Teaching Natural Computation

Colin G. Johnson
Computing Laboratory
University of Kent
Canterbury, Kent, CT2 7NF, England
Email: C.G.Johnson@kent.ac.uk

Published in *IEEE Computational Intelligence*, 4(1):24–30, 2009

Abstract

This paper consists of a discussion of the potential impact on computer science education of regarding computation as a property of the natural world, rather than just a property of artifacts specifically created for the purpose of computing. Such a perspective is becoming increasingly important: new computing paradigms based on the natural computational properties of the world are being created, scientific questions are being answered using computational ideas, and philosophical debates on the nature of computation are being formed. This paper discusses how computing education might react to these developments, goes on to discuss how these ideas can help to define computer science as a discipline, and reflects on our experience at Kent in teaching these subjects.

Keywords: Computer science education, natural computation, nature-inspired computing, university curriculum

Introduction

What is the object of study in the university discipline of *computer science*? A typical answer to this question would be that it is the study of *computers*, machines created by humans for the purpose of computing. However, an increasingly common contrasting perspective is to regard computation as a property of objects in the natural world, or of the organization of such objects. That is, we regard computational ideas as a set of tools for scientific reasoning, as well as a way of building machines.

These ideas are important for a number of reasons. Firstly, an increasing amount of work in computer science (and other sciences) is either concerned with understanding the world by using *concepts* first constructed in computer science, or else using a computational perspective on the natural world to inspire novel ways of carry out computations. If we are not aware of the scope of ideas which computer science might encompass then we are in danger of teaching a limited kind of computer science in universities. Secondly, this provides the beginnings of an answer to the ongoing debate about whether computer science is a proper science, or whether it is a branch of engineering, mathematics, or whatever. If we are to teach computer science then we need to be aware of the possible alternative ways in which the idea of computation can be made meaningful.

The remainder of the paper splits into three main sections. The first section argues why the perspective that computing is a property of the world is a valuable idea, and how these issues interact with the main computer science curriculum. The second section discusses a couple of particular points, whilst the third reflects on our experience of running courses on these topics.

Computing in the World

Where in the world do we find "computation"? A reasonable initial answer is that we see it in machines built for the purpose: computers. Nonetheless when we think about this for a moment, that conclusion seems inadequate. An easy objection is that computers exist merely to speed up the process of computation, and to make complex computations do-able on a tractable scale. Remember that before the word "computer" meant an electronic device, it was a job title; a "computer" was a person who carried out a sequence of routine calculations using a mechanical calculating machine. As computational ways of thinking [34] become more embedded in the background of scientists, computational concepts become part of the vocabulary and mental toolset across the sciences, as well as computers themselves being tools for science. As Denning [6] puts it:"Computation and information processes have been discovered in the deep structures of many fields... scientists now see information processes abundantly in nature".

In essence it would seem to be the transformation of the information, not the fact that it happens on electronic machines, which lies at the heart of the computational process. The phrase "information processing" captures this rather well. Nonetheless there is another question to be answered—is the *intention* to carry out the information processing an important part of it or not? If a non-conscious biological system carries out some transformation of its environment to increase its chance of survival, can that be regarded as "doing a computation"? Why is this process different from a traditional computing machine doing the computation—is the conscious intention to start the process and observe the answer a significant distinction?

We can usefully stop this argument at a number of points. We could regard this broader application of the word "computation" simply as a pragmatic tool which allows us to reason about processes which we see in the world and use these as inspirations for novel computational techniques. At another extreme we can consider that that computational concepts, such as information content, should be regarded as having equal status with physical concepts such as mass and energy [30, 31]. This is an interesting debate, and it is indicative of the lack of concern for this type of question in computer science that we have no framework for discussing these ideas.

In the remainder of this section I would like to unpack this idea, and look at a number of reasons (practical, scientific and philosophical) why we might care about this argument. The potential place of these in the computing curriculum will be discussed.

