COMPUTATIONAL INTELLIGENCE FOR CARBON-CENTRIC COMPUTING

John Fulcher
School of Computer Science and Software Engineering, University of Wollongong
Wollongong 2522, Australia
Email: john@uow.edu.au

ABSTRACT

The focus of this paper is twofold: firstly we make a case for the use of Computational Intelligence (CI) techniques in the modelling and/or prediction of global weather, CO_2 emissions, climate change and similar endeavours. CI exploits processes found in Nature, albeit by way of software simulations on digital computers (i.e. in silico), and excel in particular at pattern recognition and/or classification. Moreover, they are characterized as being non-algorithmic, bottom-up, data-driven, and learn-by-example. The second focus of this paper is to propose the use of carbon- rather than silicon-based computing, specifically in the form of DNA (or molecular) computing. Notwithstanding the unsolved difficulties with the latter (especially concerning Input/Output), its inherent massive parallelism has the potential to yield significant performance advantages. Finally, to come full circle, it could well eventuate that the inherent parallelism of DNA Computing could be brought to bear in the modelling/prediction endeavours mentioned previously.

INTRODUCTION

Our industry – ICT – is rightly viewed as being a "dirty" one, from the toxic chemicals used (and waste generated) during manufacture[11], all the way through to lack of re-cycling once computers reach their "use-by" date. The latter consideration is becoming more significant all the time, given that the US alone is predicted to junk around 3 billion electronic devices – primarily computers – by the year 2010[12]. In between this "birth" and "death", during their working life computers are

[11] http://pprc.org/hubs/subsection.cfm?hub=1004&subsec=11&nav=11&CFID=1133914&CFTOKEN=33073843

[12] http://www.ban.org/BANreports/10-24-05/documents/TheDigitalDump.pdf

collectively responsible for greenhouse gas emissions on a par with the aviation industry, producing around 830 million tonnes of CO_2 for example in 2007, which corresponds to ~2% of global emissions (and this level is predicted to rise to at least 1.4 billion tonnes by 2020). [13]

At the micro (local) level, emissions from Data Centres, PCs & peripherals, and networks & devices are roughly equivalent to those of Nigeria, Iran and Poland, respectively (Mankoff *et al.* 2008). In response to this, some major players (including `Google`, `Yahoo` and `Microsoft`) are currently investing a lot of effort into building more energy efficient, "green/carbon-neutral" data centres (Kurp 2008). Notwithstanding their inherently polluting nature, "dirty" computers can be used to model and predict weather patterns, greenhouse gas emissions, and by extension, climate change. Indeed, modelling of the Earth's weather system(s) has been an active area of research for several decades now[14].

Modelling of global weather (CO_2 emissions, climate change) is undertaken using one of the following approaches:
(i) precise mathematical models, or
(ii) rules (either precise or fuzzy), or
(iii) pattern recognition (we shall see in the next section that Computational Intelligence techniques are particularly suited to this latter approach).

The traditional approach to global weather modelling has involved the solution of precise, complex, non-linear mathematical describing functions (the driving functions for our climate models). Moreover, because of the heavy computational overhead required, this is often viewed as a "Grand Challenge" for High Performance Computing[15]. Not surprisingly, this usually requires the utilization of (expensive, highly parallel) super-computers. One such example is the NEC Earth Simulator, which up until the mid 2000s held the record of being the world's fastest supercomputer. It was specifically commissioned for global climate modelling, to investigate global warming, and to investigate solid earth geophysics. It operates at around 36 trillion floating point operations per second (TeraFLOPS), and its simulations boast a resolution of 10 km.

---

[13] *Economist* June 19 2008; http://www.theclimategroup.org

[14] http://www.aip.org/history/climate/GCM.htm; http://www.iop.org/activity/policy/Publications/file_4147.pdf; http://stephenschneider.stanford.edu/Publications/PDF_Papers/SunHansenJOC.pdf;

[15] http://www.nitrd.gov/pubs/200311_grand_challenges.pdf; http://www.ukcrc.org.uk/grand_challenges/index.cfm; http://www.cra.org/grand.challenges/

More recently, this performance has been superseded by both HECToR (High End Computing Terascale Resources) in the UK, and NASA's 67 TFLOP Discover supercomputer in the US. HECToR uses similar global climate models to that used in the Japanese Earth Simulator – but with finer resolution – thus enabling more realistic simulations of regional and local climate patterns, storms, and summer heat waves in Europe, as well as weather systems in tropical regions (including hurricanes, monsoons and El Niño phenomena).

