Evolution-In-Materio: Solving Function Optimization Problems Using Materials

Maktuba Mohid[1], Julian F. Miller[1], Simon L. Harding[1], Gunnar Tufte[2], Odd Rune Lykkebø[2], Mark K. Massey[3], and Michael C. Petty[3]

[1]Department of Electronics, University of York, York, UK. Emails: [mm1159, julian.miller]@york.ac.uk, slh@evolutioninmaterio.com

[2]Department of Computer and Information Science, Norwegian University of Science and Technology, 7491 Trondheim, Norway. Emails: [gunnart, lykkebo]@idi.ntnu.no

[3]School of Engineering and Computing Sciences and Centre for Molecular and Nanoscale Electronics, Durham University, UK. Emails: [m.k.massey, m.c.petty]@durham.ac.uk

Abstract—**Evolution-in-materio (EIM) 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 computational problems. We demonstrate for the *first time* that this methodology can be applied to function optimization. We evaluate the approach on 23 function optimization benchmarks and in some cases results come very close to the global optimum or even surpass those provided by a well-known software-based evolutionary approach. This indicates that EIM has promise and further investigations would be fruitful.**

Keywords—evolutionary algorithm, evolution-in-materio, material computation, evolvable hardware, function optimization

I. Introduction

Natural evolution can be seen as an algorithm which exploits the physical properties of materials. Evolution-in-materio (EIM) aims to mimic this by manipulating physical systems using computer controlled evolution (CCE) [6], [7], [8], [12]. In the main, EIM aims to exploit the properties of physical systems for solving *computational* problems. It is important to note, that one of unique features of EIM is that it aims to exploit physical processes that a designer may either be unaware of or not know how to utilize. This is discussed in more detail in a recent review of EIM [13].

EIM was inspired by the 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 (FPGA). He evolved a digital circuit that could discriminate between 1kHz or 10kHz signal [16]. When the evolved circuit was analysed, Thompson discovered that artificial evolution had exploited physical properties of the chip. Despite considerable analysis and investigation Thompson and Layzell were unable to pinpoint what exactly was going on in the evolved circuits [17].

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 solve a number of computational problems [4]:

- Two input logic gates: OR, AND, NOR, NAND, etc. [7].
- Tone Discriminator: A device was evolved which could differentiate different frequencies [4].
- Robot Controller: A controller for a simulated robot with wall avoidance behavior [5].

Previous work on evolution-in-materio has used either silicon in the form of an FPGA or a liquid crystal display [13]. Both these platforms are designed for other uses and it is unclear whether these materials are particularly suitable for evolution-in-materio. In addition, computational problems previously solved using these approaches have not been standard computational benchmarks.

In this paper, we describe the use of a purpose built platform called Mecobo that facilitates computer controlled evolution of a material [10] for solving a computational problem. The Mecobo platform has been developed within an EU funded research project called NASCENCE [2]. The computational material we have used in this investigation is a mixture of single-walled carbon nanotubes 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. In addition NASCENCE is tasked with assessing the utility of evolution-in-materio on a wide variety of computational problems (including standard benchmarks). In other very recent work, the technique has been applied to solving travelling salesman problems [3], and classification problems [15]. Here, we apply the technique, for the first time to the benchmark problem of function optimization.

Evolutionary computation has been widely used to solve complex multi-modal optimization functions. Here, we show that using the Mecobo platform it is possible to evolve solutions to benchmark function optimization problems. This is the first time EIM has been used to solve function optimization problems. Our aim is not to claim EIM as a competitive method for solving function optimization problems, we are simply trying to apply EIM to standard computational benchmark problems so that we have a yardstick to assess various aspects of EIM using the Mecobo platform. For instance, what type of signals are appropriate, what 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 (see later) 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 [13].

To assess the feasibility of using the EIM method for function optimization we needed to compare it with a software-based evolutionary technique using the same number of function evaluations. To do this we have compared its performance with that of Cartesian Genetic Programming (CGP) on the same set of optimization benchmarks. We did this because we have the CGP software to hand, and we have already shown that the latter produces reasonable results [14].

