Chapter 16
Computational Matter: Evolving Computational Functions in Nanoscale Materials

Hajo Broersma, Julian F. Miller and Stefano Nichele

Abstract Natural evolution has been manipulating the properties of proteins for billions of years. This 'design process' is completely different to conventional human design which assembles well-understood smaller parts in a highly principled way. In evolution-in-materio (EIM), researchers use evolutionary algorithms to define configurations and magnitudes of physical variables (e.g. voltages) which are applied to material systems so that they carry out useful computation. One of the advantages of this is that artificial evolution can exploit physical effects that are either too complex to understand or hitherto unknown. An EU funded project in Unconventional Computation called NASCENCE: Nanoscale Engineering of Novel Computation using Evolution, has the aim to model, understand and exploit the behaviour of evolved configurations of nanosystems (e.g. networks of nanoparticles, carbon nanotubes, liquid crystals) to solve computational problems. The project showed that it is possible to use materials to help find solutions to a number of well-known computational problems (e.g. TSP, Bin-packing, Logic gates, etc.).

16.1 Introduction

Conventional or classical computation is based on an abstract model of a machine called a Turing Machine [69]. Such a machine can write or erase symbols on a possibly infinite one dimensional tape. Its actions are determined by a table of instructions that determine what the machine will write on the tape (by moving one square left

H. Broersma (✉)
University of Twente, Enschede, The Netherlands
e-mail: h.j.broersma@utwente.nl

J.F. Miller
University of York, YO10 5DD Heslington, York, England
e-mail: julian.miller@york.ac.uk

S. Nichele
Norwegian University of Science and Technology, Trondheim, Norway
e-mail: nichele@idi.ntnu.no

© Springer International Publishing Switzerland 2017
A. Adamatzky (ed.), *Advances in Unconventional Computing*,
Emergence, Complexity and Computation 23,
DOI 10.1007/978-3-319-33921-4_16

or right) given its state (stored in a state register) and the symbol on the tape. Turing showed that the calculations that could be performed on such a machine accord with the notion of computation in mathematics. Von Neumann proposed a design for a computer architecture based on the ideas of Turing that formed the foundation of modern stored programs computers [52]. These are digital in operation. Although they are made of physical devices (i.e. transistors), computations are made on the basis of whether a voltage is above or below some threshold. Classical computers are based on a *symbolic* notion of computation.

Unconventional computing looks at systems that carry out computation which do not conform to the Turing model of computation. An obvious and highly plentiful source of such unconventional computing systems are living organisms. These carry out prodigious amounts of computation in their everyday tasks of survival and reproduction. Unlike classical programs the computational instructions that underlie living organisms have not been designed but rather have been evolved. Symbolic notions of computation have a severe drawback compared with evolving physical computational systems. The former has no obvious way of making use of the natural computational power of physical systems. According to Conrad this leads us to pay "The Price of Programmability" [12], whereby in conventional programming and design we proceed by excluding many of the processes that may lead to us solving the problem at hand. The question then emerges: Is there a way to use classical computation to *exploit* rather than exclude, the properties of physical systems to solve computational problems? This chapter describes work that answers this question by attempting to use computer controlled evolution to 'program' useful devices. This allows artificial evolution to directly manipulate a physical system. In this way it is hoped to create novel and useful devices in physical systems whose operational principles are not necessarily understood or are hitherto unknown.

Computer-controlled evolution is referred to by a variety of names: evolutionary algorithms [14], genetic algorithms [23], genetic programming [27, 56]. It is part of a wide research area known as bio-inspired computation. The main elements of an evolutionary algorithm are:

```
Generate initial population of size $p$. Set number of generations, $g = 0$
REPEAT
    Calculate the fitness of each member of the population
    Select a number of parents according to quality of fitness
    Recombine some, if not all, parents to create offspring genotypes
    Mutate some parents and offspring
    Form a new population from mutated parents and offspring
    Optional: promote a number of unaltered parents from step 4 to the new
    population
    Increment the number of generations $g \leftarrow g + 1$
    UNTIL($g$ equals the number of generations required) **OR** (the fitness is
    acceptable)
```

In evolutionary computing, the term *genotype* (or chromosome) is used to refer to the string of numbers that defines a solution to a search problem. The individual elements of the genotype are commonly referred to as *genes*. To solve a computational problem requires an assessment of how well a particular genotype represents a solution to the computational search problem. This is called a *fitness function*. The "survival-of-the-fittest" principle of Darwinian evolution is implemented by using a form of fitness-based selection that is more likely to choose solutions for the next generation that are fitter rather than poorer. *Mutation* is an operation that changes a genotype by making random alterations to some genes, with a certain probability. *Recombination* is a process of generating one or more new genotypes by recombining genes from two or more genotypes. Sometimes, genotypes from one generation are promoted directly to the next generation, this is referred to as *elitism* (see the optional step in the above Algorithm).

Evolution-in-materio (EIM) is a term coined by Miller and Downing [38] that refers to the manipulation of physical systems using computer controlled evolution (CCE) [19–21, 38, 41]. It is a type of unconstrained evolution in which, through the application of physical signals, various intrinsic properties of a material can be heightened or configured so that a useful computational function is achieved. Yoshihito discussed a closely related concept of "material processors" which he describes as material systems that can process information by using the properties of the material [75]. Zauner describes a related term which he refers to as "informed matter" [76]. It is interesting that inspection of much earlier research publications reveals that ideas similar to evolution-in-materio, albeit without computers, were conceived in the late 1950s (particularly by Gordon Pask, see [6, 55]).

The concept of EIM grew out of work that arose in a sub-field of evolutionary computation called evolvable hardware [16, 22, 61, 77], particularly through the work of Adrian Thompson [66, 68]. In 1996, Thompson famously demonstrated that unconstrained evolution on a silicon chip called a Field Programmable Gate Array (FPGA) could utilise the physical properties of the chip to solve computational problems [65].

In 2002, Miller and Downing discussed the concept of evolution-in-materio and suggested that liquid crystal might be a suitable material for attempting to evolve computation in materials [38]. They also discussed many other materials whose properties can be affected reversibly via physical signals. Utilising the evolvable motherboard concept of Layzell [32], Harding constructed a liquid crystal analogue processor (LCAP) that utilises the physical properties of liquid crystal for computation [18]. The experimental setup used by Harding was similar in concept to that used by Thompson [65, 67].

The work described in this chapter was carried out as part of an EU funded research project called NASCENCE (Nanoscale Engineering of Novel Computation Using Evolution) [5] in which various computational problems were investigated using evolution-in-materio using micro-electrode arrays.

The plan of the chapter is as follows. Section 16.2 explains the central concept of evolution-in-materio. Section 16.3 describes the used configurable nano-materials, and in Sect. 16.4 an overview of the EIM hardware control system is given. A brief introduction of the computational problems under investigation is presented in Sect. 16.5, and Sect. 16.6 gives a detailed summary of the experimental results for the solved computational problems, together with an explanation of possible emergent behaviours of carbon nanotube materials and gold nanoparticle materials. In Sect. 16.7 different models and simulation tools for nano-materials are introduced, each at a different abstraction level. Finally, Sect. 16.8 concludes the chapter and outlines directions for further work.

16.2 Conceptual Overview

The central idea of evolution-in-materio is that the application of some physical signals to a material (configuration variables) can cause it to alter how it responds to an applied physical input signal and how it generates a measurable physical output (see Fig. 16.1) [38]. Physical outputs from the material are converted to output data and a numerical fitness score is assigned based on how close the output is to a desired response. This fitness is assigned to the member of the population that supplied the configuration variables. Ideally, the material would be able to be reset before the application of new configuration instructions. This is likely to be important as without the ability to reset the material, it may retain a memory from past configurations. This could lead to the same configuration having different fitness values depending on the history of interactions with the material.

Mappings need to be devised which convert problem domain data into suitable signals to apply to the material. An *input-mapping* needs to be devised to map problem domain inputs (if any) to physical input signals. An *output-mapping* is required to convert measured variables from the material into a numerical value which can be used to solve a computational problem. Finally, a *configuration-mapping* is required to convert numerical values held on a computer into physical variables that are used to "program or configure" the material. In the course of this chapter we will see examples of a number of mappings.

A difficult question which to some extent can only be answered by experiments, concerns which types of physical variables should be manipulated to obtain the best response from the material. As will be seen, generally in the NASCENCE project, only electrical stimuli to the materials have been investigated. This was largely chosen because it is relatively straightforward to manipulate such signals.

There are two main ways that computational problems can be solved using EIM. In the first, a material is used in the mapping of genotype to a fitness value. In this approach the material is seen as an assistant in an evolutionary search process. It provides a "black-box" mapping from genotype to output data (from which fitness is assessed). The thinking behind this is that the material may provide a more evolvable genotype-to-phenotype mapping, since physical variables can be exploited that could not be exploited if a purely algorithmic mapping was used (as is standard in evolutionary computation). In this type of *hybrid* system, much of the data required for solving a particular problem would remain on a computer. The role of the material

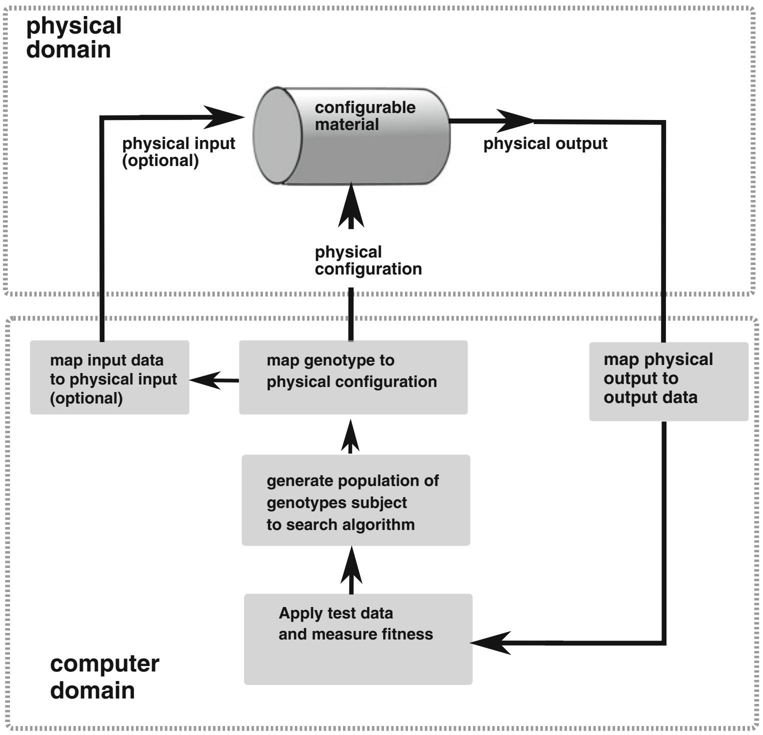
physical domain
configurable material
physical input (optional)
physical output
physical configuration
map input data to physical input (optional)
map genotype to physical configuration
map physical output to output data
generate population of genotypes subject to search algorithm
Apply test data and measure fitness
computer domain

Fig. 16.1 Concept of evolution-in-materio. There are two domains: physical and computer. In the physical domain there is a material to which physical signals can be applied or measured. These signals are either input signals, output signals or configuration signals. A computer controls the application of physical inputs applied to the material, the reading of physical signals from the material and the application to the material of other physical inputs known as configurations. A genotype of numerical data is held on the computer and is transformed into configuration variables that physically affect the material. The genotypes are subject to an evolutionary algorithm. Physical output signals are read from the material and converted to output data in the computer. A fitness value is obtained from the output data and supplied as a fitness of a genotype to the evolutionary algorithm

would be to improve the search process itself. Thus in this case the material does not necessarily require any input data. Examples of computational problems that can be tackled using this approach are: Travelling Salesman Problem (TSP). Function Optimisation and Bin-packing. The TSP is the well known problem of determining the shortest tour through a number of cities. Function optimisation is the problem of determining a vector of numbers which minimises a complex function. Bin-packing is the problem of packing a number of items into as few bins as possible, assuming that each bin has a fixed weight capacity. To obtain solutions to such problems using EIM requires that a set of configuration signals are determined that cause the material to output a suitable vector of measured values.

