Solving Even Parity Problems Using Carbon Nanotubes

Maktuba Mohid and Julian F. Miller

Department of Electronics, University of York, York, UK. Email: [mm1159, julian.miller]@york.ac.uk

***Abstract*—Natural evolution has been exploiting the properties of materials ever since life first appeared. Evolution-in-materio (EIM) is a technique that is inspired by this. In EIM, computer-controlled evolution is used to directly manipulate or tune the properties of physical materials so that computational problems can be solved. The thinking behind this is that materials have many internal physical interactions that could be exploited by computer-controlled evolution to solve a computational problem. We show that using a purpose-built hardware platform called Mecobo, it is possible to evolve voltages and signals applied to a mixture of carbon nanotubes and a polymer to build Boolean even-parity functions. We also investigate the merits of different ways of mapping the problem inputs to physical variables that can be applied to the material and different ways of mapping sampled physical outputs from the material to computational outputs. This is the first time that even-parity problems have been solved with evolution-in-materio. We compare the results with a well-known and highly effective form of genetic programming, called Cartesian Genetic Programming, with encouraging results.**

***Keywords*—evolutionary algorithm, evolution-in-materio, material computation, evolvable hardware, even parity, Boolean problem**

I. Introduction

Evolution-in-materio (EIM) was proposed by Miller and Downing [12]. It is a method that "programs" materials by evolving physical signals so that the material carries out some desired input-output transformation, without requiring a detailed understanding of the internal physics of the material. It was inspired by the earlier work of Adrian Thompson who investigated whether it was possible for unconstrained evolution to evolve working electronic circuits using a silicon chip called a Field Programmable Gate Array. He evolved a digital circuit that could discriminate between 1kHz or 10kHz signal (tone discriminator) [18]. After analysis, Thompson discovered that artificial evolution had exploited physical properties of the chip. Harding and Miller later showed that liquid crystal could be used in place of silicon and examined three types of computational problems: two-input logic gates [6], tone discriminators [4] and robot control [5]. Indeed, they found that it was easier to evolve tone discriminators with liquid crystal than with silicon. For a recent review of EIM see [13].

To carry out EIM experiments, we have used a hardware platform called Mecobo which has been purpose-built for EIM within an EU funded research project called NASCENCE [2]. A full description of the hardware is given in [10]. The computational material is a mixture of amorphous carbon nanotubes embedded in an insulating polymer. In this paper, we investigated whether we could evolve solutions to even-parity functions of three and four inputs. We chose these functions because they have often been used in the field of Genetic Programming (GP) as a benchmark. We also wanted to compare the experiments using a physical material with a well known and highly effective form of GP, known as Cartesian Genetic Programming (CGP).

EIM has been used before to build small Boolean functions. Harding et al. evolved liquid crystal to get digital circuits of up to two variables (NOT, AND, NAND, OR, NOR and XOR) [6]. Even the mixture of carbon nanotubes and a polymer was used before to build logic circuits (NOT, AND, NAND, OR, NOR XOR) using the Mecobo hardware by Lykkebø et al. [10]. Kotsialos et al. evolved some Boolean functions of three inputs and two outputs using multi-threshold gates: (AB+BC), (AB, A+B), half adder, full adder using the mixture of carbon nanotubes and a polymer, but with a different hardware platform [8]. The hardware platform was an mbed microcontroller (NXP LPC1768) and some additional electronics (A/D and D/A, converters, operational amplifier).

Other than the Boolean problems, many other computational problems have been investigated recently using EIM with a mixture of carbon nanotubes and a polymer. These investigations have considered machine learning classification [17], [14], function optimisation [16], bin packing [15] and traveling salesman problems [3]. In particular, the classification experiments (with Iris dataset) compared the performances of different mixtures of materials and varied organisations of electrodes. However, the experiments showed that the different organisations of electrodes do not matter and the choice of polymer plays no important role in computation. However, according to the investigation results using bin packing problem it has been found that no evolution is possible with the SWCNT percentage lower than 0.02% or with empty electrodes, where the values of output buffers are always zero [15].

Our aim in this paper is to show that EIM can be applied to standard computational benchmark problems so that we have a yardstick to assess various aspects of EIM using the Mecobo platform. We also have investigated what type of signals that interface with the material are appropriate and give the best results. At present, manipulating materials in an evolutionary algorithm (EA) has some drawbacks. The main one is that it is slow (see later). This means that we can only feasibly evaluate relatively few potential solutions. At present, the slowness of

the method is largely caused by the conventional interfacing hardware. The underlying computational speed of the material is extremely high. EIM is new approach to the solution of computational problems which has barely been explored in the literature and can be seen as a new form of analogue computation. As the technology is developed it could offer advantages over conventional computational methods (e.g. in speed, power). EIM has been discussed in detail in [13].

