Int. Journ. of Unconventional Computing, Vol. 0, pp. 1–37
Reprints available directly from the publisher
Photocopying permitted by license only

Evolving Solutions to Computational Problems Using Carbon Nanotubes

Maktuba Mohid* and J. F. Miller

Department of Electronics, University of York, York, UK

Received: June 20, 2015. Accepted: June 22, 2015.

Evolution-in-materio is a method that uses artificial evolution to exploit properties of materials to solve computational problems without requiring a detailed understanding of such properties. In this paper, we show that using a purpose-built hardware platform called Mecobo, it is possible to evolve voltages and signals applied to physical materials to solve two computational problems: classification and robot control. We have investigated many experimental factors that could affect the effectiveness of the device and the evolution-in-materio technique. On the classification problem (Iris) we show that the evolution-in-materio approach with analogue signals gives very good results that are competitive with a well-known effective software-based evolutionary approach. In the case of robot control, we were able to evolve a controller that successfully allowed a simulated Khepera robot to fully explore its environment without colliding with any obstacle.

Keywords: Evolutionary algorithms evolution-in-materio material computation evolvable hardware machine learning classification robot control

1 INTRODUCTION

Natural evolution could be viewed as an algorithm which exploits the physical properties of materials. Evolution-in-materio (EIM) aims to mimic the exploitation of physical properties by natural evolution by manipulating physical systems using computer controlled evolution (CCE) [11–13, 19]. In particular, EIM aims to exploit the properties of physical systems for solving

* Contact Author: E-mail: mm1159@york.ac.uk

computational problems. EIM was inspired by the work of Adrian Thompson who investigated whether it was possible to evolve working electronic circuits using a silicon chip called a Field Programmable Gate Array (FPGA). Making sure that the configuration bitstrings could not involve a clock signal or a flip-flop, Thompson evolved a digital circuit that could discriminate between 1kHz or 10kHz signal [25]. The task would normally call for a sequential digital circuit to do this task. Thompson wondered whether intrinsic evolution could somehow connect digital gates with nanosecond delays into a circuit that could solve a task which required much longer time scales. When the evolved circuit was analysed, Thompson discovered that artificial evolution had exploited subtle physical properties of the chip. Moreover, despite extensive analysis and detailed simulation, Thompson and Layzell were unable to explain how the evolved circuit worked [24]. Following on from this, Harding and Miller attempted to replicate these findings using a liquid crystal display. They found that computer-controlled evolution could utilize the physical properties of liquid crystal to help solving a number of computational problems [9].

- Two input logic gates: OR, AND, NOR, NAND, etc. [12].
- Tone Discriminator: A device was evolved which could differentiate different frequencies [9].
- Robot Controller: A controller for a simulated robot with wall avoidance behavior [10].

In this paper, we describe the use of a purpose built platform called Mecobo that facilitates computer controlled evolution of a material [15] for solving machine learning classification problem and controlling a Khepera-like robot. The Mecobo platform has been developed within an EU funded research project called NASCENCE [5]. There are two versions of Mecobo have been developed so far and they are Mecobo 3.0 and Mecobo 3.5. The computational materials we have used in these investigations are mixtures of single-walled carbon nanotubes (SWCNT) and a polymer. This new platform allows a variety of materials to be investigated in custom designed electrode arrays, using a variety of electrical signals and inputs. One of the aims of NASCENCE is to assess the ability of EIM as a methodology for solving a wide variety of computational problems. In recent work, the technique has been applied to solve traveling salesman problems [6], function optimization [22] and bin-packing [21]. Here, we show that using the Mecobo platform it is possible to evolve solutions to machine learning classification problems. To evaluate the effectiveness of the technique we have compared our results with a well-known software-based evolutionary computation

technique called Cartesian Genetic Programming (CGP). We also evaluated the performances of different mixtures (different ratios of SWCNT in polymer) of material, different organizations of electrodes and different hardware platforms based on classification experiments.

Evolutionary computation has been widely used to control robots [4, 23]. EIM has been used before to control simulated robot with wall avoidance behaviour. However, the evolvable substrate was liquid crystal [10]. In that work only two sonars, and two motors were used. In the work we present here where we are using at least 6 sonars (we used 6 sonars for four experiments and 8 sonars for one experiment) to control the simulated robot. Additionally, the robot simulator we used is for a Khepera robot. It is not yet our aim to make EIM a competitive method for solving classification problems or controlling robots, we are simply trying to show that evolving configurations of carbon nanotubes can be used to classify data or control robot. In addition, the experiments provide a yardstick to assess various aspects of EIM using the Mecobo platform. For instance, we can investigate what type of signals are appropriate, and what mixtures of materials give the best results. Using materials in the genotype-phenotype map has, at present, some drawbacks. The main one is that it is slow, this means that we can only feasibly evaluate relatively few potential solutions. However, it is a new approach to the solution of computational problems and as the technology is developed it could offer advantages over conventional computational methods [20].

The organisation of the paper is as follows. In Section 2 we give a brief conceptual overview of EIM. We describe the Mecobo EIM hardware platform in Section 3. The preparation and composition of the physical computational material is described in Section 4. Section 5 describes the machine learning classification problem. The way we have used the Mecobo platform for classification problem is described in Section 6. We describe our classification experiments and analysis of results in Section 6.6. Section 7 describes the robot controlling problem. The way we have used the Mecobo platform for controlling robots is described in Section 8. We describe our robot controlling experiments and analysis of the results in Section 9. We Finally conclude and offer suggestions for further investigation in Section 10.

2 CONCEPTUAL OVERVIEW OF EIM

EIM is a hybrid system involving 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

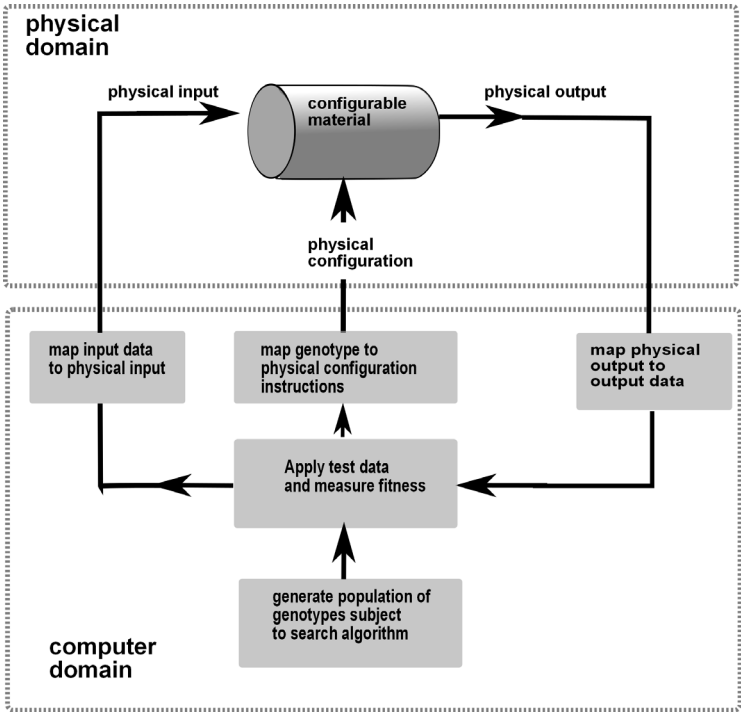


FIGURE 1
Concept of EIM [20].

material and the application to the material of other physical inputs known as physical configurations. A genotype of numerical data 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 [20]. The conceptual overview of EIM has been shown in Figure 1.

In EIM a highly indirect genotype-phenotype mapping is employed. One of its interesting features is that an evolutionary algorithm may be able to exploit hitherto unknown physical variables in a material which may increase evolvability. Software-only genotype-phenotype mappings are highly constrained. Natural evolution operates in a physical world and exploits the physical properties of materials (mainly proteins). Banzhaf et al. discussed the importance of physicality and embodiment [3]. Despite this, there have been very few attempts to date to include materials in the evolutionary process.

Not all materials may be suitable for EIM. Miller and Downing suggested some guidelines for choosing materials. The material needs to be reconfigurable, i.e., it can be evolved over many configurations to get desired response. It is important for a physical material to be able to be "reset" in some way before applying new input signals on it, otherwise it might preserve some memory and might give fitness scores that are dependent on the past behaviour. Preferably the material should be physically configured using small voltage and be manipulable at a molecular level [17, 20].

3 MECOBO: AN EIM HARDWARE PLATFORM

The Mecobo hardware platform has been designed and built within an EU-funded research project called NASCENCE [5]. There are two versions of Mecobo have been built so far and they are Mecobo 3.0 and Mecobo 3.5.

Mecobo is designed to interface a large variety of materials. The hardware allows for the possibility to map input, output and configuration terminals, signal properties and output monitoring capabilities in arbitrary ways. The platform's software component, i.e. EA and software stack, is as important as the hardware. Mecobo includes a flexible software platform including hardware drivers, support of multiple programming languages and a possibility to connect to hardware over the internet makes Mecobo a highly flexible platform for EIM experimentation [15].

It is important to appreciate that in EIM the computational substrate is piece of material for which the appropriate physical variables to be manipulated by evolution may be poorly understood (see Figure 1). This means that the selection of signal types, i.e. inputs, outputs and configuration inputs, assignment to I/O ports could easily not relate to material specific properties. Thus interactions with the materials should be as unconstrained as possible. This means that any I/O port should be allowed by the hardware to accept any signal type. In addition, the signal properties, e.g. voltage/current levels, AC, DC, pulse or frequency, should be allowed to be chosen during evolution. The Mecobo hardware interface is designed to handle all these features. Many computational problems require input data so the interface thus Mecobo has been designed to allow user-defined external input data signals.