In the second approach, the evolutionary algorithm determines a configuration which allows the material to act as a *stand alone* computational device. This is a device which provided with the evolved configuration signals, carries out the desired computational mapping. A number of such problems have been considered: digital logic gates, data classification, robot control and graph colouring. For example, suppose that one desired to carry out data classification using a material. Assuming that a stand alone device could be built that used the material and some circuitry to provide the evolved set of configuration signals, one could potentially feed data into the material at a very fast rate and obtain data classification at very high speed and low power consumption. Similarly, evolved logic gates may be able to operate at high speeds and low power etc. We will see examples of both approaches being used in this chapter. The term *configuration* of a material can have a number of meanings. It can merely be the application of physical signals to the material so that some underlying physical properties change, e.g. conductance or resistance. As a result, the material is put into a state that allows the desired computation to take place. Or alternatively, it may be that when the physical signals are applied the material physically changes in some way. For instance, the underlying (electrical) network might be rearranged, or the molecules at the nano-scale could self-organise to a desired state so that the target computational function takes place. An example of the latter is provided by the work with liquid crystal of Harding and Miller [18]. In this case, applied configuration signals caused liquid crystal molecules to twist, thus there was a physical change in the material when configuration signals were applied. Finally, both these effects may happen at the same time.

16.3 Configurable Materials and Micro-electrode Arrays

Although computational materials may be configured by different kinds of stimuli, e.g. electrical signals, magnetic fields, temperature variations, light, etc., it was decided to only manipulate electrical signals within the NASCENCE project. Two types of evolvable material systems were constructed. Both are based on electrode arrays. A material is deposited in the vicinity of the electrodes. Some of the electrodes are chosen as inputs (if the computational problem demands inputs), some are chosen as outputs, and a number of electrodes are chosen as configuration electrodes. In one system, the material deposited was a mixture of single-walled carbon nanotubes (SWCNT) randomly mixed in an insulating material. This is shown in Fig. 16.2. The insulating material was either PMMA/PBMA (Polymethy/butyl methacralate) [49]. Carbon nanotubes are conducting or semi-conducting and the role of the PMMA/PBMA is to introduce insulating regions within the nanotube network, to create non-linear current versus voltage characteristics.

In the other system, shown in Fig. 16.3, a disordered network of gold nanoparticles interconnected by insulating molecules (1-octanethiols) is trapped in a small region surrounded by electrodes [4].

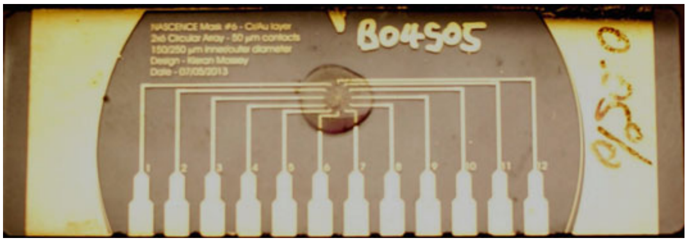

Fig. 16.2 Circular twelve electrode array. The material in the centre is a mixture of SWCNT and PMMA. The concentration of SWCNT is 0.05 % by weight. SWCNTs are mixed with PMMA or PBMA and dissolved in anisole (methoxybenzene). 20 µL of material is drop dispensed onto the electrode array. This is dried at 100°C for 30 min to leave a film over the electrodes [49]

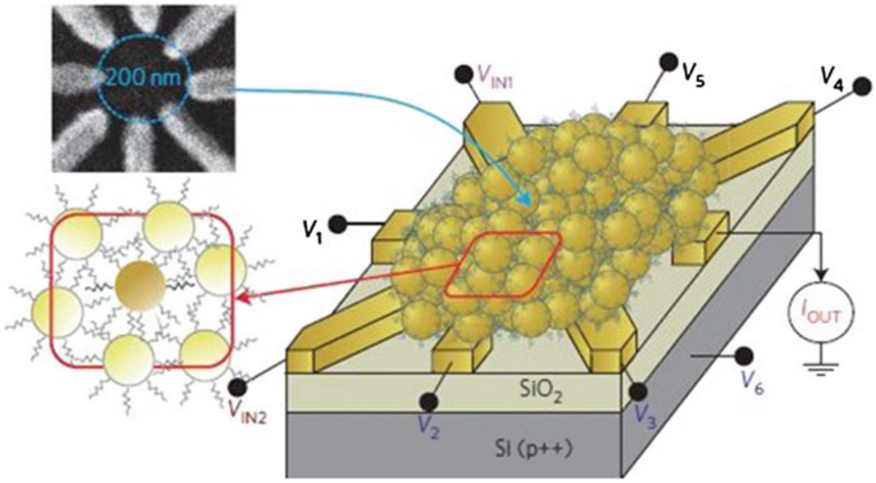

Fig. 16.3 Circular eight electrode array. The material is a disordered network of 20 nm Au NPs interconnected by insulating molecules (1-octanethiols). The NPs are trapped in a circular region (200 nm in diameter) between radial metal (Ti/Au) electrodes on top of a highly doped Si/SiO_2 substrate, which functions as a back gate. The device operates at temperatures below 1°K [4]

At low temperatures, a nanoparticle with capacitance C has a charging energy $E = e^2/C$ which is larger than the thermal energy.[1] In this case nanoparticles exhibit Coulomb blockade and act as a single electron transistor (SET). One electron at a time can tunnel when sufficient energy is available (ON state), either by applying a voltage across the SET or by electrostatically shifting its potential. Otherwise, the transport is blocked because of the Coulomb blockade (OFF state).

[1] e is the charge on an electron.

16.4 EIM Hardware Control Systems

In order to be able to apply an evolutionary algorithm to determine a set of signals that should be applied to the electrode arrays, one requires a hardware interface system between a computer and the material. The hardware system needs to allow a variety of signals to be applied to the electrodes. In the NASCENCE project the signals used were one of the following:

- Digital voltages, 0 and 3.5 V
- Analogue voltages in some range
- Square-wave signals

One also needs to be able to sample and record voltages detected on electrodes, since from these measurements a fitness value is determined. Thus one needs equipment that allows the user to choose a sampling frequency and store the values (in a buffer). Since it is not known in advance to which electrodes input signals should be applied, generally one needs a way of allowing the evolutionary algorithm to choose which electrodes will receive inputs (if the computational problem requires inputs) and which electrodes will be designated as outputs, and finally which electrodes will be the configuration inputs. A variety of different hardware systems have been explored for doing this.

- Digital acquisition cards together with programmable switch arrays [9]
- Mbed microcontrollers with digital to analogue converters [37]
- Purpose built platforms [4, 34]

16.5 Computational Problems

The NASCENCE consortium investigated a diverse range of computational problems. The list of problems is given below.

1. Logic gates
   a. Two-input single output Boolean functions (e.g. (N)AND, (N)OR, XOR) [4, 26, 34]
   b. Three/Four input single output Boolean functions (e.g. even-3 and 4 parity) [43]
   c. Two-input two-output Boolean functions (e.g. half adder) [4, 26, 37]
   d. Three-input, two-output Boolean functions (e.g. full-adder)
2. Travelling Salesman

   This has no inputs and as many outputs as there are cities [9]

3. Classification
    a. Standard machine learning benchmarks (Iris, Lens, banknote): number of inputs equals the number of attributes, number of outputs is equal to the number of classes [8, 44]
    b. Frequency classification: this requires one input for carrying the source signal whose frequency is to be classified and two outputs which are used to decide the class of the frequency (high or low) [45, 49]
    c. Tone discriminator: this has the same number of inputs and outputs as the frequency classifier [49]
4. Function Optimisation
    a. This has no inputs and as many outputs as there are dimensions in the function to be optimised [47, 49]
5. Bin-Packing
    a. This has no inputs and as many outputs as there are items to be placed into bins [46]
6. Robot control
    a. This has as many inputs as robot sensors and as many outputs as robot actuators (e.g. motors) [42]
7. Graph colouring
    a. This has been looked at with a single input (graph select) and as many outputs as there are nodes to be coloured. Each output selects the colour of the node [33]

The seven classes of problems cover many types of problems involving markedly different numbers of inputs, outputs and number of instances. Some problems like TSP, Function Optimisation and Bin-packing have no inputs. The material acts like a form of genetic programming and via evolved configurations generates a solution from its outputs. This is standard practice in genetic programming. In this type of approach a material is used in the genotype-phenotype mapping. However, one must be careful that in problems that have no inputs, evolution is not merely evolving configuration signals to produce outputs that are desired. In other words, that it is not directly wiring configuration signals to outputs, thus effectively ignoring the material.

In the work on evolving logic gates various functions present much greater difficulty due to their inherent non-linearity. It is well known that parity functions are difficult to evolve non-linear functions. Indeed, they have been used as benchmark problems in genetic programming for some time.

16.6 Experimental Investigations

Below we give more details on some of the computational tasks that have been used in our experimental work.

