Ubiquity Symposium

The Science In Computer Science

Natural Computation

by Erol Gelenbe

Editor's Introduction

In this twelfth piece of the Ubiquity symposium discussing science in computer science, Erol Gelenbe reviews computation in natural systems, focusing mainly on biology and citing examples of the computation that is inherent in chemistry, natural selection, gene regulatory networks, and neuronal systems. This article originally appeared as part of the "What is Computation" symposium.

Peter Denning
Editor-in-Chief

Ubiquity Symposium

The Science In Computer Science

Natural Computation

by Erol Gelenbe

In his opening statement to the "What is Computation" symposium, Peter Denning underlines the breadth of the concept of computation, which may not require an explicitly defined and specified algorithm, nor an identifiable computer or dedicated physical computational device. So does nature compute, and does computation actually predate its invention, or rather discovery, by human beings If it is the case, then this would actually lend credence to the claim that computer science is actually a science and not just and only a branch of engineering.

When we watch a science fiction movie, we as computer scientists know that most of the images and sound we see and hear are computer generated, and that the movie itself is therefore the result of a complex computation that combines digitized natural video sequences and still shots, with sound and video sequences that are also generated by algorithms embodied in programs that execute on powerful computers, and including yet other computations such as digital compression and decompression techniques for both sound and video [1].

When we listen to a recording of classical music, some of us are aware of the fact that the natural sound recording, in fact the "recordings" since many different takes (as in a film) may actually be used to produce the digital recording of a single piece so as to correct errors made as the artist perform the piece, or improve the interpretation. Thus the recording will typically, have been substantially transformed by digital—i.e. computational—editing, and that the output we are listening to is in fact the outcome of a computational process applied to many naturally generated sound streams.

But when we are sitting in a concert hall, the music that we are listening to and the artists that we watch and listen to while they perform, are coming to each of us individually over a physical medium, via our sensory systems and then our brains which are processing the sound and the "live videos" that reach us. Many scientists would agree that while this happens, our brains are performing computations in which the understanding and appreciation of the music itself is only a small part of the overall natural computation that each of our brains need to individually undertake, based on other sensory inputs as well as emotions.

From the earliest days of automatic computation, the analogy between the animal brain and computational devices has been discussed and exploited. Thus we in the computer science profession have long recognized that natural computation exists. This assumption has had significant consequences for the development of mathematical models of computation, such as neural networks [2], which are directly inspired from what we know— and we should perhaps say the little that we know' about animal brains. This work has now given rise to substantial applications, especially in pattern recognition and image processing [3] and the resulting techniques are very widely used.

However natural computation is much broader than neuronal computation, and this brief paper will try to illustrate both its breadth and its depth although in a few pages it is impossible to go over all of the different ideas and analogies that have been proposed between natural and artificial computation [4, 5].

Examples of Natural Computation

Going further along the path of nature, suppose that we have a detailed mathematical model of some physical process such as—say—a chemical reaction; clearly we can either organize the reaction in the laboratory and observe the outcome, or we can set up the mathematical model of the reaction on a computer either as the numerical solution of a system of equations, or as a Monte Carlo simulation, and we can then observe the outcome. We can all agree that when we "run the reaction" on the computer either as a numerical solution or a Monte Carlo simulation, we are dealing with a computation.

But why then not also consider that the laboratory experiment itself is after all only a "computational analog" of the numerical computer experiment! In fact, the laboratory experiment will be a mixed analogue and digital phenomenon because of the actual discrete number of molecules involved, even though we may not know their number exactly. In this case, the "hardware" used for the computation are the molecules and the physical environment that they are placed in, while the software is also inscribed in the different molecules species that are involved in the reaction, via their propensities to react with each other [6].

There will be "errors" in the outcome of this natural computation even in a laboratory setting, because of errors and imprecision in the initial conditions (e.g. number of molecules of each species that are involved), imprecision in other relevant physical conditions such as the temperature and the presence of impurities, and imprecision in the measurement of the outcome. Similarly, in the computer based computation using a mathematical model, we will have lack of precision due to model simplifications, convergence rates, and numerical errors; if we use a Monte Carlo simulation, some of the previous factors as well as issues of statistical convergence will lead to errors in the outcome.

