Bio-Inspired Computing Paradigms (Natural Computing)

Gheorghe Păun

Institute of Mathematics of the Romanian Academy
PO Box 1-764, 7014700 Bucureşti, Romania, and
Research Group on Natural Computing
Department of Computer Science and Artificial Intelligence
University of Sevilla
Avda. Reina Mercedes s/n, 41012 Sevilla, Spain
E-mail: george.paun@imar.ro, gpaun@us.es

Abstract. This is just a glimpse to the fruitful and constant preoccupation of computer science to try to get inspired by biology, at various levels. Besides briefly discussing the main areas of natural computing (genetic algorithms–evolutionary computing, neural computing, DNA computing, and membrane computing), we mention some of the hopes and the difficulties/limits of this enterprise.

1 From Turing to Present Days

In some sense, the whole history of computer science is the history of a series of continuous attempts to discover, study, and, if possible, implement computing ideas, models, paradigms from the way nature – the humans included – computes. We do not enter here into the debate whether or not the processes taking place in nature are by themselves "computations", or we, *homo sapiens*, interpret them as computations, but we just recall the fact that when defining the computing model which is known now as *Turing machine* and which provides the standard by now definition of what is computable, A. Turing (in 1935–1936) explicitly wanted to abstract and model what a clerk in a bank is doing when computing with numbers. One decade later, McCullock, Pitts, Kleene founded the finite automata theory starting from modelling the neuron and the neural nets; still later, this led to the area called now *neural computing* – whose roots can be found in unpublished papers of the same A. Turing (see also Section 3 below). *Genetic algorithms* and evolutionary computing/programming are now well established (and much applied practically) areas of computer science. One decade ago, the history making Adleman's experiment of computing with DNA molecules was reported, proving that one can not only get inspired from biology for designing computers and algorithms for electronic computers, but one can also use a biological support (a bio-ware) for computing. In the last years, the search of computing ideas/models/paradigms in biology, in general in nature, became explicit and systematic under the general name of *natural computing*[1].

[1] As a proof of the popularity of this syntagm, it is of interest to point out that there are conferences with this topic explicitly included in their scope, a new journal with this name

This trend of computer science is not singular, many other areas of science and technology are scrutinizing biology in the hope (confirmed in many cases) that life has polished for billions of years numerous wonderful processes, tools, and machineries which can be imitated in domains apparently far from biology, such as materials and sensor technology, robotics, bionics, nanotechnology.

2 A Typical Case: Evolutionary Computing

In order to see the (sometimes unexpected) benefits we can have in this framework, it is instructive to examine the case of genetic algorithms. Roughly speaking, they try to imitate the bio-evolution in solving optimization problems: the space of candidate solutions for a problem are encoded as "chromosomes" (strings of abstract symbols), which are evolved by means of cross-overing and point mutation operations, and selected from a generation to the next one by means of a fittness mapping; the trials to improve the fitness mapping continue until either no essential improvement is done for a number of steps, or until a given number of iterations are performed. The biological metaphors are numerous and obvious. What is not obvious (from a mathematical point of view) is why such a brute force approach – searching randomly the space of candidate solutions, with the search guided by random cross-overings and point mutations – is as successful as it happens to be (with a high probability, in many cases, the Genetic Algorithms provide a good enough solution in a large number of applications). The most convincing "explanation" is probably "because nature has used the same strategy in improving species". This kind of bio-mystical "explanation" provides a rather optimistic motivation for related researches.

3 Neural Computing

A special mentioning deserves another "classic" area included nowadays in natural computing, namely neural computing. In short, the challenge is now to learn something useful from the brain organization, from the way the neurons are linked; the standard model consists of neuron-like computing agents (finite state machines, of very reduced capabilities), placed in the vertices of a net, with numerical weights on edges, aiming to compute a function; in a first phase, the net is "trained" for the task to carry out, and the weights are adjusted, then the net is used for solving a real problem. Pattern recognition problems are typical to be addressed via neural nets. The successes (and the promises) are comparable with those of genetic algorithms, without having a similarly wide range of applications. However, the brain remains such a misterious and efficient machinery that nobody can underestimate the progresses in any area trying to imitate the

is published by Kluwer, a new series of the Elsevier *Theoretical Computer Science* journal is devoted to natural computing, a new series of books published by Springer-Verlag and a column in the *Bulletin of the European Association for Theoretical Computer Science* also have this name.

brain. (It also deserves to mention the rather interesting detail that Alan Turing himself, some years after introducing Turing machines, had a paper where he proposed a computing device in the form of a net of very simple computing units, able to learn, and then to solve an optimization problem – nothing else than neural computing *avant la lettre*. Unfortunately, his paper remained unpublished and was only recently reevaluated; see `http://www.AlanTuring.net` and [12] for details.)