16.6.1 Travelling Salesman Problem

Solutions to TSP problems were evolved using twelve electrodes (3 × 4) and sixteen electrodes (4 × 4) [9]. Figure 16.4 shows results using a 3 × 4 electrode array for

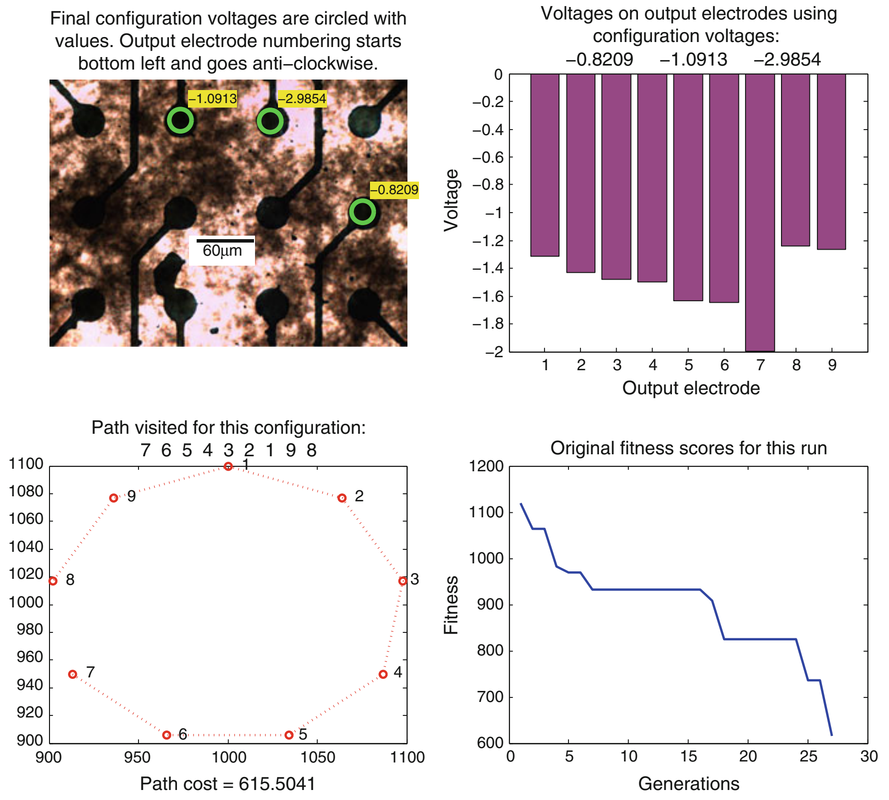

Fig. 16.4 *Top left* shows the CNT dispersal over an earlier prototype 3 × 4 grid electrodes array (×200). Note unevenness of material over electrodes and the mask fault on the third electrode does not appear to affect the evolutionary search when finding the shortest tour of the TSP (the evolutionary history is shown by the best performing genotype for each generation, *bottom right*). In this final configuration, voltages are applied to the circled electrodes and the remaining electrodes provide the floating point values into the TSP. *Top right* recorded voltages which when sorted determine the order to visit cities. *Bottom left* Optimum tour solution of the TSP [9]

a nine-city TSP problem. The particular TSP instances were generated by placing cities on a circle so that they were equidistant from one another. The genotype defined a number of real-values voltages and to which electrodes these configuration voltages would be applied. The latter was accomplished by using digitally configurable analogue cross point switches. A DAQ card first digitally configures the switch connections and then inputs analogue configuration voltages to the material and records the corresponding analogue outputs. The number of configuration voltages deployed depends on the problem being tackled and the availability of spare electrodes on the array. The configuration voltages and electrodes to which they connect were decided by a $1+4$ evolutionary algorithm. The range of voltages values was restricted to ± 3 V and all connections are one to one (i.e. one configuration voltage can only go to one electrode). Configuration voltages were applied for 1 s and a mean value of sampled voltages from the output electrodes was calculated from the last 0.2 s of sampled values. This was done to exclude any "settling periods" within the material. The time required to configure the analogue switch and set up channels on the DAQ card means that testing a configuration takes several seconds. Actually further investigations revealed that signals from the SWCNT-PMMA materials have negligible noise levels after the initial 50ms so that sampling times could be substantially reduced.

The method of obtaining a tour of cities (i.e. a permutation) is as follows. A vector of voltage values with as many elements as cities is read from the electrode array. The ith element represents city i. The vector is sorted and the city indexes form a permutation, thus defining a tour (see [9] for details). The graphic at the top right of Fig. 16.4 shows a set of voltages read from a 3×4 array. Choosing the lowest voltages (y-axis) in sequence and observing which electrode corresponds to that, one obtains the permutation: 7 6 5 4 3 2 1 9 8.

To assess the effectiveness of the technique, results using the electrode array were compared with a software-based evolutionary technique called Cartesian Genetic Programming [40] in which a graph of mathematical operations is evolved that takes a number of random real-valued inputs to produce a real-valued vector with as many elements as cities. It was found that results using the electrode array were comparable with the CGP method. It remains for future experiments to determine whether the material system scales well as the number of cities increases.

16.6.2 Classification

Using the purpose-built evolutionary platform Mecobo 3.0 and subsequently Mecobo 3.5, experiments were carried out to investigate whether well-known classification problems could be solved [44, 48]. Two relatively small problems were selected from the UCI Machine Learning repository [2]. The problems are known as Lens and Iris. We only report here on the results with the Iris dataset (see [48] for results with the Lens dataset). The Iris dataset is a list of measurements taken from one of three types of Iris flowers. It has four attributes which are classified into one of the three classes. The dataset contains 150 instances with real-valued attributes. The first fifty

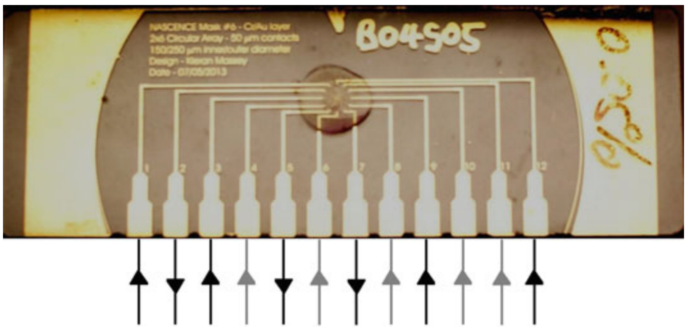
NASCENCE Mask #6 - Cr/Au layer
2x6 Circular Array - 50 μm contacts
150/250 μm inner/outer diameter
Design - Kieran Massey
Date - 07/05/2013
B04505
1
2
3
4
5
6
7
8
9
10
11
12

Fig. 16.5 Organisation of inputs, outputs and configuration inputs for a randomly chosen genotype example for Iris data classification

instances are class 1, the second fifty class two and the third set of fifty are class 3. The dataset was divided into two groups (training and test set) of 75 instances each. Each set contained exactly 25 instances of each class.

All of these experiments were performed with electrode arrays having 12 electrodes. The data inputs, outputs and configuration signals applied to the electrode array are shown in Fig. 16.5. Four electrodes were used to input data mapped from the attribute data, three electrodes were used as outputs (i.e. to define the class) and five electrodes were used as configuration inputs (shown in grey). Each output electrode was used for each output class. Each genotype defined which electrodes were outputs, inputs or received the configuration inputs. Class was decided by the leftmost output with the largest value (e.g. if the leftmost output was the largest value, the data would be designated as class one).

In the case of Mecobo 3.0, attribute information was converted by an input-mapping to the frequency of a square wave input signal. This was done by creating a linear mapping from two defined limiting square wave frequencies to the maximum and minimum attribute values [44, 48]. The output-mapping looked at the numbers of bits in the output buffers between transitions from zero to one. The length in bits between these transitions was measured and the average transition gap was used to determine the output classes. The configuration mapping took the allowed gene values and using the Mecobo 3.0 hardware, these were converted into various configuration signals. The genes defined:

- Which electrode would the signal be applied to (0–11)
- Signal type (0 or 1) indicating either a constant voltage (0 or 3.5 V) or square wave configuration signal
- Amplitude (0 or 1) deciding whether the constant applied is 0 or 3.5 V
- Frequency of square wave (500 Hz–10 KHz)
- Phase of square wave
- Duty cycle of square wave (0–100)

Thus a genotype for a twelve electrode array requires 72 genes (6 for each electrode). The first 24 genes were reserved for inputs, that is to say the first gene of each group of six would decide to which electrode the input signals would be applied to, the remaining five genes in each group are then redundant. The next 30 genes decided what kind of configuration signal would be applied to which electrode. Of the remaining three groups of six genes the first genes defined which electrode would be an output. The remaining five in each group were redundant. Thus in this way, a genotype of 72 genes could define which electrodes would receive input signals, which would supply outputs and which would receive configuration signals (of various types).

In the case of Mecobo 3.5, the amplitude of the static analogue input signal was used for input mapping by creating a linear mapping between maximum and minimum attribute values and the maximum and minimum voltages that could be applied to the electrodes. In this case, the output-mapping was simply the average of the sample values of the output buffers.

Here, the genotype is much simpler and consists of 24 genes. The first gene in each of the first four pairs defines where the inputs will be applied. The first gene in each of the last three pairs of genes defines which electrode would be chosen as an output. The remaining genes define the configuration electrodes and the voltage that is applied to those electrodes.

Twenty 1+4-evolutionary algorithms were executed over 50 generations and the two methods compared. The results are shown in Table 16.1.

The fitness calculation is based on the confusion matrix. This required counts to be made of the number of true positives *TP*, true negatives *TN*, false positives *FP*, and false negatives *FN*. The way this was done is as follows. If the predicted *p* is correct, then it is a true positive, so *TP* should be incremented. It is also a true negative for the other two classes, hence *TN* should be incremented by two. If the predicted class is incorrect, then it is a false positive for the class predicted, so *FP* should be incremented. It is also a false negative for the actual class of the instance, so *FN* should be incremented. Finally, the remaining class is a true negative, so *TN* should be incremented.

Once all instances had been classified, the fitness of a genotype was calculated using Eq. 16.1.

$$fitness = \frac{TP.TN}{(TP + FP)(TN + FN)} \tag{16.1}$$

Table 16.1 Performance results for Mecobo 3.0 and Mecobo 3.5

Accuracy	Mecobo 3.5 analogue (%)	Mecobo 3.0 digital (%)
Training	91.33	66.93
Test	86.6	60.73

Twenty evolutionary runs of 50 generations of 1+4-evolutionary algorithm were carried out using the Iris data set. Accuracy is the percentage of the training or test set correctly predicted [44]

Table 16.2 Comparative results of different percentages of SWCNT in PMMA

%SWCNT (%)	Average training accuracy (%)	Average test accuracy (%)
1.0	82.13	72.27
0.71	80.67	71.07
0.50	81.73	71.6
0.10	81.73	71.6
0.05	85.07	72.27
0.02	80.93	69.47

Ten 1+4 evolutionary runs of 500 generations were performed on the Iris dataset with Mecobo 3.0. The first column shows the weight percent fraction of SWCNT in PMMA. The second and third columns show the average training accuracy and average test accuracy found. Accuracy is the percentage of the training or test set correctly predicted. Note that no evolution was possible using percentages (by weight) of SWCNT to PMMA polymer lower than 0.02 % since output buffers contained only zeroes

So, if all instances are correctly predicted, the fitness is 1, since in this case $FP = 0$ and $FN = 0$. In the case that all instances are incorrectly predicted, $TP = 0$ and $TN = 0$, so the fitness is zero.