Pragmatic Reasons

In recent years a number of novel paradigms for computation have been proposed which take as their inspiration processes which are observed in the natural world. Examples are genetic algorithms [23], genetic programming [20] and other aspects of the broad field of evolutionary computation; simulated annealing [19]; tabu search [11, 12]; ant colony optimization [8, 7]; and artificial neural networks [22]. In each of these a phenomenon has been observed in the natural world which solves a complex problem, and the algorithmic heart of this phenomenon has been abstracted and implemented on a computer. This has provided a rich source of inspiration for computational problem-solving, however in order to carry out this kind of reasoning about *computational metaphors* it is necessary to have an outlook on the world which looks at phenomena from biology, chemistry, physics, et cetera with a computational perspective [24, 25]. An overview of the state of the art in these areas has been given by Kari&Rozenberg [16].

Traditional computer science education fails to provide this perspective. The idea that we should look at the world with the intention of seeing what information processing goes on in it is a powerful yet neglected idea in computing. The one place where it is sometimes found is in the study of information systems or "systems analysis". In this area of study students are sometimes sent out into the world to observe the information which is exchanged and transformed in the process of human interactions within the world. However the intention of this study is different from what we have discussed above. In this context, these observations are done from the perspective of *simulating* or *supporting* this information-flow process. In contrast, the ideas above were developed by looking at processes and *abstracting* from the the way in which they generate complexity, with the aim of using this to inspire novel computing techniques.

If we want students to engage on the natural science/computer science interface, is also not sufficient for computing students simply to study science subjects as part of a broad first year or two of a degree. The point of using the computational perspective on the world is to look at the natural sciences using a computational perspective. Whilst some scientific knowledge is necessary in order to gain scientific inspiration for computing, simply studying both science and computation in parallel and hoping that ideas will move between the two is not the solution. As well as just studying the two subjects, the student needs to be shown examples of how natural science has inspired computation, and to apply the ideas of computational thinking to the science that they have studied.

Another way in which the computational capacity of the natural world is used is to create novel computing paradigms is in the creation of computational systems which make direct use of a property of the world which allows us to do things which cannot be done (easily, or at all) using traditional computing hardware.

Perhaps the most well developed of such systems is *quantum computing* [9, 28, 29, 33]. This makes use of the ability of quantum systems to exist in a superposition of states. This superposition is used to represent and process many more data items in a certain amount of physical matter than is possible with classical computing techniques. It is informative to look back at the origins of quantum computation. The core idea was invented by Feynman [10] as a result of thinking about what it might mean to simulate a quantum process on a computer. He realized that certain processes which happen on a tractable timescale in real world take intractable amounts of time when simulated on classical computers. The leap of the imagination

which led to quantum computing was to realize that this could be interpreted to mean that our conventional view of computing was limited, and that computers which were built using different properties of the world could achieve more things than were possible on classical computers.

In more recent years it has been realized that a number of other materials in the world can make interesting new substrates for the construction of computational devices. Long-chain molecules (such as DNA) have been used to represent information in comptuational problems, with enzyme action used to carry out computation on this information [5, 26, 3]. Because many millions of such molecules can be contained and processed in a small space, it is possible to carry out massively parallel computations on the information represented in these molecules. Another advance is in the application of interactions between diffusing liquids to solve geometric problems [1, 2].

Again, it is insufficient simply to study science alongside computing and believe that ideas will transfer across. In addition to studying science as a description of the natural world it is important to get students to look at natural systems which they have studied and ask questions like the following:

- How does the system process information about its world?
- How does the system represent the information which it needs to function in the world?
- Are we able to initialize that information to a desired state or set of states?
- How does information within the system change?
- Are we able to affect those changes ourselves?
- Does this combination of representation and process allow us to do things which we cannot do (either practically or at all) with conventional computing systems?

One particular difficulty (which we have personally experienced) is that many computer science students have not studied pure science subjects, or relevant mathematics, to a very high level[1]. This means that it is difficult to teach such such topics. Most obviously, this is for practical reasons: for example it is difficult to get anywhere with a technical understanding of quantum computing if even the basic mathematical language of vectors and matrices has not been covered earlier in the course. More importantly, however, is the lack of appreciation of why these ideas are important: if students already have some idea of quantum mechanics or DNA before studying the application of them to computing, then there is a sense of amazement at how these ideas can be applied to computation. If they encounter them in a just-in-time fashion as part of a computing course, they can seem mundane.