Ever since Koza's work on GP, evolutionary computation has been widely used to solve n-bit even parity problems [9]. Miller solved even parity-3,4,5 successfully with CGP with function set {and, or, nand, nor} [11] and found that extremely low populations are most effective. He also showed that a form of neighbourhood search known in the field of evolutionary computation as a (1+λ)-Evolutionary Strategy (ES) was a very effective and was more efficient than the GP. Using the same EA as CGP and under as identical conditions as possible, we show that materials can be programmed to solve the parity problems with reasonable efficiency.

The organization of the paper is as follows. In Sect. II we give a conceptual overview of EIM. We describe the Mecobo EIM hardware platform in Sect. III. The preparation and composition of the physical computational material is described in Sect. IV. Sect. V describes the even parity problem. The way we have used the Mecobo platform for solving even parity problems is described in Sect. VI. We describe our experiments and the results in Sect. VII. Finally we conclude and offer suggestions for further investigation in Sect. VIII.

II. Conceptual Overview Of Evolution-In-Materio

EIM involves both a physical material and a digital 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 instructions. 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 the physical inputs known as configurations. A string of numerical data (the genotype) is held on the computer and is transformed into configuration instructions. 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 [13]. One can think of the configuration data as an evolved program that configures the material so that it can carry out the desired function. The ultimate aim is to use the programmed material in a standalone device. The conceptual overview of EIM has been shown in figure 1. To use this device to carry out computation, one must devise mappings for how the task problem inputs will be mapped to suitable physical variables (input-mapping) and how the output signals from the device will be measured, stored and converted to task problem outputs (output-mapping). In addition, one must decide what physical variables will be applied to the material so that its physical state will be altered. Various ways of doing this have been investigated in this paper.

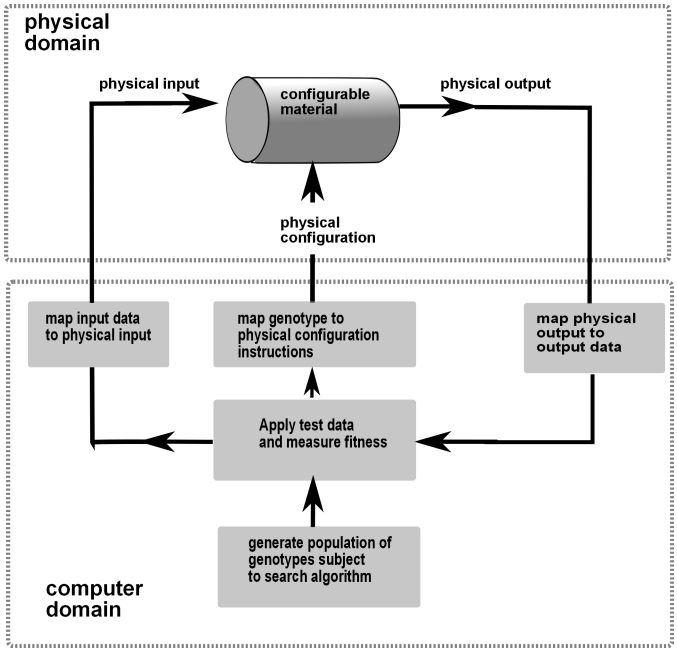

Fig. 1: Concept of EIM [13].

EIM is inspired by natural evolution which itself operates in the physical world and exploits the properties of protein complexes of enormous complexity and degrees of freedom. Traditional engineering techniques require a complete understanding of the physical properties of sub-components in order to construct high-level functions. In contrast, EIM's interesting feature is that an EA may be to exploit physical effects in the material for computational purposes *without* requiring a detailed understanding of low-level components. It is hypothesised that this may increase evolvability. In contrast, software-only systems use highly constrained representations with very limited and prescribed degrees of freedom.

III. Mecobo: an Evolution-In-Materio Hardware Platform

Mecobo is a purpose built hardware/software platform designed to allow EIM to be applied to a large variety of materials [10]. The hardware allows the possibility to map input, output and configuration terminals, signal properties and output monitoring capabilities in various ways.

The Mecobo hardware allows only two types of inputs to the material: constant voltage (0V or 3.5V) or a square wave signal of programmable frequency and cycle time. The cycle time is the percentage of the period of a square wave signal that is 1. The start time and end time of each input signal determine how long an input is applied. Mecobo only samples using digital voltage thresholds, hence the material output is interpreted as strictly high or low, (i.e. 0 or 1). Note that in the case that an electrode is chosen to be read, a user-defined output sampling frequency and the start and end times determine the size of the output buffer associated with the electrode. Analogue inputs and outputs are possible in case of next version of this hardware.

Fig. 2: Slide with two electrode arrays and two samples having 16 electrodes.