Statistical tests were carried out and they supported the hypothesis that solving the Iris classification is easier when evolution manipulates analogue voltages rather than more complex digital signals [44]. This also has relevance for creating a stand alone system. It would be relatively straightforward to build a circuit that could supply fixed configuration voltages to an electrode array. It would also be straightforward to build a system to automatically map numerical attribute information into applied voltages to be input to the device.

Experiments were performed to investigate how the classification results depended on the concentration of carbon nanotubes. The findings are detailed in Table 16.2.

It was found that when the percentage by weight of SWCNT in the polymer was above 0.02 % no statistical difference could be found in the results.

The classification results using Mecobo 3.5 were also compared with an implementation of CGP for classification under the same evolutionary conditions [44]. The implementation of CGP for this problem was similar to that used for the TSP problem. That is to say, the function set was a set of arithmetic and mathematical operators. The inputs were a fixed set of randomly chosen constants. There were as many outputs as classes. The class was decided by the leftmost largest output and fitness was calculated using (16.1). The results with the material turned out to be not statistically significantly different from those obtained with CGP. This again shows that evolving classifiers in materials is promising.

16.6.3 Logic Gates

A number of Boolean logic functions have been realised by applying evolution-in-materio using various evolvable platforms and materials.

Using a twelve electrode array similar to that shown in Fig. 16.2 and the Mecobo 3.0 evolution-in-materio hardware platform [34] an exhaustive search was carried out over the twelve electrodes. All possible combinations of choosing two input electrodes, one output electrode and nine configuration electrodes were examined. This experiment revealed that with the right configuration inputs any of the sixteen possible two-input Boolean functions can be obtained.

In other experiments the MBed experimental platform was used [26, 37]. Experiments were carried out to see if materials could implement threshold logic gates. In these logic gates one assumes continuous variables are divided into ranges and values in these ranges are designated logical one or zero based on whether the value of the variable is above or below certain real-valued thresholds. To obtain logical OR or AND requires just one threshold, however more complex gates such as XOR may require two. Using the derivative-free search methods Nelder-Mead [51] and Differential Evolution [63] it was shown to be possible to find thresholds that allowed a number of logic functions to be implemented, for example XOR, half and full one bit adders. On the more complex functions, Nelder-Mead proved to be less effective. In later work it was found that using concentrations of SWCNT to polymer (by weight) greater than 0.11 % consistently produced poorer results which worsened with increasing concentrations [37].

Using Mecobo 3.0 it was found to be possible to implement both even-3 and even-4 parity functions [43]. Even parity functions output one when an even number of inputs are one, and zero otherwise. It is well-known that such functions are extremely hard to find using random search and as a consequence have been frequently used as a benchmark for genetic programming methods [27]. A sixteen electrode array was used and a genetic representation similar to that used in the classification work (see Sect. 16.6.2). However, the phase of the applied square waves was not used in the study, as previous work had shown that manipulating the phase of applied square waves had no utility in problem solving. The parity problems were formulated as classification problems, so that two outputs were assumed, the largest leftmost value deciding whether the output was zero or one. Binary inputs were represented as one of two different frequency square waves (500 Hz represented logical zero and 10 KHz represented logical one). As with previous work on classification an average transition gap was calculated from the output buffers. In these experiments fitness was measured by computing a confusion matrix and using the Matthews Correlation Coefficient. Other experiments were performed where inputs and configuration signals were either 0 or 3.3V. It was found that using square waves inputs and evolving configurations that choose different square wave frequencies performed better under evolutionary search than using amplitudes [43].

We close this subsection with a brief description of recently published work [4] in which the nanoparticle networks from Fig. 16.3 were used to evolve all Boolean logic gates. In Fig. 16.6a we see an atomic force micrograph (AFM) image of a real nanoparticle network, where the two input electrodes and the output electrode are denoted by V_{IN1}, V_{IN2} and I_{OUT}, respectively. Time dependent signals in the order of a hundred mV are applied to the input electrodes as illustrated in Fig. 16.6b, and a time dependent current in the order of a hundred pA is read from the output

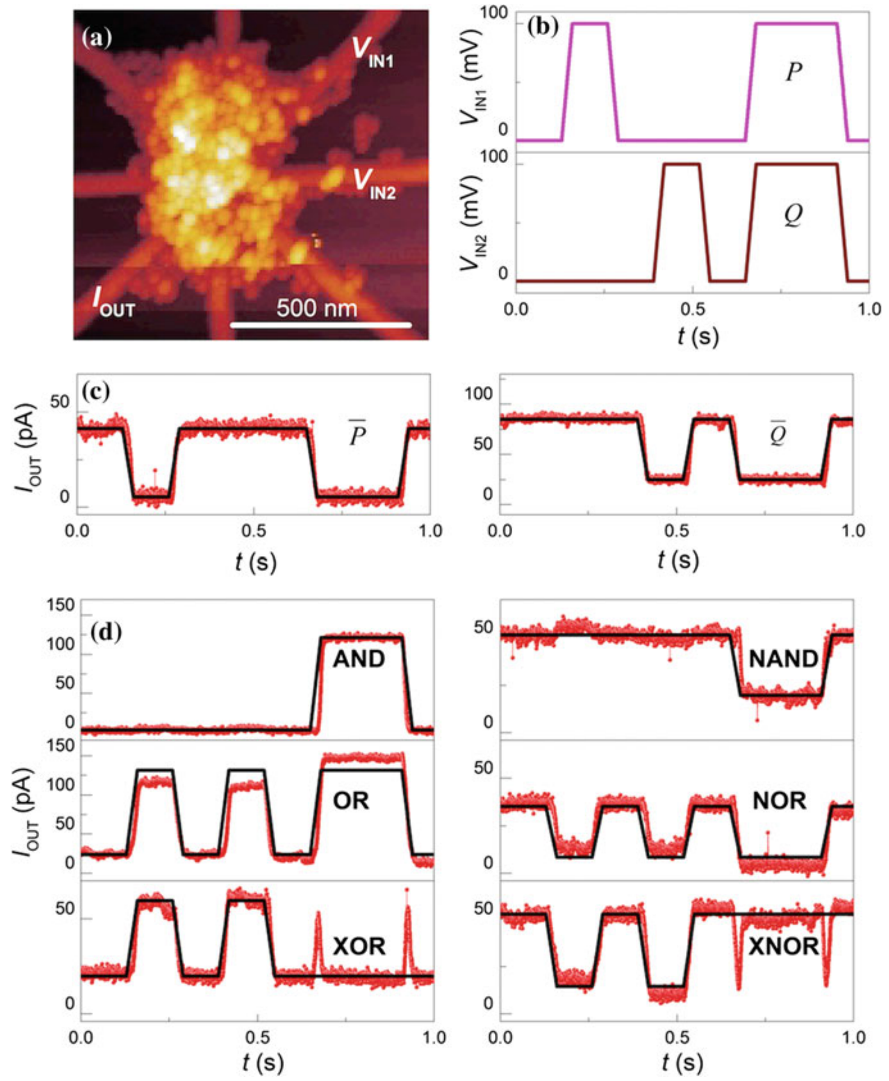

Fig. 16.6 AFM image of a nanoparticle network (**a**), the input voltages in mV applied to V_{IN1} and V_{IN2} (**b**) and the different logic outputs in pA read from I_{OUT} (**c** and **d**)

electrode. The other five electrodes and the back gate can be used to apply different sets of static configuration voltages. Using a genetic algorithm, suitable sets of configuration voltages have been found to produce the output functions of Fig. 16.6c, d. Red symbols are experimental data, solid black curves are expected output signals (matched to amplitude of experimental data). We observe two clear negators (inverters) for the input functions P and Q in Fig. 16.6c, and we observe a variety of

Boolean logic gates in Fig. 16.6d, including the universal NAND and NOR gate. All these gates show a great stability and reproducibility. For the exclusive gates (XOR, XNOR) spike-like features are observed at the rising and falling edges of the (1,1) input, as expected for a finite slope in the input signals. More details can be found in [4]. The remarkable thing here is not that we can produce logic gates using the electrical and physical properties of charge transport in neighbouring nanoparticles. It is remarkable that we can do this with one and the same sample of a disordered network of nanoparticles in a circular region of about 200 nm in diameter, and by using only six configuration voltages. A similar designed reconfigurable device based on current transistor technology would require about the same space. This shows the great potential for our approach.

16.6.4 Other Computational Problems

Below we briefly describe the results of other experimental investigations with the carbon nanotube composites.

16.6.4.1 Function Optimisation

The evolution-in-materio technique was also used to help find the minima of complex multi-dimensional mathematical functions [47, 49]. These kinds of problems are well-known in the research field of evolutionary computation and indeed there are extensive benchmark suites containing highly nonlinear multi-modal optimisation functions. Using the Mecobo 2.0 platform experiments were conducted on a suite of 23 of these functions. A similar genotype representation to that used for classification was employed in this work. However, function optimisation like TSP requires no inputs. Instead one wants to generate a vector which optimises a function. Unlike the classification work, outputs from the electrode arrays were calculated as a scaled average number of ones in the output buffers (i.e. we did not use a calculation of average transition gap). Many of the suite of benchmark functions are thirty dimensional, meaning that the vector that optimises the function in question has thirty elements. This raises an immediate problem when using an electrode array with only sixteen electrodes. In order to evolve solutions that could provide a large number of outputs a multi-chromosomal genotype was devised in which each chromosome applied configuration signals to the electrode array and an output was read. Then the next chromosome was loaded, another evolved set of configuration signals applied and the next output value was computed. Once again the results using the material were compared with the CGP technique. As before the implementation of CGP uses a small number of inputs which are random constants and the genotype represents a network of mathematical operations. It generates as many real-valued outputs as the dimension of the optimisation function. Inputs and internal node operations are defined in the interval $[-1, 1]$. The outputs are then linearly mapped into the defined ranges of the dimensions of the optimisation functions. Using this technique CGP has been compared with Differential Evolution (DE), Particle Swarm Optimization (PSO), and a Standard Evolutionary Algorithm (SEA). Comparisons showed that in 15/20 benchmarks CGP is the same or better than DE, in 19/20 cases CGP is the same or better than PSO or SEA [39].

