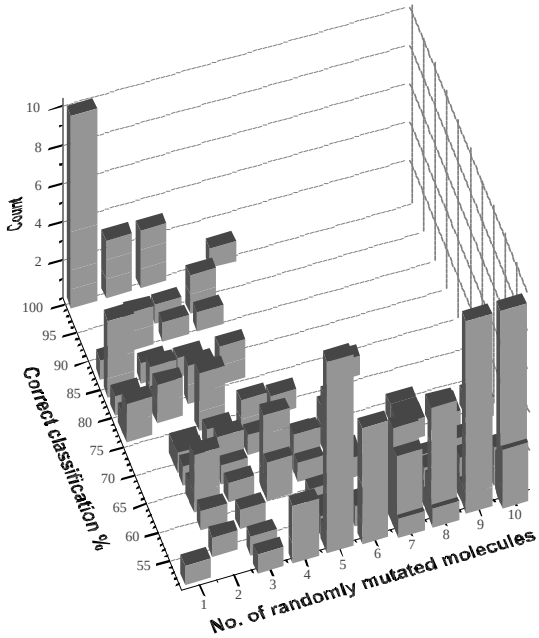

Figure 10. Performance and mutant count of sub-populations generated by random mutation from one 10-input parity organism. There are 20 mutants in each sub-population.

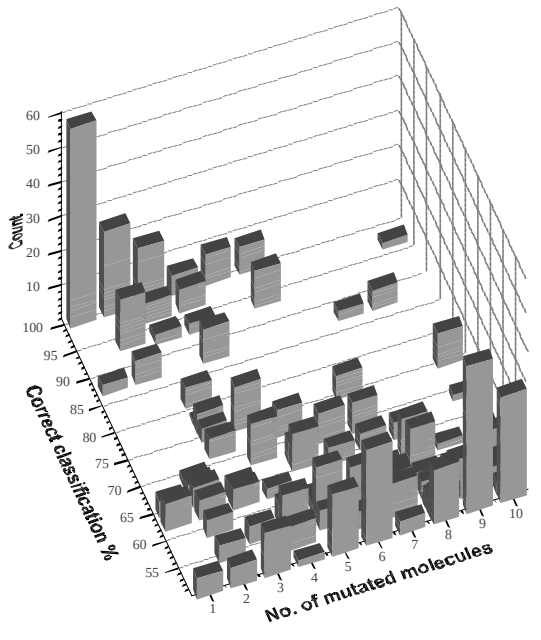

Figure 9. Performance and mutant count of sub-populations generated by molecular denaturation from five 8-input parity organisms. There are 100 mutants in each sub-population.

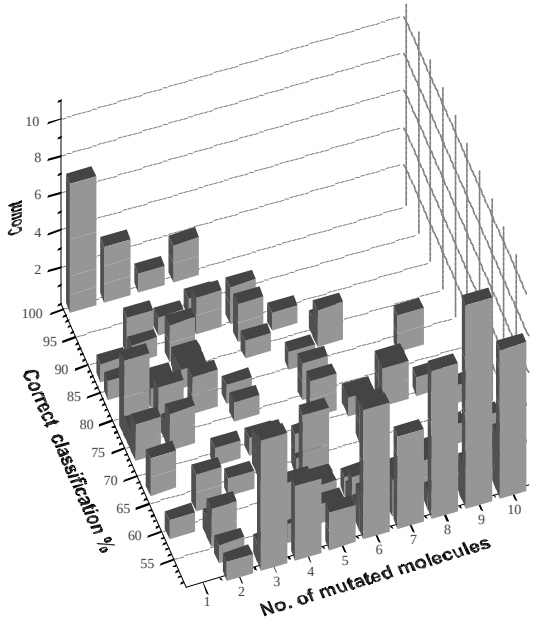

Figure 11. Performance and mutant count of sub-populations generated by molecular denaturation from one 10-input parity organism. There are 20 mutants in each sub-population.

Acknowledgements

This work was supported in part by NASA under grant NCC-2-1189, and in part by NSF under grant ECS-9704190.

References

[1] 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.

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

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

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

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

[6] 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.

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

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

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

[10] 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.

[11] 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.

[12] 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.

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

[14] Segovia-Juárez, J. (2001). *The Hypernetwork Architecture: A Hierarchical Molecular Interaction Model of Biological Information Processing*. PhD thesis, Wayne State University.

[15] Segovia-Juárez, J. and Conrad, M. (2001). Learning with the molecular-based hypernetwork model. In *Proceedings of the 2001 Congress on Evolutionary Computation. To appear.*

[16] 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.

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

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

[19] Waterhouse, S. and Robinson, A. (1994). Classification using hierarchical mixtures of experts. In *Proceedings of the 1994 IEEE Workshop on Neural Networks for Signal Processing IV.*, pages 177–186.

[20] 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.

[21] Zebulim, Ricardo, S., Pacheco, M. A., and Vellasco, M. (1997). Evolvable systems in hardware design: Taxonomy, survey and applications. In Higuchi, T., Iwata, M., and Liu, W., editors, *Evolvable Systems: From Biology to Hardware. Proceedings of the First International Conference, ICES 96. Tsukuba, Japan, October 1996.*, Lecture Notes in Computer Science; Vol. 1259, pages 344–358, Berlin, Heilderberg, New York, Barcelona, Budapest. Springer.