In many instances of natural computation, research on the physical or biological computational process itself has given rise to algorithmic procedures that mimic these natural computations. Evolution itself can be viewed as a computational process whose outcome is to seek to continually optimize a living species to the conditions of their environment and to other species that share the same environment. The computational processes of evolution were recognized early on by scientists working at the interface between biology and computation [7, 8] leading to the field of evolutionary computation where existing strings of letters, representing DNA, can combine to form new strings, and small random changes in strings are allowed so as to model mutation. The effect of natural selection is then represented by a "fitness operator," which will eliminate strings, which appear to be less successful.

Another computational model that has emerged from studies of natural computation are Gene Regulatory Networks (GRN) [9], which represent the manner in which a collection of DNA segments interact with each other and with other substances in a cell via their RNA; the outcome controls the rate at which genes are transcribed into mRNA, which in turn will make a specific protein or set of proteins, which can serve to provide structural properties to the cell,

or to act as an enzyme to catalyze certain chemical reactions, or simply to activate other genes which act as transcription factors that can activate or inhibit other genes. These GRNs can also respond to conditions, which are external to the cell so as to improve its survival or its ability to reproduce. For cells with identical genomes, the precise temporal state of the gene regulatory network can differ so as to represent different stages of development of the cell or of the organism to which the cells belong. Formal models of GRNs are not dissimilar from models of neural networks since they constitute interconnected systems of activation (excitation) and inhibition.

Similar questions can be raised when we consider some of the foundations of the physical world based on quantum mechanics and its artificial counterpart that would use a quantum system to conduct artificial computations in [10, 11], and how such a machine would be able to act as a simulator for quantum physics. In principle, the superposition principle will allow quantum computers to solve NP-complete problem by "squeezing" information of exponential complexity into polynomially many quantum states, the real problem may then be to find ways to retrieve this information efficiently.

Desirable Properties of a System for Natural Computation

These different examples, and the experience that has spanned over half a century of research on these questions, can give provide us with some insight about what the desirable properties of such natural computational systems may be. These desirable properties would include: simplicity in the basic principles that are used, malleability to many different possible uses in nature without requiring some form of reprogramming, and the ability to adapt so that the same system can progressively respond to different needs with relatively small parameter changes. In order to illustrate these ideas we will focus on neuronal networks, which combine the complexity in their usage both by nature and in engineering applications, and greater maturity in our understanding of their underlying biological and computational principles.

There is an interesting analogy between a neuronal model that is used as an associative memory [2] and a quantum system [10]. The neuronal system can store many different patterns simultaneously in a large machine that has a single well defined set of parameters, and each of the individual patterns, or an approximate rendition of each pattern, can be provided as

the output, in response to a limited cue or prompt that is provided as input. This is similar to a quantum system that reveals one of its possible states at the time that a measurement is made. Another interesting observation first developed by J. Hopfield, arises when we consider the impressive success of neuronal models as fast approximate solvers of NP-hard problems [12] based on the fact the a neuronal network can encode a very large number of problem solutions simultaneously and that its internal state will stochastically gravitate towards lower cost instances of the problem. A neuronal system with N cells, each of which has say M states, can be used to store as many as MN patterns, and even more if one considers that it can represent a probability distribution over these MN states. Since all the details of the internal state of a natural neuron can only be represented by perhaps several continuous variables, the number of states that such a system may store can indeed be even greater. Models of GRNs capture some of the features of neuronal models, so that many of the things that one may say about neuronal models can also carry to GRNs.

A desirable feature of natural computational systems is that they should capture subtleties of different forms of representation. This is formally represented by an ability to approximate the input-output represented by certain classes of mathematical functions. Turing computability is often cited as a desirable property for computational models. In the case of neuronal models [13], and hence also for certain GRN models, a fundamental property of interest is their ability to approximate arbitrarily closely the class of bounded and continuous functions [14]. Thus for any multi- dimensional bounded and continuous function that may represent, for instance the manner in which a neuronal system responds to some sensory pattern to provide motor controls, there will be a neuronal system of bounded size that can approximate this behavior in a well defined manner. As the size of the neuronal system in—say—the number of cells grows, then the precision of the approximation can become better.