IV. Description Of Physical Computational Material

The material sample is a mixture of single-walled carbon nanotubes and an insulator called polybutyl methacrylate (PBMA). Initially, the sample is dissolved in a surfactant called anisole (methoxybenzene) [1]. The sample is baked causing the anisole to evaporate. The concentration of carbon nanotube used in our experiments is 1.0% (weight% fraction of PBMA).

Why carbon nanotubes? Carbon nanotubes are conducting or semi-conducting and role of the PBMA is to introduce insulating regions within the nanotube network, to create non-linear current versus voltage characteristics. The idea is that this might show some interesting computational behavior. We see this material as a kind of random analogue electronic circuit with a large range of possible behaviours. However, to obtain *desired* behaviour one must electrically stimulate the material in the right way. One must also choose appropriate electrical signals to present to the material and one must read appropriate signals. Since such signal are unknown in advance we require a form of automated search. The preparation of experimental material is given below:

- $\approx$20 μL of material are dispensed onto the electrode array;
- This is dried at $\approx 85^o$ C for $\approx$30 min to leave a "thick film";
- To prevent stress built up in the material, this is allowed to cool to room temperature on the hotplate for $\approx$1.5 h.

Two electrode arrays are placed on each microscope slide. One drop of experimental material is placed in the middle of each electrode array. Sixteen gold electrodes (eight electrodes on each side) are connected directly with each sample on the electrode array. This electrode arrangement is shown in Fig. 2. The electrode array is wired directly with the Mecobo board via a suitable connector.

V. Even Parity Problem

The n-bit parity function has n binary inputs and a single binary output. In the case of even parity problem, the output is one only when there are an even number of ones in the input stream. The problem gets harder as the number of inputs increases. It is well known in the field of GP that finding functions that represent even parity functions are very difficult to find when carrying out a random search of all GP trees with function set {and, or, nand, nor} [9].

[1]Kieran Massey and Mike Petty of Durham University prepared the materials used as substrates and the electrode masks for our experiments

VI. Solving Even Parity Problems Using EIM

A. Methodology

We represent Boolean problems as two class problem in which we associate each output value with a particular class. If the output of even parity functions is 0, it is interpreted to be class 1, otherwise, it is class 2.

The experiments investigate the effectiveness of a variety of input and output mappings. We have mapped the Boolean inputs, '0' or '1' of the targeted parity functions to two different values of three types of physical inputs to the material sample: square-wave frequency, cycle time of a fixed frequency square-wave and voltage (amplitude). We have also investigated two output-mappings whereby an output can be determined using measurements of sampled physical outputs from the material sample (stored in an output buffer). One output-mapping uses the average gap between transitions from zero to one in the output buffer to determine an output value, the other uses the percentage of ones in the output buffer.

Eight sets of experiments were performed. The first six sets were aimed at solving the even parity-3 problem (A-F) and last two sets were aimed at the even parity-4 problem (G-H). This means, in first six sets of even parity experiments (A-F), the number of inputs were 3 and in last two sets (G and H), the number of inputs were 4. In case of set A-D and F, three electrodes (three inputs of the problem) were used for inputting the signal to be classified, 2 electrodes were used as outputs and 11 electrodes were used as configuration inputs. In case of set E, three electrodes were used as inputs, one electrode was used for output and the remaining 12 electrodes were used as configuration inputs. In case of set G and H, four electrodes (four inputs of the problem) were used as inputs, 2 electrodes were used as outputs and 10 electrodes were used as configuration inputs. Each evolved string (chromosome) defined which electrodes were either outputs, inputs (receive square waves or constant voltage) or received the configuration data (square waves or constant voltage).

The results of the first three sets (A-C) were used to compare the performances of different input mappings and output determination methods, where set A, B and C used frequency, cycle time and amplitude for input mapping respectively. Set A used average transition gap and set B, C used percentages of ones for classifying outputs.

The results of set C and D were used to compare the performances of different types of configuration inputs, where set D used only static voltages and set C used a mixture of static voltages and square waves for configuration inputs. In both of these cases, the input signals were static voltages. The same type of comparison (performances of different types of configuration inputs) was performed using set B and F, where set F used only square waves and set B used a mixture of static voltages and square waves for configuration inputs. In both of these cases, the input signals were square waves.

The results of set D and E were used to compare the merits of using different numbers of electrodes to classify the Boolean output values, where set D used two electrodes and set E used only one electrode for outputs. Both set D and E used static voltages for inputs and configuration inputs.

Experiments G used square-wave frequency for the input mapping, and the average transition gap for classifying outputs. Experiments H used amplitudes for the input mapping, and percentages of ones for classifying outputs. Set H used only static voltages and set G used a mixture of static voltages and square waves for configuration inputs.