Figure 2 shows an overview of the hardware interface. In the figure an example set up is shown in the dotted box. The example genome defines pin 2 to be the output terminal, pin 1 to be the data input and pin 3 - 12 to be configuration signals. The architecture is controlled by a scheduler controlling the following modules: Digital I/O can output digital signals and sample responses. Analogue output signals can be produced by the DAC module. The DAC can be configured to output static voltages or any arbitrary time

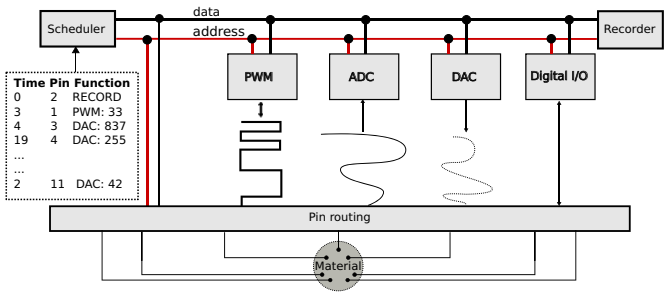


FIGURE 2
Overview of the complete system.

dependent waveform. Sampling of analogue waveforms from the material is performed by the ADC. Pulse Width Modulated (PWM) signals are produced by the PWM module.

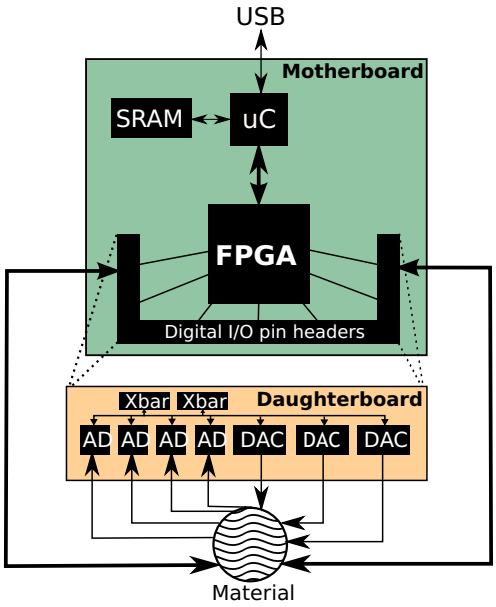
The system's scheduler can set up the system to apply and sample signals statically or produce time scheduled configurations of stimuli/response. The recorder stores samples, digital discrete values, time dependent bit strings, sampled analogue discrete values or time dependent analogue waveforms. Note that the recorder can include any combination of these signals.

In the interface all signals pass a crossbar, i.e. pin routing. Pin routing is placed between the signal generator modules and the sampling buffer (PWM, ADC, DAC, Digital I/O and Recorder) making it possible to configure any terminal of a material to be input, output or receive configuration signals.

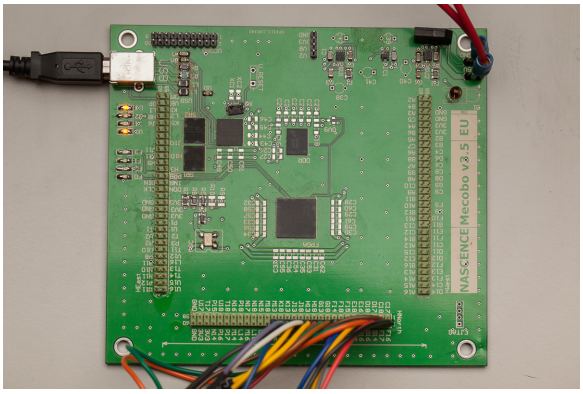
The material signal interface presented in Figure 2 is very flexible. It not only allows the possibility to evolve the I/O terminal placement but also a large variety of configuration signals are available to support materials with different sensitivity, from static signals to time dependent digital functions. At present, the response from materials can be sampled as purely static signals (digital or analogue), digital pulse trains. Mecobo 3.0 only supports digital input and output, wherever, Mecobo 3.5 allows the direct input and output of analogue signals. Further the scheduler can schedule time slots for different stimuli when time dependent functions are targeted or to compensate for configuration delay, i.e. when materials need time to settle before a reliable computation can be observed.

3.1 Hardware implementation

The hardware implementation of the interface is shown as a block diagram in Figure 3(a). Mecobo is designed as a PCB with an FPGA as the main component. The system shown in Figure 2 is part of the FPGA design together with communication modules interfacing a micro controller and shared memory.



(a) Mecobo block diagram.



(b) Picture of Mecobo.

FIGURE 3
Hardware interface implementation overview.

As shown in Figure 3(a) the digital and analogue designs are split into two. All analogue components are placed on a daughter board; such as crossbar switches and analogue-digital converters. This allows the redesign of the analogue part of the system without changing the digital part of the motherboard. The system shown in Figure 3(a) is an example of the current system. The

Parameter Name	Description	Note
Amplitude	0 or 1 corresponding to 0V or 3.5V (Mecobo 3.0), Range [0, 255] corresponding to [-5V, 5V] (Mecobo 3.5)	Wave signal amplitude must be 1
Frequency	Frequency of square wave signal	Irrelevant if fixed voltage input
Cycle Time	Percentage of period for which square wave signal is 1	Irrelevant if fixed voltage input
Phase	Phase of square wave signal	Irrelevant if fixed voltage input
Start time	Start time of applying voltage to electrodes	Measured in milliseconds
End time	End time of applying voltage to electrodes	Measured in milliseconds

TABLE 1
Adjustable Mecobo input parameters.

micro controller stands as a communication interface between the FPGA and the external USB port.

Figure 3(b) shows the motherboard with the Xilinx LX45 FPGA, Silicon Labs ARM based EFM32GG990 micro controller connected to a 12 terminal material sample.

Mecobo 3.0 hardware allows only two types of inputs to the material, which are digital. The input is either a constant voltage (0V or 3.5V) or a square wave signal. In case of Mecobo 3.5, a constant analogue voltage is possible other than the digital square wave signal as an input. It has analogue input in a range [-5V, 5V]. The amplitude of the input signal determines the voltage level. The amplitude of the analogue input signal has a range [0, 255], where value 0 corresponds to -5V and value 255 corresponds to 5V.

Different characteristics or input parameters associated with the inputs of Mecobo 3.0 and Mecobo 3.5 can be chosen. These input parameters are described in Table 1.

The start time and end time of each input signal determine how long an input is applied. Mecobo 3.0 only samples using digital voltage thresholds, hence the material output is interpreted as strictly high or low, (i.e. 1 or 0 respectively). Mecobo 3.5 supports analogue output in a range [-5V, 5V]. The output value of Mecobo 3.5 has a range [-4096, 4096], where value -4096 corresponds to -5V and value 4096 corresponds to 5V.

The output is recorded in a buffer. Three output parameters, i. e., a user-defined output sampling frequency along with the start time and end time of reading output electrode determine the buffer size of output samples. If the output frequency is F_{out}, start time $Time_{start}$ (start time of reading electrode)

and end time is $Time_{end}$ (end time of reading electrode), then the buffer size is Buf_{size} is given by:

$$Buf_{size} = F_{out}(Time_{end} - Time_{start})/1000 \quad (1)$$

Here, $Time_{start}$ and $Time_{end}$ are measured in milliseconds.

However, in practice due to pin latency, the real buffer size is generally smaller. It should be noted that in all experiments described here, inputs are applied for a number of milliseconds and the outputs are accumulated in a buffer for the same number of milliseconds. This has been referred as input-output timing in this paper.

In case of both versions of Mecobo platform, the sequenced actions do not take place at the same instant of time. It maintains a schedule. It happens even if the same start times and end times are used for all inputs, configuration inputs and output(s). It takes some time for Mecobo to circulate a signal to each electrode, which is $\approx$ 1 millisecond. So, for 12 electrodes, it will take $\approx$ 12 milliseconds (more than 12 milliseconds) to circulate inputs (inputs and configuration inputs) to electrodes and read the output buffer(s) from electrode(s) due to the fact of scheduling, even if the same start and end times are used for all electrodes. According to observation, for 12 electrodes, it took 16 milliseconds to get a reasonable buffer size. The response time 16 milliseconds is a long time.

4 THE COMPUTATIONAL MATERIAL

The experimental material consists of SWCNT mixed with polymethyl methacrylate (PMMA) or polybutyl methacrylate (PBMA) and dissolved in anisole (methoxybenzene)* . The sample is baked causing the anisole to evaporate. This results in material which is mixture of SWCNT and PMMA/ PBMA. Different mixtures of material have been used in experiments.

Carbon nanotubes are conducting or semi-conducting and role of the PMMA/ 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. Another benefit of the polymer is to help with dispersion of the nanotubes in solution. The preparation of experimental material is given below:

- 20 μL of material are dispensed onto the electrode array;
- This is dried to leave a 'thick film"

* Mark K. Massey and Michael C. Petty prepared the materials used as substrates and the electrode masks for our experiments

FIGURE 4
Slide with one electrode array and one sample having 12 electrodes.

FIGURE 5
Slide with two electrode arrays and two samples having 16 electrodes

The experimental material is placed in the middle of a plate of the electrode array. Two different arrangements of electrode array in slides have been used in the experiments. In one arrangement, a single electrode array is placed on the slide. This is prepared by placing one drop of the experimental material in the middle of the slide. Twelve gold electrodes arranged on one side are connected directly with the drop. This electrode arrangement is shown in Figure 4. In another arrangement, two electrode arrays are placed in each 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 Figure 5. The electrode array is wired directly with the Mecobo board via a suitable connector.

The materials that have been used in classification and robot controlling experiments are described in Table 2.

