Training Single Walled Carbon Nanotube based Materials to perform computation

QAISER, FAWADA

Training Single Walled Carbon Nanotube based Materials to perform computation

Fawada Qaiser

A Thesis presented for the degree of
Doctor of Philosophy

School of Engineering
University of Durham
England

January 2018

Dedicated to

This thesis is dedicated to my children, Tahreem and Ibrahim.

You have made me stronger, better and more fulfilled person than I could have ever imagined.

Training Single Walled Carbon Nanotube based Materials to perform computation

Fawada Qaiser

Submitted for the degree of Doctor of Philosophy
January 2018

Abstract

This thesis illustrates the use of Single Walled Carbon Nanotube based materials for the solution of various computational problems by using the process of computer controlled evolution. The study aims to explore and identify three dimensions of a form of unconventional computing called, 'Evolution-in-materio'. First, it focuses on identifying suitable materials for computation. Second, it explores suitable methods, i.e. optimisation and evolutionary algorithms to train these materials to perform computation. And third, it aims to identify suitable computational problems to test with these materials.

Different carbon based materials, mainly single walled carbon nano-tubes with their varying concentrations in polymers have been studied to be trained for different computational problems using the principal of 'evolution-in-materio'. The conductive property of the materials is used to train these materials to perform some meaningful computation. The training process is formulated as an optimisation problem with hardware in loop. It involves the application of an external stimuli (voltages) on the material which brings changes in its electrical properties. In order to train the material for a specific computational problem, a large number of configuration signals need to be tested to find the one that transforms the incident signal in such a way that a meaningful computation can

be extracted from the material. An evolutionary algorithm is used to identify this configuration data and using a hardware platform, this data is transformed into incident signals. Depending on the computational problem, the specific voltages signals when applied at specific points on to the material, as identified by an evolutionary algorithm, can make the material behave as a Logic gate, a tone discriminator or a data classifier.

The problem is implemented on two types of hardware platforms, one a more simple implementation using mbed (a micro- controller) and other is a purpose-built platform for 'Evolution-in-materio" called Mecobo.

The results of this study showed that the single walled carbon nanotube composites can be trained to perform simple computational tasks (such as tone discriminator, AND, OR logic gates and a Half adder circuit), as well as complex computational problems such as Full Adder circuit and various binary and multiple class machine learning problems.

The study has also identified the suitability of using evolutionary algorithms such as Particle Swarm Optimisation algorithm (PSO) and Differential evolution for finding solutions of complex computational problems such as complex logic gates and various machine learning classification problems.

The implementation of classification problem with the carbon nanotube based materials also identified the role of a classifier. It has been found that K-nearest neighbour method and its variant kNN ball tree algorithm are more suitable to train carbon nanotube based materials for different classification problems.

The study of varying concentrations of single walled carbon nanotubes in fixed polymer ratio for the solution of different computational problems provided an indication of the link between single walled carbon nanotubes concentration and ability to solve computational problem.

The materials used in this study showed stability in the results for all the

considered computational problems. These material systems can compliment the current electronic technology and can be used to create a new type of low energy and low cost electronic devices. This offers a promising new direction for evolutionary computation.

Declaration

I hereby declare that the work carried out in this thesis has not been previously submitted for any degree and is not currently being submitted in candidature for any other degree

Publications

0.1 Publications from this work

- **F. Qaiser**, A. Kotsialos, M. K. Massey, D. A. Zeze, C. Pearson, M. C. Petty, "Training disordered carbon nanotubes to solve machine learning problems using kNN and PSO algorithms", International Journal of Unconventional Computing, January, 2018 *under review*

- M. K. Massey, A. Kotsialos, **F. Qaiser**, D. A. Zeze, C. Pearson, M. C. Petty, "Computing with Carbon Nanotubes: Optimization of Threshold Logic Gates using Disordered Nanotube/Polymer Composites", Journal of Applied Physics, March, 2015

- D. Volpati, M. K. Massey, D. W. Johnson, A. Kotsialos, **F. Qaiser**, C. Pearson, K. S. Coleman, G. Tiburzi, D. A.Zeze, and M. C. Petty, "Exploring the alignment of carbon nanotubes dispersed in a liquid crystal matrix using coplanar electrodes", Journal of Applied Physics,February 2015

- A. Kotsialos, M. K. Massey, **F. Qaiser**, D. A. Zeze, C. Pearson, M. C. Petty, "Logic Gate and Circuit Training on Randomly Dispersed Carbon Nanotubes", International Journal of Unconventional Computing, 10(5):473 - 497, 2014

0.2 Conferences and poster presentations

- **F. Qaiser**, A. Kotsialos, M. K. Massey, D. A. Zeze, C. Pearson, M. C. Petty, "Training disordered carbon nanotubes based materials as tone discriminators", School of Engineering & Computing Sciences Durham University, annual research day, October, 2017 (Poster presentation)

- **F. Qaiser**, A. Kotsialos, M. K. Massey, D. A. Zeze, C. Pearson, M. C. Petty, "Manipulating the conductance of single-walled carbon nanotubes based thin films for evolving threshold logic circuits using particle swarm optimisation", IEEE Congress on Evolutionary Computation (CEC), July, 2016

- **F. Qaiser**, A. Kotsialos, M. K. Massey, D. A. Zeze, C. Pearson, M. C. Petty, "Solving simple computional tasks using disordered carbon nanotubes", The Royal Society meeting, THEO MURPHY INTERNATIONAL SCIENTIFIC MEETING, Heterotic computing: exploiting hybrid computational devices, November, 2013 (Poster presentation)

Acknowledgements

I would like to express my deepest gratitude to my supervisors Dr. Apostolos Kotsialos, Professor Dagou Zeze and Professor Michael Petty for their collegial guidance and mentorship throughout this project. They have been a constant source of inspiration, teaching and support from the first day I arrived at Durham. I strongly believe they have provided me with the best start of my research career. I am also grateful for their moral and emotional support along the way. My earnest thanks to Dr.Kotsialos, for his valuable advice, constructive criticism, positive appreciation and counsel throughout the course of the research which led to the successful completion of my PhD. Under his guidance, I successfully overcame many difficulties and learned a lot.

I would also like to thank Dr. Mark Kieran Massey, Dr. Chris Groves and Dr. Chris Pearson, who have assisted me with my PhD work.

A special gratitude to the research fund. The research is supported by European's Community Seventh Framework Programme (FP7/2007-2013) under the grant agreement No. 317662 (NAno Scale Engineeing for Novel COmputaion using Evolution - NASCENCE (http://www.nascence.eu)).

Finally, I would like to express my extra special gratitude to my parents who have provided me with moral and emotional support in my life. Without their prayers, love and support, I would not have got nearly as far as possible. I am also grateful to my husband, family members, my siblings and friends who have

supported me along the way.

And last but by no means least, also to everyone in the office, it was great sharing office with all of you during last four years.

Thank you for all your encouragement!

Contents

List of Figures

List of Tables

Abbreviations

ADC	Analogue-to-Digital. 44
CNTs	Carbon Nanotubes. 5, 7, 40, 184
DAC	Digital-to-Analogue. 44, 45
DE	Differential Evolution. 9, 62–65
EA	Evolutionary Algorithms. 8, 18, 21, 22
EHW	Evolvable Hardware. xi, 22, 24
EIM	Evolution-in-materio. xvi, 5, 6, 8–11, 29–31, 36, 53, 69, 71, 103, 106, 119, 120, 151, 152, 164, 165, 174, 177, 179, 182–185
FPGA	Field Programmable Gate arrays. 24, 25, 27, 52, 53
FPMA	Field Programmable Array Matter. 30, 53
GA	Genetic Algorithm. 8, 24, 25
GO	Graphene oxide. 33, 34

kNN	k-Nearest Neighbour. xv, 152–154, 157, 158, 160, 162–166, 171–177
LCAP	Liquid Crystal Analogue Processor. 28
NASCENCE	NAnoSCale Engineering for Novel Computation using Evolution. 32
NM	Nelder-Mead algorithm. 16, 181
NN	Nearest Neighbour. 153
PBMA	Poly Butyl Methacrylate. xix, xx, xxii, xxiii, 37, 108, 109, 113, 116, 117, 136, 144–149, 171, 176
PC	Personal Computer. 24
PCB	Printed Circuit Board. 44
PCE	Percentage classification error. xxiii, 164, 166, 168, 170, 171, 174, 175, 177
PMMA	Poly MethylMethacrylate. xii, xiii, xvi, xxii, 37, 38, 40–42, 69, 70, 72, 78, 100–105, 107, 119, 136, 140, 141, 149, 162
PSO	Particle Swarm Algorithm. 7–10, 17, 53, 58–61, 119, 120, 125, 126, 135, 137–140, 143, 161, 162, 167, 172, 174, 175, 181, 183
rGO	reduced graphene oxide. 34, 37

SPV	Shortest Position Value. 7, 61, 62, 125, 135, 137, 181
SWCNTs	Single Walled Carbon Nanotubes. xii, xiii, xix, xx, xxii, xxiii, 7, 8, 10, 31–33, 37, 38, 42, 66, 67, 69–72, 77, 78, 100–105, 107–110, 112, 113, 116–119, 135, 136, 140, 141, 144–150, 162, 171, 172, 174, 180–184

Chapter 1

Motivation

Computation is defined as any type of calculation that is performed using arithmetic or non-arithmetic steps and follows a well defined method. It is referred as a physical phenomenon that occurs within an enclosed physical system, called a computer [9]. For example, a physical system can be digital computer that can perform billions of functions in a second; or it can be a biological cell, which has information processing capabilities such as, self assembly, protein-protein interaction networks, gene regulation and biological transport. Where as, computing is the procedure of calculating and/or determining an outcome by mathematical or logical methods [9].