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 function optimization problem. The way we have used the Mecobo platform for function optimization is described in Sect. VI. We describe our experiments and analysis of results in Sect. VII. Finally we conclude and offer suggestions for further investigation in Sect. VIII.

II. Conceptual Overview Of Evolution-In-Materio

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 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 [13]. 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

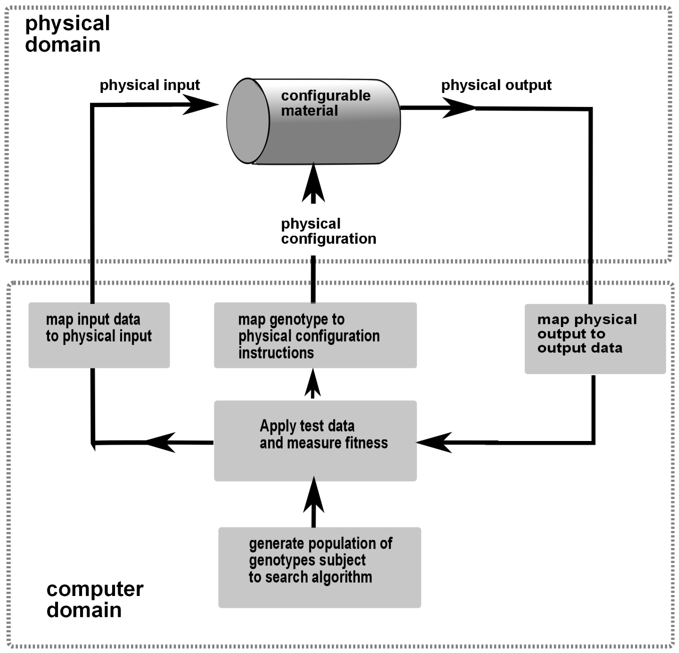

physical domain
physical input
configurable material
physical output
physical configuration
map input data to physical input
map genotype to physical configuration instructions
map physical output to output data
Apply test data and measure fitness
generate population of genotypes subject to search algorithm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

Fig. 1: Concept of evolution-in-materio [13].

exploits the physical properties of materials (mainly proteins). Banzhaf et al. discussed the importance of physicality and embodiment [1]. 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 [12], [13].

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

The Mecobo hardware platform has been designed and built within an EU-funded research project called NASCENCE [2].

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 [10].

It is important to appreciate that in EIM the computational substrate is piece of material for which the appropriate physical

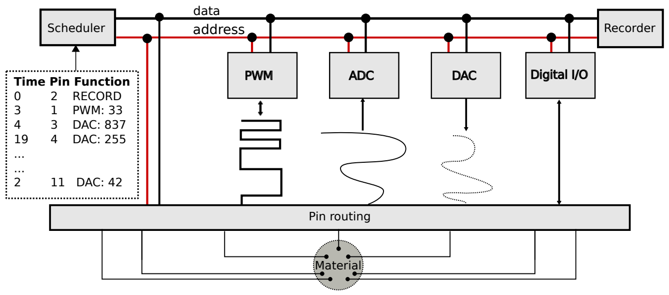


Fig. 2: Overview of the complete system.

variables to be manipulated by evolution may be poorly understood (see Fig 1). This means that the selection of signal types, i.e. inputs, outputs and configuration data, 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 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 digital signals, digital pulse trains. The next version of Mecobo will allow 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.

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

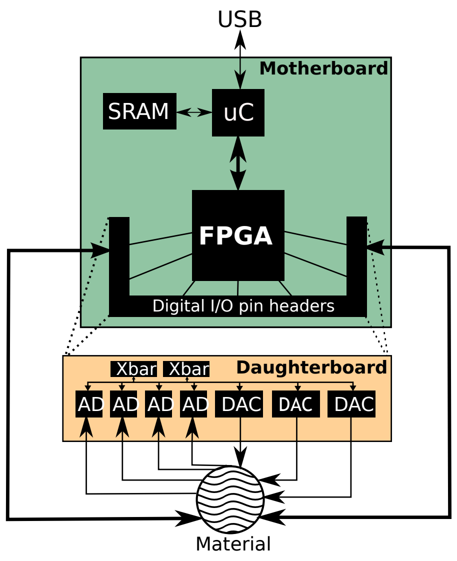
Figure 3(b) shows the motherboard with the Xilinx LX45 FPGA, Silicon Labs ARM based EFM32GG990 micro controller connected to a 12 terminal material sample.