5 MACHINE LEARNING CLASSIFICATION PROBLEM

Classification is an important class of problems in machine learning. The objective is to correctly decide which class a set of data instances belongs to. The EIM approach has been evaluated here on one classification problem obtained from [2]: Iris [7]. The Iris dataset has four attributes which are classified into one of the three classes. The dataset contains 150 instances with real-valued attributes. The first fifty instances are class 1, the second fifty class two and the third set of 50 are class 3. The dataset was divided into two

Material Sample Number	Arrangement of Electrodes	Mixture of Material (weight% fraction of PMMA/PBMA)
Sample 1	2 samples, 16 electrodes connected with each sample (8 electrodes in each side of one sample)	1.0% SWCNT in PBMA
Sample 2	1 sample, 12 electrodes connected in 1 side	1.0% SWCNT in PBMA
Sample 3	1 sample, 12 electrodes connected in 1 side	1.0% SWCNT in PMMA
Sample 4	1 sample, 12 electrodes connected in 1 side	0.71% SWCNT in PMMA
Sample 5	1 sample, 12 electrodes connected in 1 side	0.50% SWCNT in PMMA
Sample 6	1 sample, 12 electrodes connected in 1 side	0.10% SWCNT in PMMA
Sample 7	1 sample, 12 electrodes connected in 1 side	0.05% SWCNT in PMMA
Sample 8	1 sample, 12 electrodes connected in 1 side	0.02% SWCNT in PMMA
Sample 9	1 sample, 12 electrodes connected in 1 side	0.01% SWCNT in PMMA
Sample 10	1 sample, 12 electrodes connected in 1 side	Only PMMA

TABLE 2
Description of materials used in experiments.

groups (training and test set) of 75 instances each. Each set contained exactly 25 instances of each class.

6 CLASSIFYING DATA USING EIM

6.1 Methodology

Twelve different sets of experiments were performed. The first (A) and second (B) sets of experiments were performed with Mecobo 3.5 and the remaining ten sets (C-L) of experiments were performed with Mecobo 3.0. Results of sets A and B were used to compare the performances of different types of configuration inputs (comparison between all static analogue voltages and mixtures of static analogue voltages and digital square waves). The first five sets (A-E) of experiments were performed with material sample 1 (according to Table 2). Of these, the results of sets B and C were used to compare the performances of two Mecobo platforms, i.e. the performance of using all analogue inputs, outputs and configuration inputs against the performance of using all digital inputs, outputs and configuration inputs. The experiments of sets C and D were used to compare results using different input-output timings (the input-output timings of sets C and D were 32 milliseconds and 128 milliseconds respectively). Experiments of sets E and F were used to investigate whether different organisations of electrodes play any role or not. Set F used material sample 2, i.e. both sets E and F used the same mixture of material, but the organisation of electrodes was different. An investigation was performed using experiments of sets F and G to see whether different polymers matter or not. Set G used material sample 3, i.e sets F and G used

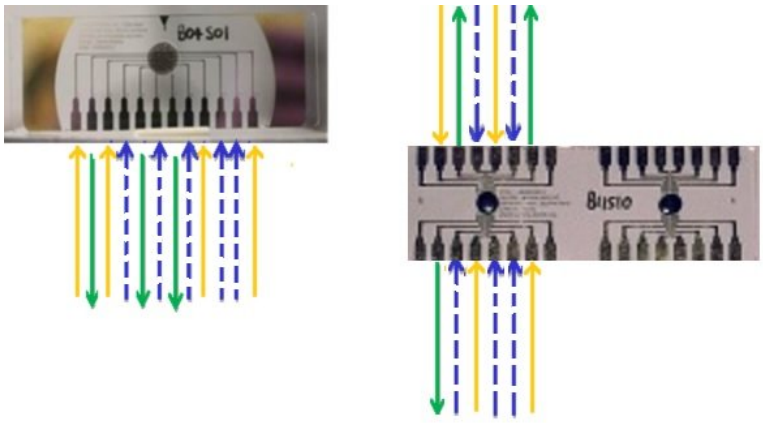


FIGURE 6
The examples of genotype used in classification problem using two different material samples having different organisations of electrodes, where green arrows (away from device) are used to indicate reading outputs from the output electrodes, yellow arrows (solid toward device) are used to show inputs to input electrodes and blue arrows (hashed and toward device) are used to show configuration inputs being sent to 5 electrodes.

the same percentage of SWCNT but in the different polymer. Experiments of sets G-L compared results obtained by different percentages of SWCNT in PMMA, where set H used material sample 4, set I used sample 5, J used sample 6, K used sample 7 and set L used sample 8. The results of experiments of set B was compared with the results of CGP to evaluate the effectiveness of the EIM method for solving classification problems.

All of these experiments were performed with electrode arrays having 12 electrodes. However, in the case of material sample 1, one electrode array was used, where only 12 electrodes were used from the 16 electrodes of that electrode array, these were the middle 6 electrodes from each side of one sample. Four electrodes were used as inputs (i.e. are instance-related), 3 electrodes were used as outputs (i.e. defining the class) and 5 electrodes were used as configuration inputs. Each output electrode was used for each output class. Each chromosome defined which electrodes were outputs, inputs (received square waves) or received the configuration inputs (square waves or constant voltages). The examples of genotype used in classification problem are shown in Figure 6, where green arrows (away from device) are used to indicate reading outputs from the output electrodes, yellow arrows (solid toward device) are used to show inputs to input electrodes and blue arrows (hashed and toward device) are used to show configuration inputs being sent to 5 electrodes.

The input-output timing was 32 milliseconds for the experiments of sets A-C and 128 milliseconds for sets D-L. A 25 KHz output sampling frequency was used for all sets of experiments.

In case of Mecobo 3.0, the frequency of the square wave input signal was used for input mapping and the average transition gap (this is described in Section 6.4) was used for determining output classes. In case of Mecobo 3.5, the amplitude of the static analogue input signal was used for input mapping and average of the sample values of the output buffer was used to determine output classes.

The fitness calculation in the EA only used training data. Once the EA was finished, the configurations of electrodes having the best fitness were tested with the test data to determine their ability to predict correctly unseen data (the test set).

In case of sets A-D, the experiments were carried out up to 50 generations and in case of sets E-L, the experiments were carried out up to 500 generations. Ten independent runs were carried out for the sets A and D-L. Twenty independent runs were carried out for the sets B and C. It should be noted that twenty evolutionary runs were carried out only in those experiments that dealt with the comparisons with CGP and the two Mecobo platforms. Other experiments were carried out up to 10 runs as many sets of experiments were needed to perform all these different types of investigations.

The experimental settings of all sets of classification experiments are summarised in Table 3 to get a clear overview of the experiments. The purposes of performing the classification experiments are described in Table 4.

6.2 Genotype Representation

Each chromosome used $n_e = 12$ electrodes at a time. In cases of experiments A and C-L, associated with each electrode there were six genes which defined 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, phase, cycle time (Section 3). This means that each chromosome required a total of $12 \times 6 = 72$ genes. Mutational children were created from a parent genotype by mutating a single gene (i.e. one gene of 72).

In case of experiments B, associated with each electrode there were two genes which defined which electrode was used as an input or output or configuration input, or amplitude of the input applied to the electrode (Section 3). This means that each chromosome required a total of $12 \times 2 = 24$ genes. Mutational children were created from a parent genotype by mutating a single gene (i.e. one gene of 24).

The values that genes could take are shown in Table 5. i takes values 0, 1, ... 11.

Set	No. of gen.	No. of run	Mat. sam.	Mecobo vers.	In. sig.	Out. sig.	Conf. volt.	In. map.	Out. map.	Time
A	50	10	1	3.5	AS	A	M	A	SB	32
B	50	20	1	3.5	AS	A	AS	A	SB	32
C	50	20	1	3.0	DW	D	MD	F	TG	32
D	50	10	1	3.0	DW	D	MD	F	TG	128
E	500	10	1	3.0	DW	D	MD	F	TG	128
F	500	10	2	3.0	DW	D	MD	F	TG	128
G	500	10	3	3.0	DW	D	MD	F	TG	128
H	500	10	4	3.0	DW	D	MD	F	TG	128
I	500	10	5	3.0	DW		MD	F	TG	128
J	500	10	6	3.0	DW	D	MD	F	TG	128
K	500	10	7	3.0	DW	D	MD	F	TG	128
L	500	10	8	3.0	DW	D	MD	F	TG	128

TABLE 3
The experimental settings of all sets of classification experiments. The “No. of gen.” and “No. of run” columns show the number of generations and number of runs of the experiments respectively. The “Mat. sam.” and “Mecobo vers.” columns show the material sample number (according to Table 2) and version of Mecobo hardware respectively. The “In. sig.”, “Out. sig.” and ”Conf. volt.” columns show the types of used input signals (AS=analogue static voltages, DW=digital square waves), output signals (A=analogue, D=digital) and configuration inputs (AS=analogue static voltages, MD=mixtures of digital square waves and digital static voltages, M=mixtures of digital square waves and analogue static voltages) respectively. The “In. map.” and “Out. map.” columns show the used input mapping (A=amplitude mapping, F=frequency mapping) and output mapping (SB=average of sample values of the output buffer, TG=average transition gap) respectively. The last column shows the input-output timing (measured in milliseconds) used in the experiments. It should be noted that all sets of experiments used 12 electrodes of the electrode array and a 25 KHz output sampling frequency.