Modern digital computers are built using the classical computational techniques, which require an exact knowledge of what needs to be built and clear operational control of the system. These systems are built on the paradigm of Turing machine [10] and follow the Von Neumann architecture [11]. There is a clear distinction between the permanent structure (the hardware) and the set of instructions (the software) to run this kind of system. Their operation is sequential and the information within these machines is stored at special memory addresses and is retrieved using these memory addresses.

These systems face the challenges of being highly flexible and adaptable due to ever-changing demands placed on them. The traditional approach to building these systems brings in the inherent centralisation and inflexibility. The Von Neumann architecture requires these machines to be controlled by a central processing system and to this extent, these systems are essentially 'top-down' systems. This centralised approach is strongly contrasted with the complex systems found in nature.

The Natural systems such as, neural networks, eco-systems, immune systems, ant colonies finding the shortest path to food, bird swarms coordinating to find food are very few examples of naturally occurring biological systems that compute. These systems, as compared to Turing machines are highly complex and self-organised systems. The resilience of natural systems can be characterised by the following main factors [12]:

- Natural systems are autonomous, every entity within the system works independently of others and there is no central control for the direction or coordination.

- Each entity in natural systems has their own decision making capability. They can not only share their state but also have the ability to change the state of other elements in the system. Conventional computing processes (such as optimisation processes or scheduling processes), on the other hand, continuously share their state information with other entities in the system according to predefined rules.

- The collective behaviour of various entities can be very complex that was not present in individual entities.

- Each entity in the system constantly uses its adaptive mechanisms fine tune its behavioural attributes.

- The elements of natural systems can self organise. They can interact among themselves in order to achieve a desired outcome or a certain goal.

These characteristics can be encompassed in modern computers to allow flexibility and adaptability in their processes and scalability. The desire to achieve computational capabilities of natural systems has urged the researchers to study natural systems and take inspiration in the design of computer systems.

Unconventional computing or alternative computing is a field of computing that studies non-classical models of computation [13]. Example of unconventional computing paradigms include quantum computing, optical computing, molecular computing, and chemical computing [14]. It addresses the paradigm of natural computing and its various philosophical aspects [15]. These may encompass theoretical and philosophical views of unconventional computing (such as understanding of novel principles of computation in natural systems i.e. physical, chemical or biological systems)) and the differences between conventional and unconventional computing [16].

Natural computing also investigates the computing designed by humans that is inspired by nature such as computing that takes place in nature and the computation using natural materials [15]. It also focuses on building frameworks to model and interpret natural computing. Generally, the field of unconventional computing is divided into three main categories [16].

- Computation inspired by nature: It is concerned with the study of natural phenomenons and take inspiration from them in order to develop methods and algorithms to solve complex problems [17]. For example, computational models such as neural computation [18], DNA computation [19], Quantum computing [20], evolutionary computation [21], swarm intelligence [22], developmental and cellular systems [23], molecular computing [24], behaviour inspired computation [25] and amorphous computing [26] are all different forms of

computation that are inspired by the nature.

- Simulation and emulation of natural systems: It is basically a synthetic process to mimic life forms, processes or natural phenomenon to create patterns, forms or behaviours in order to have better understanding of natural systems [27].

- Computation using materials: It make uses of natural materials to perform computation that constitutes a novel paradigm that can supplement or replace conventional silicon based computing.

Unconventional computing is thus a field of study that is inspired by the novel principles of computation in natural systems, that are used to develop non-novel methods and algorithms and artificial computing architectures and to implement computation in nature inspired materials.

This field of study is predominantly occupied by theoretical research such as quantum computing , membrane computing, amorphous computing etc. Various philosophical, theoretical, logical and technical aspects have been described by some of predominant scientists in the field of unconventional computing. For example, Stepney states that unconventional computation is described in terms of physical laws of Newton and Lagrangian mechanics [28], [12]. Some new computational paradigms are presented by Copper [14], Martin emphasised on computation by mimicking nature [29], MacLennan argued that the power of analogue computation should be expressed in its own terms rather compared or in context of Church Turing thesis [30] and Burgin presented the idea of super recursive algorithms [31].

Despite the profound potential of unconventional computing, only handful of prototypes are available such as reaction-diffusion processor [32], hot ice computer [33], extended analogue computers [34], molecular computation [35], DNA based

computers [36]etc.

Due to an underlying need to find alternatives to the classical model of computation, unconventional computing has the potential to offer enormous capabilities. One recent field of study in unconventional computing is called EIM [37]. It implements computer controlled evolution to manipulate the properties of the materials in such a way that some useful computation can be extracted from them. The idea presented some powerful examples to prove that artificial evolution can be used to configure materials for computation, such as liquid crystals to perform simple computational tasks [38] as tone discrimination [39], logic gates [40] and robot controller [41]. The study proposes that computation can be extracted from the materials if highly specific signals are applied to it. EIM suggests that evolutionary algorithms can manipulate the highly complex properties of the physical systems that are either difficult to understand or can be manipulated by other means.

However, the study is in its very initial phase. It requires identification of suitable materials for computation, identification of signals that can manipulate material properties for purpose of computation and suitable methods to extract computation.

1.1 Research problem

The study presented in this thesis is based on the idea of EIM. It studies different Carbon Nanotubes (CNTs) based composites as well as Graphene-based materials for the purpose of extraction of computation using the principle of EIM. In general, it aims at answering the following questions:

- What are the suitable nano-material systems for extracting computation
- What kind of hardware and software setup is appropriate to achieve the desired outcomes

- What are the suitable tasks/methods for extracting computation from the nano material systems

- What are the suitable investigative methods for evolving the material? i.e. what kind of evolutionary/optimisation processes/algorithms can be used for different computational tasks? The computational tasks themselves are different and independent of the material used. Some materials and algorithms combinations may be most suitable for a computational task while some are suitable for others. This is a major part of this study that is investigated.

1.2 Hypothesis

Physical properties of nano-material systems can be altered by means of computer-controlled signals; these changes can then be used to perform some meaningful computation.

1.3 Outline

In this thesis, different carbon-based material are studied for their computational capabilities using the principal of EIM. Chapter 2, discusses the literature review in detail. In chapter 3, the detail description of experimental methods is presented. It also describes the materials that are used in experiments, the initial hardware/software platform, the purpose-built platform, called Meccobo (used in later stages of experiments) and the algorithms that are used to train the materials for specific computational problems. Chapter 4 outlines the optimisation and evolutionary algorithms that are used in experiments. This chapter also describes different computational problems that are implemented and presented in this thesis. Chapter 5, discusses the initial experiments with SWCNTs based materials to solve the

problem of logic gates with the initial hardware/software set-up. It also discusses the investigations with different algorithms and the effect of different concentration of SWCNTs on computational capabilities. It also reports on an investigation of evolving graphene based materials to behave as logic gate/circuit. These experiments shed light on what are the best concentrations of graphene that produce an ideal non linear response for logic gates.

Chapter 6 emphasises the relationship of polymers and SWCNTs and their effect on computational capabilities. The results showed that conductance is the main property that plays an important role to make the material behave as a logic gate.

Chapter 7 discusses the implementation of complex logic gate in CNTs base materials, using Mecobo. Particle swarm algorithm Particle Swarm Algorithm (PSO) with Shortest Position Value (SPV) rule is used to manipulate material's conductive property. This method allowed the flexibility to put the signal application terminals under evolutionary control. The successful implementation of the problem demonstrates that SWCNTs have the potential to solve other complex computational problems. It also highlights the suitability of use of Mecobo hardware for other computational problems and different kinds of materials.