Another important feature of neuronal systems is their ability to adapt based on ongoing experience. This allows a neuronal system to better perform its frequently performed activities, to refresh its ability to perform a given activity, and also to retrain itself to take up new functions. In computer science terminology, this can also be viewed as a form of reprogramming based on actual usage, rather than based on a priori design. In neuronal systems this property is called “learning.”

For neuronal systems, many different views of how learning is actually carried out have been advanced. Some, such as Hebbian learning and synaptic plasticity, are well founded in biology. Others, such as back-propagation and gradient decent learning [2, 3], are still the subject of discussion although one knows for the mammalian brain that the "forward" flow of signaling that back-propagation requires can be carried out using the neuronal spiking activity. Although much slower, a backward flow of signaling along the paths that carry the forward flow of spikes, in response to the spiking activity has been identified at the biochemical level [15], and this flow may explain some of the slower learning processes that we are so familiar with in the activities of the mammalian brain.

Conclusions

In this brief paper, we have reviewed computation in natural systems, focusing mainly on biology, and have touched on the computation that is inherent in chemistry, natural selection, gene regulatory networks and neuronal systems. We have tried to point out to analogies in each case with artificial computation and have discussed some of the superficial relationships between such systems. The mathematical models used to represent these different forms of natural computation are actually quite similar and are based on discrete mathematics and probability theory.

Most of the topics we have touched upon are still the subject of research as indicated by the relatively recent dating of most of the references that we cite. We therefore hope that these few lines will motivate further research and discussion both across disciplines, and within computer science.

References

[1] E. Gelenbe, M. Sungur, C. Cramer, P. Gelenbe. Traffic and video quality in adaptive neural compression. *Multimedia System* 4, 6 (1996), 357-369.

[2] D. H. Ballard. *An Introduction to Natural Computation*. MIT Press, Cambridge, 1997.

[3] E. Gelenbe and K. Hussain. Learning in the multiple class random neural network. *IEEE Transactions on Neural Networks* 13, 6 (2002), 1257-1267.

[4] C. Calude and Gh. Paun. *Computing with Cells and Atoms*. Taylor and Francis, London, 2000.

[5] Gh. Paun. Membrane computing: History and brief introduction. In *Fundamental Concepts in Computer Science*, E. Gelenbe, J.-P. Kahane, eds., 17-41, Imperial College Press, London and Singapore, 2009.

[6] E. Gelenbe Network of interacting synthetic molecules in equilibrium. In *Proc. Royal Society* A 464, 2096 (2008), 2219-2228.

[7] N.A. Barricelli. Symbiogenetic evolution processes realized by artificial methods. *Methodos* 9, 35-36 (1957). 143–182.

[8] A. Fraser. Simulation of genetic systems by automatic digital computers. *Aust. J. Biol. Sci*. 13, 2 (1957).

[9] E. Gelenbe. Steady-state solution of probabilistic gene regulatory networks. *Phys. Rev. E.* 76, 1 (2007), 031903.

[10] R. Feynman. Simulating physics with computers. *International Journal of Theoretical Physics* 21, 6-7 (1982), 467–488.

[11] V. A. Adamyan, C. Calude and B. S. Pavlov. Transcending the limits of Turing computability. In T. Hida, K. Saitô, S. Si (ed.). *Quantum Information Complexity*. Proceedings of Meijo Winter School 2003. World Scientific, Singapore, 2004, 119-137.

[12] E. Gelenbe, A. Ghanwani, V. Srinivasan. Improved neural heuristics for multicast routing. *IEEE Journal of Selected Areas of Communications* 15, 2 (1997), 147-155.

[13] E. Gelenbe. Stability of the random neural network model. *Neural Computation* 2, 2 (1990), 239-247.

[14] E. Gelenbe, Z.H. Mao, and Y.D. Li. Function approximation by random neural networks with a bounded number of layers. *Journal of Differential Equations and Dynamical Systems* 12, 1-2 (2004), 143-170.

[15] K.D. Harris. Stability of the fittest: organizing learning with retroaxonal signals. *Trends in Neurosciences* 31, 3 (2008), 130-136.

About the Author

Erol Gelenbe (e.gelenbe@imperial.ac.uk) is the Dennis Gabor Chair Professorship in the Electrical and Electronic Engineering

DOI: 10.1145/2576894.2576895

Reprinted with Permission from* Ubiquity *February 2011
DOI: 10.1145/1940721.1940722