Experiments	Purpose
Sets A, B	Comparison of the performance of using all static analogue voltages against the performance of using mixtures of static analogue voltages and digital square waves.
Sets B, C	Comparison of the performance of using all analogue inputs, outputs and configuration inputs against the performance of using all digital inputs, outputs and configuration inputs (comparison of the performances of two Mecobo platforms).
Sets C, D	Comparison of results using different input-output timings.
Sets E, F	Comparison of results using different organisations of electrodes.
Sets F, G	Comparison of results using different polymers.
Sets G-L	Comparisons of results using different percentages of SWCNT in PMMA.
Set B	Comparison of experimental results against the results of CGP.

TABLE 4
The purposes of performing the classification experiments. The first column shows the sets of experiments. The second column shows the purpose.

Gene symbol	Signal applied to, or read from i^{th} electrode	Allowed values
p_i	Which electrode is used	0, 1, 2 . . . 11
s_i	Type (Irrelevant for set: B)	0 (constant), 1(square wave)
a_i	Amplitude	0 , 1 (For sets: C-L) 1, 2 . . . 254 (for sets: A, B)
f_i	Frequency (Irrelevant for set: B)	500 ,501 . . . 10K
ph_i	Phase (Irrelevant for set: B)	1, 2 . . . 10
c_i	Cycle time (Irrelevant for set: B)	0, 1, 2 . . . 100

TABLE 5
Description of genotype for classification experiments

The genotype for a chromosome of an individual consists of the 72 genes is shown below:

$p_0 s_0 a_0\, f_0 ph_0 c_0 \ldots p_{11} s_{11} a_{11}\, f_{11} ph_{11} c_{11}$

The genotype for a chromosome of an individual consists of the 24 genes is shown below:

$p_0 a_0 \ldots p_{11} a_{11}$

For experiments A, C-L, the first 24 gene values of a chromosome are related to inputs. These are:

$p_0 s_0 a_0\, f_0 ph_0 c_0 \ldots p_3 s_3 a_3\, f_3 ph_3 c_3$

For experiments B, the first 8 gene values of a chromosome are related to inputs. These are:

$p_0 a_0 \ldots p_3 a_3$

For experiments A, C-L, the last 18 gene values of a chromosome are related to outputs. These are:

$p_9 s_9 a_9\, f_9 ph_9 c_9 \ldots p_{11} s_{11} a_{11}\, f_{11} ph_{11} c_{11}$

For experiments B, the last 6 gene values of a chromosome are related to outputs. These are:

$p_9 a_9\, p_{10} a_{10}\, p_{11} a_{11}$

In these input and output genes, only the first p_i (here the value of i is 0-3 and 9-11) has any effect, the remainder are redundant. The gene p_i decides which electrode will be used for the input or output of the device. Thus, mutations in this gene can choose a different electrode to be used as an input or output.

6.3 Input Mapping

In experiments C-L, each of the inputs to the electrode array was a square wave 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, 4. Denote the maximum value and minimum value taken by this attribute in the whole dataset by $I_{i_{max}}$ and $I_{i_{min}}$ respectively. Denote the maximum allowed frequency and minimum allowed frequency by F_{max} and F_{min} respectively. Then the linear mapping given in Equation 2 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.

$$F_i = a_i I_i + b_i \tag{2}$$

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.

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

and

$$b_i = (F_{min} I_{i_{max}} - F_{max} I_{i_{min}})/(I_{i_{max}} - I_{i_{min}}) \tag{4}$$

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

In experiments A and B, each of the inputs to the electrode array was a static voltage of a particular amplitude. The amplitude was determined by a linear mapping of attribute data. That means, amplitude was used instead of frequency for the input mapping, where maximum amplitude value was 254 and minimum amplitude value was 1. The mapping equation is the same as other experiments (Equation 2), but instead of frequency, the amplitude was used in the equation.

6.4 Output Mapping

The class that an instance belongs to was determined by examining the output buffers which contain samples taken from the output electrodes. Mecobo 3.0 can only recognise binary values, so the output buffers contain bitstrings. So, in experiments C-L, the *transitions* from 0 to 1 in the output buffers were used to calculate the class that an instance belongs 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 frequencies of

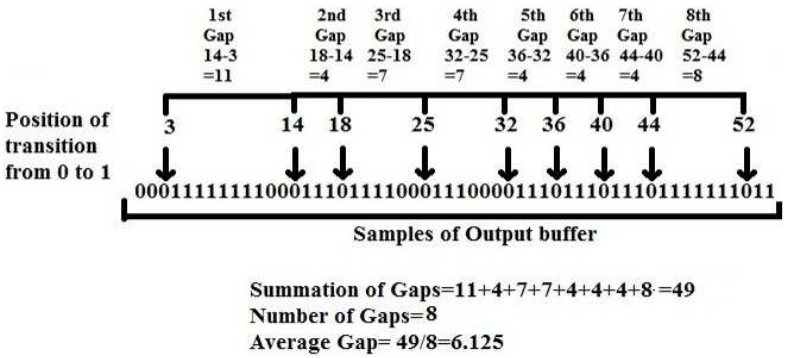


FIGURE 7
An example of average transition gap calculation for an output electrode

applied signals, it seemed natural to use the method of reading output buffer bitstrings, which is itself frequency related. An example of average gap calculation for an output electrode is shown in Figure 7

The output class was determined by the output buffer with the largest average transition gap. If two or more buffers had the same average gap, then the class was determined by the first such buffer encountered (in precedence order: 1, 2, 3). For instance, if the second output buffer had the highest average gap, the output class would be predicted to be class 2.

Mecobo 3.5 supports analogue outputs. So, in case of experiments A and B, average values of output buffers were used to calculate the class that an instance belongs to. The output class was determined by the output buffer with the largest average value. If two or more buffers had the same average value, then the class was determined by the first such buffer encountered (in precedence order: 1, 2, 3). This mapping was used as it seems more closely related to amplitude.

6.5 Fitness Score

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. For an instance, having a class c, according to the dataset and a predicted class p, the TP, TN, FP, and FN can be calculated using Equation 5. The explanation of this 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.

$$\begin{aligned} &\textbf{if } p = c \textbf{ then } TP = TP + 1;\ TN = TN + 2 \\ &\textbf{if } p \neq c \textbf{ then } FP = FP + 1;\ FN = FN + 1;\ TN = TN + 1 \end{aligned} \tag{5}$$

Once all instances have been classified, the fitness of a genotype can be calculated using Equation 6 [1].

$$\textit{fitness} = \frac{TP.TN}{(TP + FP)(TN + FN)} \tag{6}$$

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 fitness is zero.

6.6 The Experimental Details

In case of all sets of experiments, a $1 + \lambda - ES$ with $\lambda = 4$ was used. The $1 + \lambda - ES$ evolutionary algorithm has a population size of $1 + \lambda$ and selects the genotype with the best fitness to be the parent of the new population. The remaining members of the population are formed by mutating the parent.

As said before that in case of sets A-D, the experiments were carried out up to 50 generations and in case of sets E-L, the experiments were carried out up to 500 generations. Ten independent runs were carried out for the sets A, D-H and I-L and twenty independent runs for the sets B and C. It took more than 12 hours (one evolutionary run) to run 500 generations on the Iris training set with input-output timing 128 milliseconds.

To evaluate the effectiveness of the EIM method for solving classification problems we compared results with those obtainable using CGP using the same $1 + 4$ evolutionary algorithm over the same number of generations using the same fitness function. CGP is a graph-based form of genetic programming [18]. The genotypes encode directed acyclic graphs and the genes are integers that represent where nodes get their data, what operations nodes perform on the data, and where the output data required by the user is to be obtained. CGP was applied to solve classification problems before. Völk et al. applied CGP to classify mammograms [27]. Harding et al. presented a version of CGP that could handle multiple data types and then applied it to find solutions to multiple classification tasks [8]. In classification problems the number of outputs, n_O is chosen to be equal to the number of classes in the dataset. The class of a data instance is defined as the class indicated by the maximum numerical output. The function set chosen for this study was defined over the real-valued interval [0.0, 1.0] and consisted of the following

primitive functions of their inputs. The functions were assumed to have three inputs, z_0, z_1, z_2 (but some are ignored):

$(z_0 + z_1)/2$; $(z_0 - z_1)/2$; $z_0 z_1$;
if $|z_1| < 10^{-10}$ **then** 1 **else if** $|z_1| > |z_0|$ **then** z_0/z_1 **else** z_1/z_0;
if $z_0 > z_1$ **then** $z_2/2$ **else** $1 - z_2/2$.

We used *three* mutation parameters. A percentage for mutating connections, μ_c and functions, μ_f. Mutation of outputs μ_o, is done probabilistically. In all experiments $\mu_c = 3\%$, $\mu_f = 3\%$, and $\mu_o = 0.5$. The output mutation probability was set as 0.5 because there are only as many outputs as there are classes. We chose a linear CGP geometry by setting the number of rows, $n_r = 1$ and the number of columns, $n_c = 100$ with nodes being allowed to connect to any previous node.

In case of all experiments of the experimental material and CGP of classification problems, a child replaced the parent if its fitness was greater than or equal to the parent.