Chapter 8 discusses the investigations of evolving the material as a tone discriminator. The problem is implemented using purpose built platform Mecobo and the Particle swarm algorithm using SPV rule. Square waves with varying input frequencies are used as input signals along with the static voltages as configuration signals to train the material as a tone discriminator. Different concentrations of SWCNTs were used in these experiments. The results of these experiments showed that all the considered concentrations of SWCNTs were successfully trained for this problem.

Chapter 9 demonstrates that SWCNTs/polymer based materials can be trained

as data classifiers. Various binary data classification and multiple class data classification problems are discussed and results are reported. Various data classification rules are studied for this problem and it has been found that k-nearest neighbour and its different variations are more suitable for data classification problems with these materials using PSO and EIM. The results of these investigations were very close to the results found in the literature for these data classification problems. The initial findings suggested the different areas of improvement for successful implementation of the classification problem. Based on the findings it can be suggested that the materials with more non-linear response and materials that can be reconfigurable can be considered for further investigations with classification problem. Also, its worth trying methods which can implement more flexibility and adaptability in order to exploit the physical properties of the material.

Finally, chapter 10 outlines the main conclusions drawn from this work, and suggests the possible directions of future investigations.

1.4 Original contribution

Evolution in materio is a new field of unconventional computing that uses Evolutionary Algorithms (EA) to train a bulk matter for special purpose computation. The principal of EIM has been implemented with Liquid Crystals and Genetic Algorithm (GA) to solve various problems of computing. However, the field of study is in its very initial phase. There are four main key areas as outlined in research problem that need to be explored in detail in order to successfully implement this study in future computational and electronic devices.

In order to find suitable novel materials for this study, this thesis focused on studying mainly different single walled carbon nanotube based materials and graphene based materials for different computational problem. It is outlined in in

this thesis that certain combinations of SWCNTs/polymers and some Graphene based materials can be used to solve simple computational problems such as Logic gates and tone discrimination

It is the first time that SWCNTs based materials were used to solve simple computational problems such as Logic gates (Chapter 5). Later, in this study, these material systems were found suitable for complex computational problems such as complex Logic circuits (Chapter 8) and machine learning problems (Chapter 9).

The Logic gate problem is further implemented to identify the effect of concentration of SWCNTs on solving a computational problem. It has been highlighted in Chapter 6, that there is a certain threshold of SWCNTs in a fixed polymer ratio which is more suitable for solving a Logic gate/circuit problem. This has been observed for other computational problems presented in this thesis, such as tone discrimination (Chapter 7) and data classification problems (Chapter 9).

A more flexible and versatile hardware platform is necessary for EIM. This fact is highlighted in Chapter 8, where Mecobo hardware platform has been found more suitable for the solution of complex Logic circuits. This hardware platform is used to further study other computational problems such as data classification problems (Chapter 9).

The three optimisation algorithms, namely, Nelder-mead optimisation algorithm, Differential evolution and Particle Swarm Optimisation algorithm are studied with different computational problems for their suitability for EIM. Nelder-Mead solved simple logic gates problem, but it was not successful with complex logic circuits as compare to Differential Evolution (DE) and PSO (Chapter 5) which solved some complex logic circuit problems. PSO has been successfully implemented for complex Logic circuits (Chapter 8), tone discrimination problem (Chapter 7) and data classification problems (Chapter 9).

To summarise, SWCNTs based materials are successfully used for the solution of different computational problems using optimisation and evolutionary algorithms following EIM. The successful use of these materials and evolutionary algorithms stands up as an important contribution to the field of unconventional computing, specially EIM.

Chapter 2

Literature review

The study presented in this thesis used computers running search algorithms (optimisation and evolutionary algorithms) to find the values of incident signals and other variables that should be applied to the materials in order to change their physical properties to carry out some meaningful computation. These signals can be static voltages and variables may dictate the location of these signals. This approach does not involve any top-down sequence to carry out a specific computation. Instead, these materials are programmed by the computer controlled evolution [42] that utilises the physical properties of the material to perform computation which a human programmer is not aware of.

The literature review presented in this thesis discusses the use of materials in unconventional computing, optimisation and evolutionary algorithms, evolvable hardware, evolvable motherboard that lead to the concept of EIM. It also includes a brief historical background of carbon nanotubes and graphene that are used in the experiments described in this thesis.

2.1 Materials for computation

Natural systems [43] such as living organisms, are parallel architectures that are linked with the environment through interaction [44]. They can adapt to the changes in environment by making full use of system and environment interaction. The interaction is a result of exploitation of physical properties of the materials they are composed of and by taking full advantage of physical and information processes of the system [45]. Such systems, if programmed, can perform computations with great speed and efficiency and with remarkably reduced computational time. For example, an analogue physical system has demonstrated that partial differential equations can be solved with much faster computational capability as compared to traditional machines [46].

It is argued that one kilogram of matter can have a capacity of an ultimate laptop which can perform 5.4258×10^{58} computations per second and can have a storage capacity of 10^{31} bits [47].

Despite the enormous potential the material systems can offer, it is still unclear what physical material systems are most suitable for the extraction of computation and how they can be used for computational purposes. There is lot of research going on to find suitable materials for computing and few categories of materials that are used in unconventional computing are listed below [48].

- *Biological materials*

  There is an active research by the name of *'wet'* unconventional computing [49] which uses biological material substrates for the purpose of computation. Few examples include the use of slime moulds [50], bacterial consortia [51], [52] and biological cells such as neurons [53] [54].

  The research using biological materials for computation is productive but it does not provides a deep insight into computation performed by these

material systems. This is due to the fact that biological systems have achieved their complex properties by undergoing the process of evolution over billions of years. Moreover, these substrates are not suitable for digital and classical computational problems due to their analogue nature. For these reasons, it is difficult to describe or model the computation performed by them and in turn we are provided with no information on how their properties can be exploited for computation. Therefore, it has been argued that before considering natural material systems for computation, non-biological materials should be considered [48].

Hence, more simpler, laboratory synthesised materials can be more suitable candidates for computation. The chemical and physical properties of these materials can be explored for computational purposes [48].

- *Non-biological materials*

  Early research using physical materials for computation goes back to reaction-diffusion systems [55]. Reaction-diffusion systems are introduced as a morphological mechanism that may arise from homogeneous state of chemicals [56]. These are now generally, considered as material computational systems [57]. Reaction-diffusion systems are chemical systems in which chemicals react with each otherand transform into each other and/or they get distributed at different rates through the system. These processes result in the waves or wave like phenomenon and other self organising patterns [57]. For example, these systems are used to demonstrate the implementation of logic gates [58], the solution of a 2D Voronoi diagram problem [59], a reaction-diffusion robot controller [32].

  In another example, an unconventional super-computer has been proposed by using variety of materials substrates such as conductive plastic sheets

and gelatin doped with table salt to implement Rubel's extended analogue computer [60].

- *Novel materials*

  The notion of 'edge of chaos' [61] provides an indicative direction in the search of novel substrates for unconventional computing. This idea suggests that physical systems have most computational power at the edge of the transitions, for example, transition of matter from solid state to liquid state. A proposal that enhance this idea, suggested that physical materials which are configurable, robust and are provided with properties that effect the applied signals, could be suitable for computation [62]. By configurable means it should configure over many iterations to give the desired response. The material would be able to reset to its previous state other wise it will retain its memory and may use the stored information in it for future computations, which is undesirable.

  The use of smart material for a robotic arm is another example of using novel substrates for computational purposes [63]. An alloy called Nitinol (made of Nickel and Titanium) is used as a muscle for robotic arm. The alloy is super elastic and has shape memory. It has different morphologies at different temperature. The implementation used an evolutionary algorithm that controlled the activation of wires to produce a snake like movement. If the wire breaks in the middle of experiment, evolutionary algorithm can activate new sequences in the wire to compensate the damage.

  Another example used switchable glass [64], which is also referred to as 'smart windows' to control the amount of light through the windows when an electric field is applied. The study used three existing technologies namely: electro-chromic devices, suspended particle devices and liquid crystal devices

to vary the amount of light with varying voltages with the help of evolvable hardware.

A number of materials have been identified to have a potential to be exploited and configured for computational capabilities, such as liquid crystals, Langmuir-Blodgett films, and materials based on nano-particles [62]. It has already been demonstrated that a chip made form liquid crystal can act as a tone discriminator [39] and a robot controller [41]. Liquid crystal has a structure of both a solid and liquid. It can flow like a liquid as well as has a structure in which its molecules can be oriented in crystal form.