Higher resolution modelling is also a feature of Discover, which is currently being used to predict the Earth's climate into the next century, to analyse global satellite weather observations, to model solar activity (and the affect this has on both the Earth's climate and on telecommunications transmissions), as well as simulating the merging of black holes, the formation of solar systems, and so on. It should be pointed out in passing that this system was commissioned with green computing in mind – the IBM DataPlex servers that comprise this system were designed to reduce power and cooling costs by mounting them sideways, and by using a liquid-cooled heat exchanger.

A fundamental need in the above pursuits is for the recognition of weather patterns (both past and present) – especially if we are attempting to characterize climate change, and make predictions in both the short- and longer term. There are complementary needs in this context for optimization and predictive ability. The focus of the present paper is on an approach to computing that potentially offers much by way of performance in terms of pattern recognition, optimization, modelling and/or prediction. This group of techniques is referred to collectively as "Computational Intelligence" (Fulcher & Jain 2008).

COMPUTATIONAL INTELLIGENCE (CI)

In earlier times, CI has gone by various names, including "intelligent systems", "soft computing", "Nature-inspired computing", and "Natural Computing" (Kari & Rozenberg 2008), to name but four. The common consensus these days is that CI encompasses (i) artificial neural network approaches, (ii) evolutionary methods, and (iii) fuzzy techniques. Other approaches can also be incorporated, especially if combined with one (or more) of the former methods into a hybrid system, three such typical techniques being immunity-based computing, swarms, and agents.

Computers	**Brains/CI**
Digital	Analog
Sequential	*(massively)* Parallel
(fast) Switches	*(slow)* Neurons
Symbolic processing	Sub-symbolic processing
(top-down) Model-driven	*(bottom-up)* Data-driven
Programmed *(algorithmic)*	Trained *(learn-by-example)*
Crisp, precise *(brittle)* 2-valued logic	Fuzzy logic
Precise, exact *(fault intolerant)*	Fault tolerant
Intolerant of noise & errors	Noise tolerant

Table 1 – Computers *versus* Brains/Computational Intelligence *(after Fulcher & Jain 2008)*.

CI approaches usually exhibit the following characteristics, apart that is from being inspired by Nature: (a) bottom-up, (b) data-driven, (c) non-algorithmic, and (d) learning-by-example. As such, they stand in direct contrast to the top-down, model-driven, logic-based nature of traditional algorithmic approaches to problem solving (Table 1).

The following is a (partial) list of phenomena in the natural world which have served as inspiration for CI approaches:

- The evolutionary process *(natural selection)*;
- Our brains *(biological neural networks)*;
- Insect swarms/flocks of birds;
- *(biological)* immune systems; and
- Our DNA structure.

These will be elaborated upon in the following sections.

ARTIFICIAL NEURAL NETWORKS (ANNS)

The inspiration for ANNs comes from the workings of our brain (biological neural network). There are around 10^{11} neurons residing in the human brain, with approximately 10^{14} connections between them. Inputs are received (on dendrites) from (up to 10^4) neighbouring neurons via these connections (synapses), but at a comparatively slow rate (1—100m/sec) – several orders of magnitude in fact slower than the speeds commonly encountered within modern-day (silicon) computers. In response to these inputs, an individual neuron will "fire", producing a pulse train of around 100Hz on its axon (output), which in turn connects to other neurons. Given an operating rate of $\sim 10^7$ operations per second, this amounts to an effective overall computational rate of $\sim 10^{18}$, which is around 1,000,000 times faster than present-day (TeraFLOP) supercomputers. Note however that individual neurons operate with only *milli*second response times. It appears that somehow (no-one can quite explain exactly how), the massive parallelism (albeit of combinations of slow processing elements) is what gives the brain its enormous computational ability. Moreover, brains are excellent at pattern recognition (a large proportion of the brain – around one third – being dedicated to vision).

ANNs are simplified models of the brain, but in which the number of neurons and interconnections (weights) number only in the hundreds and thousands, respectively. They are trained rather than programmed, using labelled pairs of input/output patterns. Such training procedures can take a long time in practice, but once trained an ANN can respond almost instantaneously to newly encountered inputs, producing a "best match" output corresponding to this new input.

The above description fits that of supervised, feedforward ANNs (Figure 1) – the most commonly used model being the Multi-Layer Perceptron/Backpropagation – many other models are in common usage, including unsupervised models such as Kohonen's Self-Organizing Map.