In case of all sets of experiments, the input-output timing was 25 milliseconds and the output sampling frequency was 25KHz.

In the evaluation of each chromosome, 8 test cases were used in set A-F and 16 test cases were used in set G-H. A summary of the experiments is shown in Table I.

B. Genotype Representation

For all experiments, we used $n_e = 16$ electrodes. In experiments A-C and G, associated with each electrode there were five values (genes) which either define which electrode was used as an input or output or configuration input, or characteristics of the input applied to the electrode: signal type, amplitude, frequency, cycle (Section III). So, each collection of genes (known as a chromosome) required a total of 5x16=80 genes.

In experiments F, four genes were associated with each electrode and these defined which electrode was used as an input or output or configuration input, or characteristics of the input applied to the electrode: amplitude, frequency, cycle (only the square waves were used for inputs and configuration inputs). This requires 4x16=64 genes.

In case of experiments D, E and H, associated with each electrode there were two genes which either define which electrode was used as an input, output, configuration, or amplitude (either value 0 or 1) of the input applied to the electrode. So, each chromosome required a total of 2x16=32 genes.

In all experiments, the mutational offspring were created from a parent genotype by mutating a single gene (i.e. one gene of 80 in experiments A-C, G; one gene of 64 in experiments F; one gene of 32 in experiments D, E, H).

The values that genes could take are shown in Table II. i takes values 0, 1, ... 15.

For instance, the genotype for a 80 gene chromosome is shown below:

$p_0 s_0 a_0 f_0 c_0 \ldots p_{15} s_{15} a_{15} f_{15} c_{15}$

C. Input Mapping

In experiments A and G, the inputs to the electrode array were square waves of a particular frequency. The frequency was determined by a linear mapping of attribute data. This is described as follows.

Denote the i^{th} attribute in a dataset by I_i, where i takes values $1, 2, 3$ or $1, 2, 3, 4$ in case of set A and G respectively. Denote the maximum and minimum value taken by this attribute in the whole data set by $I_{i_{max}}$ and $I_{i_{min}}$ respectively. Denote the maximum and minimum allowed frequencies by F_{max} and F_{min} respectively. Then the linear mapping given in Equation 1 allows the i^{th} attribute of an instance, I_i to map to a square-wave frequency F_i which was applied to a given electrode.

$$F_i = a_i I_i + b_i \tag{1}$$

where the constants a_i and b_i are found by setting I_i and F_i to their respective maximum and minimum and solving for a_i and b_i.

$$a_i = (F_{max} - F_{min})/(I_{i_{max}} - I_{i_{min}}) \tag{2}$$

and

$$b_i = (F_{min} I_{i_{max}} - F_{max} I_{i_{min}})/(I_{i_{max}} - I_{i_{min}}) \tag{3}$$

In the experiments, $F_{min} = 500$Hz and $F_{max} = 10000$ Hz. The cycle time and amplitude of input signal were set to 50% and 1 respectively.

In experiments B and F, cycle time was used instead of frequency for input mapping, where maximum cycle time was 75 and minimum cycle time was 25. The mapping equation was the same as Equation 1, but instead of frequency, cycle time was used in equation. The frequency of input square wave signal was set to 5KHz and its amplitude was set to one (i.e. 3.5 V).

In experiments C-E and H, amplitude was used instead of frequency for input mapping, where maximum amplitude was 1 and minimum amplitude was 0. The mapping equation was the same as Equation 1, but instead of frequency, amplitude was used in the equation. The input signals were static voltages.

D. Output Mapping

The class that an instance belonged to was determined by examining the output buffers which contain samples taken from the output electrodes. Mecobo can only recognize binary values, so the output buffers contain a bitstring. So, in experiments A and G, the *transitions* from 0 to 1 in the output buffers were used to calculate the class that an instance belonged to. For each output buffer the positions of transitions were recorded and the gaps between consecutive transitions were measured and an average calculated. A transition based mapping was used as it is frequency related. Since, instance data affects the frequencies of applied signals, it seemed natural to use method of reading output buffer bitstrings that is itself frequency related. An example of average gap calculation for an output electrode is shown in Figure 3

However, in experiments B-F and H, percentage of ones in the output buffer was used to determine output classes. The thinking behind using a percentage of ones is that may be useful when inputs are cycle time or amplitude related.