At present the Mecobo hardware allows only two types of inputs to the material: constant voltage (0V or 3.5V) or a square wave signal. However, different characteristics or input parameters associated with these inputs can be chosen. These input parameters are described in Table I.

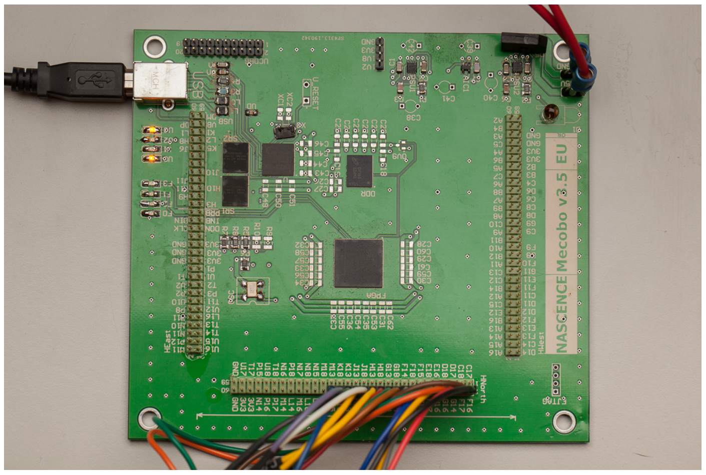
TABLE I: Adjustable Mecobo input parameters.

Parameter Name	**Description**	**Note**
Amplitude	0 or 1 corresponding to 0V or 3.5V	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.

The start time and end time of each input signal determines how long an input is applied. Mecobo only samples using digital voltage thresholds, hence the material output is interpreted



(a) Mecobo block diagram.



(b) Picture of Mecobo.

Fig. 3: Hardware interface implementation overview.

as strictly high or low, (i.e. 0 or 1). In later versions of this hardware, analogue inputs and outputs will be possible.

Also, in the case that an electrode is chosen to be read, a user-defined output sampling frequency determines the buffer size of output samples. If the output frequency is F_{out}, start time $Time_{start}$ and end time is $Time_{end}$, then the buffer size is Buf_{size} is given by:

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

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

However, in practice due to pin latency, the real buffer size is generally smaller.

Fig. 4: Electrode array with sample.

IV. Description Of Physical Computational Material

The experimental material consists of single-walled carbon nanotubes mixed with polymethyl methacrylate (PMMA) and dissolved in anisole (methoxybenzene) [1]. The sample is baked causing the anisole to evaporate. This results in material which is mixture of carbon nanotube and PMMA. The concentration of carbon nanotube is 0.71% (weight% fraction of PMMA).

Carbon nanotubes are conducting or semi-conducting and role of the PMMA 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:

- A M3-sized nylon washer was glued on the electrode array to contain the material whilst drying;
- 20 μL of material were dispensed into the washer;
- This was dried at $\approx 100^o$ C for $\approx$1 h to leave a "thick film".

The experimental material is placed in the middle of a plate of the electrode array. Twelve gold electrodes are connected directly with the experimental material in the plate. The electrode array is connected directly with the Mecobo board via wires. The electrode sample is shown in Fig. 4.

V. Function Optimization

Benchmark function optimization problem are functions, $f(x_i)$ of a number (n) of real-valued variables, where $i = 1, 2, \ldots n$. The aim is to obtain the values of x_i which cause $f(x_i)$ to be a minimum. In evolutionary computation many complex, multi-modal functions have been designed whose minima are known, but are challenging functions to minimise using search algorithms. An example of such a function is given in Eqn. 2 [19] and the two-dimensional version is illustrated in Fig. 5. In general these functions have many dimensions (typically 30).

$$f_8(x) = \sum_{i=1}^{d} -x_i \sin(\sqrt{|x_i|}) \qquad (2)$$

Optimization functions are typically defined over a variety of ranges for each variable, x_i. For instance, f_8 is defined over $-500 \leq x_i \leq 500$ and has a global minimum given by $min(f_8(x)) = f_8(420.9687, ..., 420.9687) = -12,569.5$ [19].