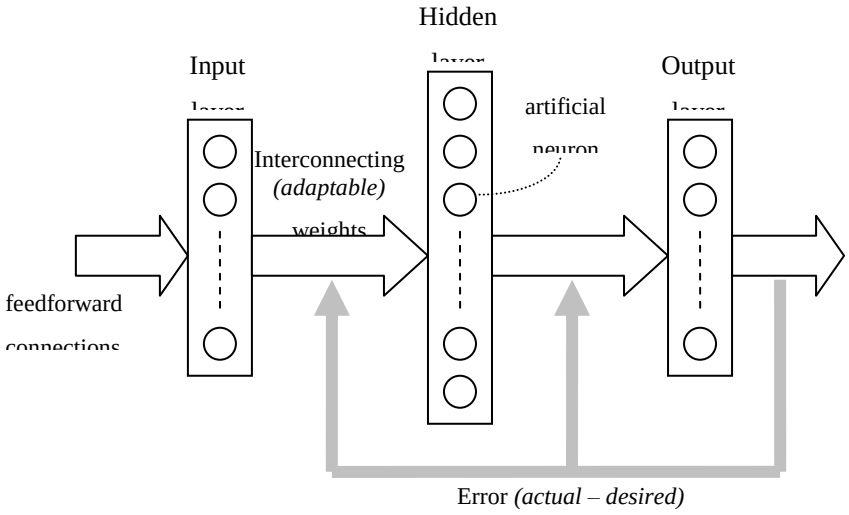

Figure 1 – Artificial Neural Network (ANN).

ANNs constitute a bottom-up, data-driven approach, in contrast to the top-down, model-driven nature of algorithmic approaches to problem solving. They have the added advantages of being both noise- and fault tolerant, however they suffer from certain well known limitations common to optimization techniques. Apart from the long training times already mentioned, they can become stuck in "local minima", and fail to converge to a trained, usable state (since network training can be likened to an optimization problem in which we seek to minimize an error function based on the difference between the actual and desired network outputs) (Haykin 1999).

Generally speaking, ANNs excel at pattern recognition and/or classification (irrespective of the specific application domain – i.e. pattern – of interest), optimization, modelling and prediction (even time series). A key consideration in applying ANNs in practice is pre-processing; data must be first converted into a suitable form prior to launching into a lengthy period of training. Accordingly, in the context of climate change and greenhouse gas emissions (in which past, present and future patterns are of major concern), they have a lot to offer.

In this context, one modest example of such an application of ANNs is in satellite weather prediction (Zhang & Fulcher 2004). Inputs to the `ANSER` system are satellite-derived measures such as cloud growth temperature, rainburst factor, storm speed, and the like; outputs are half-hourly rainfall estimations, achieved using an ANN reasoning network (one-of-*n*, the most appropriate one being selected by the Expert System in Figure 2). The system encapsulates knowledge derived from a human expert with over 30 years experience; one such knowledge item can be expressed by the following (precise) rule:

IF cloud temperature is -46^{o}C **AND** growth latitude = 2/3

THEN 0.48" of rain is expected in the ensuing ½-hour period.

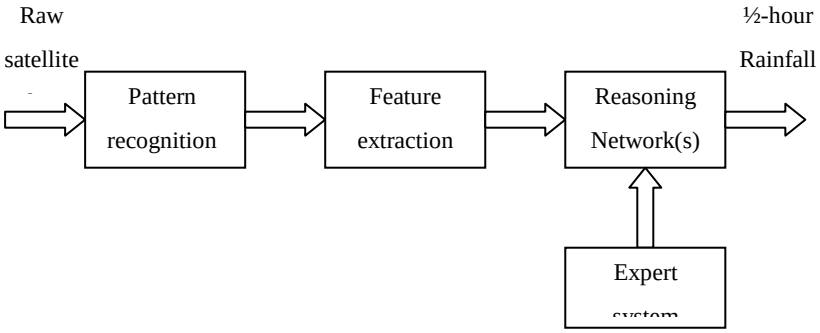

Figure 2 – Artificial Neural System for the Estimation of Rainfall (ANSER).

A typical rainfall event predicted by the ANSER system is shown in Figure 3. ANNs can be used for both pattern recognition (feature and knowledge extraction) as well as the Expert System proper. With regard to the latter, ANNs can replace not only the (rule-based) Inference Engine, but also the Knowledge Base with which the Inference Engine interacts (i.e. knowledge being stored/retrieved as patterns, rather than as explicit rules, be they precise or fuzzy).