The experiments show that in 7/23 functions the best results with the experimental material are equal to optimum results and in case of 11/23 functions the best results are very close to optimum results. In four cases the average results with the experimental material are equal to optimum results and in thirteen cases average results are very close to optimum results. In 10/23 functions the best results of experimental material are better than or equal to the best results of CGP. Given the competitiveness of the CGP technique, the results for the material are very encouraging.

16.6.4.2 Bin Packing

Bin-packing is a well-studied NP-hard problem [11]. In the bin packing problem, a total number of n_0 items, consisting of items with different weights (or sizes), have to be placed in bins. Each bin however has a maximum weight (size) capacity c_j. The objective is to place all the items in the least number of bins such that no bin has its weight (size) limit exceeded.

Scholl and Klein have collected bin-packing benchmarks [59]. The datasets are divided into three classes, according to difficulty. The best result for each dataset has been obtained by Scholl et al. [60] using an algorithm called BISON which combines a successful heuristic meta-strategy tabu search and a branch and bound procedure.

Experiments were performed with an electrode array having twelve electrodes [46]. The material consisted of PMMA and with a concentration of SWCNT equal to 0.71 % (expressed as a weight % fraction of the PMMA).

Like TSP and function optimisation bin-packing problems require no inputs. The total number of outputs required is equal to the number of items that need to be packed into bins. Since bin-packing problems typically have 50 or more items multiple chromosomes must be used. Each chromosome defines a number of configuration signals to the electrode array and the remaining outputs supply recorded values in output buffers. The number of chromosomes required is given by the number of items to be packed into bins divided by the number of outputs chosen. For instance for a problem with 50 items, if two outputs are chosen the genotype requires 25 chromosomes. Thus in this case, there will be 10 configuration signals applied for each chromosome processed.

The genotype representation was similar to that used for classification experiments. A series of output values (0 or 1) were read from a buffer of samples taken from output electrode(s). The output values read from electrode(s) were linearly mapped between values -1.0 and 1.0. These values were then used to define the index of the bin (bin_i) in which the object i of the bin packing problem would be placed. So, the total number of outputs must be equal to the total number of objects (n_o).

The linearly mapped output values, x_i corresponding to each chromosome were used to decide the bin index, bin_i which denotes which bin item, i will be placed in. Assuming the number of items is n_o, the bin index is given by Eq. 16.2.

$$bin_i = \left\lfloor n_o \frac{(x_i + 1.0)}{2 + \varepsilon} \right\rfloor \tag{16.2}$$

The floor function $\lfloor z \rfloor$ returns the nearest integer less than or equal to its argument, z. Here $epsilon$ is a very small positive quantity. Essentially, Eq. 16.2 divides the interval $[-1, 1]$ into n_o equal intervals corresponding to bins, so that the mapped output values decide which bin an item will be placed in. For instance, assuming the number of items, $n_o = 50$ if x_i is -1.0, bin_i is 0, and if x_i is 1.0, bin_i is 49.

The fitness was assessed in two stages. First the total bin overflow was subtracted from the bin capacity. This results in a negative value when some bins overflow. However, as soon as a solution is reached in which no bin is overfull, the fitness is the number of unused bins.

Twenty evolutionary runs of 5000 generations were carried out using a 1+4 evolutionary algorithm. This took more than two days to complete. On a suite of bin packing benchmarks a range of results were obtained. In some cases (on easier benchmarks) it was possible to find solutions that were near to the known minimum number of bins. Some experiments were done over more generations (25,000) and much better results were obtained. This indicates that the evolution was not stuck in a local optimum and continued to improve with more generations.

16.6.4.3 Robot Control

Experiments were also undertaken to investigate both simulated and actual robot control using a micro-electrode array [42]. Electrode arrays were used with either 12 or 16 electrodes and PMBA (polybutyl methacralate). The sample used in the experiments consisted of 1.0 % carbon nanotubes by weight (99 % by PBMA).

Robots used either six or eight IR distance sensors. The distance values were linearly mapped to determine the duty cycle of a square wave with frequency 5KHz. The duty cycle varies from 0 to 100, where a value 0 means a constant 0 V signal is applied and 100 means a constant 3.3 V signal is applied. A duty cycle of 50 means that a regular square wave of 5 KHz is applied. High values of duty cycle produce a more separated series of 3.3 V pulses.

Two outputs were taken from the electrode array corresponding to two robot motors. The fraction of ones in two output buffers sampled from the electrode array were linearly mapped to determine motor speeds.

The genotype representation was similar to classification except that applied signal phase was not used.

Robot controllers were evolved that allowed both simulated and actual robots to continuously explore mazes with as little obstacle collision as possible. In some cases, the robot controllers showed good generalisation ability in that the evolved controller could control a robot placed in a maze not seen during training (evolution). However, robot controllers transferred to real robots did not work as well as their simulated counterparts. This is not surprising as real robots differ in a number of ways. Direct evolution of control hardware with real robots was too computationally expensive to be practicable.

16.6.4.4 Graph Colouring

Using the purpose-built Mecobo platform, solutions were evolved to the well known graph colouring problem [33]. Graph colouring in our experiments was formulated as follows: given 3 different colours, colour a graph such that no two neighbouring nodes of the graph are assigned the same colour. The target device was the one capable of producing two valid 3-colourings of a simple graph with 4 nodes. Two such colourings are shown in Fig. 16.8. A Genetic Algorithm was used to construct (evolve) a device that produced such colourings in the material connected to the Mecobo board. A conceptual overview of the sought devices is depicted in Fig. 16.7. Graph select was used as input, in order to define which instance ought to be evolved. Eight configuration signals were used to configure the material and 4 outputs signals were mapped to the four graph nodes. The output voltage range was divided in 3 intervals and the highest output voltage observed on each pin was defined as the chosen colour.

The chosen problem instances are fairly simple. The goal of the experiments described herein was to investigate which type of configuration signals produced best results on the chosen problem rather than comparing the performances against known benchmarks. As such, three different groups of configuration signals were tested: only static analogue voltages as configuration signals, only square waves

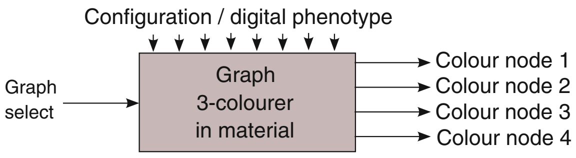

Fig. 16.7 The target device: the graph select input is used to select which of the two instances of the problem, i.e. two different colourings in Fig. 16.8, should be solved in output [33]

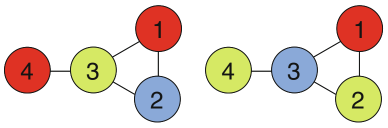

Fig. 16.8 Two valid 3-colourings of the same graph. Both are valid solutions because there are no two neighbours with the same colour. The numbers are node labels, which are mapped to outputs from the device [33]

as configuration signals, mixed configuration signals, i.e. both the previous were allowed. Each of the different signal forms produced working devices. However, the most successful signal representation was square waves, both in terms of successful working devices and number of generations required to obtain a working device.

16.6.5 Electrical Behaviour of Carbon Nanotubes

As mentioned earlier, carbon nanotube-based material may be configured by different kinds of stimuli, e.g. electrical signals, magnetic fields, temperature variations, light, etc. CNT materials have been investigated within the NASCENCE project by means of different electric signals. In particular, the following electrical signals have been considered and explored:

- Static voltages;
- Square waves;
- A mixture of the above.

In order to be able to produce any kind of computation within the underlying nanotubes network, the input data and the configuration data must allow the exploitation and manipulation of underlying physical properties in the CNT-polymer material. Moreover, such manipulated properties must be observable and give a measurable response, i.e. output response. As such, the choice of the input signal and configuration types play an important role and define which physical properties are available and utilised. In [33], a comparison of different signals have been investigated for the evolution of solutions to a well known computational problem, i.e. graph colouring (see Sect. 16.6.4.4).

In the case of static voltages, the parameters under evolutionary control are typically the physical pin to which the signal is applied to, the starting time, the ending time, and the voltage amplitude. In the experiments in [33], the range of amplitude was limited to 0–3.3 V. In the case of square wave signals, the evolved parameters are the physical pin to which the signal is applied to, the starting time, the ending time, the frequency, and the duty cycle. In the experiments in [33], the square wave amplitude was fixed at 0 and 3.3 V. The results show that it was possible to evolve working solutions using all three types of signals (static voltages only, square waves only, a mixture of static voltages and square waves). However, the choice of signal types influenced the evolutionary results. Square wave signals showed promising potential and the ability to produce rich dynamics [53].

One model of the CNT material suggested that if only static DC voltages are applied it behaves as a network of resistors [35]. It was shown that TSP problems [9] could be solved using a SPICE model of the material as a ‘cloud’ of resistors [50]. In case of applied square wave signals, the CNT material could be seen as an RC circuit, i.e. the CNT material holds capacitance. It is then possible to create macroscopic pin-to-pin models for every pin couple and thus model the material as a whole. Inspection of evolved solutions in [54] showed that the exploited physical properties are often unanticipated. In particular, it was observed that evolution was able to create and exploit signal delays, signal inversions and signal canceling. All the mentioned properties may provide a source of non-linearity and rich dynamics that may be potentially exploited for physical implementations of reservoir computing [24, 36] in CNT materials.

16.6.6 Behaviour of Gold Nanoparticles

Unlike the electrical properties and behaviour of the CNT materials we have been using within the NASCENCE project, the electrical properties and physical effect of Coulomb blockade behind the charge transport in the used gold nanoparticle networks