The Results

As mentioned before, experiments A and B were performed using Mecobo 3.5 and the Iris dataset with material sample 1. Mecobo 3.5 is located in Norway and the experiments were performed via the internet by connecting with Norway Mecobo board. Usually, it takes some extra time to communicate and then takes more time to perform the experiment. That is why the input-output timing was decreased to 32 milliseconds and also each run was carried out up to 50 generations. As mentioned before that Mecobo 3.5 supports static analogue input, digital square wave input and analogue output. In addition, no more than 8 static analogue inputs (inputs and configuration inputs) can be sent via Mecobo 3.5. It should be noted that Mecobo 3.5 stops working if more than 8 analogue static inputs are used. If more than 8 analogue static inputs are required to use, it is done by sending 8 static analogue inputs and making the remaining inputs undefined by the program. If any electrode is connected with the material and left undefined by the program, Mecobo 3.5 connects static voltage of -2.3V using that electrode by default. That means, in any case if more than 8 static analogue inputs are needed to be sent to the material, the remaining inputs are set as static -2.3V by default by Mecobo 3.5. In experiments B, 4 static analogue inputs and 5 static analogue configuration inputs were used, i.e. in total 9 static analogue inputs were needed to be sent to the material via Mecobo 3.5, the last 1 configuration input was set as static -2.3V by Mecobo 3.5 irrespective of the voltage level set by the genotype for that configuration input. In experiments A, it was restricted via the program that no more than 4 configuration inputs would have static

Accuracy	Static analogue voltages (set B)	Mixtures of static analogue voltages and digital square waves (set A)
Training	93.33%	82.8%
Test	87.87%	73.2%

TABLE 6
Comparative results of different input signal combinations (static analogue voltages against mixtures of static analogue voltages and digital square waves) of Mecobo 3.5. Both of these experiments were performed using Iris dataset with material sample 1 up to 50 generations (the input-output timing was 32 milliseconds). The comparison was performed over ten evolutionary runs. Accuracy is the percentage of the training or test set correctly predicted.

analogue voltages, i.e. at least 1 configuration input must be a digital square wave signal. The results of experiments A and B were compared over ten runs as experiments A were performed up to 10 runs. It should be noted that the results of first ten runs from the twenty runs of experiments B were used in this comparison. The detailed comparison results are shown in Table 6.

After analysis of the results of experiments A and B, it has been found that the input signals (inputs and configuration inputs) having only static analogue voltages show better results than the results obtained using mixtures of static analogue voltages and digital square waves for inputs and configuration inputs. The results of the experiments A and B have been compared using the non-parametric two-sided Mann-Whitney U-test and the two-sample Kolmogorov-Smirnov (KS) test [14]. The effect size [26] statistic has also been computed. It has been found that the difference of the results of experiments A and B is statistically significant according to U-Test (< 0.05) and KS-Test (< 0.05) and also the effect size is large. A U-or KS test value of < 0.05 indicates that the difference between two datasets is statistically 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 have been performed using total number of corrected instances (combined on training and test sets) of all runs and the same ranges have been used for determining the effect size in case of all comparisons in these machine learning classification experiments.

The results of experiments B and C show the comparison of the performance of Mecobo 3.5 against the performance of Mecobo 3.0, i.e. the comparison of the performance of using analogue inputs, outputs and configuration inputs against the performance of using digital inputs, outputs and

Accuracy	Mecobo 3.5 (analogue inputs, outputs and configuration inputs) (set B)	Mecobo 3.0 (digital inputs, outputs and configuration inputs) (set C)
Training	91.33%	66.93%
Test	86.6%	60.73%

TABLE 7
Comparative results of performances of Mecobo 3.5 and Mecobo 3.0. Both of these experiments were performed using Iris dataset with material sample 1 up to 50 generations (the input-output timing was 32 milliseconds). Twenty evolutionary runs were carried out. Accuracy is the percentage of the training or test set correctly predicted.

configuration inputs. The comparison was performed over 20 runs with 50 generations for each run, where the input-output timing was 32 milliseconds. The detailed results are shown in Table 7.

After analysis of the results of experiments B and C, it has been found that the performance of using analogue inputs, outputs and configuration inputs is far better than the performance of using digital inputs, outputs and configuration inputs, i.e. the performance of Mecobo 3.5 is far better than the performance of Mecobo 3.0. Statistical significance tests have also been performed, and it has been found that the difference is statistically significant according to U-Test (< 0.05), KS-Test (< 0.05) and also the effect size is large.

As mentioned before that the experiments A-C used 32 milliseconds as input-output timing, where the rest used 128 milliseconds. It was investigated whether different input-output timings show any significant difference in results or not. Experiments C and D were used for this comparison. Both of these used material sample 1 with Mecobo 3.0. The experiments were carried out up to 50 generations and the results were compared using 10 runs. It has been found from the results that the difference is not statistically significant according to U-Test (< 0.05) and KS-Test (< 0.05). The results are shown in detail in Table 8.

Accuracy	Input-output timing 32 milliseconds (set C)	Input-output timing 128 milliseconds (set D)
Training	65.87%	68.4%
Test	61.2%	63.6%

TABLE 8
Comparative results of experiments C and D having different input-output timings (32 milliseconds and 128 milliseconds respectively). Both of these experiments were performed using Iris dataset with material sample 1 up to 50 generations. Ten evolutionary runs were carried out. Accuracy is the percentage of the training or test set correctly predicted.

Accuracy	Material sample 1 (set E)	Material sample 2 (set F)
Training	84.8%	84.4%
Test	72.13%	78.27%

TABLE 9
Comparative results of experiments E and F using different material samples (sample 1 and sample 2 respectively) having different organisations of electrodes. Both of these experiments were performed using Iris dataset with Mecobo 3.0 up to 500 generations (the input-output timing was 128 milliseconds). Ten evolutionary runs were carried out. Accuracy is the percentage of the training or test set correctly predicted.

Experiments E and F used the same material mixture (material sample 1 and material sample 2 respectively) and the same hardware platform (Mecobo 3.0), but different organisations of electrodes (Table 2). The results of these two experiments were compared to investigate whether different organisations of electrodes matter or not. Both of these experiments were carried out up to 10 runs with 500 generations (the input-output timing was 128 milliseconds). The statistical significance tests have showed that the difference of results is insignificant according to U-Test (< 0.05) and KS-Test (< 0.05). The detailed results are shown in Table 9.

The same percentage (1.0%) of SWCNT was used in the different polymer (PBMA and PMMA) in material sample 2 and material sample 3 (Table 2). Material sample 2 (contains PBMA) and material sample 3 (contains PMMA) were used in experiments F and G respectively. The results of these two sets were compared to investigate whether different polymers play any role in computation or not. Both of these experiments were performed up to 10 runs with 500 generations (the input-output timing was 128 milliseconds). The statistical significance tests have showed that the difference of results is insignificant according to U-Test (< 0.05) and KS-Test (< 0.05). The detailed results are shown in Table 10.

Accuracy	SWCNT in PBMA (set F)	SWCNT in PMMA (set G)
Training	84.40%	82.13%
Test	78.27%	72.27%

TABLE 10
Comparative results of experiments F and G using different material samples (sample 2 and sample 3 respectively) having the same percentage (1.0%) of SWCNT in the different polymer (PBMA and PMMA respectively). Both of these experiments were performed using Iris dataset with Mecobo 3.0 up to 500 generations (the input-output timing was 128 milliseconds). Ten evolutionary runs were carried out. Accuracy is the percentage of the training or test set correctly predicted.

Set	Material sample no.	SWCNT in PMMA	Average training accuracy	Average test accuracy
G	3	1.0%	82.13%	72.27%
H	4	0.71%	80.67%	71.07%
I	5	0.50%	81.73%	71.6%
J	6	0.10%	81.73%	71.6%
K	7	0.05%	85.07%	72.27%
L	8	0.02%	80.93%	69.47%

TABLE 11
Comparative results of different percentages of SWCNT in PMMA. All of these experiments were performed using Iris dataset with Mecobo 3.0 up to 500 generations (the input-output timing was 128 milliseconds). The comparisons were performed over ten evolutionary runs. The first column shows the set of experiments. The second column shows the material sample number (Table 2). The third column shows the weight percent fraction of SWCNT in PMMA. The fourth and fifth columns show the average training accuracy and average test accuracy of the experimental material. Accuracy is the percentage of the training or test set correctly predicted. It has been observed by investigation that the lowest percentage of SWCNT required for computation is 0.02% (approximately) (weight percent fraction in polymer). Lower than this percentage or empty electrode does not perform any evolution at all, where the output buffers are always full of zeroes.

Different percentages of SWCNT were used in PMMA in material samples 3-8 (Table 2), which were used in experiments G-L respectively. The performances of different percentages of SWCNT were compared using the results of experiments G-L. The comparisons were performed over ten runs up to 500 generations for each run (the input-output timing was 128 milliseconds). The detailed results are shown in Table 11.

It has been found from the results of Table 11 that material sample 7 (0.05% SWCNT in PMMA) shows the best result among the results obtained by all other percentages of SWCNT in PMMA according to the average training accuracies and the average test accuracies. Statistical significance tests have also been performed using the results of each of the pairs of the material samples 3-8. It has been found that the difference of the results of each of the pairs of material samples 3-8 (sets G-L respectively) is statistically insignificant according to U-Test (< 0.05) and KS-Test (< 0.05). It has been observed by investigation that the lowest percentage of SWCNT required for computation is 0.02% (approximately) (weight percent fraction in polymer). Lower than this percentage or empty electrode does not perform any evolution at all, where the output buffers are always full of zeroes.