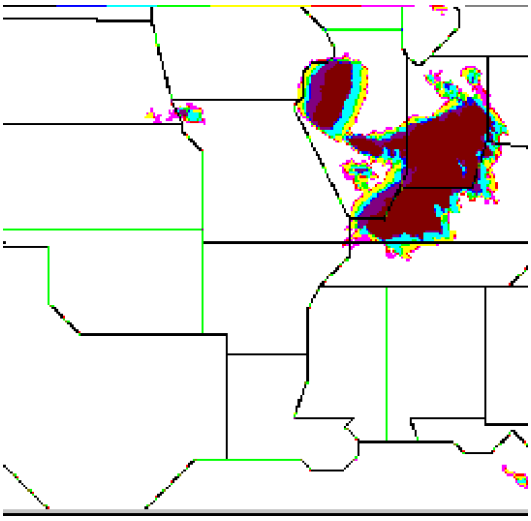

Figure 3 – Typical ANSER Output *(30 minute rainfall prediction)*.

EVOLUTIONARY COMPUTING (EC)

Evolutionary techniques, as the name suggests, take their inspiration from Nature – more specifically, Darwinism, evolution, and survival-of-the-fittest. As with ANNs, pre-processing is a

critical consideration with evolutionary techniques, since (as with the former technique) long training times are involved. Firstly, we need to encode in genetic string or "chromosome" form candidate solutions to the problem of interest. We also need to decide on both a genetic fitness measure and termination criteria – either in terms of number of evolved generations, or as a function of the system error falling to an acceptable level. Evolution proper comprises repeated cycling through "mating" (the crossover of chromosomes from genetically fit individuals – parents) ➔ the addition of (a small amount of) mutation ➔ measurement of the fitness of the resulting offspring ➔ mating…and so on, until the termination criterion is reached (Fogel 2005 – Figure 4).

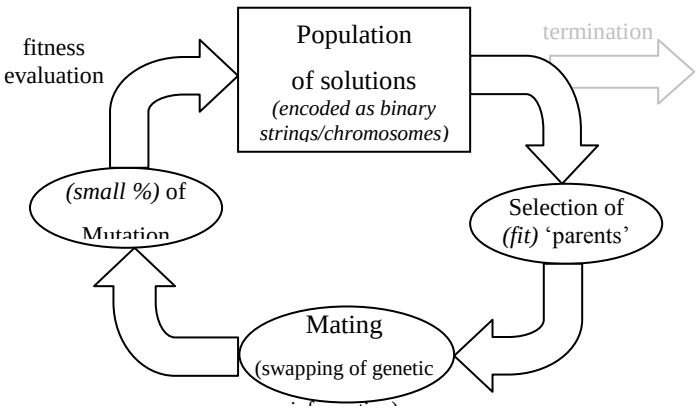

Figure 4 – The Basic Steps in Evolutionary Computing

Actually EC encompasses several variations on the above general principle, including Genetic Algorithms (GA), evolutionary programming, evolution strategies and Genetic Programming (GP) – the latter involves the evolution of computer programs, represented as syntax trees (Langdon *et al.* 2008).

It is important to realize that EC methods possess no knowledge of the domain (solution) space. Nevertheless, the hope is that over time (sometimes over *very long* times) more fit population members evolve. This pre-supposes sufficient diversity in the (solution) population as a whole, and more especially with the appropriateness of the initial encoding in chromosome (bit string) form. Thus, as with ANNs, pre-processing is critical in practice; likewise, acceptable solutions to problems of interest can often take a long time to evolve (just as networks can take a long time to train).

EC is particularly effective for optimization – i.e. discovering optimal solutions to under-constrained problems such as scheduling, timetabling, graph colouring, bin packing, travelling salesman, and the like.