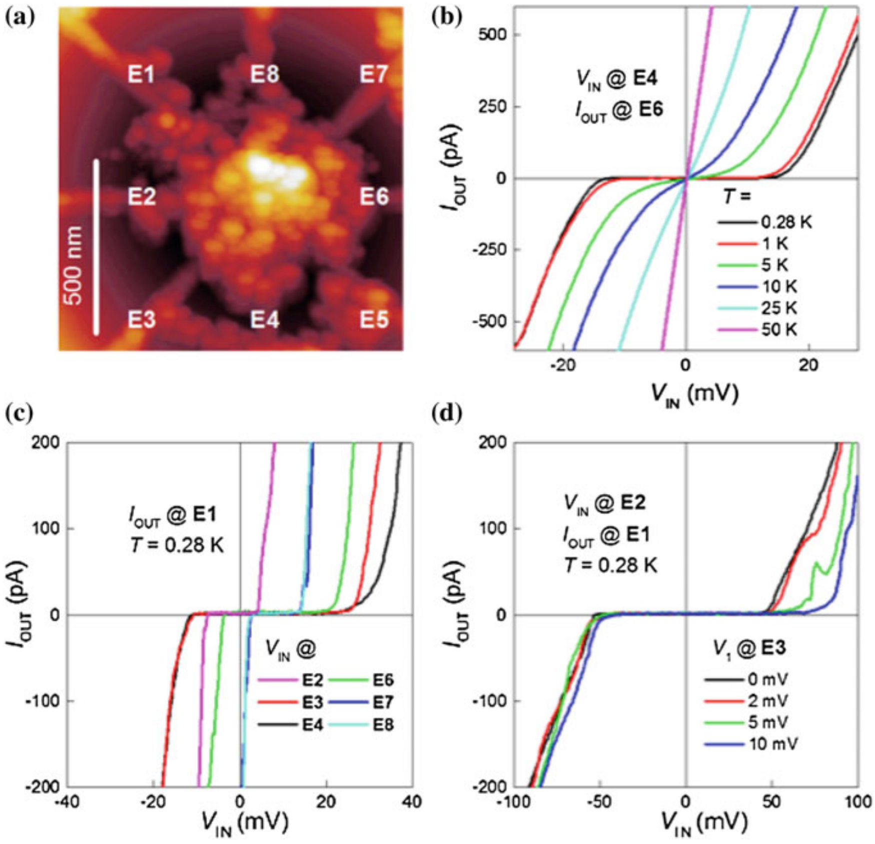

Fig. 16.9 AFM image of a nanoparticle network (**a**), the input voltages in mV applied to electrode E_4 and the output current in pA read at electrode E_6 at different temperatures (**b**) the outputs in pA read from E_1 for different input electrodes at 0.28 K (**c**), and the effect of a static voltage applied at E_3 on the I-V curves of a fixed pair at 0.28 K (**d**)

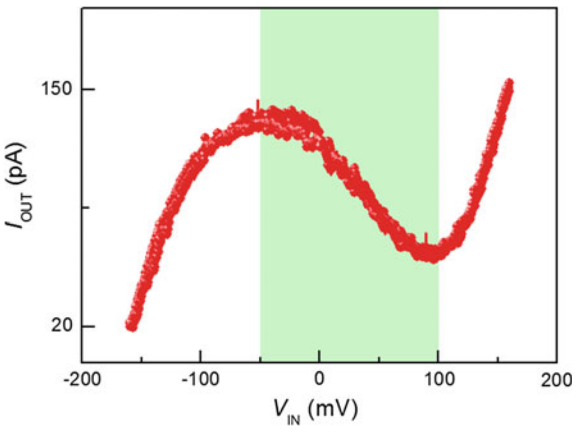

Fig. 16.10 NDR behaviour of a gold nanoparticle network: the output current I_{OUT} increases with an increasing input voltage V_{IN} in the interval $-150\,\text{mV} < V_{IN} < -50\,\text{mV}$, but the I-V curve bends down in the region of $-50\,\text{mV} < V_{IN} < 100\,\text{mV}$

are pretty well understood. As described in Sect. 16.3, under suitable energy conditions and restrictions, the charge transport is governed by the Coulomb blockade effect [25, 71]. The particles act as single electron transistors (SETs) with a high ON/OFF ratio and strong non-linear behaviour. This makes them potentially good candidates for interesting nontrivial functionalities. As an illustration, in Fig. 16.9 we included some I-V characteristics of one of the nanoparticle network we used in [4]. The nonlinear behaviour is very clear from the figures. We also observed a special form of nonlinearity usually referred to as negative differential resistance (NDR), as shown in Fig. 16.10: a gate that was evolved to be a negator (inverter) for $0\,\text{mV} < V_{IN} < 100\,\text{mV}$ exhibits NDR within the considerably larger range $-50\,\text{mV} < V_{IN} < 100\,\text{mV}$.

This behaviour is interesting and plays a key role in the evolvability of more complex functions. For instance, if we compare an XOR with an OR, then it can be observed that the OR could in principle be based on simple linear behaviour, where a high input signal gives rise to a high output signal, no matter whether both input signals are high or just one of the input signals. In case of an XOR this is different: we should only have a high output signal if precisely one of the input signals is high and the other is low; two high input signals should yield a low output signal. This is clearly possible if the evolvable system exhibits NDR behaviour.

16.7 Simulations

Apart from the experimental work and results, the NASCENCE consortium has also worked on the theoretical underpinning of the experimental work and on simulations. The latter are based on physical or mathematical models of the material systems. In case of the nanoparticle networks the physics is pretty well understood, but for the composites of nanotubes the situation is quite different and much more complex.

We start this subsection by describing four different approaches to modeling the nanotube composites, reflecting four different levels of abstraction. The subsection will be completed by a short description of the physical model that underlies the behaviour of jumping electrons in nanoparticle networks, as well as an alternative approach to simulating these networks using neural networks.

16.7.1 Physical Models

16.7.1.1 Models of Carbon Nanotube Materials

Modeling of the physical nanoscale structures may be performed at several abstraction levels, ranging from the low level local interactions between neighboring particles to the high level "black box" behavioral models. Other intermediate levels are also important, particularly for describing emergence of properties at different intermediate scales. Four different models are described here:

1. Model of computation based on collective property of wave functions which describe electrons moving within the material;
2. Model of electrical properties based on DC or AC circuits;
3. Model of conductivity based on dynamical hierarchies and cellular structure;
4. Model of abstract behavior and computational classes using cellular automata.

Model of Computation Based on Collective Electrodynamics in Aggregates of Carbon Nanotubes

Information processing performed by the CNT material is described in [29] within the framework of Ashby's systems theory [1], as introduced in classical cybernetics. Electrical properties exploited for computation arise as an emergent property of the stimulated material. At the lower level of the hierarchy, the wave functions of electrons are manifested as an electromagnetic field, which is one of the main physical phenomena manipulated for computation in an EIM setting. Even if the computation happens at the nanoscale and quantum level, what is captured by the measuring instrumentation is an approximation of the true physicality of computations in the material. As such, the electric field in the nanocomposite can be considered as a manifestation of a collective emergent property of electrons in the computing substrate. In the proposed framework, a future research direction is proposed that may consider the manipulation of quantum properties of electrons in the material so that the emerging electromagnetic properties can be used for computation. Another aspect of the proposed framework is to allow the manipulation of parameters, e.g. temperature, in the description of the system state. This may allow control of parameters during the computation and the system may be described with a bigger choice of variables. This is close to polymorphic electronics [62], where there may be different functionalities for different operating temperatures.

Model of Electrical Properties Based on DC/AC Circuits

Observing the behavior of CNT materials under varying inputs, e.g. static voltages or square wave signals, allows macroscopic modeling of pin-to-pin characteristics with simple RC circuits. In [33], two SPICE models [50] are presented, one for describing the electrical behavior of carbon nanotube materials when stimulated with static voltages applied to input pins, and one for capturing the behavior when square waves are used as manipulation signals. A simple SPICE model, consisting of a 'cloud' of resistors and connectors between them, has been successfully used to replicate results by Clegg et al. [9] for solving an instance of the traveling salesman problem. In this case, using only DC voltages as configuration parameters, the material behaves as a network of resistors. It must be noted that each sample of nanotube material contains a wide variety of such networks and the different configuration signals allow the selection of suitable networks to solve a wide variety of problems [9, 42, 46, 47]. Another model that captures the behavior of CNT materials under the influence of square waves has been proposed in [33]. This model consists of simple circuit elements such as capacitors and resistors. As such, each pin pair can be modeled by a simple RC circuit and by using one such model for each pin pair, a complete model of the material slide can be constructed.

Model of Conductivity Based on Dynamical Hierarchies and Cellular Structure

The approach in [30] aims at modeling conductivity dependence on the concentration of carbon nanotubes and varying electric potential in the material. The approach is based on two main paradigms: dynamical hierarchies [57] and cellular computation [10]. Each material sample is divided in a grid where each cell can represent a sub area of the sample, with relative content, i.e. polymer molecules, nanotubes bundles, electrodes. Each cell behaves according to the physics of the material it contains and interactions with neighboring cells. Results show that higher concentration of CNTs lead to more percolation paths and consequently more current flow. Different cell shapes may be considered for future works, e.g. dodecahedron.

Model of Abstract Behavior Using Cellular Automata

The wide variety of problems solved in CNT materials does not give any direct indication of the computational properties and computational power of the materials used. However it is clear that the materials can be exploited at the computational level required to solve the given task. Cellular automata (CA) offer a broader knowledge of different complexity levels and computational classes, e.g. Wolfram classes [74]. As such, CA models of the material may allow a framework to be established that relates measurable physical properties to abstract CA behaviors. In [15], cellular automata transition tables of different complexities have been evolved in-materio.

An interesting future direction is the possibility to evolve universal cellular automata [3, 13] in the CNT material. In addition, ongoing work attempts to relate the evolved in-materio cellular automata with CA parameters, e.g. lambda [31], and connect material computation with the notion of edge of chaos.

16.7.1.2 Models of Nanoparticle Materials

As we explained earlier, the charge transport in the nanoparticle networks we have been using in the experiments that have led to [4] is based on a physical phenomenon that is known as the Coulomb blockade effect [25, 71]. The individual gold nanoparticles act as single electron transistors (SETs). Electrons can jump between neighbouring particles when the energy conditions are favourable. One electron at a time can tunnel between two particles if sufficient energy is available (ON state), either by applying a voltage across the particle or by electrostatically shifting its potential; otherwise, the transport is blocked due to Coulomb blockade (OFF state). These disordered assemblies of nanoparticles therefore provide an almost random network of interconnected robust, non-linear, periodic switches, as a result of the Coulomb oscillations of the individual nanoparticles. We have observed experimentally that electron transport below 5 K is dominated by Coulomb blockade, and strongly depends on the used input and output electrodes, as well as on the static voltages applied to the remaining electrodes.

Due to the high costs and time consuming experiments involved in the experimental work, it was highly desirable to develop a simulation tool to explore the potential functionalities of such nanoparticle networks without the burden of spending many hours in the lab and wasting expensive resources to look for such functionalities experimentally. In addition, the simulations can also inform us on the minimum requirements that are needed for obtaining the targeted functionality if we were able to produce these nanoparticle networks according to a predetermined design. This could lead to new devices for the digital industry, possibly replacing purpose-built assemblies of transistors. Moreover, simulations can provide us with evidence concerning the scalability of our approach. Simulations can also give us new insights into the dynamics of the charge transport that might lead to a better understanding as to why and how the networks reveal the functionalities we observe. Furthermore, there are many questions on the use of these networks that are difficult to answer experimentally, because there are serious challenges in fabricating examples with smaller central gaps or with more control electrodes using the same area.

The simulation tool we developed in [70] is an extension of existing tools for simulating nanoparticle interactions, like SPICE [50] or SIMON [73]. Since the dynamics of our nanoparticle networks is governed by stochastic processes: electrons on particles can tunnel through junctions with a certain probability, there are basically two simulation methods to our disposal: Monte-Carlo Methods and the Master Equation Method [71, 72]. Since the number of particles is large, this rules out the second approach, hence the Monte-Carlo Method is the only suitable candidate. This method simulates the tunnelling times of electrons stochastically. To get meaningful results, one needs to run the algorithm in the order of a million times. Doing so, the stochastic process gives averaged values of the charges, currents, voltages, etc. More details on the simulation tool can be found in [70].