If all the results of all sets of experiments are considered, it has been found that the results of experiments B are the best since they have acquired the highest accuracies in case of both training and test data. As the results of experiments B are the best, the results were compared with the results of the

Set	Data set	Avg. train. acc. of exp. mat.	Avg. test acc. of exp. mat.	Best acc. of exp. mat.	Avg. train. acc. of CGP	Avg. test acc. of CGP	Best acc. of CGP
B	Iris	91.33%	86.6%	97.33%	87.2%	84.4%	96.67%

TABLE 12
Comparative results of experimental material (experiments of set B) with CGP on machine learning classification problem using Iris dataset. The experiments B were performed using material sample 1 with Mecobo 3.5 (the input-output timing was 32 milliseconds). The comparison was performed over twenty evolutionary runs and up to 50 generations for each run. The first column shows the set of experiments and second column shows the dataset on which the experiments were performed. The third, fourth and fifth columns show average training accuracy, average test accuracy and best accuracy of experimental material respectively. The sixth, seventh and eighth columns show average training accuracy, average test accuracy and best accuracy of CGP respectively. Accuracy is the percentage of the training or test set correctly predicted. "U-Test", "KS-Test" and "Ef. sz." columns show results of statistical significance tests (L = large, M = medium, S = small). The statistical significance tests have been performed using total number of corrected instances (combined on training and test sets) 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.

well-known CGP. It has been found that the average results (accuracies) of experiments B are better than the average results (accuracies) of CGP in case of both training and test data. The best accuracy of the experimental material is also better than that of CGP. This at least shows that evolving configurations of materials holds promise for classification. The experiments B used Mecobo 3.5, which used analogue signals for inputs, outputs and configuration inputs. The statistical significance tests have also been performed, which show that the difference of the results is statistically insignificant according to U-Test (< 0.05) and KS-Test (< 0.05). The detailed comparison results are shown in Table 12

A summary of outcomes from these experiments is shown below:

- The results of the experimental material (material sample 1) are better than the results of CGP, where analogue inputs, outputs and configuration inputs were used with Mecobo 3.5. The experiments were carried out up to 50 generations.
- The performance of using analogue inputs, outputs and configuration inputs is better than the performance of using digital inputs, outputs and configuration inputs according to comparison results of Mecobo 3.5 and Mecobo 3.0.
- In case of Mecobo 3.5, it has been found that the input signals (inputs and configuration inputs) having only static analogue voltages show

better results than the results obtained by mixtures of static analogue voltages and digital square waves for inputs and configuration inputs.

- It has been found that different organisations of electrodes do not matter in computation. The difference of results is statistically insignificant.
- The choice of polymer (PMMA or PBMA) in material mixture plays no important role in computation. The difference of results is statistically insignificant.
- An experimental investigation examined which weight percentage of SWCNT in PMMA appears to be the most effective for classification. It appeared that the best results were obtained with 0.05% SWCNT. However, the results with various mixtures appeared not to be statistically distinguishable.

7 WALL AVOIDING ROBOT

The task for the robot is to travel around a closed environment avoiding the walls and obstacles and to cover as much floor space as possible. In this scenario, the robot has to navigate around an unknown environment avoiding collision with the walls. The control system is able to use the distance sensors on the robot, which perform some form of signal processing and use this information in the control of the motion of the robot using motors. We have used the Khepera robotic platform, which has 8 short range infra red sensors and two motors. Generally in evolutionary robotics evolution is performed in simulation. Solutions based on simulation can be run in faster than using real robot, as it can ignore the physical properties of the robot and its hardware.

8 EIM CONTROLLED ROBOT

We have adapted a Khepera robot simulator (version 2.0) written by Marcin Pilat[†] . Pilat rewrote a Unix based Khepera written by Olivier Michel [16].

Many sets of experiments have been performed here with different maps, different numbers of input sensors, different input and output mappings etc. The robot has diameter of 55 nominal units and obstacles or walls are made from small bricks having width and height 20 unit. The map is 1000 X 1000 $unit^2$.

8.1 Methodology

The simulated robot that has been used in the experiments has 8 infrared sensors (S0-S7) and 2 motors (M1-M2). The robot together with the placement

[†] http://www.pilat.org/khepgpsim/

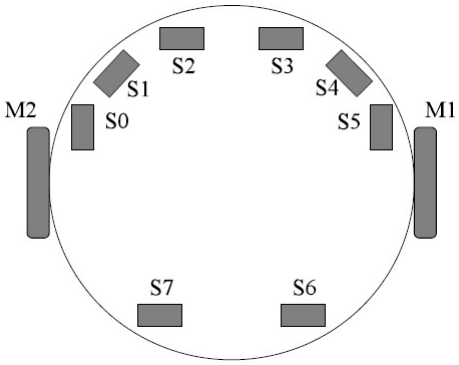


FIGURE 8
Schematic view of the Khepera Robot with the positions of the IR proximity sensors (S0-S7) and motors (M1, M2).

of sensors and motors that have been used in the experiments is shown in Figure 8. The sensor distance value (in a range [0, 1023], where 0 means no object is found and 1023 means an object is in the nearest position) is calculated as a function of the presence (or the absence) of obstacles (see footnote 1). Random noise corresponding to $\pm 10\%$ of its amplitude is added to the distance value of the sensor. In experiments, the distance values of sensors have been used as inputs to the material. The robot moves according to the speeds (in a range $[-10, 10]$) of the motors. Random noise of $\pm 10\%$ is also added to the amplitude of the motor speed while a random noise of $\pm 5\%$ is added to the direction resulting from the difference of the speeds of the motors. The motor speed is decided by the output of the carbon nanotube material.

In case of each individual of the population in each evolutionary run, the robot controller is allowed to execute for up to 5000 time steps. But if the robot collides with an obstacle before completing the 5000 time steps, it is stopped immediately. Also, if robot rotates in the same place for a long time, it is also stopped. The latter is assessed using the x and y coordinates of robot's position from all the timesteps in the past. If the differences between old and new x and y coordinates are ≤ 30 units (approximately half the diameter of the robot), it is assumed that the robot visits the same place as before, otherwise it is assumed that the robot is exploring a new area of the map. The previous (immediate) 50 moves of the robot is not used to prevent a slowly moving robot being penalized. If the robot rotates in approximately the same place for 1000 consecutive moves, it is stopped immediately and its overall fitness is assessed. If robot is not stopped early and it is exploring a new area of the map, the distance between previous move and the new move is added

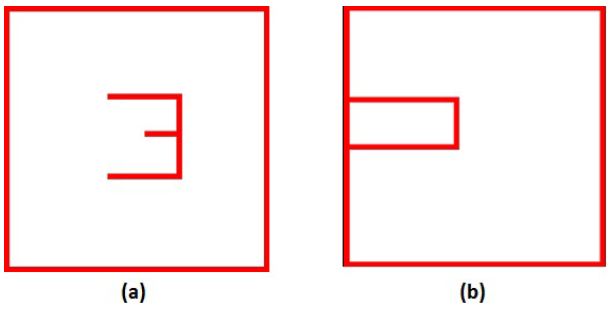


FIGURE 9
Task environments used in the robot experiments. In (a) and (b), the obstacles or the walls are shown in red and the white area of the map is the area where the robot is allowed to move.

to the fitness score. The distance is calculated using Euclidean distance with x and y coordinates of previous and new move. The better individuals are decided by higher fitness values.

Five sets (A-E) of experiments have been performed. In all sets of experiments were performed with material sample 1 (according to table 2) and Mecobo 3.0. In first (A) and fourth (D) sets, the map shown in image 9 (a) was used. In other sets (B, C and E) of experiments, the map shown in image 9 (b) was used. In all sets of experiments except the set E, only 12 electrodes from the 16 electrodes have been used (the middle 6 electrodes from each side of one material sample). These 12 electrodes were used in the following manner: 6 electrodes have been used as inputs, 2 electrodes have been used as outputs and the remaining 4 electrodes have been used for configuration inputs. The 6 inputs were provided by sensors S0, S2, S3, S5, S6 and S7 (see image 8). In fifth (E) set of experiments all 16 electrodes have been used, where 8 electrodes (all 8 sensors) have been used as inputs, 2 electrodes as outputs and the remaining 6 as configuration inputs.

For all sets of experiments, each chromosome defined which electrodes were either outputs, inputs (receive square waves) or received the configuration inputs (square waves or constant voltage). The amplitude of input electrode voltage was set to 1 (3.5 V).

In all sets of experiments, we used a 25KHz output sampling frequency. Ten independent runs were carried out for all sets of experiments with 100 generations for each run.

In case of sets A-D, we accumulated output values in a buffer for 20 milliseconds. However, in case of set E, we accumulated output values in a buffer for 25 milliseconds due to using a greater number of electrodes. Sampling over longer times is necessary as scheduling in Mecobo is serial. This means that several sequences of actions (i.e., sending inputs signals, configuration

Gene Symbol	Signal applied to, or read from i^{th} electrode	Allowed values
p_i	Which electrode	0, 1, 2 ... 11 (for sets:A-D)
	is used	0, 1, 2 ... 15 (for set E)
s_i	Type	0 (constant) or
		1(square-wave)
a_i	Amplitude	0 , 1
f_i	Frequency	500 ,501 ... 10K
c_i	Cycle	0, 1, ... 100

TABLE 13
Description of genotype.

inputs etc) do not take place at the same instant. Mecobo maintains a schedule. Thus, it takes some time for Mecobo to circulate signals to each electrode.

8.2 Genotype Representation

In case of all sets of experiments, associated with each electrode there were five genes which defined 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 time (Section 3).

As we discussed in the sets A-D, each chromosome used $n_e = 12$ electrodes at a time. This means that each chromosome required a total of $12 \times 5 = 60$ genes. In all experiments, mutational children were created from a parent genotype by mutating a single gene. In the first four sets of experiments, this means that one of the 60 parent genes was mutated to create a child. Since in the set E, each chromosome used $n_e = 16$ electrodes, genotypes have $16 \times 5 = 80$ genes (so one gene in 80 was mutated). The values that genes could take are shown in Table 13 where i takes values 0, 1, ... 11 for first four sets of experiments and values 0, 1, ... 15 for fifth set of experiments.