The study presented in thesis also used novel materials substrates such as Carbon nanotubes dissolved in polymer and reduced graphene oxide composites for the purpose of unconventional computing. Such material system have never been explored earlier for their computational capabilities. These material systems have been chosen after studying their electrical properties, because a material with a non-linear response to the signal applied can be suitable for the extraction of computation [48]. The properties of material systems are exploited using optimisation algorithms and evolutionary techniques for the purpose of computation extraction. The following section provides a detail description of these techniques.

2.2 Optimisation algorithms

An optimisation algorithm involves a process that is executed iteratively to compare different solutions within a specific search space until an optimum or satisfactory solution is obtained. Optimisation algorithms are divided into two categories: gradient-based methods and population based methods. A gradient-based method finds the solution of the problems of the form:

$$\underset{x \in \mathbb{R}}{\text{minimize}}\ f_0(x) \tag{2.2.1}$$

where the search directions are defined by the gradient of the function at the current point. For example, gradient descent [65] and the conjugate gradient [66] are examples of such methods.

Gradient-based methods include direct search methods [67] [68], which minimise the function $f(x)$

$$\begin{aligned} &\underset{x\in\mathbb{R}}{\text{minimize}}\ f(x) \\ &\mathbb{R} \rightarrow \mathbb{R}^n \end{aligned} \tag{2.2.2}$$

but there is no information present about the gradient of the function $f(x)$. The population based methods include genetic algorithms [69] and simulated annealing [70] [71]. Gradient-based methods are helpful in finding optimal solution of continuous problems (where small change in input results in small change in output) and differentiable functions (a differentiable function for a variable is a function whose derivative exits at every point in domain). Due to the analogue nature of the materials and the fact that the applied signals are discrete in nature, direct search methods are first chosen in initial experiments. These are straightforward to implement and do not use derivative information of an objective function but only its values for a guided search process. For the current study, a well-known direct search algorithm known as Nelder-Mead algorithm (NM) [72] has been used as an optimisation tool. The algorithm, also known as the Downhill simplex optimisation method, minimises an objective function in a multi-dimensional space [73]. The detailed overview of the algorithm can be found in [73], [74], [72], [75]. The experiments with NM demonstrated that it was not successful to solve complex problems, hence, after some initial experiments with NM, evolutionary algorithms were implemented to extract the computation from the materials. A brief background of these methods is described below.

2.3 Evolutionary computation

Evolutionary computation is a family of algorithms that are inspired by nature and are used for global optimisation. It is also a subset of the field of artificial intelligence that studies these algorithms. These algorithms are population based optimisation or meta-heuristic methods [76].

The history of evolutionary computing goes back to late 1940s, when Turing proposed 'genetic and evolutionary search' [77]. In 1960's different techniques by the names of evolutionary programming [78], genetic algorithms [79] and evolutionary strategy [80] were introduced. Whilst, structurally different these techniques are inspired by the nature and particularly used the idea of evolution to solve the problems. These fields are now categorised under a general category of 'evolutionary computing'.

The main paradigms of evolutionary computing are, evolutionary programming, genetic algorithms, evolutionary strategies, classifier systems and related fields such as swarm intelligence (Ant Colony Optimization and Particle Swarm Optimization). Many other hybrid systems have been introduced that implement various features of evolutionary computing with other methods [81]. For example use of harmony search [82] with PSO [83].

These methods are widely used in the design and implementation of unconventional computing systems. It is argued that evolution has optimised the processes in nature, therefore the adoption of evolutionary paradigm can lead to finding optimal solutions to many computational problems [76]. The nature of computational problem requires search through many possibilities and sometimes even the search space is quite large. This requires a parallel approach to solve these problems. Also, these problems require algorithms that are adaptive in nature. Evolution on the other hand is a parallel process that can produce solutions with the changing environment and evolutionary algorithms can provide highly optimised solutions

to these problems in different problem settings.

The evolutionary computing works by generating initial set of candidate solutions. The process is iteratively repeated and each new generation of solution is the outcome of removal of less desired solutions and some random changes. In biological terms, it can be stated that population of solutions is a result of natural selection and mutation. As a result of this process the population gradually increase in fitness and in case of particular problem it improves according to the fitness function of the algorithm.

2.4 Evolutionary algorithms

Evolutionary algorithms are the subset of evolutionary computing that model biological mechanisms such as reproduction, mutation, recombination, natural selection and survival of the fittest [84]. EA start with the population of candidate solutions and using the principle of 'survival of the fittest' find better and better solutions to the problem. Candidate solutions to the optimization problem play the role of individuals in a population, and the fitness function determines the environment within which the solutions will 'thrive'.

EAs starts with the random individuals and the objective function is evaluated for these individuals. This results in the first generation of individuals. If the optimisation criteria is not met the next generation of individuals is produced. Just like optimisation algorithms, an EA selects new sets of individuals at every generation that have been evolved according to their level of fitness in the problem domain. The evolution takes place after the repeated process of 'selection', 'recombination' and 'mutation'. These new individuals also referred as offspring and are inserted into population replacing the previous individuals (parents). This cycle continues until the termination criteria is met.

Figure 2.1 shows the structure of a simple evolutionary algorithm. During this

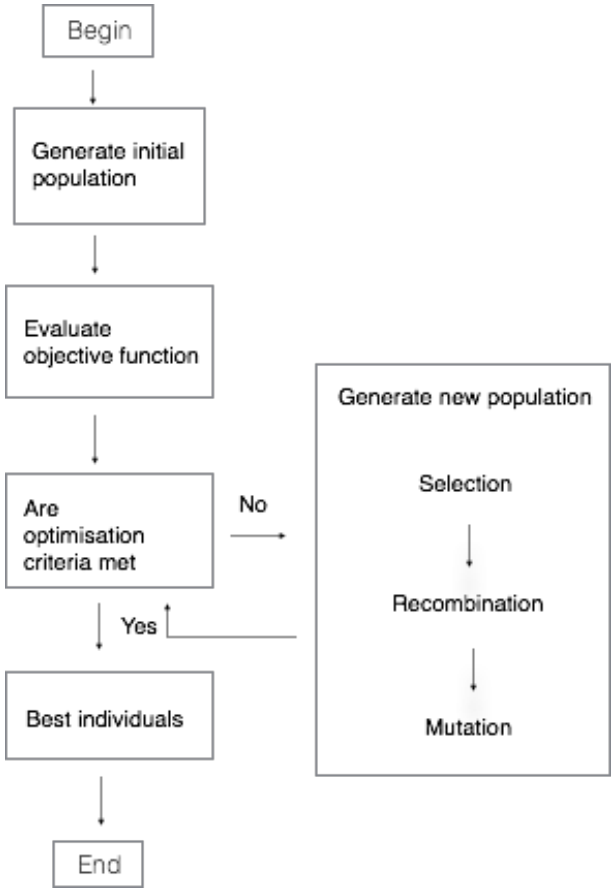


Figure 2.1: The general structure of an evolutionary algorithm

process, the two main forces form the basis of an evolutionary process:

- Recombination and mutation: These processes create the necessary diversity among individuals and facilitate novelty; while
- Selection: serves the purpose for bringing in quality.

Many aspects of evolutionary computing are stochastic. The information exchange during recombination and mutation is random. While the selection operators can be either deterministic, or stochastic. During the selection process

those individuals which have a higher fitness score have a higher chance to be selected than individuals with lower fitness score, but sometimes the weak individuals can have a chance to become a parent or to survive.

Following sections list some main parts of Evolutionary algorithms [85] [86] .

1. **Population:** Population is the set of solutions that are chosen to be evolved during an EA run. Generally, the initial population is selected randomly, but sometimes population is initialised manually in order to have a good starting point at the start of an evolutionary run.

2. **Fitness function:** Fitness-based selection improves the quality of an EA. Hence, the design of fitness function (or evaluation function)is crucial. Any available and usable knowledge about the problem domain should be used when designing a fitness function. This is due to the fact that it represents 99% of the total computational cost of evolution in most real-world problems and mostly it is the only information about the problem in the algorithm.

3. **Selection:** Selection chooses the individuals that can be recombined to produce new offspring. It also selects how many offspring each individual will produce. During the process each individual in selection pool receives the production probability with respect to its own objective function values and the objective function value of other individuals in the pool. the next step is to select the parents according their their fitness by means of different algorithms, such as roulette-wheel selection, stochastic universal sampling and local selection etc.