Scientific Reasons

The previous section considered how a computational stance towards the natural world could help us in finding inspiration or technology for the creation of novel computing paradigms inspired by computational phenomena in the world. A second reason for taking this stance is to consider whether concepts from computer science can help us to understand the natural world.

Clearly computers can be used as tools for scientific research. This is not the topic of interest in this paper. Instead consider the role that computational concepts could play in understanding scientific phenomena. It is commonly assumed that ideas in computer science have no relevance beyond the production of computing devices:

> *"Hypothesising how long an algorithm will take and testing that hypothesis is sharpening our computing tools, not testing a scientific hypothesis which would tell us more about the real world."* [14]

However, this idea rests on the assumption that computing is only to be found in artifacts specifically constructed for the purpose. If we take the stance that computation is found elsewhere in the world, then a computational result has implications for our understanding of a wide range of phenomena.

One immediate way in which computational and informational issues impact upon our understanding of the world is by providing constraints on what is possible in the world. How fast can a brain (regarded as a physical system) process a certain amount of information? How fast can the immune system learn to recognize a new kind of antigen? These sorts of questions would seem to be more about the way in which the physical structures are *organized* (i.e. the information they encode and process) rather than because of

[1] This may be particularly acute in the English education system, which has high levels of subject specialization from the age of 16.

anything to do with what they are made of. Change the way in which the information is represented and processed and the answers to the above questions might change in turn.

Consider the old saw that "we only use 10% of our brains". What sort of statement is this? Could we take the brain and cut out the 90% that we don't use? Is this an average over time? What is the other 90% doing? What would it be like to be using 100% of our brains? Would we be able to think 10 times at quickly? Store 10 times as much stuff? Is it meaningful to think of memory in such linear terms anyway? Why should there be a limit on the ultimate capability anyway?

If we want to answer these questions in a meaningful way it is important to get away from the idea of this being a "physical" 10% and approach the question differently. One approach is *computational*; in a computational context questions about memory structure and speed of computation become formalizable. The extent to which this is useful depends on the extent to which the brain is a computational system.

Another related issue is how biological systems exploit efficient computational structures which exist *in potentio* in the world. Consider the following statement by Kauffman [17]:

> *"We all know that oil droplets in water manage to be spherical without the benefit of natural selection and that snowflakes assume their evanescent sixfold symmetry for spare physiochemical reasons."*

Or, consider the role of the genetic code in creating structures in the world. A common view is that the DNA acts as the blueprint for the organism produced, i.e. it acts like a computer program running on a fixed computer. However the situation is more complex; a more sophisticated view is that the DNA is specifying the parameters for certain natural structures. These ideas have interesting applications both for computer science and for our understanding of the natural world.

These are not the sort of issues which are usually tackled in learning computer science. That said, neither are they the sort of issues which are tackled by scientists in biology, chemistry, physics, et cetera. Nonetheless such considerations provide a powerful way of tackling questions about the world, and so far scientists working in these areas have simply stumbled into the area by a mixture of curiosity and serendipity.

Where could this sort of thing fit into the university curriculum? One problem with attempting to fit this into the core of the undergraduate computing curriculum is that undergraduate computer science is currently moving further away from the natural sciences, and computer science students seem less likely to define themselves as "scientists" than perhaps they might once have done. Yet natural science students are unlikely to have the experience with comptuational concepts to deal with this. Perhaps its natural place is at the postgraduate level; it is certainly the case that bioinformatics, another area requiring a combination of science and computing, has found its natural place as a largely postgraduate subject, studied (in the UK) mainly on taught masters degrees.

Philosophical Reasons

A final reason for considering these issues is to consider what we might mean by the *foundations of computer science*. This phrase usually means an axiomatic study of some particular model of computing, i.e. we define what we mean by "a computer" and proceed to derive properties of that model.