In case of all experiments except the E, two electrodes were used for outputs. The class associated with an output electrode was determined by the output buffer with lower value (average transition gap or percentage of ones). If first electrode had lower value, it was designated to be class one, otherwise it was designated to be class two. So, the first output electrode was expected to have lower value only if the output class was 1. Thus, if the solution works as desired, it would have class

TABLE I: The experimental settings of all sets of even parity experiments. The "In. Sig." and "Conf. Volt." columns show the used input signal (SW=square wave, S=static voltages) and configuration inputs (SW=square wave, S=static, M=mixture of static voltages and square waves) of the experiments respectively. The "No. Of In.", "No. Of Out." and "No. Of Conf." columns show the number of inputs, number of outputs and number of configuration inputs respectively. The "In. Map." and "Out. Map." columns show the used input mapping (C=cycle time mapping, F=frequency mapping, A=amplitude) and output determination method (PO=percentages of ones, TG=average transition gap) respectively. Sixteen electrodes were used in total. The input-output timing was 25 milliseconds and output sampling frequency was 25 KHz.

Set	Problem	In. Sig.	Conf. Volt.	No. Of In.	No. Of. Out.	No. Of Conf.	In. Map.	Out. Map.
A	Even Parity-3	SW	M	3	2	11	F	AT
B	Even Parity-3	SW	M	3	2	11	C	PO
C	Even Parity-3	S	M	3	2	11	A	PO
D	Even Parity-3	S	S	3	2	11	A	PO
E	Even Parity-3	S	S	3	1	12	A	PO
F	Even Parity-3	SW	SW	3	2	11	C	PO
G	Even Parity-4	SW	M	4	2	10	F	AT
H	Even Parity-4	S	S	4	2	10	A	PO

TABLE II: Description of genotype

Gene Symbol	**Signal applied to, or read from i^{th} electrode**	**Allowed values**
p_i	Which electrode is used	0, 1, 2 ... 15
s_i	Type (Irrelevant for set: D-F, H)	0 (constant), 1(square-wave)
a_i	Amplitude	0 , 1
f_i	Frequency (Irrelevant for set: D, E, H)	500 ,501 ... 10K
c_i	Cycle (Irrelevant for set: D, E, H)	0, 1, 2 ... 100

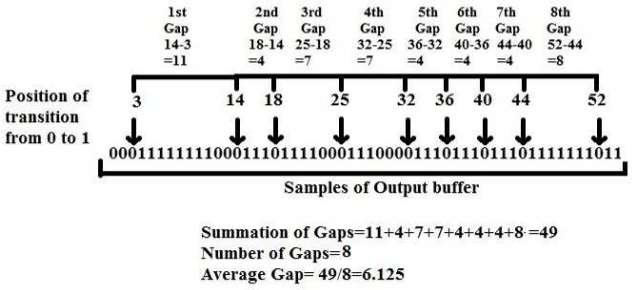

Fig. 3: Example of average transition gap calculation for an output electrode

one, when first electrode has a lower value whenever the output value of the even parity problem is 0. It would have class two, when the first electrode has a higher value at the time when the output value of the problem is 1.

In experiments E, only one electrode was used for output. In this case the percentage of ones in the output buffer was used to decide the output class. The class was determined using a threshold value, set to 50%. If the percentage of ones in the output was less than or equal to 50, the output was designated to be class one, otherwise it was designated to be class two. Thus, if the solution works as desired, it would have class one, when percentage of ones in the output electrode buffer is less than or equal to 50 whenever the output value of the even parity problem is 0. It would have class two, when the percentage of ones in the output electrode buffer is more than 50 at the time when the output value of the problem is 1.

E. Fitness Score

In case of all experiments, the fitness calculation required counts to be made of the number of true positives TP, true negatives TN, false positives, FP and false negatives, FN. There are four possible cases. In all experiments except where E is mentioned:

- If the output of even parity problem is 0 and first electrode has lower value (average transition gap or percentage of ones), it is a correct behaviour. In experiments E, if the output of even parity problem is 0 and the output electrode has value (percentage of ones) less than or equal to 50, it is a correct behaviour. In both of these cases, TN = TN + 1;

- If the output of even parity problem is 1 and second electrode has lower value (average transition gap or percentage of ones), it is a correct behaviour. In experiments E, if the output of even parity problem is 1

and the output electrode has value (percentage of ones) more than 50, it is a correct behaviour. In both of these cases, TP = TP + 1;

- If the output of even parity problem is 0 and second electrode has lower value (average transition gap or percentage of ones), it is an incorrect behaviour. In experiments E, if the output of even parity problem is 0 and the output electrode has value (percentage of ones) more than 50, it is an incorrect behaviour. In both of these cases, FP = FP + 1;

- If the output of even parity problem is 1 and first electrode has lower value (average transition gap or percentage of ones), it is an incorrect behaviour. In experiments E, if the output of even parity problem is 1 and the output electrode has value (percentage of ones) less than or equal to 50, it is an incorrect behaviour. In both of these cases, FN = FN + 1;

In even parity problems the set of value TP, TN, FP, FN accumulated over all instance data (test cases) define the so-called confusion matrix. To obtain the fitness value the Matthew's correlation coefficient (MCC) was used. The MCC is recognized as one of the best single number measures of the quality of a classification algorithm based on the confusion matrix [1]. It can be applied to balanced or unbalanced datasets. The MCC is calculated using Equation 4 and was the fitness function adopted in the experiments.

$$MCC = \frac{TP.TN - FP.FN}{\sqrt{(TP+FP)(TP+FN)(TN+FP)(TN+FN)}} \quad (4)$$

If all results are correct, the fitness is 1, since in this case $FP = 0$ and $FN = 0$. In the case that all results are incorrect, then $TP = 0$ and $TN = 0$, so fitness is -1.