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

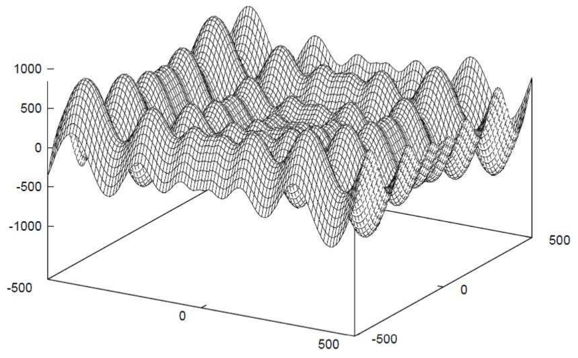


Fig. 5: Function optimization problem $f_8(x_1, x_2)$.

Here in this experiment, we have chosen 17/23 benchmark functions (Function 1, 3 - 11, 14 - 16, 18, 21 - 23) from [19] and 6/23 functions (Function 2, 12 - 13, 17, 19 - 20) from [20].

VI. Optimizing Functions Using Evolution-In-Materio

A. Methodology

The experiments were performed with an electrode array having twelve electrodes. In the experiment, reported here, one electrode has been used as output and the remaining electrodes have been used as configuration voltages. No inputs were needed. The configuration voltages affect the electrical behaviour of the carbon nanotube-polymer material and the interaction induces certain voltages on the output electrode. It is this unknown mapping that is being exploited by computer-controlled evolution.

We read a series of output values (0 or 1) from a buffer of samples taken from a single electrode. These values were used to define the value of a variable 'x_i' in function optimization problem (see Sect. V). As the optimization functions have more than one dimension, more than one output from the device was needed. In those cases, a split genotype technique has been used. In this technique, we used a genotype consisting of multiple chromosomes, each of which was applied to eleven electrodes. On each application of a chromosome we read the binary values in an output buffer of samples from the remaining electrode. For instance, for a 30 variable optimization problem we used 30 chromosomes. Each chromosome defined which electrode would be read and which electrodes would receive the configuration data (square waves or constant voltage). There can be many other ways of using the twelve electrodes. We could have read two electrodes and applied evolved configuration data to the remaining ten and other choices are possible. Examining other choices remains for future work.

Using the Mecobo platform we can control the time (in milliseconds) that a signal is applied to the material (see Sect. III). Here, we accumulated output values in a buffer for 128 milliseconds. The number of samples of output buffer can be controlled by start time, end time and frequency of output electrode. In experiment, we used a 25KHz buffer sampling frequency.

TABLE II: Description of genotype.

Gene Symbol	**Signal applied to, or read from i^{th} chromosome and j^{th} electrode**	**Allowed values**
$p_{i,j}$	Which electrode is used	0, 1, 2 ...11
$s_{i,j}$	Type	0 (constant) or 1(square-wave)
$a_{i,j}$	Amplitude	0 , 1
$f_{i,j}$	Frequency	500 ,501 ...10K
$ph_{i,j}$	Phase	1, 2 ...10
$c_{i,j}$	Cycle	0, 1, ...100

B. Genotype Representation

Each chromosome used $n_e = 12$ electrodes at a time. Associated with each electrode there were six genes which either define which electrode was used as an output, or characteristics of the input applied to the electrode: signal type, amplitude, frequency, phase, cycle (see Sect. III). So each chromosome required a total of 72 genes and a genotype of 30 chromosomes required a total of 2106 (72x30) genes. Mutational offspring were created from a parent genotype by mutating a single gene (i.e one gene of 2106). The values that genes could take are shown in Table II. The chromosome index, i takes values $0, 1, \dots d-1$, where d is the number of dimensions of the function optimization problem and the electrode index, j takes values $0, 1, \dots n_e - 1$.
The i^{th} chromosome, C_i is defined by:

$$C_i = p_{i,0}s_{i,0}a_{i,0}f_{i,0}ph_{i,0}c_{i,0} \dots p_{i,11}s_{i,11}a_{i,11}f_{i,11}ph_{i,11}c_{i,11}$$