4. **Recombination:** Recombination produces new offspring by combining the information present in the parents. The process is done by combining variable values of the parents. Different methods of recombination are used according to representation of variables. For example, discrete recombination

can be use to recombine all types of variables. Real value recombinations are applied to real valued variables, where as, binary values are recombined using binary valued recombinations. Binary value recombination is also called 'crossover' and are typically used in genetic algorithms.

5. **Mutation:** Stochastic transformations of an individual is referred as mutation. The trade off of exploration and exploitation should be maintained during mutation. Some time large mutations are necessary as it introduces genetic diversity but it can lead the algorithm to a random walk, hence an offspring close to parent must be generated. For this reason the individual variables are mutated by small perturbations (size of mutation step) and low probability (e.g., for binary variables flip one bit of a bit string; and, in case of real variables, add zero-mean Gaussian noise with carefully tuned standard deviation)

6. **Reinsertion:** After the reproduction of a new offspring they are inserted in to population. This is especially important if less offspring reproduce than necessary, not all offspring are to be use at each generation or if more offspring are reproduced than necessary. Thus a reinsertion scheme is placed to determine which offspring will be included in the new population. For example the local insertion scheme for local insertion and global insertion scheme for global insertion and all other methods.

From above discussion it can be concluded that EA are substantially different from traditional search and optimisation methods and offer various advantages. Some of them are listed below:

- EA perform parallel search, not the single point search
- No derivative information is required, and only objective function and the fitness value is needed that will affect future searches.

- EA use probabilistic transition rules instead of deterministic rules.
- Several parameters of an objective function can be simultaneously optimised
- They are well suited for the problems that are difficult to describe mathematically.
- They are well suited when it is difficult to understand how to approach the problem.

Evolutionary algorithms are now used to solve multi-dimensional problems more efficiently than software produced by human designers, and also to optimise the design of systems [85].

2.5 Evolvable Hardware

Evolutionary algorithms have found extensive acceptance for design and implementation of wide range of unconventional computing systems. One such stream of implementation is called evolvable hardware [87]. In its most fundamental form an Evolvable Hardware (EHW) is a circuit that can improve its performance by changing its architecture and behaviour according to the changes in environment [88].

The history of EHW can be traced back to 1990's. The concept of 'evolvable hardware' was pioneered in 1993 [89] and [88]. These workers envisioned that the process of circuit design should rely on natural evolution. One of the first experiments involving 'evolvable hardware' was conducted in 1996. This experiment deployed computer controlled manipulation of a material's properties using artificial evolution [90].

The fundamental steps of working of an EHW are highlighted in Figure 2.2. First, the randomly generated population of circuits is evaluated against a fitness function. If the desired solution is not found a new generation is generated by applying the genetic operators (e.g. mutation crossover). The cycle continues

until the desired fitness value is achieved. The two terminologies are commonly used for classifying the operation of evolvable hardware [91].

- **Extrinsic evolution [92]** In this technique simulation is used to evolve the circuit. The best solution found by simulation is implemented within the hardware.

- **Intrinsic evolution [93]** The technique directly implements and evaluates the candidate solution in the hardware. The evolutionary algorithm uses the evaluations received from the hardware and generates new solution if fitness criterion is not met. This process is iterated until the desired solution is found or termination criteria is met.

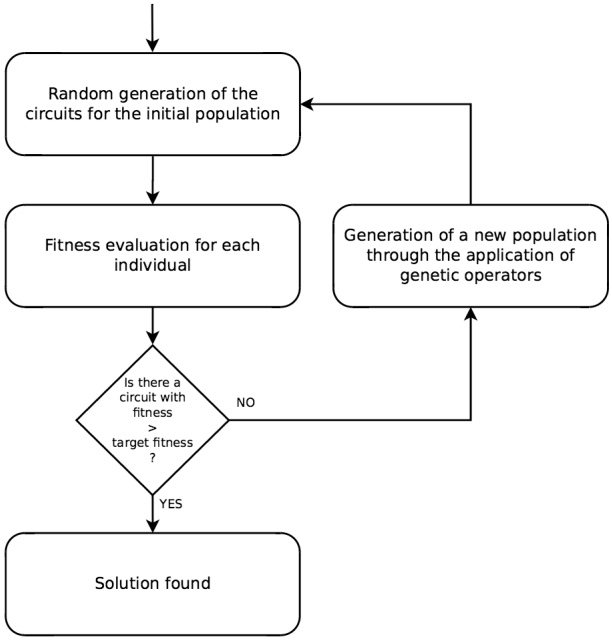


Figure 2.2: A general flow chart of an evolvable hardware system [4]

Moreover, the operation of evolvable hardware can be classified in to categories based at which level the evolution is performed. The complexity of basic building

blocks (also called granularity) of evolvable hardware can be divided into

- **Gate level:** The building blocks of evolvable hardware correspond to logic gates such as, AND, OR, NOT etc. For gate level evolution evolvable hardware utilises the pattern recognition system where the system can recognize noisy binary input patterns [94]. Another example of gate level evolution is an EHW comparator used in an industrial welding robot [94].

- **Function level:** This is a further level of abstraction of gate level evolution where a more complex function is implemented using the basic building blocks. The example of function level evolution is the adaptive equalization in digital mobile communication and data compression [95].

The first experiments that demonstrated the possibilities offered by evolvable hardware were conducted by Adrian Thompson in 1996 [96]. Thompson used a special chip called Field Programmable Gate arrays (FPGA) to evolve electronic circuits [96]. This chip itself had no specific built-in function but can be programmed by the user [97], [98]. The chip is a grid of 64 x 64 simple logic cells, each of which can perform basic boolean operations such as AND, OR and NOT. Different configurations can be applied to FPGAs to form electronic circuits.

Thompson used 10×10 grid of the FPGA using GA to design an electronic circuit. An initial population of 50 individuals called, 'genotypes' was created by the GA. In terms of theFPGA, the genotype is the 1800 bit-string that encodes the chip's configuration. The inputs were 10 frequencies of $1kHz$ and $100KHz$ each in 500 milliseconds. The fitness of each genotype was evaluated by constructing a circuit that can produce an output of 1 volt for one frequency and 0 volt for another. The next step was to produce new 'offspring' by using selection, recombination and mutation. The GA running on a Personal Computer (PC) generated the genotypes that were transferred to the chip as soon as they were

generated. After almost 5000 iteration the GA was able to find perfect genotype that can construct a circuit that can discriminate between $1kHz$ and $10kHz$ tones. The combination of FPGA and GA produced an electronic circuit that occupied less circuitry and in far less time than could be achieved manually.

In other set of experiments, a robot controller was evolved using the same system [99]. Some other works for evolving Robot controllers using FPGA are [100], [101], [102].

Later, FPGAs were used for evolving oscillators [103]. These studies identified that the configuration settings applied to one area of the chip have not produced the same results when applied to the other area of the chip. Furthermore, the results were temperature sensitive [103]. In later experiments, circuits were evolved under different physical conditions [104], [105]and it was revealed that the evolved circuits were robust to most but not all environmental conditions. To overcome these limitations a new kind of evolvable hardware was proposed and it was named as evolvable motherboard. The following section briefly describe Evolvable motherboard.

2.5.1 Evolvable motherboard

The elements of FPGAs can be configured virtually to allow various different types of circuits. However, evolving circuits with FPGAs did not give the flexibility to test the elements of the final circuits as the test equipment only cover very small areas [103].

Furthermore, the components of FPGAs allow no control over them as they are predefined by the manufacturers and follow a conventional modular circuit design. This restriction has not allowed the evolution to fully exploit the FPGA circuit as it may have exploited different circuitry in an arbitrary way. It is also not clear which type of building block of FPGA (the configurable logic blocks (CLBs) or

its analogue equivalent analogue) is most suitable for hardware evolution.

To overcome these drawbacks another platform was designed to perform intrinsic evolution [106] and was named as an 'evolvable motherboard'. The success of these experiments was demonstrated by evolving a 'NOT' gate.