However computers (whether deliberately designed artifacts or computers identified as such in the natural world) are fundamentally *physical* devices. One way in which a view of computing as a property of the natural world influences our view of the foundations of computer science is to alter our view of the "limits of computation". Instead of saying that computation is limited by the properties of some logical model of computation, this perspective makes us ask what constraints the *physical* world places on our ability to compute [21]. A famous early application of these ideas (which predates modern concepts of computing) is the resolution of the *Maxwell's Demon* problem. This thermodynamic problem concerns an imaginary intelligent being (or information processing system) which "looks" at molecules which are travelling in one half of a divided chamber and lets high-energy molecules pass through a small door to the other side of the chamber, thus increasing the entropy of the system without spending energy. It was shown by Szilard [32] that the energy loss is in the information processing stage—there is a lower bound on the amount of energy which need be committed in order to carry out a particular computation.

This leads onto the issue of whether we want to regard a computational concept such as information as a fundamental property of the universe, in the same way that we might regard mass, energy and time as such. This argument has been extensively explored by Stonier [30]; the following quote outlines his core argument:

> *"Every time we pick up a pencil from the floor and place it on the desk, we expend energy to alter the organization of the universe. In placing the pencil on the desk, we have engaged in work; we have also create a thermodynamically less probable situation—hence we have* increased *the information content of the universe. Traditionally, it has always been a mystery as to what happened to the energy expended when we picked up the pencil. To explain it, physicists were forced to invent an accounting device:* potential energy*."* [30]

If information *is* such a property, then why haven't we noticed this before? Why have our conservation laws not been broken by energy disappearing into information? Stonier explains this by saying that we have conventionally wrapped up these concepts under terms such as "potential energy", "latent heat" and "entropy changes". It may be the case that any conserved quantity can be meaningfully broken down in a number of different ways, depending on what we want to explain. We can imagine a fictional history in which the concept of mass has been heavily developed but the concept of energy has only been implicitly developed. Thus scientists study "potential mass", "dispersed mass", et cetera, in place of energy. This is not to say that any arbitrary division of such a quantity is equal in explanatory power, neither is this *carte blanche* to do what we like. It remains a solecism to attribute more than one explanation *simultaneously* to a single phenomenon. Dividing explanation into mass, energy and information is no different to dividing the explanation into just mass, energy—there are no "new phenomena" which are informational in content which have been overlooked by previous explanations and which now need to be included.

Clearly such issues are far from the current concerns of computer science degrees. Should they be? Do we consider it important to include such issues in computing courses?

Implications for Computer Science Education

If computation-in-the-world is a significant area of study, and likely to become more important in future years, what role should it play in computer science education? This section outlines a couple of arguments and potential debates surrounding this issue.

Analytic and Synthetic Subjects

Science traditionally has both an *analytic* perspective, in which we look at the world and attempt to analyse its properties and regularities, and a *synthetic* perspective, in which we study ways of modifying the world to achieve certain end-results. For most areas of science, there is a balance between the synthetic and analytic, with a "pure science" giving us the analytic perspective, and a corresponding engineering subject giving us the synthetic perspective. This is clearly illustrated by the relationship between chemistry and chemical engineering.

Computer science stands out in this framework as being a wholly synthetic discipline. The role of theory in computer science is not to explain the properties of the "stuff" used to create computers; it is simply a more mathematised synthetic discipline.

Is it simply a historical accident that computer science has arisen as a synthetic subject, with no corresponding analytic discipline? If computer science moves more towards a balance between the two, what will students need to study as part of their degrees?

Computer Science: Science, Engineering or What?

Is computer science really a science? It has often been argued that it is not, because in contrast to traditional sciences computer scientists study the products of human ingenuity rather than the products of the natural world, and because computer science is concerned with the principles and practice of *creating* computational artifacts rather than studying existing artifacts.