The genotype for a d dimensional problem is a collection of d chromosomes: $C_0C_1 \dots C_{d-1}$

C. Output Mapping

To determine a real-valued output from a collection of ones in an output buffer it was decided to use the fraction of ones. However, initial findings revealed that the output buffer never contained more than 40% ones. As a result, before the function optimization experiment, an initial evolutionary investigation was performed to discover the typical contents of an output buffer under various conditions. The fraction of number of ones in the output buffer was calculated to obtain the values of the variable required to optimize functions. However, because the buffer contained a maximum of 40% ones, the fraction of ones was multiplied by 2.5 so that a real-valued output would take values between 0 and 1. We denote this value by q.

In the initial investigation, evolutionary runs were carried out to find the electrode configurations (which electrode is used as output or configuration voltage, signal type, amplitude, phase, cycle, frequency of configuration voltages) that gave different percentages of ones in the output buffer. The different percentages were 0%, 10%, 20%, 30% and 40%. The evolved electrode configurations that gave these percentages were used to seed the initial populations for the evolutionary runs for the function optimization problems.

These real values determined from the fraction of ones in the output buffer were linearly mapped to the ranges that particular variables were allowed to take in various optimization functions. This was done as follows. The value of the fraction was then mapped using minimum and maximum values of benchmark function. Let, max_i and min_i be the ranges allowed for a variable, x_i in a function optimization problem. Then the equation used for calculating the linearly mapped output value, x_i is given by:

$$x_i = min_i + (max_i - min_i)q \tag{3}$$

The linearly mapped output values x_i were determined corresponding to each chromosome to obtain the measured vector minimizing the optimization function.

VII. Experiments

Twenty-three benchmark functions of function optimization problem were investigated. A $1 + \lambda - ES$, evolutionary algorithm with $\lambda = 4$ was used [11] and run for 5000 generations. 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. The experiment was performed over 10 independent runs in case of each benchmark function. Only 10 runs were undertaken as it took over 7 days for these experiments. Different functions took different time due to different number of dimensions. Elapsed time increased with the number of dimensions.

A. CGP experimental details

To evaluate the effectiveness of the EIM method for solving function optimization problems we compared results with an evolutionary search technique called Cartesian Genetic Programming using a $1 + 4$ evolutionary algorithm over the same number of generations [2].

Cartesian genetic programming is a graph-based form of genetic programming [11]. 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. Five constant inputs (terminals) are generated randomly in the interval [-1, 1] at the start of each evolutionary run. The function set chosen for this study is defined over the real-valued interval [-1.0, 1.0] and is shown in Table III.

The number of outputs is $n_o = d$, where d is the dimensionality of the optimization problem. Since the terminals and functions all return numbers in the interval [-1, 1] the program outputs, q_i also have values defined in this range. However, as noted previously, the optimization functions are defined over a variety of intervals. Thus the program outputs, q_i, need to be mapped to the intervals defined in the optimization problem, x_i. Equation 4 gives the mapping.

$$x_i = \frac{max_i - min_i}{2} q_i + \frac{max_i + min_i}{2}. \tag{4}$$

We used *three* mutation parameters. A probability of mutating connections, μ_c, functions, μ_f and outputs, μ_o. In all experiments $\mu_c = 0.01$, $\mu_f = 0.03$, and $\mu_o = 0.04$.

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.

TABLE III: Node function gene values and their definition.