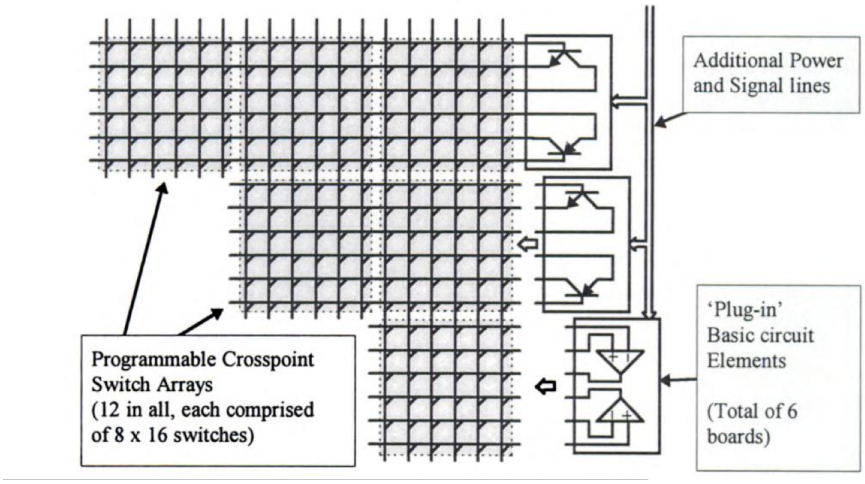


Figure 2.3: The simplified representation of an evolvable motherboard

Figure 2.3 represents a simplified evolvable motherboard. In its simplest form an evolvable motherboard is a matrix of analogue switches that can be connected to 6 daughter-boards that contain all the basic components necessary for evolution. All the basic component of electronic circuitry (from transistors to capacitors, logic functions etc) were present to allow maximum exploitation by evolution. The switches can be controlled by programming and are connected to computer via interface card plugged in to host PC's ISA bus. This allowed the genotypes to be instantiated in hardware in very short time ($< 1ms$).

Each daughter-board has eight lines on a switch matrix in addition to eight connections for various input/output paths. The switch matrix is designed with 1500 switches, to allow a search space of 10^{420} possible circuits.

The motherboard allowed any point of the circuit to be accessible for the measurement. This allowed an easy access to see which properties of FPGA are

used during the intrinsic evolution. In order to test the capabilities of the evolvable motherboard and investigate its potential to use as a tool for Intrinsic hardware evolution, a NOT gate was evolved using a bipolar transistor as an evolutionary building block and a genetic algorithm. These were the first circuits that were evolved intrinsically at transistor level.

Similarly, Thomson developed a new evolutionary circuit known as '*evolvatron*' to overcome the limitations of FPGA and the external conditions [107]. These studies have helped to figure out that during intrinsic evolution the physical properties of the system are utilised which is why the circuit evolved in one part of chip cannot be implemented in other parts.

In some other experiments, a much simpler version of evolvable mother board was implemented by use of a low cost micro processor and a rectangular bus approach [108]. The microprocessor bridged the gap between the computer and configurable units and lowered the cost. The bus approach made the safety and validation easier and simpler and an on-board signal generator was able to generate square, sine and saw tooth signal waves. This system successfully implemented a potential divider and full wave rectifiers.

In 2002, Miller and Dowing presented the idea of *'evolution in materio'* [37]. It was proposed that the liquid crystal can be a suitable material for the extraction of computation from the materials. A similar platform used by Layzell [106] was utilised to extract the computation in material. The setup was similar to Thompson [104], [105] and used a layered technique for parallel evolution. The platform included four cross-switch matrix devices. Each switch has 64 connections that can be connected to 8 external connections. The external connections can be input signals, ground voltage and/or can be a connection to other external devices. The external connections are connected to an input/output card of the computer, where the connections are dedicated to incident signal, configuration

signals and the measurement signals. The value of configuration signals are provided by the genetic algorithm and remain constant during an evolutionary run. The device was named as Liquid Crystal Analogue Processor (LCAP). The schematic diagram of cross-switch matrix is presented in Figure 2.4.

A detail description of the concept and use of evolvable motherboard for material computation is given in the following section 2.6.

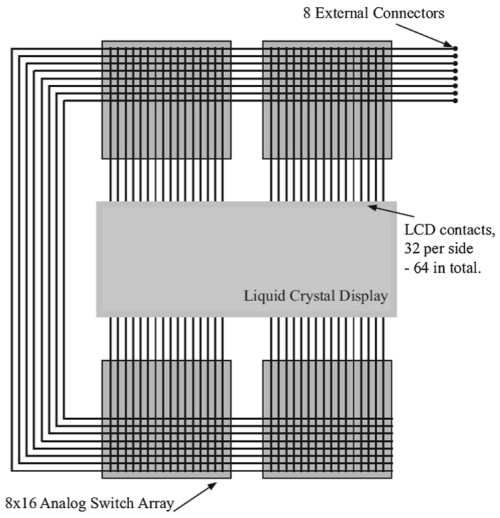


Figure 2.4: Schematic of LCAP [5]. The part of genotype decides which external connectors can act as input, output or configuration. The magnitude of these voltages is also provided by the genotype received form computer.

2.6 Evolution in Materio

New possibilities were pointed out by Adrian Thomson in his work of using intrinsic evolution in hardware [104], [105]. This approach was applied to a broader set of materials using the concept of 'evolution in materio' [109], [54]. The idea states that the computer controlled evolution can be used to manipulate the physical properties of different materials with an aim to extract computation

from them without having a detail understanding of these properties. The idea was demonstrated by using liquid crystals as substrates and genetic algorithm as a mechanism of exploiting the material's physical properties. The conceptual

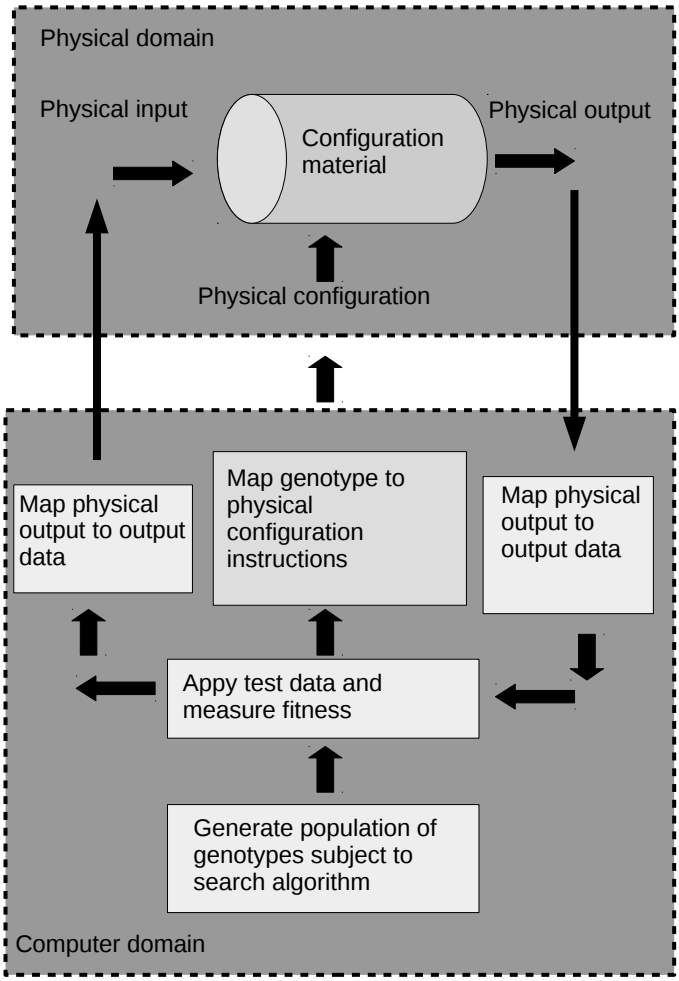


Figure 2.5: Conceptual overview of EIM [5]

overview of EIM is shown in Figure 2.5. EIM is a hybrid system, which consist of a digital computer and the analogue material system [37] [5]. The physical signals are applied onto the material system and the response is gathered from it. An

evolutionary algorithm running on the computer generates the candidate solution. The digital data in the candidate solution is translated into analogue signals and applied on the material as input or configuration signals. Similarly, the analogue response gathered from the material is translated to digital information and passed on to evolutionary algorithm for evaluation and a fitness score is assigned to it. The cycle continues until the termination criteria is met.

In order to identify the suitable materials for computation, Miller [62] introduced the idea of Field Programmable Array Matter (FPMA). It states that highly specific signals (e.g. voltages) applied to the material bring physical changes in it. These changes can exploit the material in number of ways. The suggested characteristic of the materials which can act as substrate to FPMA is their ability to configure when a voltage is applied. Also, they should return to their original state when the signal is removed. Initially, conductive polymers and conductive liquid crystals were pointed out as suitable candidates. The idea was demonstrated by its application to liquid crystals [110], [109], [52]. Later in 2004, it was demonstrated that if physical properties of liquid crystals are exploited by means of unconstrained search of evolution, the liquid crystals can act as tone discriminators [111]. In 2005, the results of further studies showed that it is possible to evolve logic gates [52] and a robot controller [109] in liquid crystals. However, while working with the liquid crystals, it was found out that their solutions were highly unstable. The response of the system varied and the results obtained could not be used for practical purposes [111]. This lead to exploration of other materials.