We have validated our tool for designed systems with small numbers of particles that are experimentally known from literature, and that have also been simulated before [72]. We have also used our tool to examine other structures of nanoparticle networks. Interestingly, we have shown through simulations that all Boolean logic gates that we evolved experimentally in [4] can be evolved in a regular 4×4 grid consisting of only 16 nanoparticles. We refer to [70] for more details. Currently, we are not aware of any production techniques for constructing these regular grids of nanoparticles.

Although our simulation tool can in principle handle arbitrary systems of any size, scalability is a serious issue if we consider the computation time. Even a parallellized CUDA code we have developed for a GPU does not really solve the problem if we want to simulate networks consisting of hundreds of particles. Moreover, as the networks in [4] cannot be produced according to a predefined specific design, it is not possible to use an accurate physical model for such systems.

With these drawbacks in mind, in the next subsection we present an alternative approach. This novel approach is based on training artificial neural networks in order to model and investigate the nanoparticle networks.

16.7.1.3 Neural Network Simulation Model

To support future experimental work on our evolvable systems, we developed simulation tools for predicting candidate functionalities. One of these tools that we have briefly described in the previous subsection is based on a physical model, but the one we present here is based on a neural network model.

Neural networks have proven to be powerful function approximators and have been successfully applied in a wide variety of domains [7, 28, 58, 64]. Being essentially black-boxes themselves, neural networks do not facilitate a better understanding of the underlying quantum-mechanical processes. For that purpose the physical models we described before are more appropriate. But in contrast to physical models, neural networks provide differentiable models and thus offer interesting possibilities to explore the computational capabilities of the nano-material.

Before this exploration can take place, a neural network must first be trained, using data collected from the material. In our case, since we already have a physical model and an associated validated simulation tool for the nanoparticle networks, to show that this approach is useful we can restrict ourselves in the first instance to training data obtained from the simulated material. This gives us the opportunity to predict functionalities in small nanoparticle networks, also networks that have not been fabricated yet, like the 4×4 grid structure we mentioned above. This in turn can inform electrical engineers on the minimum requirements necessary for obtaining such functionalities without the burden of costly and time-consuming fabrication and experimentation.

One of the advantages of the neural network approach is that we do not need to have any detailed information on the structure or physical properties of the material. We only need as many input-output data combinations as we can get from the simulation tool or from measurements on a particular material sample, in order to train a neural network that models this specific sample. The more independent data we use, the more accurate the trained neural network is expected to model the sample.

Another advantage of the neural network approach is that one can optimise the input configuration through gradient descent instead of performing a black-box optimisation. In other words, as soon as we have trained the neural network with sufficiently many input-output combinations, searching for arbitrary functions is very fast and can happen independently of the material or the physical model.

To show that this approach is worthwhile, in [17] we used data obtained from the physical-model-based simulations of the previous subsection to train a neural network. We show there that the neural network can model the simulated nanomaterial quite accurately. The differentiable neural network model of the evolvable nanoparticle network is then used to find logic gates, as a proof of principle.

This shows that the new approach has great potential for partly replacing costly and time-consuming experiments. We are currently using the neural network approach on real data collected from samples of the nanoparticle networks and the carbon nanotube composites. It is too early to report on the results of this approach here.

16.8 Conclusions

Evolution-in-materio is a bottom-up approach where the intrinsic underlying physics of materials is exploited as a computational medium. In contrast to a traditional design process where a computational substrate, e.g. silicon, is precisely engineered, EIM uses a bottom-up approach to manipulate materials with the aim of producing computation. This idea is rather old. Gordon Pask pioneered this work in the late 1950s, by growing neural structures (dendritic wires) in ferrous sulphate materials by electrical stimulation without computers. The EIM ideas became popular again with the work of Adrian Thompson in 1996. Thompson demonstrated that artificial evolution could utilize physical properties of an FPGA chip to solve computational problems. Miller and Downing suggested that many materials could be exploited and coined the term "evolution-in-materio" [38]. The work described in this chapter was carried out within the EU funded project NASCENCE. The goal of the project was to demonstrate that computer-controlled evolution could exploit the physical properties of carbon nanotubes / polymer nano-composites and networks of gold nanoparticles for solving difficult computational problems. Experimental results have shown that EIM is a plausible, competitive and efficient method for solving computational functions. Proof of concept has been given on several instances of problems within various complexities, and different number of inputs and outputs. In particular, solutions have been evolved in-materio for logic gates, travelling salesman problem, machine learning classification, frequency classification, tone discrimination, function optimization, bin-packing, robot control, and graph colouring. The results outlined herein are very promising and lay the foundation for further work. Future work includes the investigation of novel materials and bigger instances of the solved problems. Being able to scale-up the instances of problems tackled may allow real world applications to be targeted. The long term goal of the EIM research community is to build information processing devices by exploiting bottom-up architectures without reproducing individual components. We envision that such devices will be potentially very fast, energy efficient and rather cheap compared to traditional von Neumann-based computers.

Acknowledgments The research leading to these results has received funding from the European Community's Seventh Framework Programme (FP7/2007-2013) under grant agreement number 317662.

References

1. Ashby, W.R.: Design for a Brain, the origin of adaptive behaviour. Chapman & Hall Ltd., New York (1960)
2. Bache, K., Lichman, M.: UCI machine learning repository (2013). http://archive.ics.uci.edu/ml
3. Berlekamp, E.R., Conway, J.H., Guy, R.K.: Winning ways for your mathematical plays, vol. 4. AMC 10, p. 12 (2003)
4. Bose, S.K., Lawrence, C.P., Liu, Z., Makarenko, K.S., van Damme, R.M.J., Broersma, H.J., van der Wiel, W.G.: Evolution of a designless nanoparticle network into reconfigurable boolean logic. Nat. Nanotechnol. (2015). doi:10.1038/NNANO.2015.207
5. Broersma, H., Gomez, F., Miller, J.F., Petty, M., Tufte, G.: Nascence project: nanoscale engineering for novel computation using evolution. Int. J. Unconv. Comput. **8**(4), 313–317 (2012)
6. Cariani, P.: To evolve an ear: epistemological implications of Gordon Pask's electrochemical devices. Syst. Res. **3**, 19–33 (1993)
7. Ciresan, D.C., Meier, U., Masci, J., Schmidhuber, J.: A committee of neural networks for traffic sign classification. In: International Joint Conference on Neural Networks (IJCNN), pp. 1918–1921 (2011)
8. Clegg, K., Miller, J., Massey, M., Petty, M.: Practical issues for configuring carbon nanotube composite materials for computation. In: Proceedings of the 2014 IEEE International Conference on Evolvable Systems (ICES), pp. 61–68 (2014)
9. Clegg, K.D., Miller, J.F., Massey, M.K., Petty, M.C.: Travelling salesman problem solved 'in materio' by evolved carbon nanotube device. In: Proceedings of bthe 13th International Conference on Parallel Problem Solving from Nature - PPSN XIII. LNCS, vol. 8672, pp. 692–701. Springer (2014)
10. Codd, E.F.: Cellular Automata. Academic Press, New York (1968)
11. Coffman Jr., E.G., Garey, M.R., Johnson, D.S.: Approximation algorithms for bin packing: a survey. In: Hochbaum, D.S. (ed.) Approximation Algorithms for NP-hard Problems, pp. 46–93. PWS Publishing Co., Boston (1997)
12. Conrad, M.: The price of programmability. In: Herken, R. (ed.) The Universal Turing Machine A Half-Century Survey, pp. 285–307. Oxford University Press, Oxford (1988)
13. Cook, M.: Universality in elementary cellular automata. Complex Syst. **15**(1), 1–40 (2004)
14. Eiben, A.E., Smith, J.E.: Introduction to Evolutionary Computing. Springer, New York (2003)
15. Farstad, S.: Evolving cellular automata in-materio. In: Master Thesis Semester Project, Norwegian University of Science and Technology, Supervisor: Stefano Nichele, Gunnar Tufte. NTNU (2015)

16. Greenwood, G., Tyrrell, A.M.: Introduction to Evolvable Hardware. IEEE Press, New Jersy (2007)
17. Greff, K., van Damme, R., Koutník, J., Broersma, H., Mikhal, J., Lawrence, C., van der Wiel, W., Schmidhuber, J.: Unconventional computing using evolution-in-nanomaterio: neural networks meet nanoparticle networks. Preprint (2015)
18. Harding, S., Miller, J.F.: Evolution in materio: a tone discriminator in liquid crystal. In: Proceedings of the Congress on Evolutionary Computation 2004 (CEC'2004), vol. 2, pp. 1800–1807 (2004)
19. Harding, S.L., Miller, J.F.: Evolution in materio: evolving logic gates in liquid crystal. Int. J. Unconv. Comput. **3**(4), 243–257 (2007)
20. Harding, S.L., Miller, J.F., Rietman, E.A.: Evolution in materio: exploiting the physics of materials for computation. Int. J. Unconv. Comput. **4**(2), 155–194 (2008)
21. Harding, S., Miller, J.F.: Evolution in materio. In: Meyers, R.A. (ed.) Encyclopedia of Complexity and Systems Science, pp. 3220–3233. Springer, Berlin (2009)
22. Higuchi, T., Liu, Y., Yao, X.: Evolvable hardware. Springer, New York (2006)
23. Holland, J.H.: Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control and Artificial Intelligence. MIT Press, Cambridge (1992)
24. Jaeger, H.: The "echo state" approach to analysing and training recurrent neural networks-with an erratum note. Bonn, Germany: German National Research Center for Information Technology GMD. Technical Report 148, 34 (2001)
25. Korotkov, A.: Coulomb Blockade and Digital Single-Electron Devices, pp. 157–189. Blackwell, Oxford (1997)
26. Kotsialos, A., Massey, M.K., Qaiser, F., Zeze, D.A., Pearson, C., Petty, M.C.: Logic gate and circuit training on randomly dispersed carbon nanotubes. Int. J. Unconv. Comput. **10**, 473–497 (2014)
27. Koza, J.: Genetic Programming: On the Programming of Computers by Natural Selection. MIT Press, Cambridge (1992)
28. Krizhevsky, A., Sutskever, I., Hinton, G.E.: Imagenet classification with deep convolutional neural networks. In: Advances in Neural Information Processing Systems (NIPS 2012), p. 4 (2012)
29. Laketić, D., Tufte, G., Lykkebø, O.R., Nichele, S.: An explanation of computation - collective electrodynamics in blobs of carbon nanotubes. In: Proceedings of 9th EAI International Conference on Bio-inspired Information and Communications Technologies (BIONETICS), IN PRESS. ACM (2015)
30. Laketić, D., Tufte, G., Nichele, S., Lykkebø, O.R.: Bringing colours to the black box - a novel approach to explaining materials for evolution-in-materio. In: Proceedings of 7th International Conference on Future Computational Technologies and Applications. XPS Press (2015)
31. Langton, C.G.: Computation at the edge of chaos: phase transitions and emergent computation. Phys. D: Nonlinear Phenom. **42**(1), 12–37 (1990)
32. Layzell, P.: A new research tool for intrinsic hardware evolution. In: Proceedings of The Second International Conference on Evolvable Systems: From Biology to Hardware. LNCS, vol. 1478, pp. 47–56 (1998)
33. Lykkebø, O., Tufte, G.: Comparison and evaluation of signal representations for a carbon nanotube computational device. In: Proceedings 2014 IEEE International Conference on Evolvable Systems (ICES), pp. 54–60 (2014)
34. Lykkebø, O.R., Harding, S., Tufte, G., Miller, J.F.: Mecobo: A hardware and software platform for in materio evolution. In: Ibarra, O.H., Kari, L., Kopecki, S. (eds.) Unconventional Computation and Natural Computation. LNCS, pp. 267–279. Springer International Publishing, Cham (2014)
35. Lykkebø, O., Nichele, S., Tufte, G.: An investigation of square waves for evolution in carbon nanotubes material. In: Proceedings of the 13th European Conference on Artificial Life (ECAL2015), pp. 503–510. MIT Press (2015)
36. Maass, W., Natschläger, T., Markram, H.: Real-time computing without stable states: a new framework for neural computation based on perturbations. Neural Comput. **14**(11), 2531–2560 (2002)