In addition to the fitness calculation, the number of classes (test cases) that were predicted correctly were recorded by summing up the values of TP and TN.

VII. Experimental Details

For each of these experiments, a $1 + \lambda$-ES with $\lambda = 4$ was used for a maximum of 5000 generations. However, the run was terminated if the fitness score reached at value 1.0 (when all the test cases were correctly predicted). Twenty independent runs were carried out. The total time required for 5000 generations was more than 5 hours in case of experiments B, C, E-F and more than 9 hours in case of experiments G and H. The later took a longer time due to the fact that the experiments G and H solved even parity-4 problem which has twice as many test cases as the even parity-3 problem. It should be noted than none of the experiments of set A and D actually required 5000 generations as they all reached a 100% correct result before completing the full evolutionary run.

To evaluate the effectiveness of the EIM method for solving even parity-3,4 problems, the results of experimental material were compared with the results of the CGP method using the same $1 + 4$-ES over the same number of generations (5000) and same number of runs (20), but the evolutionary run was terminated when accuracy reached 100%. The function set chosen for this study was {and, or, nand, nor}. The maximum number of nodes were 200 and the mutation rate was 2.00%, i.e. 12 genes per chromosome were mutated. The comparison result is shown in detail in Section VII-A.

In case of all experiments of the experimental material and CGP of even parity problems, an offspring replaced the parent if its fitness was greater than or equal to the parent.

A. The Results

The average accuracies of all sets of experiments are given in Table III. Accuracy is the percentage of the test cases correctly predicted.

TABLE III: The accuracy averaged over 20 runs for all experiments. Accuracy is the percentage of the test cases correctly predicted.

A	B	C	D	E	F	G	H
100%	97.5%	99.38%	100%	78.75%	99.38%	94.38%	87.5%

The results of experiments A-C were used to compare the performances of different input mappings and output determination methods, where experiments A, B and C used frequency, cycle time and amplitude for input mapping respectively. The experiments A used average transition gap and experiments B, C used percentages of ones for classifying outputs. It has been found from the comparisons that the result of experiments A was the best of all. Thus it appears that using frequency for input mapping and average transition gap for classifying outputs performed the best. Since the results of experiments C were better than the results of experiments B, it appears that an amplitude-based input-mapping performs better than an input-mapping based on cycle time.

Each pair of set A-C was compared using the non-parametric two sided Mann-Whitney U-test and the two-sample Kolmogorov-Smirnov (KS) test [7]. The effect size [19] statistic was also computed. A U-or KS test p-value of $p < 0.05$ indicates that the difference between two dataset is statistically significant. If $p \geq 0.05$ the differences are not significant. The effect size, A value shows the importance of this difference considering the spread of the data; with values $A < 0.56$ showing small importance, $0.56 <= A < 0.64$ medium importance and $A >= 0.64$ large importance. Therefore if a comparison between results is shown to be statistically significant with a medium or large effect size, then it can be said reasonably that any difference is not due to under sampling. It should be noted that the statistical significance tests were performed using the number of correct instances of all runs, and the same ranges were used for determining the effect sizes in case of all comparisons of the even parity experiments. The statistical significance test results comparing experiments A-C are given in Table IV

The results of experiments C and D were used to compare the performances of different types of configuration inputs,

TABLE IV: The statistical significance test on results of even parity experiments of set A-C. The first column shows the pair of sets on which comparison result is given. "U-Test", "KS-Test" and "Effect Size" columns show results of statistical significance test. The statistical significance tests were performed using the number of corrected instances of all runs. '✓' of "U-Test", "KS-Test" columns indicates that the difference between the two data samples is statistically significant and 'X' indicates that the difference is not statistically significant.

Pair of sets	U-Test	KS-Test	Effect Size
Set A - Set B	✓	X	Medium
Set B - Set C	X	X	Medium
Set A - Set C	X	X	Small

where experiments D used only static voltages and experiments C used a mixture of static voltages and square waves for configuration inputs. In both of these cases, the input signals were static voltages. It has been found from the results that the static voltages worked slightly better than the mixture of static voltage and square waves for configuration inputs. Statistical significance tests have also been performed, and it has been found that the difference is statistically insignificant ($p \geq 0.05$) according to U-Test and KS-Test. The same type of comparison (performances of different types of configuration inputs) was performed using experiments B and F, where experiments F used only square waves and experiments B used a mixture of static voltages and square waves for configuration inputs. In both of these cases, the input signals were square waves. It has been found from the results that the square waves worked better than the mixture of static voltage and square waves for configuration inputs. Again statistical significance tests were performed, and it was found that the difference is statistically insignificant according to U-Test and KS-Test.