A similar point has been made by Grundy [13]

> *"The gist of this criticism is that computing in no way attempts to explain natural phenomena as traditional experimental science does. This can be seen by focusing on its subject matter rather than on its methodology. All sciences have a subject matter and one is entitled to ask what things any science deals with. For instance, a child might ask what seismology is about and be told it is the scientific study of earthquakes. If we ask in this way what the subject matter of computer science is, the answer cannot be that it is the study of computers, if one means by 'computers' simply assemblages of things like metal and plastic."*

Nonetheless we can read this comment two ways. The first is to argue that this means that computing, and the teaching of computing, is different from the teaching of traditional science subjects. However an

alternative is to say that there is a subject of "computation" which is the subject of study of computer science—that is the subject matter of computer science is a class of *processes* in the world (in the way that a lot of physics is the study of processes), rather than being defined by the study of a particular class of *objects* (as is the case with seismology).

Later in the same article an attempt is made at elucidating the possible relationship between science and computing

> *"They can only be understood with reference to the human purposes for which they have been designed and built. Any link between science and computing must be* the application of various sciences *to the ongoing project of designing better and better machinery cum software."*

Again this assumes that the only valid application of computational concepts is to the creation of computing machines.

An alternative perspective is that computer science is a branch of engineering, and that the presence of the word *science* in its name is a historical accident. What are reasonable questions to ask of an engineering discipline? One is "what problems in the world does it set out to solve?" This sort of question is easy to answer for computer science. However another valid question which we might ask is "what science is the engineering subject called computer science grounded in?". It is a characteristic of engineering subjects that they are grounded in (one or more) science subjects. This is what distinguishes an engineering subject from a craft based around empirical rules of thumb. However this is a hard question for computer science.

Some people might answer "mathematics", yet this seems a weak answer; the other engineering disciplines take some things which are out there in the world and make use of them in building their engineering artifacts. Given that computers are made of physical stuff and not mind stuff, mathematics is at best a partial answer.

One answer would be to suggest that there is an inchoate, as yet to be formalized science which is the analytic counterpart to the synthetic engineering discipline. This would be a science of how objects-in-the-world store and process information, and some of the ideas reviewed earlier in the paper point towards a conception of what this science might be. The relationship here would be similar to that of the relationship between chemistry and chemical engineering. Another answer is to say that computer science is not an engineering discipline at all, nor a science, and not a branch of mathematics; it is a subject in its own right, with links to all of these, but with no obligation to take on the methodological mantle of any of them.

Some Initial Attempts at Teaching Natural Computation

Over the last few years we have introduced the teaching of these topics in two courses that form a part of our computer science programme: as part of the *Introduction to Intelligent Systems* and as a more focused course for final year students, entitled *Natural Computation*.

The aim of these courses has been to provide a broad overview of these topics, rather than specializing in one or two areas. The topics covered have included:

- Genetic algorithms and genetic programming
- Neural networks
- Artificial immune systems
- Swarm systems
- DNA and reaction-diffusion computing
- Quantum computing
- Using computation to understand biology and physics
- Philosophical issues concerning the interactions between computation and natural science

Perhaps our teaching has leaned too far towards issue of nature inspired computing, and has not covered the applications of computational thinking in the sciences. This is as issue that we proposed to address in future versions of the course.

Some of our experiences in teaching these courses and listening to student feedback include the following.

- There has been a general enthusiasm from students about the course content: the courses have been identified by a number of students in course feedback as amongst the most interesting and innovative in their degree.
- Some topics were very hard to teach beyond an initial, superficial coverage. This was particularly noticeable with quantum computing, where the lack of student background in mathematics and physics provided difficulties both of a practical nature and with motivating and contextualizing the material.