37. Massey, M.K., Kotsialos, A., Qaiser, F., Zeze, D.A., Pearson, C., Volpati, D., Bowen, L., Petty, M.C.: Computing with carbon nanotubes: optimization of threshold logic gates using disordered nanotube/polymer composites. J. Appl. Phys. **117**(13), 134903 (2015)
38. Miller, J.F., Downing, K.: Evolution in materio: looking beyond the silicon box. In: Proceedings of NASA/DoD Evolvable Hardware Workshop, pp. 167–176 (2002)
39. Miller, J.F., Mohid, M.: Function optimization using Cartesian genetic programming. In: Genetic and Evolutionary Computation Conference(GECCO) Companion, pp. 147–148 (2013)
40. Miller, J.F., Thomson, P.: Cartesian genetic programming. In: Poli, R., Langdon W.B., et al. (eds.) Proceedings of EuroGP 2000. LNCS, vol. 1802, pp. 121–132. Springer (2000)
41. Miller, J.F., Harding, S.L., Tufte, G.: Evolution-in-materio: evolving computation in materials. Evol. Intell. **7**, 49–67 (2014)
42. Mohid, M., Miller, J.: Evolving robot controllers using carbon nanotubes. In: Proceedings of the 13th European Conference on Artificial Life (ECAL2015), pp. 106–113. MIT Press (2015)
43. Mohid, M., Miller, J.: Solving even parity problems using carbon nanotubes. In: 2015 15th UK Workshop on Computational Intelligence (UKCI). IEEE Press (2015, in press)
44. Mohid, M., Miller, J.: Evolving solution to computational problems using carbon nanotubes. Int. J. Unconv. Comput. (2016, in press)
45. Mohid, M., Miller, J., Harding, S., Tufte, G., Lykkebø, O., Massey, M., Petty, M.: Evolution-in-materio: a frequency classifier using materials. In: Proceedings of the 2014 IEEE International Conference on Evolvable Systems (ICES): From Biology to Hardware, pp. 46–53. IEEE Press (2014)
46. Mohid, M., Miller, J., Harding, S., Tufte, G., Lykkebø, O., Massey, M., Petty, M.: Evolution-in-materio: solving bin packing problems using materials. In: Proceedings of the 2014 IEEE International Conference on Evolvable Systems (ICES): From Biology to Hardware, pp. 38–45. IEEE Press (2014)
47. Mohid, M., Miller, J., Harding, S., Tufte, G., Lykkebø, O., Massey, M., Petty, M.: Evolution-in-materio: solving function optimization problems using materials. In: 2014 14th UK Workshop on Computational Intelligence (UKCI), pp. 1–8. IEEE Press (2014)
48. Mohid, M., Miller, J.F., Harding, S.L., Tufte, G., Lykkebø, O.R., Massey, M.K., Petty, M.C.: Evolution-in-materio: solving machine learning classification problems using materials. In: Proceedings of the 13th International Conference on Parallel Problem Solving from Nature - PPSN XIII. LNCS, vol. 8672, pp. 721–730. Springer (2014)
49. Mohid, M., Miller, J., Harding, S., Tufte, G., Massey, M., Petty, M.: Evolution-in-materio: Solving computational problems using carbon nanotube-polymer composites. Soft Comput. (2016, in press)
50. Nagel, L., Pederson, D.: Simulation program with integrated circuit emphasis. Memorandum ERL-M382, University of California, Berkeley (1973)
51. Nelder, A., Mead, R.: A simplex method for function minimization. Comput. J. **7**, 308–313 (1965)
52. Neumann, J.v.: First draft of a report on the EDVAC. Technical report, University of Pennsylvania (1945)
53. Nichele, S., Laketić, D., Lykkebø, O.R., Tufte, G.: Is there chaos in blobs of carbon nanotubes used to perform computation? In: Proceedings of 7th International Conference on Future Computational Technologies and Applications. XPS Press (2015)
54. Nichele, S., Lykkebø, O.R., Tufte, G.: An investigation of underlying physical properties exploited by evolution in nanotubes materials. In: Proceedings of 2015 IEEE International Conference on Evolvable Systems. IEEE Symposium Series on Computational Intelligence, IN PRESS. IEEE (2015)
55. Pask, G.: Physical analogues to the growth of a concept. Mechanisation of Thought Processes, no. 10 in National Physical Laboratory Symposium, pp. 877–922. Her Majesty's Stationery Office, London, UK (1958)
56. Poli, R., Langdon, W.B., McPhee, N.F.: A Field Guide to Genetic Programming. Lulu Enterprises, UK Ltd (2008)

57. Rasmussen, S., Baas, N.A., Mayer, B., Nilsson, M., Olesen, M.W.: Ansatz for dynamical hierarchies. Artif. Life **7**(4), 329–353 (2001)
58. Sak, H., Senior, A.W., Beaufays, F.: Long short-term memory based recurrent neural network architectures for large vocabulary speech recognition. CoRR **abs/1402.1128** (2014). http://arxiv.org/abs/1402.1128
59. Scholl, A., Klein, R.: Bin packing. http://www.wiwi.uni-jena.de/Entscheidung/binpp/index.htm
60. Scholl, A., Klein, R., Jürgens, C.: Bison: a fast hybrid procedure for exactly solving the one-dimensional bin packing problem. Comput. Oper. Res. **24**(7), 627–645 (1997)
61. Sekanina, L.: Evolvable components: From Theory to Hardware Implementations. Natural Computing. Springer (2004)
62. Sekanina, L.: Design methods for polymorphic digital circuits. In: Proc. of the 8th IEEE Design and Diagnostics of Electronic Circuits and Systems Workshop DDECS, pp. 145–150 (2005)
63. Storn, R., Price, K.: Differential evolution — a simple and efficient heuristic for global optimization over continuous spaces. J. Glob. Opt. **11**(4), 341–359 (1997)
64. Sutskever, I., Vinyals, O., Le, Q.V.: Sequence to sequence learning with neural networks. In: Ghahramani, Z., Welling, M., Cortes, C., Lawrence, N.D., Weinberger, K.Q. (eds.) Advances in Neural Information Processing Systems 27: Annual Conference on Neural Information Processing Systems 2014, 8–13 December 2014, Montreal, Quebec, Canada, pp. 3104–3112 (2014). http://papers.nips.cc/paper/5346-sequence-to-sequence-learning-with-neural-networks
65. Thompson, A.: An evolved circuit, intrinsic in silicon, entwined with physics. In: T. Higuchi, M. Iwata, L. Weixin (eds.) Proceedings of the 1st International Conference on Evolvable Systems (ICES'96). LNCS, vol. 1259, pp. 390–405. Springer (1997)
66. Thompson, A.: Hardware evolution: automatic design of electronic circuits in reconfigurable hardware by artificial evolution. Distinguished dissertation series. Springer (1998)
67. Thompson, A., Harvey, I., Husbands, P.: Unconstrained evolution and hard consequences. In: Sanchez, E., Tomassini, M. (eds.) Towards Evolvable Hardware: The Evolutionary Engineering Approach. LNCS, vol. 1062, pp. 136–165. Springer (1996)
68. Thompson, A., Layzell, P., Zebulum, R.S.: Explorations in design space: unconventional electronics design through artificial evolution. IEEE Trans. Evol. Comput. **3**(3), 167–196 (1999)
69. Turing, A.M.: On computable numbers, with an application to the entscheidungsproblem. Proc. Lond. Math. Soc. **42**(2), 230–265 (1936)
70. van Damme, R., Broersma, H., Mikhal, J., Lawrence, C., van der Wiel, W.: A simulation tool for evolving functionalities in disordered nanoparticle networks. Preprint (2015)
71. Wasshuber, C.: Computational Single-Electronics. Springer, Berlin (2001)
72. Wasshuber, C.: Single-Electronics – How it works. How it's used. How it's simulated. In: Proceedings of the International Symposium on Quality Electronic Design, pp. 502–507 (2012)
73. Wasshuber, C., Kosina, H., Selberherr, S.: A simulator for single-electron tunnel devices and circuits. IEEE Trans. Computer-Aided Des. Integr. Circuits Syst. **16**, 937–944 (1997)
74. Wolfram, S.: Universality and complexity in cellular automata. Phys. D: Nonlinear Phenom. **10**(1), 1–35 (1984)
75. Yoshihito, A.: Information processing using intelligent materials - information-processing architectures for material processors. Intell. Mater. Syst. Struct. **5**, 418–423 (1994)
76. Zauner, K.P.: From prescriptive programming of solid-state devices to orchestrated self-organisation of informed matter. In: Banâtre, J.P., Fradet, P., Giavitto, J.L., Michel, O. (eds.) Unconventional Programming Paradigms: International Workshop UPP 2004, vol. 3566, pp. 47–55. Springer (2004)
77. Zebulum, R., Pacheco, M., Vellasco, M.: Evolutionary Electronics – Automatic Design of Electronic Circuits and Systems by Genetic Algorithms. The CRC Press International Series on Computational Intelligence (2002)