Comparing the results of experiments C with D and experiments B with F, it has been found that for configuration inputs using only static voltages worked better than using a mixture of square waves and static voltages when input signals were static voltages. Furthermore, only square waves worked better than the mixture of square waves and static voltages when input signals were square waves. Thus, it appears that the performances of different types of configuration inputs may be influenced by the input signals. However, the difference of the results is statistically insignificant and the outcome might be influenced by some other factors, such as input mappings, output determination methods and types (digital or analogue) of input signals. This requires further investigation.

The results of experiments D and E were used to compare the performances of using different numbers of electrodes for outputs to classify Boolean output values, where experiments D used two electrodes and experiments E used only one electrode for outputs. It has been found from the results that two electrodes worked far better than one electrode as outputs in case of even parity problems. Statistical significance tests have also been performed, and it has been found that the difference is statistically significant according to U-Test and KS-Test and also effect size is large.

After analysis of the results of all sets (A-F) of even parity-3 experiments, it has been found that both experiments A and D achieved 100% accuracy on average. In case of experiments A, the 100% correct result was obtained on average within 275.75 generations (averaged over 20 runs) and the lowest number of generations to get 100% correct result was 27, where in case of experiments D, the 100% correct result was obtained on average within 899.9 generations (averaged over 20 runs) and the lowest number of generations to get 100% correct result was 43, which shows that the performance of experiments A was better than the performance of experiments D. That means, the performance of using frequency for input mapping, average transition gap for output mapping, square waves for input signals, mixture of static voltages and square waves for configuration inputs was better than the performance of using amplitude for input mapping, percentage of ones for output mapping, static voltages for input signals and configuration inputs. The same outcome has been obtained in case of experiments G and H, where experiments G achieved 94.38% accuracy on average, 100% accuracy in case of 6 out of 20 runs and experiments H achieved 87.5% accuracy on average, but none of the 20 runs of experiments H achieved 100% accuracy.

The results that the performance of experiments G and H for even 4-parity was not as good as their counterparts for even 3-parity experiments A and D. As in the former case, not all runs ended in 100% correct results. This is probably due to an insufficient maximum number of generations for the harder even 4-parity problem.

Experiments A and G gave the best performance of all of the experiments of even parity-3 and even parity-4 problem respectively. Thus the results obtained in these experiments were compared with the results of CGP. The comparison result is shown in Table V. It should be noted that both of these experiments (A and G) used frequency for input mapping, average transition gap for classifying outputs, square waves for input signals and mixture of static voltages and square waves for configuration inputs.

It has been found that in case of even parity-3 problem, both CGP and experimental material achieved accuracy 100% on average. In case of experimental material, the 100% correct result was obtained on average within 275.75 generations (averaged over 20 runs) and the lowest number of generations to obtain 100% correct result was 27. In case of CGP, the 100% correct result was obtained on average within 845 generations and the lowest number of generations to obtain 100% correct result was 142. So, for the even parity-3 problem, the experimental material performed slightly better than CGP. However, in case of even parity-4 problem, CGP performed marginally better than the experimental material.

VIII. Conclusions And Future Outlook

EIM is a form of hybrid digital-analogue computing where digital computers are used to stimulate and configure materials to carry out computation. Using this technique, it may be possible to develop entirely new computational devices. A purpose-built evolutionary platform called Mecobo, has been used to evolve configurations of a physical system to obtain the

TABLE V: Comparative results of experimental material with CGP on even parity-3,4 problem. The average results for both CGP and the experimental material are computed from 20 independent evolutionary runs with 5000 generations. However, in both of these cases, the evolutionary run was terminated when the accuracy reached 100%. The input-output timing of experiment was 25 milliseconds. The second and third columns show the average accuracy of experimental material and CGP respectively. Accuracy is the percentage of the test cases correctly predicted. "U-Test", "KS-Test" and "Effect Size" columns show results of statistical significance test. The statistical significance tests are performed using the number of corrected instances of all runs. '✓' of "U-Test" and "KS-Test" columns indicates that the difference between the two results is statistically significant and 'X' indicates that the difference is not statistically significant.