- The students generally seemed more interested in those aspects of the course that apply these ideas to computing than those which have scientific aims; overall, the students appear to see themselves as practical software creators, not as scientists.
- A number of students have been inspired to follow up these topics further, e.g. by carrying out individual projects on these topics for the final year of their degree, or going on to graduate study in these areas.
- We experimented with a number of alternative assessment forms. In particular, we asked students (individually or in pairs) to produce short podcasts/radio programmes, by taking a scientific paper of their choice and talking about it for a few minutes, in a way that was accessible to the general, though scientifically literate, public. The aim of this was to get students early on in their degree to engage with the research literature, without having to go into too much technical detail. This assessment somewhat divided the students, with some showing enthusiasm, with others being too shy and introverted to even tackle the assessment (more analysis of this in [15]).
- Another form of assessment was the essay, which is unusual for computing students; most students had not written an essay elsewhere during their degree. This was received initially with low enthusiasm, but many students found themselves working hard on the topic and, in the end, produced excellent work. However, there was (due to lack of experience) a lack of knowledge about how to effectively present a structured discussion of a topic, and a lack of knowledge about different forms of references (e.g. a lot of essays referenced only informal websites, course notes, course slides from this university and elsewhere, et cetera, rather than more authoritative sources). This needs to be addressed more directly in future iterations of the course.
- We considered it important that students learn to work with existing software and extend that software. Therefore we set assignments that were based on extending existing packages, e.g. for genetic programming. This was valuable, but some of the students with weaker programming skills found just understanding the code difficult and so did not manage to even begin the assessment, let alone engage with the technical material.
- We experimented with seminars, again based on (accessible) research papers. These were largely not successful: students did not do the reading in advance, and providing time to read during the seminar took up too much time. One approach which was moderately successful was to split up papers into sections and get pairs of students to do a short presentation on each section. However, our experience was that without assigning marks to the seminars, getting students to participate and do the reading was very difficult. This participation is particularly important in this area, as engagement with the research literature is important to understand these areas.
- Whilst there are lots of books and articles on the individual topics, we found it hard to find good overview material that addressed the more general issues. We did find that a number of popular science books were useful as ways to provide an initial, quick overview for students.

It would be interesting to compare our experiences with experiences in teaching this material at other universities. In a previous issue of this magazine, Xin Yao has described an entire MSc course programme on these topics [36] (see also [35]); he also comments on the difficulties that some students have experienced with the mathematical aspects of quantum computing and machine learning. More focusedly, Poli [27] has also given a detailed syllabus, and some commentary on the motivation behind, teaching evolutionary computation. An MSc course focused on the application of computational intelligence techniques has been discussed by Kaymak [18]. However, these papers do not address the wider task of applying "computational thinking" [34] across the sciences, instead being focused solely on the teaching of computational intelligence methods and their applications.

Concluding Questions: Natural Computing and the University Curriculum

Where could topics such as those discussed in this paper fit into the university curriculum? Could we look forward to a new generation of scientists who have an appreciation of the role that computational thinking, as well as computational skills, can have in understanding the natural world? Or is this area of increasing importance simply going to continue to be occupied by scientists who have blundered into this area from an arbitrary background by sheer force of curiosity? It will be interesting to see how these issues develop in the near future.

References

[1] A. Adamatsky and D. Tolmachiev. Chemical processor for computation of Voronoï diagram. *Advanced Materials for Optics and Electronics*, 6:191–196, 1996.

[2] A. Adamatsky and D. Tolmachiev. Chemical processor for computation of skeleton of planar shape. *Advanced Materials for Optics and Electronics*, 7:135–139, 1997.

[3] Martyn Amos. *Theoretical and Experimental DNA Computation*. Springer, 2003.

[4] P. Angeline, Z. Michalewicz, M. Schoenhauer, X. Yao, and A. Zalzala, editors. *Proceedings of the 1999 Congress on Evolutionary Computation*. IEEE Press, 1999.

[5] Cristian Calude and Gheorghe Paun. *Computing with Cells and Atoms*. Taylor and Francis, 2000.

[6] Peter J. Denning. Computing is a natural science. *Communications of the ACM*, 50(7):13–18, July 2007.

[7] M. Dorigo, M. Birattari, and T. Stutzle. Ant colony optimization. *Computational Intelligence Magazine, IEEE*, 1(4):28–39, Nov. 2006.

[8] M. Dorigo, V. Maniezzo, and A. Colorni. The ant system: Optimization by a colony of cooperating agents. *IEEE Transactions on Systems, Man and Cybernetics—Part B*, 26(1):29–41, 1996.