The idea of EIM brings back the analogue computing and argues that by using the idea of computer controlled evolution, new devices can be build that are impossible to imagine otherwise. The work presented in this thesis is based on the principal of EIM, and explores different SWCNTs composites. Some different

concentrations of reduced graphene oxide (RGO) in polymer are also used in initial experiments for their computational capabilities using computer controlled evolution.

2.7 Recent work using EIM

This section highlights the work that has been carried out using EIM while this thesis was being written. Various SWCNT material samples have been used to solve different computational mainly using Genetic Algorithms on Mecobo 3.0 platform. For instance, it has been demonstrated that simple logic gates (AND, OR, XOR) can be evolved in a SWCNTs/PMMA based material sample using square wave and static voltage signals [2].

The Bin packing problems have been studied using EIM, Mecobo 3.0 platform and Genetic Algorithms. The results of this experiment were implemented with various benchmark problems and it showed that the EIM technique can produce results which are closer to global optima and are competitive with state-of-the-art bin packing problem [112]. The same approach has been used to successfully solve the Function optimisation problem [113], Frequency classification problem [114], Machine Learning classification problems [115], even parity problem [116] and to evolve a robot controller [117].

In an another approach the travelling salesman problem has been solved for 9, 10 and 11 cities [118]. The experiment was implemented with 0.1% SWCNT/PMMA. However, instead of Genetic Algorithms and Mecobo platform, a (1 + 4)-ES algorithm was used to find suitable configurations to evolve the material to solve the problem and a data acquisition card was used as a hardware platform for the EIM. This allowed the 4×4 electrode array to be connected to a 16×16 analogue crosspoint switch to control connections from data acquisition card to the

material. This allowed configuration inputs to be placed anywhere on the electrode array. The same setup and methodology was used to solve Machine Learning classification problems. Although this setup has not yielded any promising results but it highlighted to further study and understand the complex relationship between the material and the voltage signals applied on it [119].

The choice of input parameters can influence the search space in EIM and can be critical for configuring the material samples to solve a computational task. In order to understand which input parameters are well suited to exploit the underlying properties of chosen material, a study was focused on using square wave signals with SWCNTs based materials. The experiment provided common measurements such as power spectrum and phase plots which were taken using Mecobo platform. These experiments also linked the frequency of square waves to the comparability of outputs from the material [120].

In an another experimental approach three different signal representations have been used to solve the graph colouring problem. The results showed that the static voltages, square wave signals ad mixed signals (involving static voltages and square waves were capable of producing a working device. However it was found that square wave signals always produced the best results as compare to other signal representations [121].

In the very recent experiments, Carbon nanotubes and Liquid Crystal based materials have been used to solve Binary classification problems. The experiment used Particle Swarm Optimisation algorithm and Differential Evolution algorithm to solve the problem. The experiment provided satisfactory results for the Binary classification problem and highlighted that Differential Evolution algorithm produced better results than the Particle Swarm Optimisation algorithm [122], [123], [124]. The study presented in this thesis uses the concept of EIM used different Carbon nanotube based materials (e.g with different polymers and Graphene) to study

various computational problems of varying difficulties. As mentioned earlier this thesis implement EIM in three dimensions i.e different SWCNTs based materials, different optimisation and evolutionary algorithms and different computational problems.

2.8 Carbon based materials

2.8.1 Carbon nanotubes

Carbon nanotubes [125] are allotrope of carbon, having a cylindrical nano-structure formed by an atom thick sheet of graphene. Due to their unique electronic properties [126], [127], researchers have high hopes with CNT-based materials in a way that they could be an exciting alternative to the present semiconductor industry [128]. CNT-based materials have been used as energy efficient transistors [129], [130] and logic gates [131]. Recently a CNT-based computer has been invented that has its own operating system and can perform integer sorting and counting simultaneously [132].

CNTs are capable of carrying an electrical current density, which is 1000 times higher than the copper. Depending on the process of their fabrication they are divided in to two categories [133]:

- Single-Walled Carbon Nanotubes (SWCNTs)
- Multi-Walled Carbon Nanotubes (MWCNTs)

A SWCNT is viewed as a rolled up sheet of graphene (i.e. two dimensional graphite plane) like a cylinder having a diameter of approximately $1nm$ and length greater than $1\mu m$. They can be metallic or semi conducting by changing their diameter and helicity only [134], [135]. It has been experimentally verified that their electronic properties change without changing in their internal bonding [136],

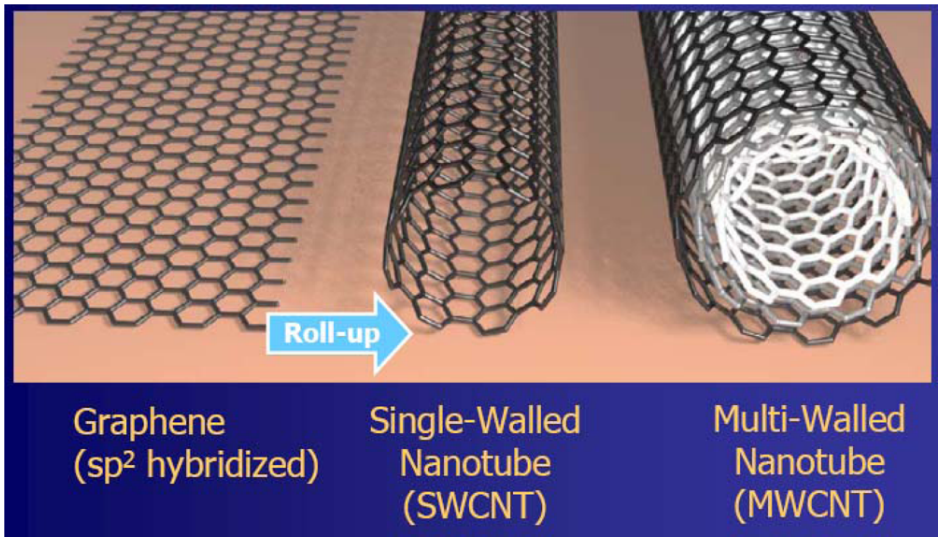


Figure 2.6: Conceptual model of SWCNTs and MWCNTs obtained form graphene sheets (courtesy of K. Banerjee/California University, Santa Barbara). [6]

[127]. The roll-up direction (geometric configuration) of SWCNTs changes their electronic properties [137].

The geometric structure of SWCNT can be identified by a circumferential vector (chiral vector or also known as a "wrapping vector") of two integer components (a, b). A nanotube can change from metallic to semiconducting if a or b changes only by 1. If $a = b$ the tubes are metallic; for $a - b = 3c$ tubes are semi-metallic, where c is a non-zero integer; and for all others $a - b = 3c \pm 1$ the tubes are semiconducting.

In order to extract computation from any material, the material should possess a complex and non-linear response to external stimuli [54]. SWCNTs form a complex conductive network and, owing to their unique electrical and conductive properties [138], they have been chosen as substrates for computation extraction. The substrates were provided by the NAnoSCale Engineering for Novel Computation using Evolution (NASCENCE) [42] project team and the experiments in this thesis demonstrate the extraction of computation from SWCNTs based materials and

effect of concentrations of SWCNTs and polymers with regards to suitability for extraction of computation using different optimisation and evolutionary algorithms.

2.8.2 Carbon nanotubes and polymers

Carbon nanotubes posses high electrically conductive properties. The exceptional properties of these nano-structures can only be exploited if they are homogeneously embedded into light-weight matrices such as polymers [139]. The presence of SWCNTs in polymers not only improves their mechanical properties but also their electrical properties [140]. The concentration of SWCNTs in the polymer as well as the type of polymer effect SWCNTs electrical properties [141]. The experiments in this thesis study the effect of different concentration of SWCNTs in the polymer on their computational capabilities. It also reports the relation between SWCNTs in different polymers and their computational capabilities.

2.8.3 Reduced graphene oxide

Graphene is a single layer of carbon atoms which are so^2 bonded and form a hexagonal 2-dimensional lattice [7]. It is the building block of $0-$dimensional buckyballs, $1-$dimensional Carbon nanotubes, or $3-$dimensional stack sheets of graphite as shown in Figure 2.7 [7]. The single layer of graphene possess exceptional chemical, thermal, optimal and electrical properties which makes it idea material to study for various purposes [7].