The genotype for a chromosome of an individual consists of the 60 genes shown below:

$$p_0 s_0 a_0 f_0 c_0 \ldots p_{11} s_{11} a_{11} f_{11} c_{11}$$

And, the genotype for a chromosome of an individual consists of the 80 genes shown below:

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

For solution with 12 electrodes, the first 30 gene values of a chromosome are related to inputs and they are:

$p_0 s_0 a_0 \, f_0 c_0 \ldots p_5 s_5 a_5 \, f_5 c_5$

and, the last 12 gene values of a chromosome are related to outputs and they are:

$p_{10} s_{10} a_{10} \, f_{10} c_{10} \, p_{11} s_{11} a_{11} \, f_{11} c_{11}$

For solution with 16 electrodes, the first 40 gene values of a chromosome are related to inputs and they are:

$p_0 s_0 a_0 \, f_0 c_0 \ldots p_7 s_7 a_7 \, f_7 c_7$

and, the last 12 gene values of a chromosome are related to outputs and they are:

$p_{14} s_{14} a_{14} \, f_{14} c_{14} \, p_{15} s_{15} a_{15} \, f_{15} c_{15}$

In these input and output genes, only the p_j (here, the values of j are 0-5 and 10-11 for solutions with 12 electrodes and values of j are 0-7 and 14-15 for solutions with 16 electrodes) has any effect, the remainder are redundant. The gene p_j decides which electrode will be used for the inputs and outputs of the device. Thus, mutations in these genes can choose a different electrode to be used as an input or output.

8.3 Input Mapping

In sets A-C and E, the inputs to the electrode array were square waves with a fixed duty cycle. The cycle was determined by a linear mapping of the distance value of a sensor. Denote the distance value of i^{th} sensor by I_i, where i takes values 0-5 (corresponding to 6 sensors) or 0-7 (corresponding to 8 sensors). Denote the maximum and minimum distance values of a sensor by $I_{i_{max}}$ and $I_{i_{min}}$ respectively. Denote the maximum and minimum allowed cycle times be denoted by C_{max} and C_{min} respectively. Then the linear mapping given in Equation 7 allows the distance value of i^{th} sensor, I_i to map to a square-wave cycle time C_i which was applied to a given electrode.

$$C_i = a_i I_i + b_i \tag{7}$$

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

$$a_i = (C_{max} - C_{min})/(I_{i_{max}} - I_{i_{min}}) \tag{8}$$

and

$$b_i = (C_{min} I_{i_{max}} - C_{max} I_{i_{min}})/(I_{i_{max}} - I_{i_{min}}) \tag{9}$$

In the experiments we chose $I_{i_{min}}$=0, $I_{i_{max}}$=1023, $C_{min} = 0$ and $C_{max} = 100$. And, frequencies of input signals were 5000Hz and amplitudes were 1.

But, in the set D, frequency has been used instead of cycle time for input mapping, where maximum frequency value was 10KHz and minimum frequency value was 500Hz. The mapping equation was same as other sets, but instead of cycle time, frequency has been used in equation. The cycle time used in that set was 50 (each square wave was divided equally into on or off) and amplitude was 1.

8.4 Output Mapping

We determined the output, by examining the output buffers containing samples taken from the output electrodes. Since Mecobo 3.0 platform can only recognize binary values, the output buffers contain bitstrings. In sets A, B and E, the fraction of ones has been used to get output values. This fitness was used as it is cycle related. The fraction of ones in the output buffer was linearly mapped to motor speed in the ranges [-10, 10]. The Khepera simulator assumes motor speeds defined in the range [-10, 10]. The mapping is shown in Equation 10.

$$o_i = -10 + 20one_i/num_i \tag{10}$$

Where, one_i is the number of 1's in i^{th} output buffer and num_i is the total number of samples of i^{th} output buffer, o_i is the output value used to determine the i^{th} motor speed of the robot and i takes values 0 or 1 (corresponding to two motors).

In case of sets C and D, we used a different output mapping that used the *transitions* from 0 to 1 in the output buffers to calculate output value. This was done so that we could ascertain the relative merits of the two mappings. In the transition-based mapping the transitions in each output buffer were recorded and the gaps between consecutive transitions were measured and an average calculated. The thinking behind using a transition based fitness is that may be useful as it is cycle and frequency related. An example of average gap calculation for an output electrode is shown in Figure 7

The average transition gap was linearly mapped to the ranges [-10, 10]. This was done using Equation 11.

$$o_i = -10 + 20avg_i/(num_i - 2) \tag{11}$$

Where, avg_i is the average transition gap of i^{th} output buffer, num_i is the total number of samples of i^{th} output buffer, o_i is the output value to determine i^{th} motor speed of robot and i takes values 0 or 1 (corresponding to two motors). The highest average transition gap can be $(num_i - 2)$ when first transition

happens at index 0 (output value 0 at index 0 and value 1 at index 1) of output buffer and last transition happens at index ($num_i - 2$) (output value 0 at index ($num_i - 2$) and value 1 at index ($num_i - 1$)) and no other transition happens in the middle.

9 EXPERIMENTS

For each of the experiments a $1 + \lambda - ES$ evolutionary algorithm with $\lambda = 4$ was used. In case of all experiments, a child replaced the parent if its fitness was greater than or equal to the parent.

In case of first map (a), the starting position of the robot was center of the map and in case of second map (b), the starting position of the robot was upper left corner of the map.

9.1 Analysis of Results

In case of experiments A, in five cases a robot controller was evolved that could explore the complete map. In one of these cases however the robot eventually collided with wall. It was quite challenging for the robot to escape from the middle of the map, where there are lots of walls (note that robot started from the center position of the map). In three runs, the robot could not escape from the middle zone of the map because of colliding with the wall. In the case of the remaining two runs the robot could escape from the middle zone, but could not explore the full map because constantly exploring a limited region of the map (roughly half). The minimum number of generations required to produce a robot that could explore the full map without colliding with wall was 21.

In case of experiments B, for seven evolutionary runs, a robot controller was evolved that could explore the full map. However, in four cases the robot could explore the full map but eventually collided with a wall. Three other runs produced robot controllers that did not explore the full map. In one of these cases the robot could explore 4/5 of the map and was continuing its journey without colliding, but was stopped because it reached 5000 time steps. The minimum number of generations required to evolve a robot controller that allowed the robot to explore the full map was 20, however that robot collided with a wall later. The minimum number of generations it took to produce a controller capable of allowing a robot to explore the full map without colliding was 33.

In case of experiments C, it was found that in all ten runs robots could not even escape from upper portion from the map because of colliding with the obstacles, so none of these robots could explore full map. The highest moves that the robot could continue without colliding with an obstacle in those 10 runs was 247 and after the move 247 it collided with an obstacle.

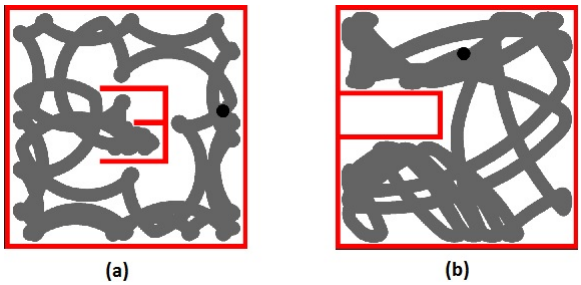


FIGURE 10
Paths of robots exploring the full map without colliding with obstacles. (a) The Path of the robot in case of one run of experiments A. (b) The Path of the robot in case of one run of experiments B. In (a) and (b), obstacles are shown in red, robot's current position is shown using a black ball and the path through which the robot has already visited the map is shown in grey.

In case of experiments D, we found that in all ten evolutionary runs robots collided with walls. In case of only 1 run, robot could escape from middle zone, but then collided and could not proceed further. So none of these robots could explore the full map. The highest moves that the robot could continue without colliding with an obstacle in those 10 runs was 91 and after the move 91 it collided with an obstacle.

In case of experiments E, in 4 evolutionary runs, the evolved robot could explore the full map. Of these, two evolved robots could explore the full map without colliding with a wall. In case of one run, the robot could explore 4/5 of the map and was continuing its journey without colliding, but was stopped because it reached 5000 time steps. The minimum number of generations to explore the full map without colliding was 32.

The third (C) and fourth (D) sets of experiments indicate that an output mapping based on average transition gap does not provide good results, which show that average transition gap is not suitable method for linear output mapping.

The result of the fifth (E) set of experiments with 6 configuration inputs were not as good as the result of the second (B) set of experiments with 4 configuration inputs because more configuration inputs might require a larger number of generations to find better solutions.

The paths of the robot exploring the full map without colliding with obstacles are shown in Figure 10 using two results of experiments A-E on two different maps (Figure 9 (a)-(b)). The movements of the robot that could not explore the full map due to colliding with obstacles are shown in Figure 11 using two results of experiments A-E on two different maps (Figure 9 (a)-(b)).

After all five sets of experiments two generalization experiments were performed. In first generalization experiment, the 4 solutions of experiments A,

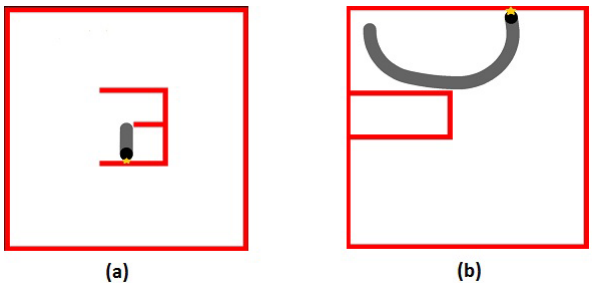


FIGURE 11
Paths of robots colliding with obstacles. (a) The Path of the robot in case of one run of experiments A. (b) The Path of the robot in case of one run of experiments C. In (a) and (b), obstacles are shown in red, robot's current position is shown using a black ball, the path through which the robot has already visited the map is shown in grey and the place where the robot has collided is shown as a yellow star.

which were able to explore the full map (the first map) without colliding with walls, were transferred to a second map to observe their behaviors. The starting position of the robot in second map was the upper left corner. It was found that only one solution allowed the robot to explore the full map without colliding with the wall. Two robots collided with walls and could not even escape from upper left corner of the map. One robot rotated in the same place for a long time and could not even escape from upper left corner of the map. In second generalization experiment, the three solutions, which could explore the full map (the second map) without colliding with walls from the experiments B, were transferred to first map to observe their behaviors. The starting position of the robot in the first map was the center of the map. It has been found that none of the robots could explore the full map and all of them collided with walls. In detail, two robots could not get out from the middle zone of the map and one robot could escape from the middle zone, but eventually collided with a wall and could not explore the full map. The greatest time that the robot survived without colliding with walls was 2222. As before, in the two generalization experiments, the robots were allowed to move up to 5000 timesteps, however, they were stopped as soon as they collided with walls or rotated in the same place for a long time (see sub-section 8.1).