Fuzzy Systems

Unlike ANNs and EC, Fuzzy Systems don't take Nature as their inspiration, unless one regards human thought processes and reasoning as being "natural". The "fuzziness' in question here refers to the inexact, imprecise (even at times contradictory) meanings of commonly used linguistic terms. Examples of such fuzzy terms might be "almost", "a little", "somewhat", "hot", "cold", "soon", and so forth. Moreover, unlike with crisp (precise, Boolean) logic, such terms can belong to more than one set – i.e. not just "black" or "white", but simultaneously 30% black and 60% white say. Fuzzy Logic allows us to map commonly used, linguistic expressions into fuzzy rules, and as such can be viewed as a generalization of 2-valued logic that supports operations on fuzzy sets. In this regard, it can be viewed as a form of "approximate reasoning" – one that sits more comfortably with us humans than the world of crisp, precise, 2-valued logic.[16]

Using the above concepts it is quite a straightforward exercise to construct a Fuzzy Inference System (FIS), comprising Input Fuzzification + Fuzzy Inference Engine + Output De-Fuzzification, in the usual Expert System configuration. The role of the Fuzzy Inference Engine in this FIS is to evaluate fuzzy rules, with reference to a Knowledge Base (in which knowledge about a specific problem domain is stored in terms of fuzzy rules) (Yan *et al.* 1995 – Figure 5).

In the context of CI, fuzzy systems are seen as complementing to ANN and EC approaches.

Other CI Approaches

Apart from the three "pillars" of CI – neural, evolutionary and fuzzy – many other techniques have been successfully applied to real-world problems in recent times. For the sake of brevity, we mention here but four: swarms, agents, immune-based computing, and membrane-based computing.

Swarms take as their inspiration the behaviour of social insects such as flocks of birds, shoals of fish, swarms of bees, nests of ants, etc. Such groups of identical, relative un-intelligent(?) individuals are nevertheless capable of behaving in quite an intelligent manner, over and above what individuals in the group are capable on their own – i.e. the whole is truly greater than the sum of its parts. It is this resulting collective or "swarm" intelligence that is simulated on digital computers in this approach to CI (Bonabeau *et al.* 1999; Englebrecht 2005; Kennedy *et al.* 2001). Swarms differ from other evolutionary approaches in that only *current* position, velocity and indirect, local interactions between swarm members are taken into consideration; there is no interest in evolving fitter population members over successive generations (indeed, all individuals are regarded as having identical form, function, status and ability). Variants of this approach include the Particle Swarm algorithm (Hendtlass 2008) and Ant Colony Optimization (Dorrigo & Stutzle 2004).

[16] http://www.fuzzy-logic.com/

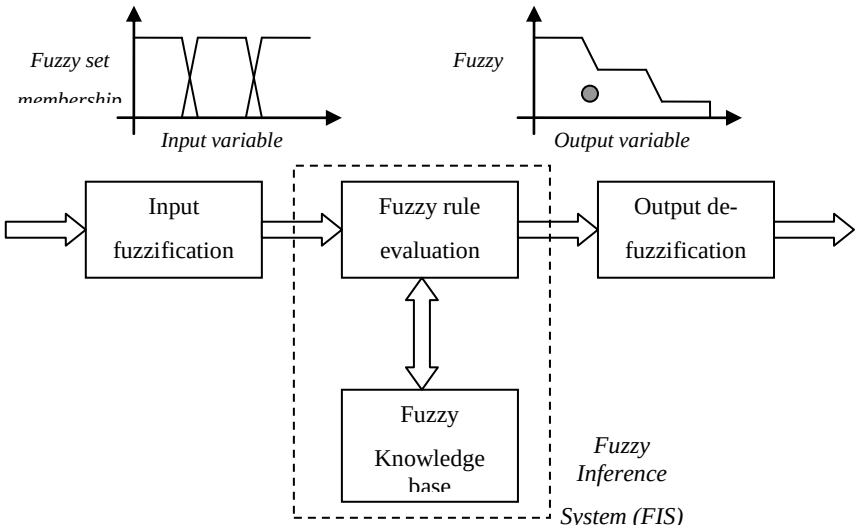

Figure 5 – Fuzzy Expert System

Intelligent Agents are an outgrowth of work in Distributed AI during the 1980s (Lesser 1995; Wooldridge & Jennings 1995). At its most fundamental level, an agent is an entity that is capable of perceiving its environment (via sensors), and taking "appropriate" actions (via actuators/effectors) in an automatic, autonomous fashion. Moreover, they are able to adapt and learn over time, as a result of the success/failure of taking various actions – hence they can be regarded as "intelligent" (Wooldridge 2002).

The inspiration for Immunity-Based Computing (IBC) is the memory, self-organization and learning abilities exhibited by invertebrate immune systems. In such systems, anti-bodies are able to discriminate between "self" and "non-self" in responding to cancerous cells or other threats to the biological entity. Software simulations encode the attributes of "antigens" and "antibodies" in appropriate string form. Furthermore, immunity-based systems are distributed, autonomous, and capable of self-repair (Segel & Cohen 2001; Ishida 2004, 2008).