4 DNA Computing

Coming back to the history making Adleman's experiment mentioned above [1], it has the merit of opening (actually, confirming, because speculations about using DNA as a support for computations were made since several decades, while theoretical computing models inspired from the DNA structure and operations were already proposed in eighties, see, e.g., [7]) a completely new research vista, not looking for better algorithms for existing computers, but for a new type of hardware, based on bio-molecules. Specifically, Adleman has solved in a lab, just handling DNA by techniques already standard in bio-chemistry, a computationally hard problem, the well-known Hamiltonian Path problem. The problem is **NP**-complete, among those considered intractable for the usual computers, but Aldeman has solved it in linear time (the number of lab operations carried out was linear in terms of the number of nodes). The graph used in the experiment had only 7 nodes, a toy-problem by all means, while the actual working time was of seven days, but the *demo* (in terms of [6]) was convincing: we can compute using DNA!

It is worth emphasizing the fundamental novelty of this event: the dream is to find an essentially new type of computers – sometimes called "wet computer". The great promise is to solve hard problems in a feasible time, by making use of the massive parallelism made possible by the very compact way of storing information on DNA molecules (bits at the molecular level, with some orders of efficiency over silicon supports). In this way, billions of "computing chips" can be accommodated in a tiny test tube, much more than on silicon. The possible (not yet very probable for the near future...) "DNA computer" also has other attractive features: energetical efficiency, reversibility, evolvability.

5 The Marvelous DNA Molecule

For the practical computer science, DNA computing fuels several hopes, mainly related to the massive parallelism mentioned above; on this basis, we can simulate non-determinism (which is anyway present in biochemistry), so that one can address in this framework computationally hard problems, with the possibility to push with some steps the feasibility barriers – at least for certain problems.

There are mentioned also other good features of DNA as a support for computations (energy efficiency, stability, reversibility of certain processes), but we

switch here to a purely theoretical observation, which is simply spectacular from a general computability point of view: in certain sense, *all Turing computable languages are "hidden" in the DNA molecules, and any particular language can be "read off" from this blue print of computability by the simplest transducer, the finite state one!*.

This newspaper-style statement has a precise mathematical counterpart, first mentioned in [11]. Everything starts with an old characterization of recursively enumerable (RE) languages, as the projection of the intersection of a twin-shuffle language with a regular language. However, both the projection and the intersection with a regular language, and the decodification of the symbols of an arbitrary alphabet from codes over a binary alphabet can be computed by a sequential transducer. Therefore, every RE language is the image through a sequential transducer of the twin-shuffle language over the alphabet with two symbols. Now, a clever observation from [11] relates the twin-shuffle language over two symbols with "readings" of DNA molecules (one goes along the two strands of a molecule, step by step but with non-deterministically varying speed, and producing a single string, by interleaving the visited nucleotides; this reading can be done either started from the same end of a double stranded molecule, or from opposite ends, for instance, according to the directionality of the two strands). Thus: every RE language can be obtained through a finite state transducer from the pool of readings of DNA molecules! The double stranded data structure, with the corresponding nucleotides related by the complementarity relation, is intrinsically universal from a computational point of view!

This observation (a presentation and variants of the mathematical details can also be found in [10]) should bring to theoretical DNA computing a similar degree of optimism as genetic algorithms bring to practical natural computing.

6 Recent Attempts

Another component of this general intellectual enterprise is membrane computing, which starts from the observation that the cell is the smallest living thing, and at the same time it is a marvellous tiny machinery, with a complex structure, an intricate inner activity, and an exquisite relationship with its environment – the neighboring cells included. Then, the challenge is to find in the structure and the functioning of the cell those elements useful for computing. Distribution, parallelism, non-determinism, decentralization, (non)synchronization, coordination, communication, robustness, scalability, are only a few keywords related to this challenge. For instance, a problem which cannot be easily solved in terms of silicon engineering, but which was misteriously and very efficiently solved by nature at the level of the cell is related to the coordination of processes, the control pathways which keep the cell alive, without a high cost of coordination (in parallel computing the communication complexity is sometimes higher than the time and space complexity). Then, interesting questions appear in connection with the organization of cells into tissues, and this is also related to the way the neurons cooperate in the brain.

Similar issues are addressed by several other recent research directions belonging to natural computing, for instance, trying to learn computing ideas/models/paradigms from the way certain colonies of insects are organized and work together, the way bacteria populations develop in a given environment, the way flocks of birds maintain their "organization", the (amazing) way ciliates unscramble their chromosomes after reproduction [5], and so on. Most of these areas still wait for producing a *demo*, many of them are still in the stage of "craftsmanship", with *ad-hoc* ideas involved in *ad-hoc* models/tools handling *ad-hoc* problems, but the whole approach is both intellectually appealling and practically promising (sometimes through "by-products", useful for biology, medicine, robotics, etc).