Value	Definition
0	$\sqrt{\lvert z_0 \rvert}$
1	${z_0}^2$
2	${z_0}^3$
3	$(2exp(z_0+1) - e^2 - 1)/(e^2 - 1)$
4	$\sin(z_0)$
5	$\cos(z_0)$
6	$\lvert z_0 \rvert^{\lvert z_1 \rvert}$
7	$\sqrt{({z_0}^2 + {z_1}^2)/2}$
8	$(z_0 + z_1)/2$
9	$(z_0 - z_1)/2$
10	$z_0 z_1$
11	**if** $\lvert z_1 \rvert < 10^{-10}$ **then** 1 **else if** $\lvert z_1 \rvert > \lvert z_0 \rvert$ **then** z_0/z_1 **else** z_1/z_0
12	**if** $z_0 > z_1$ **then** $z_2/2$ **else** $1 - z_2/2$

B. Analysis of Results

The results of the two sets of experiments are compared using the non-parametric two sided Mann-Whitney U-test and the two-sample Kolmogorov-Smirnov (KS) test [9]. We also computed with effect size [18] statistic . A U-or KS test value of < 0.05 indicates that the difference between two dataset is statistically significant. The effect size, A value shows the important 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 statically significant with a medium/large effect size, then we can be reasonably sure that any difference is not due to under sampling.

The experiments show that in 7/23 functions the best results with the experimental material are equal to optimum results and in case of 11/23 functions the best results are very close to optimum results. In 4 cases the average results with the experimental material are equal to optimum results and in 13 cases average results are very close to optimum results. In 10/23 functions the best results of experimental material are better than or equal to the best results of CGP. In case of 6/23 functions the average results of experimental material are

[2]In both cases of experimental material and CGP, offspring replaced parents if their fitness was greater than *or equal to* the parent

statistically better or at least equal to the average results of CGP. For details see table IV.

It should be noted that CGP has been compared [3] with Differential Evolution (DE), Particle Swarm Optimization (PSO), Evolutionary Algorithm (SEA). Comparisons showed that in 15/20 benchmarks CGP is same or better than DE, in 19/20 cases CGP is same or better than PSO or SEA [14]. So CGP itself is a competitive technique for function optimization.

VIII. Conclusions And Future Outlook

Evolution-in-materio 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 obtain the minima of complex, multi-modal mathematical functions. 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 function optimization problems using EIM is competitive with state-of-the-art function optimization algorithms 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 benchmark function optimization problems. In some cases, we found that we could find a solution closer to the global optimum using the EIM approach than an effective software-based evolutionary technique. In other work using Mecobo it has been shown that digital logic functions can be implemented [10]. We are currently obtaining encouraging results on machine learning classification problems. In these cases, 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 standalone device. This could have utility in robot control.

There, of course, remain 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 evolution-in-materio? The Mecobo platform is currently under development and the next version will be able to allow the utilization of analogue voltages. This may make some types of computational problems more readily solved using evolution-in-materio.

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.

[3] Based on average results over 30 independent runs, and 500000 evaluations for each run

References

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

[2] Broersma, H., Gomez, F., Miller, J.F., Petty, M., 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., Miller, J.F.: Evolution in materio. In: R.A. Meyers (ed.) Encyclopedia of Complexity and Systems Science, pp. 3220–3233. Springer (2009)

[7] 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)

[8] Harding, S.L., Miller, J.F., Rietman, E.A.: Evolution in materio: Exploiting the physics of materials for computation. International Journal of Unconventional Computing **4**(2), 155–194 (2008)

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

[10] Lykkebø, O.R., Harding, S., Tufte, G., Miller, J.F.: Mecobo: A hardware and software platform for in materio evolution. In: O.H. Ibarra, L. Kari, S. Kopecki (eds.) Unconventional Computation and Natural Computation, LNCS, pp. 267–279. Springer International Publishing (2014)

[11] Miller, J.F. (ed.): Cartesian Genetic Programming. Springer (2011)

[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] Miller, J.F., Mohid, M.: Function optimization using cartesian genetic programming. In: GECCO (Companion), pp. 147–148 (2013)

[15] 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)

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

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

[18] 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)

[19] Vesterstrom, J., Thomsen, R.: A comparative study of differential evolution, particle swarm optimization, and evolutionary algorithms on numerical benchmark problems. In: Evolutionary Computation, 2004. CEC2004. Congress on, vol. 2, pp. 1980 – 1987 (2004)

[20] Yao, X., Liu, Y.: Fast evolutionary programming. In: In L. J. Fogel et al. (Eds.), Proceedings of the 5th Annual Conference on Evolutionary Programming, pp. 451–460. MIT Press (1996)