[9] Richard Feynman. *The Feynman Lectures on Computation*. Penguin, 1999. Edited by Anthony J.G Hey and Robin W. Allen.

[10] Richard P. Feynman. Simulating physics with computers. *International Journal of Theoretical Physics*, 21:467–488, 1982.

[11] Fred Glover. Tabu search—part I. *ORSA Journal on Computing*, 1(3):190–206, 1989.

[12] Fred Glover. Tabu search—part II. *ORSA Journal on Computing*, 2(1):4–32, 1990.

[13] Frances Grundy. Computer engineering : Engineering what? *AISB Quarterly*, 100:24–31, 1998.

[14] Frances Grundy. Where is the science in computer science? In *Proceedings of Women, Work and Computerization*. Springer, 2000. Simon Fraser University.

[15] Colin G. Johnson. Student-produced podcasts for learning and assessment. In Lauri Malmi and Arnold Pears, editors, *Proceedings of the Eighth Baltic Sea Conference on Computing Education Research*, 2008.

[16] L. Kari and G. Rozenberg. The many facets of natural computing. *Communications of the ACM*, 51(10):72–82, October 2008.

[17] S. Kauffman. *The Origins of Order*. Oxford University Press, 1993.

[18] U. Kaymak. An msc program in computational economics with a focus on computational intelligence. *Computational Intelligence Magazine, IEEE*, 1(2):41–41, May 2006.

[19] S. Kirkpatrick, C.D. Gellat, and M.P. Vecchi. Optimization by simulated annealing. *Science*, 220:671–680, 1983.

[20] John R. Koza. *Genetic Programming : On the Programming of Computers by means of Natural Selection*. Series in Complex Adaptive Systems. MIT Press, 1992.

[21] Seth Lloyd. Ultimate physical limits to computation. *Nature*, 406:1047–1054, 31st August 2000.

[22] Kishan Mehrotra, Chilukuri K. Mohan, and Sanjay Ranka. *Elements of Artificial Neural Networks*. MIT Press, 1996.

[23] Melanie Mitchell. *An Introduction to Genetic Algorithms*. Series in Complex Adaptive Systems. Bradford Books/MIT Press, 1996.

[24] Ray Paton. Towards a metaphorical biology. *Biology and Philosophy*, 7:279–294, 1992.

[25] R.C. Paton, H.S. Nwana, M.J.R. Shave, and T.J.M. Bench-Capon. An examination of some metaphorical contexts for biologically motivated computing. *British Journal for the Philosophy of Science*, 45:505–525, 1994.

[26] G. Paun, G. Rozenberg, and A. Salomaa. *DNA Computing*. Springer, 1998.

[27] Riccardo Poli. Evolutionary computation teaching at Birmingham. In Angeline et al. [4], pages 1689–1695.

[28] E.G. Rieffel and W. Polak. An introduction to quantum computing for non-physicists. *ACM Computing Surveys*, 32(3):300–335, September 2000.

[29] Andrew Steane. Quantum computing. *Reports on Progress in Physics*, 61:117–173, 1998.

[30] Tom Stonier. *Information and the Internal Structure of the Universe*. Springer, 1990.

[31] Tom Stonier. Information as a basic property of the universe. *Biosystems*, 38:135–140, 1996.

[32] L. Szilard. Über die Entropieverminderung in einem thermodynamisches System bei Eingriffen intelligenter Wesen. *Zeitschrift für Physik*, 53:840–856, 1929.

[33] Colin P. Williams and Scott H. Clearwater. *Explorations in quantum computing*. Springer, 1998.

[34] Jeanette M. Wing. Computational thinking. *Communications of the ACM*, 49(3), 2006.

[35] Xin Yao. How does evolutionary computation fit into IT postgraduate teaching? In Angeline et al. [4], pages 1707–1713.

[36] Xin Yao. A research-led and industry-oriented MSc program in natural computation. *IEEE Computational Intelligence Magazine*, 1(1):39–40, February 2006.