Membrane-based Computing (MBC) is inspired by some basic features of biological membranes, such as:

- objects reside in compartments defined by the membrane structure, and evolve by means of "reaction rules", in a maximally parallel, non-deterministic manner;
- objects are able to pass through membranes, dissolve, and divide; and
- membranes can change their permeability.

These features can be used to define transitions between different system configurations; sequences of such transitions can in turn be used to define computations (Calude & Paun 2001; Paun 2002).

Hybrid CI Systems

Of particular interest in recent times has been the combination of more than *one* CI technique in order to realize hybrid systems. The rationale behind this approach is that if a single technique is unable to deliver the desired performance in a specific application, then perhaps a combination of

techniques can. The converse is that the techniques could interfere with each other and actually lead to performance degradation, so we need to exercise caution in such undertakings. Apart from choosing the most appropriate techniques *per se*, there is also the issue of how best to combine them (Ovaska 2004; Negoita *et al.* 2005).

Without going into details, we simply cite here for reference purposes some instances where researchers have had success in enhancing performance by resorting to hybrid CI systems: Neuro-Fuzzy (Jang *et al.* 1997), Fuzzy-Neural (Pedrycz 1993), Evolution of ANNs (Schalkof 1997), Evolution of Fuzzy (Karr 1991), and Swarm optimization of Fuzzy (Khosla *et al.* 2006).

DNA OR MOLECULAR COMPUTING

All of the CI techniques discussed up to now, whist being inspired by Nature, are implemented in practice by way of software simulations on conventional (silicon-based) computers. Now we turn our attention to the *converse* approach, namely the realization in so-called "wet-ware" of conventional algorithms – a "computer-in-a-test-tube", as it were. This is the approach taken with DNA Computing. In the present context, we can characterize this as being a "carbon-centric", rather than the more familiar and usual "silicon-centric", approach to computing.

One of the appeals of DNA Computing is its potentially huge storage capacity (~10^{18} or one Exabyte per gram, equivalent to ~1Gbit per square inch – compared with ~7GBits/inch^2 for present day Hard Disk Drives). Another potential advantage is its massive parallelism (~10^{20} operations per second; by contrast, modern-day supercomputers operate in the TeraFLOPs range). Massive potential parallelism, by the way, is also a strong selling point with Quantum Computing (Ovaska 2004; Reifel & Polak 2000; Williams & Clearwater 2000).

DeoxyriboNucleic Acid (DNA)

DNA is structured as a linear strand of four different nucleotides/bases (only): Adenine, Cytosine, Guantine, and Thymine. A single strand of DNA has an orientation, referred to as 5'-AAACC-3', for example. Pairwise bonds can be formed only between A and T, and between G and C, leading to (A,T) and (G,C) base pairs. Watson-Crick complementarity dictates that the single strands 5'-AAACC-3' and 3'-TTTGG-5' will bind together to form a 5 base-pair double (helix) strand of DNA. Furthermore, the nucleotides in such a double helix can be broken and re-formed in order to realize solution(s) to problems of interest.

Replication of DNA within bacteria occurs at the rate of around 500 base pairs per second, and within human cells at around 50 base pairs/sec. The former corresponds to a computation rate of ~1000 bits/sec, but this increases dramatically when multiple copies of the replication enzymes work on the DNA concurrently, increasing exponentially (i.e. 2^n after n iterations), such that after 30 iterations, say, the combined computation rate leaps to 1 Tbit/sec.

Now a major stumbling block with DNA computing, at least at the present time, is Input/Output (likewise this remains a substantial challenge with QC). In short, how do we accurately encode problems of interest (in our case, weather modelling, greenhouse gas emissions and climate change) into DNA strand (nucleotide) form in the first instance, and more importantly, how do we *decode* the results of mixing together the constituent nucleotides in our "test tube computer" (detachment) (Amos 2005; Calude & Paun 2001; Paun *et al.* 1999)?

DNA Computations

The realization that DNA molecules could be used for computation is not new – indeed Von Neumann's work on Cellular Automata during the 1940s could be formulated in molecular terms (Von Neumann 1966). Adelman was the first to report on how molecular biology could be used to perform computations with DNA *in vitro* (i.e. chemically, within a test tube, as opposed to *in vivo* – within cellular life forms) (Adelman 1994). In such a manner he solved the 7-point Hamiltonian path problem.

