Molecular Computing

edited by T Sienko, A Adamatzky, M G Rambidi and M Conrad
MIT Press, 2003
£29.95 cloth

On your desk you've probably got a computer. It will be a general purpose digital computer, able to do anything from word processing to searching for extra terrestial life. The idea that computers are universal is enshrined in a fundamental doctrine, the Church-Turing Thesis, claiming that basically all computers are equivalent. Any proper computer can be programmed to simulate any other; they are universal. In particular any desktop computer can simulate being a word processor or anything else we can program it to do. If the Church-Turing Thesis is pushed to the limits, hard-liners would argue that the human brain is also a computer.

I'm a hardliner, so I was shaken awake by an idea from this collection of papers, *Molecular Computing*. A universal computer has to be simple enough to program, and the tradeoff to being simple means that programs are fragile. This sounds arcane, but is borne out by experience: a small change to a working program often stops it working at all; it introduces a bug and the program fails to do what it should do. On the other hand, if we try to design conventional programs to be robust, then by definition they will be hard to change. If we want programs to be simple then we cannot have them easy to change as well. Thus universal computers will never be good at learning or evolving, both of which require gradual change.

Biological systems like immune systems, evolutionary systems, visual systems, brains and so on, all solve complex problems efficiently, and they involve a lot of learning to do so. Immune systems, for example, learn to recognise new diseases with surprising efficiency (partly because of their enormous parallelism). They must thus work in ways different from universal computers. They cannot be programmed in any conventional way. They can mutate, learn and evolve with an ease conventional computers might only envy.

After Len Adelman showed in 1994 how DNA could solve a tricky conventional problem, finding a Hamiltonian Tour, the field of molecular computing blossomed. Molecular computers promise fantastic speed and new ways of approaching problems. They have become an exciting research field for chemists and biologists. There are some challenges. Molecular computers are not universal; sure, an immune system solves an enormously complex problem extremely fast, but that's all it does. Setting up a wet computer so that you can tell it the problem and read off the solution requires a lot of additional work. Adelman took about a week doing his experiment for just one case of his problem (and his practical results have since been contested).

Another example from the book is to find a Voronoi diagram. This is a hard problem, and it is very impressive to see a chemistry experiment solving it at all. Unfortunately, thinking of the chemistry setup and then getting the data in and the answer out is a lot harder than programming a conventional computer to do the whole job. The fourth problem is that you can only use your molecular computer to solve a specific problem, rather than program it to solve any problem, and you can typically only use it once, on one problem case. Molecular computers may solve problems with an alarmingly high error rate.

In contrast to being universal computers, molecular computers are called instance computers: they are good at one instance of a problem, and one problem type. Ironically, if you set up a molecular computer to simulate a universal computer that can be programmed easily, you still lose out against the tradeoff — so probably don't bother building Turing Machines with any hope of utilising parallelism to speed them up. The practical advantages of molecular computers, at least for the time being, are outweighed by the difficulties of using them to do anything useful. As this book says, there is more theory than practical success at the moment: molecular computers are more exciting than effective.

But we are only just beginning. We are rather fixated on conventional problems, which conventional computers are better at handling, but molecular computers are instance computers and we need to think differently. One trick is to use molecules to solve problems that they are naturally good at solving. Thus molecular DNA computers are good at doing DNA sorts of things, and indeed some experiments have shown how the well-known baterium *E. Coli* can be persuaded to be 32,000 times better at antibiotic resistance! Another example is to make a fish freshness sensor. Somehow, you have to measure lots of compounds, and assess odour and taste to say whether a fish is fresh. This is the sort of problem that animals solve all the time, so there is good reason to hope that molecular computers could be built to do it very well too.

The excitement is that molecular computing has enormous potential, and it can synergise with conventional biocomputing. It is clear that there are an enormous potential for applications in medicine, farming, food and areas like finger print recognition, forensic science.

It's an inspiring subject, but the book is a collection of independent chapters, written in a rather formal style. You can't read far without hitting a reference to some literature that isn't explained. You've just got excited about fish freshness, say, but the paragraph finishes with "a novel identification of freshness was performed" followed by *eight* literature citations you'd have to track down to understand it. The book sticks to molecular computers, avoiding much of the background on conventional computing or on quantum computers, neural computers and other examples of natural computing. This isn't a book to read; it is a book to use.

In summary, this book is a call to join in the research for any chemist or biochemist and, simultaneously, a hard-going overview of a ground-shaking area for a computer scientist. It sets a high entry barrier: if you aren't going to do research in this area, the practical ideas, and certainly the deeper ideas of the book will remain out of reach.

The field is young, though, and there is really no alternative to a book like this. The ideas are barely twenty years old, and are already challenging our view of what we can do in the world, and what dreams we can entertain. Digital computers and molecular biology are two of the greatest scientific achievements of the twentieth century; wet computers bring them together, and will change the twenty-first century in ways we haven't started to comprehend yet. I'd recommend reading the book and getting on the bandwagon if you can.

Harold Thimbleby is Gresham Professor of Geometry and Director of UCLIC, the UCL Interaction Centre.