TABLE IV: Comparative results of experimental material with CGP on 23 benchmark optimization functions. The best and average results for both CGP and experimental material are computed from 10 independent evolutionary runs. All results are for 5000 generations. The "Res." column shows whether the results of experimental material is equal to or close to optimum or not. '✓' indicates the result is equal to or close to optimum and 'X' indicates the result is not close to optimum. The first results of this column are according to the best results of experimental material and the second results are according to the average results of experimental material. The "Com. Res." column shows the comparison between the best and average results of the experimental material with CGP. The '+' indicates the result with the material is equal or better than the result of CGP and '-' indicates the result of experimental material is worse. The first result of this column shows comparison of best results and second result of this column shows comparison of average results. "U-Test", "KS-Test" and "Effect Size" columns show results of statistical significance test. The statistical significance tests are performed over the results of all 10 runs of all 23 functions. '✓' of "U-Test", "KS-Test" columns indicates that the difference between the two datasets is statistically significant and 'X' indicates that the difference is not statistically significant. 'n/a' of "U-Test" column indicates that the test is not applicable for the datasets. The tests are not applicable when all the data of both of these datasets are same. In these cases all evolutionary runs in both cases (in-materio and CGP) converged to the known global optimum.

No.	Expected Output	Best Results of Experimental Material	Average Results of Experimental Material	Best Results of CGP	Average Results of CGP	Res.	Com. Res.	U-Test (<0.05)	KS-Test (<0.05)	Effect Size
1	0	1.547937E-05	3.261765E-05	0	0	✓ ✓	- -	✓	✓	Large
2	0	1.013977E-02	2.372435E-02	0	0	✓ ✓	- -	✓	✓	Large
3	0	127.902	1614.87	3.938	3.064647E+03	X X	- +	X	X	Large
4	0	2.038043E-01	5.600358E-01	0	0	✓ ✓	- -	✓	✓	Large
5	0	1.136536E-01	3.871166E-01	27.690461	37.668384	✓ ✓	+ +	✓	✓	Large
6	0	0	0	0	0	✓ ✓	+ +	n/a	X	Small
7	0	3.813224E-03	1.071498E-02	1.398843E-03	1.082964E-02	✓ ✓	- +	X	X	Small
8	-12569.487	-12451.028	-12255.608	-12569.450	-12569.330	✓ ✓	- -	✓	✓	Large
9	0	2.992018	5.634806	0	0	X X	- -	✓	✓	Large
10	0	1.166181E-02	3.490709E-02	1.439820E-16	1.439820E-16	✓ ✓	- -	✓	✓	Large
11	0	3.395043E-03	8.345206E-02	0	0	✓ ✓	- -	✓	✓	Large
12	0	1.047303E-01	1.151369E-01	3.072479E-01	6.513926E-01	✓ ✓	+ +	✓	✓	Large
13	0	4.336251E-05	1.587237E-04	7.415736E-06	1.176390E-03	✓ ✓	- +	X	X	Small
14	0.9980038	0.9980038	0.9980038	0.9980038	0.9980038	✓ ✓	+ +	n/a	X	Small
15	0.0003075	3.084965E-04	3.430931E-04	4.437455E-04	1.059346E-03	✓ ✓	+ +	✓	✓	Large
16	-1.0316284	-1.0316284	-1.0316229	-1.0316284	-1.0307720	✓ ✓	+ +	X	X	Medium
17	0.397887	0.397887	0.397916	0.397887	0.397994	✓ ✓	+ +	X	X	Medium
18	3.0	3.0	21.906419	3.0	3.0	✓ X	+ -	✓	X	Large
19	-3.86	-3.86	-3.86	-3.86	-3.86	✓ ✓	+ +	n/a	X	Small
20	-3.32	-3.32	-3.32	-3.32	-3.286059	✓ ✓	+ +	n/a	X	Small
21	-10.153196	-5.100772	-5.100771	-10.153199	-8.636950	X X	- -	✓	✓	Large
22	-10.402819	-5.128822	-5.128822	-10.402864	-8.820119	X X	- -	✓	✓	Large
23	-10.536284	-5.175645	-5.175644	-10.536290	-9.463113	X X	- -	✓	✓	Large