More specifically, sequences of 8-to 20 base pairs can be used as information storage media. Moreover, the following "wet-ware" techniques can be utilised as computational operators for copying, sorting, splitting and/or concatenating information:

- Ligation;
- Hybridization;
- Polymerase chain reaction;
- Gel electrophoresis; and
- Enzyme reaction.

The following describes a (*super*-parallel) DNA Computing algorithm:

- Unique letter codes are assigned to samples *(e.g. ATCG, GTAC, CAAC…etc.);*
- DNA sequences corresponding to the number of possible combinations are prepared;
- Sequences are hybridized in *super*-parallel fashion; and
- The remaining DNA fragments are amplified to obtain an answer sequence.

At this juncture we should emphasize the natural affinity for DNA computers for solving optimisation problems. The fact that *all* solutions to a problem of interest can be realized concurrently means that even brute force algorithms suddenly become feasible. Indeed, Calude and Paun have made the case for DNA computers behaving as Turing machines (Calude & Paun 2001). Further, Amos distinguishes between so-called "weak" and "strong" models of DNA Computing, which are capable of O(n) and O(n^2) performance, respectively, for generating all permutations of the integers {1,2,…,n}, say (Amos 2005). Applications of DNA computers include game playing (tic-tac-toe, using `Maya-II` – Molecular Array of Yes And logic gates – a co-development of Columbia & New Mexico Universities[17]), and the scheduling of multiple elevators in high-rise buildings (Watada 2008). More recently, researchers at the Weizmann Institute, Israel have fabricated autonomous programmable molecular modules (input, computation & output), which collectively form a "biological automaton" for the diagnosis and treatment of certain cancers (Benenson *et al.* 2004; Shapiro & Benenson 2006)[18].

[17] http://medgadget.com/archives/2006/10/computer_with_d.html

[18] http://news.nationalgeographic.com/news/2003/02/0224_030224_DNAcomputer.html

CONCLUSION

A common characteristic shared by all of the aforementioned CI techniques is their ready applicability to pattern recognition, optimization, modelling and/or prediction. As such they can be utilized to respond to the ever-increasing challenges inherent in modelling climate change, reducing CO2 emissions, and so forth. In addition, this silicon-based approach can be complemented by DNA Computing – a truly carbon-based (molecular) approach – in the realization of carbon-centric computing.

REFERENCES

Adelman, L.M. 1994. Molecular Computation of Solutions to Combinatorial Problems. *Science*. 266, 1021—1024.

Amos, M.(2005) *Theoretical and Experimental DNA Computation*. Springer-Verlag, Berlin.

Benenson, Y., Gil, B., Ben-Dor, U., Adar, R., Shapiro, E. 2004. An Autonomous Molecular Computer for Logical Control of Gene Expression. *Nature*. 429, 423—429.

Bonabeau, M., Dorigo, M., Theraulaz, G. (1999). *Swarm Intelligence.* Morgan Kaufmann, San Francisco, CA.

Calude, C.A., Paun, G. (2001) *Computing with Cells and Atoms: An Introduction to Quantum, DNA and Membrane Computing*. Taylor and Francis, London.

Dorrigo, M., Stutzle, T. (2004) *Ant Colony Optimization.* MIT Press, Cambridge, MA.

Engelbrecht, A.P. (2005) *Fundamentals of Computational Swarm Intelligence.* Wiley, London.

Fogel, D.B. (2005) *Evolutionary Computation: Toward a New Philosophy of Machine Intelligence*. Wiley, New York, NY.

Fulcher, J., Jain, L.C. (eds.) (2008) *Computational Intelligence: A Compendium*. Springer-Verlag, Berlin.

Haykin, S.Y. (1999) *Neural Networks: A Comprehensive Foundation (2nd ed.)*. Prentice Hall, Upper Saddle River, NJ.

Hendtlass, T. 2008. The Particle Swarm Algorithm. In: Fulcher, J., Jain, L.C. (eds.): *Computational Intelligence: A Compendium*. Springer-Verlag, Berlin, 1029—1062.

Ishida, Y. (2004) *Immunity-Based Systems: A Design Perspective.* Springer-Verlag, Berlin.

Ishida, Y. 2008. The Next Generation of Immunity-Based Systems. In: Fulcher, J., Jain, L.C. (eds.): *Computational Intelligence: A Compendium*. Springer-Verlag, Berlin, 1091—1119.