In case of each of the five sets of simulated robot experiments (sets A-E), the elapsed time of an evolutionary run varied according to the length of the time that a robot could survive without colliding with a wall. Individuals of a population would take longer to run if they did not rotate in the same place or collide with an obstacle. In case of set A, if robot could not explore the full map within 100 generations, it took on an average less than 6 hours to

Set	No. Of Elec.	No. Of. In.	No. Of Conf.	In. Map.	Out. Map.	Time (ms)	Map	Res.
A	12	6	4	C	PO	20	Figure 9 (a)	4
B	12	6	4	C	PO	20	Figure 9 (b)	3
C	12	6	4	C	TG	20	Figure 9 (b)	0
D	12	6	4	F	TG	20	Figure 9 (a)	0
E	16	8	6	C	PO	25	Figure 9 (b)	2

TABLE 14
Experimental settings and the results of all sets of robot controlling experiments. All experiments were carried out up to 100 generations and 10 runs. The second, third and fourth columns show total number of electrodes, number of inputs and number of configuration inputs used in the experiments respectively. In case of all experiments, 2 electrodes were used for outputs. The fifth and sixth columns show the used input mapping (C=Cycle time mapping, F=Frequency mapping) and output mapping (PO=Percentages of ones, TG=Average transition gaps) respectively. The seventh column shows the input-output timing used for the experiments. The eighth column shows the map used in the experiments. The last column shows the result, where number of robots are given that explored the full area of the map without colliding with obstacles.

complete the evolutionary run, but if it was able to explore the full map, it took on an average of 9 hours to complete the run. In case of set B, if robot could not explore the full map within 100 generations, it took on an average 8 hours to complete the evolutionary run, but if it was able to explore the full map, it took on an average of less than 15 hours to complete the run. In case of set C, it took on an average less than 1 hour to complete the evolutionary run, but none of the robot could explore the full map within 100 generations. In case of set D, it took on an average less than 4 hours to complete the evolutionary run, but none of the robot could explore the full map within 100 generations. In case of set E, if robot could not explore the full map within 100 generations, it took on an average less than 5 hours to complete the evolutionary run, but if it was able to explore the full map, it took on an average of less than 9 hours to complete the run.

9.2 Summary Of Robot Controlling Experiments

The experimental settings and the results of all sets of robot controlling experiments have been summarized in table 14 to get a clear overview of the experiments.

The outcomes obtained by the analysis of the results of these robot controlling experiments are shown below:

- Average transition gap is not suitable method for linear output mapping.
- The results of the experiments with 6 configuration inputs were not as good as the results of the experiments with 4 configuration inputs. This may be because more configuration inputs require a larger number of generations to find better solutions.

- After the generalization experiments, one robot has been found that could explore full area of a map in case of both maps without colliding with an obstacle.

10 CONCLUSIONS

EIM is hybrid of digital and analogue computing where digital computers are used to configure materials to carry out analogue computation. This holds the promise of developing entirely new computational devices. A purpose-built evolutionary platform called Mecobo, has been used to evolve configurations of a physical system to classify data and control a robot. The material used is a mixture of SWCNT and a polymer. The aim of the paper is not to show that the experimental results of controlling robots using EIM is currently competitive with state-of-the-art but rather to start a new beginning in the world of computation. In some cases, we found that the robot could explore the full map without colliding with walls. In other experiments using Mecobo we have obtained encouraging results on a machine learning classification problem. In principle, a classifier can be implemented using an electrode array and a material sample on a microscope slide and some interfacing electronics. Such a system could act as a low power standalone data classification device. In addition, we carried out many experimental investigations on factors that could influence the efficiency of the technique in solving computational problems. We tried to find out which ratio of SWCNT to polymer material is the most effective for computation. We found that if the percentage by weight of SWCNT was greater than or equal to a certain value (0.02), the computational efficiency of all mixtures were statistically indistinguishable. We also found that different organizations of electrodes do not matter in computation and the choice of polymer in material mixture plays no important role in computation. We also showed that analogue signals (input, output and configuration inputs) perform considerably better than digital signals. Experiments on robot control indicated that the output mapping of data read from the electrode array for the speeds of robot's motors is an important factor.

There are many questions for the future. Does evolutionary computation in materio scale well on larger problem instances. What other classes of computational problems are solvable using this technique? What are the most suitable materials and signal types for EIM? How much computation can we extract from a small amount of material?

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] V. Akbarzadeh, Alireza Sadeghian, and M.V. dos Santos. (June 2008). Derivation of relational fuzzy classification rules using evolutionary computation. In *IEEE Int. Conf. on Fuzzy Systems*, pages 1689–1693.

[2] K. Bache and M. Lichman, (2013). UCI machine learning repository.

[3] W. Banzhaf, G. Beslon, S. Christensen, J.A. Foster, F. Képès, V. Lefort, J. F. Miller, M. Radman, and J. Ramsden. (2006). Guidelines: From artificial evolution to computational evolution: a research agenda. *Nature Reviews Genetics*, 7:729–735.

[4] J. Bongard. (2013). Evolutionary robotics. *Communications of the ACM*, 56(8):74–85.

[5] H. Broersma, F. Gomez, J. F. Miller, M Petty, and G. Tufte. (2012). NASCENCE Project: Nanoscale Engineering for Novel Computation Using Evolution. *International Journal of Unconventional Computing*, 8(4):313–317.

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

[7] R. A. Fisher. (1936). The use of multiple measurements in taxonomic problems. *Annual Eugenics*, 7(2):179–188.

[8] Simon Harding, Vincent Graziano, Jürgen Leitner, and Jürgen Schmidhuber. (2012). MT-CGP: mixed type Cartesian genetic programming. In *Proceedings of the 14th Annual Conference on Genetic and Evolutionary Computation*, pages 751–758. ACM.

[9] Simon Harding and Julian F. Miller. (2004). Evolution in materio: A tone discriminator in liquid crystal. In *In Proceedings of the Congress on Evolutionary Computation 2004 (CEC'2004)*, volume 2, pages 1800–1807.

[10] Simon Harding and Julian F. Miller. (2005). Evolution in materio : A real time robot controller in liquid crystal. In *Proceedings of NASA/DoD Conference on Evolvable Hardware*, pages 229–238.

[11] Simon Harding and Julian Francis Miller. (2009). Evolution in materio. In Robert A. Meyers, editor, *Encyclopedia of Complexity and Systems Science*, pages 3220–3233. Springer.

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

[13] Simon L. Harding, Julian F. Miller, and Edward A. Rietman. (2008). Evolution in materio: Exploiting the physics of materials for computation. *International Journal of Unconventional Computing*, 4(2):155–194.

[14] M. Hollander and D.A. Wolfe. (1973). *Nonparametric statistical methods*. Wiley.

[15] Odd Rune Lykkebø, Simon Harding, Gunnar Tufte, and Julian F. Miller. (2014). Mecobo: A hardware and software platform for in materio evolution. In Oscar H. Ibarra, Lila Kari, and Steffen Kopecki, editors, *Unconventional Computation and Natural Computation*, LNCS, pages 267–279. Springer International Publishing.

[16] Olivier MIchel, (1996). Khepera simulator version 2.0 user manual".

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

[18] Julian F. Miller, editor. (2011). *Cartesian Genetic Programming*. Springer.

[19] Julian F Miller and Keith Downing. (July 2002). Evolution in materio: Looking beyond the silicon box. *Proceedings of NASA/DoD Evolvable Hardware Workshop*, pages 167–176.

[20] Julian F. Miller, Simon L. Harding, and Gunnar Tufte. (2014). Evolution-in-materio: evolving computation in materials. *Evolutionary Intelligence*, 7:49–67.

[21] M. Mohid, J.F. Miller, S.L. Harding, G. Tufte, O.R. Lykkebø, M.K. Massey, and M.C. Petty. (2014). 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.*, pages xx–xx. IEEE Press.

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

[23] Stefano Nolfi and Dario Floreano. (2001). *Evolutionary Robotics. The Biology, Intelligence, and Technology of Self-organizing Machines*. MIT Press, Cambridge, MA, USA.

[24] A. Thompson and P. Layzell. (April 1999). Analysis of unconventional evolved electronics. *Communications of the ACM*, 42(4):71–79.

[25] Adrian Thompson. (1998). *Hardware Evolution - Automatic Design of Electronic Circuits in Reconfigurable Hardware by Artificial Evolution*. Springer.

[26] András Vargha and Harold D Delaney. (2000). 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.

[27] Katharina Völk, Julian F. Miller, and Stephen L. Smith. (2009). Multiple network CGP for the classification of Mammograms. In *Proceedings of the EvoWorkshops 2009 on Applications of Evolutionary Computing*, EvoWorkshops '09, pages 405–413. Springer-Verlag.