Problem	Average Accuracy Of Experimental Material	Average Accuracy Of CGP	U-Test ($p < 0.05$)	KS-Test ($p < 0.05$)	Effect Size
Even Parity-3	100%	100%	X	X	Small
Even Parity-4	94.38%	97.19%	✓	✓	Large

solutions of even parity-3,4 problems. The material used is a mixture of single-walled carbon nanotubes and a polymer. The aim of the paper is not to show that the experimental results of solving even parity problems using EIM is competitive with the even parity results using the state-of-the-art methods but rather to start a new beginning in the world of computation. This is the first time it has been shown that such an approach can be used to solve well-known even parity problems. In some cases, we found that the EIM method could find a solution with 100% accuracy and the results were comparable to the results of a well-known effective software-based evolutionary technique (CGP).

There remain many questions for the future. How does evolutionary computation in materio scale on larger problem instances. What other classes of computational problems can be solved using this technique? What are the most suitable materials and signal types for EIM? What is the computational power of a piece of material using EIM?

IX. Acknowledgements

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] Baldi, P., Brunak, S., Chauvin, Y., Andersen, C.A.F., Nielsen, H.: Assessing the accuracy of prediction algorithms for classification: An overview. Bioinformatics **16**, 412–424 (2000)

[2] Broersma, H., Gomez, F., Miller, J.F., Petty, M.C., Tufte, G.: Nascence project: Nanoscale engineering for novel computation using evolution. International Journal of Unconventional Computing **8**(4), 313–317 (2012)

[3] Clegg, K.D., Miller, J.F., Massey, M.K., Petty, M.C.: Travelling salesman problem solved 'in materio' by evolved carbon nanotube device. In: Parallel Problem Solving from Nature - PPSN XIII - 13th International Conference, Proceedings, *LNCS*, vol. 8672, pp. 692–701. Springer (2014)

[4] Harding, S., Miller, J.F.: Evolution in materio: A tone discriminator in liquid crystal. In: In Proceedings of the Congress on Evolutionary Computation 2004 (CEC'2004), vol. 2, pp. 1800–1807 (2004)

[5] Harding, S., Miller, J.F.: Evolution in materio : A real time robot controller in liquid crystal. In: Proceedings of NASA/DoD Conference on Evolvable Hardware, pp. 229–238 (2005)

[6] Harding, S.L., Miller, J.F.: Evolution in materio: Evolving logic gates in liquid crystal. International Journal of Unconventional Computing **3**(4), 243–257 (2007)

[7] Hollander, M., Wolfe, D.: Nonparametric statistical methods. Wiley (1973)

[8] 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. International journal of unconventional computing. **10**(5-6), 473–497 (2014)

[9] Koza, J.R.: Genetic Programming: On the Programming Computers by Means of Natural Selection. MIT Press (1992)

[10] Lykkebø, O.R., Harding, S.L., Tufte, G., Miller, J.F.: Mecobo: A hardware and software platform for in materio evolution. In: Proceedings of the Conference on Unconventional Computation and Natural Computation, *LNCS*, vol. 8553, pp. 267–279. Springer-Verlag (2014)

[11] Miller, J.F.: An empirical study of the efficiency of learning boolean functions using a cartesian genetic programming approach. In: Proceedings of the Genetic and Evolutionary Computation Conference, vol. 2, pp. 1135–1142. Morgan Kaufmann (1999)

[12] Miller, J.F., Downing, K.: Evolution in materio: Looking beyond the silicon box. In: NASA/DOD Conference on Evolvable Hardware, pp. 167–176. IEEE Comp. Soc. Press (2002)

[13] Miller, J.F., Harding, S.L., Tufte, G.: Evolution-in-materio: evolving computation in materials. Evolutionary Intelligence **7**, 49–67 (2014)

[14] Mohid, M., Miller, J.F.: Evolving solutions to computational problems using carbon nanotubes. International journal of unconventional computing (2015). In Press

[15] Mohid, M., Miller, J.F., Harding, S.L., Tufte, G., Lykkebø, O.R., Massey, M.K., Petty, M.C.: 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)

[16] Mohid, M., Miller, J.F., Harding, S.L., Tufte, G., Lykkebø, O.R., Massey, M.K., Petty, M.C.: Evolution-in-materio: Solving function optimization problems using materials. In: Computational Intelligence (UKCI), 2014 14th UK Workshop on, pp. 1–8. IEEE Press (2014)

[17] 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: Parallel Problem Solving from Nature - PPSN XIII - 13th International Conference, Proceedings, *LNCS*, vol. 8672, pp. 721–730. Springer (2014)

[18] Thompson, A.: Hardware Evolution - Automatic Design of Electronic Circuits in Reconfigurable Hardware by Artificial Evolution. Springer (1998)

[19] Vargha, A., Delaney, H.D.: A critique and improvement of the cl common language effect size statistics of mcgraw and wong. Journal of Educational and Behavioral Statistics **25**(2), 101–132 (2000)