Jang, J.-S.R., Sun, C.-T., Mizutani, E. (1997). *Neuro-Fuzzy and Soft Computing: a Computational Approach to Learning and Machine Intelligence.* Prentice Hall, Englewood Cliffs, NJ.

Kari, L., Rozenberg, G. 2008. The Many Facets of Natural Computing. *Communications of ACM*. 51(10), 72—83.

Karr, C. (1991) Applying Genetics to Fuzzy Logic. *AI Expert.* 6(3), 38—43.

Kennedy, J., Eberhart, R.C., Yuhui, S., Shi, Y.: (2001) *Swarm Intelligence.* Morgan Kaufmann, San Francisco, CA.

Khosla, A., Kumar, S., Aggarwal, K.K. 2006. Swarm Intelligence and the Taguchi Method for Identification of Fuzzy Models. In: Fulcher, J. (ed.) *Advances in Applied Artificial Intelligence*. Idea Group, Hershey, PA, 273—295.

Kurp, P. 2008. Green Computing. *Communications ACM*. 51(10), 11—13.

Langdon, W.B., Poli, R., McPhee, N.F., Koza, J.R. 2008. Genetic Programming: An Introduction and Tutorial, with a Survey of Techniques and Applications. In: Fulcher, J., Jain, L.C. (eds.): *Computational Intelligence: A Compendium*. Springer, Berlin, 1927—1028.

Lesser, V. 1995. Multiagent Systems: An Emerging Subdiscipline of AI. *ACM Computing Surveys.* 27(3), 340—342.

Mankoff, J., Kravets, R., Blevis, E. 2008. Some Computer Science Issues in Creating a Sustainable World. *IEEE Computer*. 41(8), 102—105.

Negoita, M.G., Neagu, D., Palade, V. (2005) *Computational Intelligence: Engineering of Hybrid Systems.* Springer-Verlag, Berlin.

Ovaska, S.J. (ed.) (2004) *Computationally Intelligent Hybrid Systems: The Fusion of Soft Computing and Hard Computing.* Wiley, New York, NY.

Paun, G. (2002) *Membrane Computing: An Introduction.* Springer-Verlag, Berlin.

Paun, G.H., Rozenberg, G., Salomaa, A. (1999) *DNA Computing: New Computing Paradigms*. Springer-Verlag, Berlin.

Pedrycz, W. 1993. Fuzzy Neural Networks and Neurocomputations. *Fuzzy Sets and Systems.* 56, 1—28.

Rieffel, E.G., Polak, W. 2000. Quantum Computing for Non-Physicists. *ACM Computing Surveys.* 32(3), 300—335.

Schalkof, R.J. (1997) *Artificial Neural Networks: Application to Ecology and Evolution*. McGraw-Hill, New York, NY.

Segel, L.A., Cohen, I.R. (eds.) (2001) *Design Principles for the Immune System and Other Distributed Autonomous Systems.* Oxford University Press, New York, NY.

Shapiro, E., Benenson, Y. 2006. Bringing DNA Computers to Life. *Scientific American*. 294(5), 45—51.

Von Neumann, J. (1966) *Theory of Self-Reproducing Automata*. Burks, A.W. (ed.) University of Illinois Press, Urbana, IL.

Watada, J. 2008. DNA Computing and its Application. In: Fulcher, J., Jain, L.C. (eds.): *Computational Intelligence: A Compendium*. Springer-Verlag, Berlin, 1065—1089.

Williams, C.P., Clearwater, S.H. (2000) *Ultimate Zero and One: Computing at the Quantum Frontier.* Springer-Verlag, Berlin.

Wooldridge, M., Jennings, J.R. 1995. Intelligent Agents: Theory and Practice. *The Knowledge Engineering Review.* 10(2), 115—152.

Wooldridge, M. (2002) *An Introduction to Multiagent Systems.* Wiley, Chichester, UK.

Yan, J., Ryan, M., Power, J. (1995). *Using Fuzzy Logic: Towards Intelligent Systems*. Prentice Hall, Englewood Cliffs, NJ.

Zhang, M., Fulcher, J. 2004. Higher-Order Neural Networks for Satellite Weather Prediction. In: Fulcher, J., Jain, L.C. (eds.) *Applied Intelligent Systems: New Directions* Springer-Verlag, Berlin, 17—57.