7 Hopes and Limits

As mentioned above, there are many convincing achievements of natural computing, many bio-inspired areas of computer science have important practical applications, or/and they are appealing from a theoretical point of view. Sometimes, the usefulness of the bio-inspired models and tools has a somewhat misterious source/explanation, in other cases the matter is simpler and more transparent. Anyway, we try here to compose a list of attractive features of this attempt, of learning from the living nature to the benefit of computer science (most of these features can be called "hopes", as not being confirmed by current natural computing): in many cases, we look for ideas for improving the use of the existing computers, for new types of algorithms; in other cases, a new hardware is sought for; as new ideas to be found in nature, we can learn new data structures (such as the double strand with complementary pairs of symbols), or new operations (crossovering and point mutations, splicing, annealing, and so on and so forth); bio-computing can make available a massive parallelism, reversible computations, non-determinism, energy efficiency, maybe also evolvable hardware/software, self-healing, robust; new ideas learnt from biology can lead to a complete reconstruction of computability theory, on non-standard bases (e.g., using the splicing operation, quite different from the rewriting operation, which is standard in computability); nature can suggest new computer architectures, ways to cope with such difficulties of parallel computing as communication, (de)centralization, synchronization, controlling distributed processes, etc.

The list might be probably continued, but we want to make here a point which we find important: when discussing about new computing paradigms inspired from biology, most authors are enthusiastic or even over-enthusiastic. For promoting a young research area, this is understandable – but natural computing is no longer a young area. A more lucid position is similarly helpful as a blindly optimistic one, so that we balance here the previous list with another one, of difficulties of implementing bio-ideas in computer science: nature has (in certain sense, unlimited) time and resources, nature is cruel, kills what is not fit (all these are difficult to incorporate in computers, let them be based on electronic hardware or on a hypothetic bio-ware); nature has other goals than computing;

many bio-chemical processes have a degree of non-determinism which we cannot afford in our computations; the life processes are complex, with a high degree of redundancy; biology seems to deal with non-crisp mathematics, with probabilities, with fuzzy estimations, which are not fully manageable in computations. And, last but not least, maybe we dream too much even from a theoretical point of view. First, the space-time trade-off specific to molecular computing, cannot redefine complexity classes, and it is sometimes too costly in space (in the size of used bio-ware). Then, M. Conrad [4] warned us that programmability (universality), efficiency, and evolvability are three contradictory features of any computing model... Both these observations indicate that there is no free lunch in computer science, even in the bio-inspired one.

References

1. L.M. Adleman, Molecular Computation of Solutions to Combinatorial Problems. *Science*, 226 (November 1994), 1021–1024.
2. B. Alberts, A. Johnson, J. Lewis, M. Raff, K. Roberts, P. Walter, *Molecular Biology of the Cell*, 4th ed. Garland Science, New York, 2002.
3. J.A. Anderson, *An Introduction to Neural Networks*. The MIT Press, Cambridge, MA, 1996.
4. M. Conrad, The Price of Programmability. In *The Universal Turing Machine: A Half-Century Survey* (R. Herken, ed.), Kammerer and Unverzagt, Hamburg, 1988, 285–307.
5. A. Ehrenfeucht, T. Harju, I. Petre, D.M. Prescott, G. Rozenberg, *Computations in Living Cells*. Springer-Verlag, Berlin, 2004.
6. J. Hartmanis, About the Nature of Computer Science. *Bulletin of the EATCS*, 53 (June 1994), 170–190.
7. T. Head, Formal Language Theory and DNA: An Analysis of the Generative Capacity of Specific Recombinant Behaviors. *Bulletin of Mathematical Biology*, 49 (1987), 737–759.
8. J.H. Koza, J.P. Rice, *Genetic Algorithms: The Movie*. MIT Press, Cambridge, Mass., 1992.
9. Gh. Păun, *Computing with Membranes: An Introduction*. Springer-Verlag, Berlin, 2002.
10. Gh. Păun, G. Rozenberg, A. Salomaa, *DNA Computing. New Computing Paradigms*. Springer-Verlag, Berlin, 1998.
11. G. Rozenberg, A. Salomaa, Watson-Crick Complementarity, Universal Computations, and Genetic Engineering. *Techn. Report* 96–28, Department of Computer Science, Leiden Univ., Oct. 1996.
12. C. Teuscher, *Alan Turing. Life and Legacy of a Great Thinker*. Springer-Verlag, Berlin, 2003.

This article was processed using the LaTeX macro package with LLNCS style