Graphite oxide [142] is a compound made up of carbon, hydrogen and oxygen molecules and is commonly used material to extract Graphene based materials. Graphite oxide is oxidised to produce mass quantities of graphene. Graphene oxide (GO) is a by-product of this oxidisation. The completely oxidised compound can then be dispersed in a base solution such as water, and Graphene oxide (GO) is then produced. GO is a graphene-like nano-sheet and is normally defective

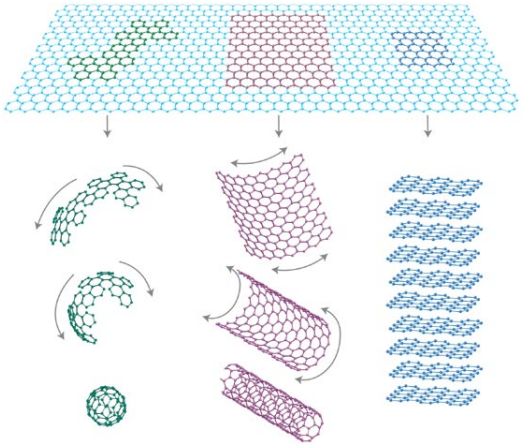


Figure 2.7: Graphene as a building block of Bucky balls, carbon nanotubes, graphite (from left to right) [7]

and requires some additional treatments to reduce it to reduced graphene oxide (rGO) [143]. rGO can be obtained by reducing Graphene Oxide (GO) chemically, thermally or via irradiation (UV or IR) to get a powder form. RGO is also known as chemically modified graphene, chemically converted graphene, or reduced graphene [144]. The process of conversion of graphite to graphene oxide to reduced graphene oxide is shown in Figure 2.8. rGO has great electrical conductivity properties that make it suitable for various research studies, hence used in experiments described in this thesis.

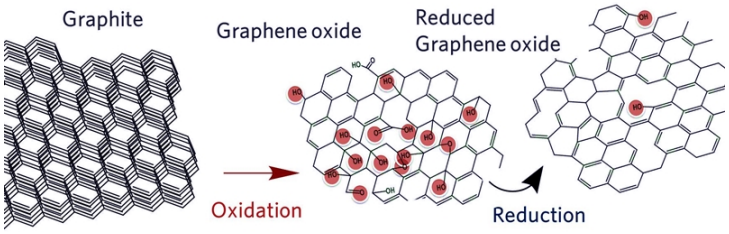


Figure 2.8: Reduced graphene oxide [8]

2.9 Conclusions

The study in this thesis is inspired by the recent research in exploiting material for computational purposes like liquid crystals, computer controlled manipulation techniques and evolutionary computing (*'evolution in materio'*) [110], [145]. Evolutionary computing techniques can exploit the properties of the materials in order to train them to perform a computation. This study focuses on nano-material systems such as CNT/PBMA-based materials, as candidate materials for extraction of computation and the use of different computer controlled optimisation techniques. The detailed description of materials that are used in the study presented in this thesis are describe in next chapter.

Chapter 3

Experimental methods

This chapter describes in detail the preparation of SWCNTs and Graphene composites with two different polymers. It also outlines the experimental hardware platforms that are used with different material composites to implement the concept of EIM. The experiments described in following the chapters used these experimental methods. The material composites and the hardware platforms are designed and produced within the NASCENCE project [42].

3.1 Single walled carbon nanotube composites

In order to identify suitable materials systems for extraction of computation using EIM methodology, several characteristics have been reported for a suitable material [54]. These are:

- The material should have a complex, configurable structure.
- It should responds instantly and consistently to a wide range of signals (e.g. voltages).
- It should have an ability to return into an un-configured or random state

when the signal (e.g voltage) is removed.

- The material should be robust to changes in the environment (light, heat, electromagnetic signals etc).
- It should be consistent between the samples.

Considering these points various carbon nanotube based materials have been prepared with a variety of formulations. Initially, electrical signals were intended to be applied on the substrates, hence the composites were supplied after studying their current- voltage ($I - V$) characteristics [135], in order to observe the non linear response. The experiments presented in this thesis used varying concentrations of SWCNTs with two different polymers : PMMA and PBMA. Later, three different concentrations of reduced graphene oxide rGO in polymer PMMA were used for the purpose of identification and extraction of computation. The following sections describe these material systems in terms of their electrical properties and methods of preparation.

3.1.1 SWCNTs / PMMA [1]

The first batch of materials provided by the project team consist of SWCNTs/PMMA based composites. The initial experiments described in chapter 5 were performed with the composites of SWCNTs and PMMA. The formulations were prepared by following the same methods in [146] and [147]. The SWCNTs were dispersed in Anisole (VWR, analytical reagent grade (methoxybenzene)) with an aid of an ultrasonic probe at a power of 20% (Cole-Parmer 750W ultrasonic homogeniser). The PMMA (Aldrich, $M_w = 93;000$) was then added and additional sonification was performed for a more uniform dispersion. The material was then deposited on chromium/gold micro-electrode arrays (section 3.2, Figure 3.7) by drop casting. In order to promote quick drying and a more uniform coverage, the substrate

was heated to $100°C$ and left for 30 minutes in order to dry any remaining solvent. These SWCNTs were distributed randomly over the electrodes to form a conductive network, as shown in Figure 3.1. The SWCNTs used were unsorted, which means they are mixture of semi-conducting and metallic varieties so that at higher concentrations, they can yield more metallic percolating pathways, which results in significantly higher current. Figure 3.2 shows the optical micrograph of different concentration of SWCNTs random dispersion.

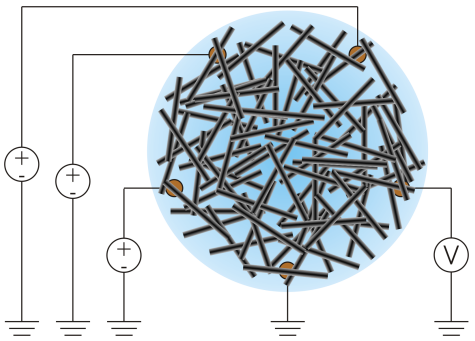


Figure 3.1: The dispersion of carbon nanotubes over the electrodes

Different concentrations of SWCNTs have been used due to the fact that the concentration of SWCNTs affects the electrical properties of the composite materials. Table 3.1 lists the different concentration of SWCNTs in PMMA used for experiments described in this thesis.

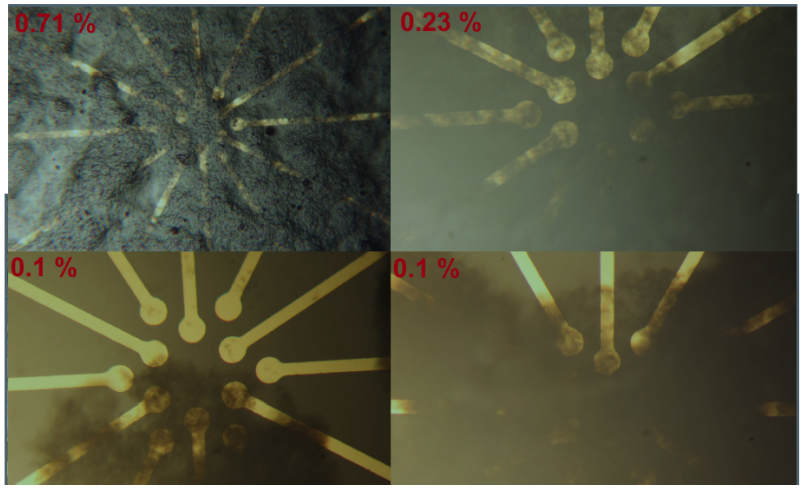


Figure 3.2: Optical micro-graphs of the various nanotube concentrations deposited on gold electrode arrays

Table 3.1: List of different concentrations of SWCNT/PMMA composites used for initial experiments

CNT (wt.% fraction of PMMA)	Solvent	PMMA wt%
1.3	Anisole	5.0
4.7	Anisole	5.2
0.71	Anisole	4.9
0.012	Anisole	16.8
0.23	Anisole	16.5
0.10	Anisole	14.8
0.05	Anisole	24.5

It is one of the desirable characteristic for the material that it should return to its previous state when signals are removed from it. Hence, the non-linearity of the response was suggested as a key property for the suitable computational material. The more conductive materials samples ($>$ 1 wt% SWCNT) showed

linear response in terms of current versus voltage relationships. The I/V (current vs voltage) plots for 3 different concentrations are given in Figure 3.3. The very low concentrations (0.012 wt%) showed a non-linear response, whereas, higher concentrations (0.1 and 0.23 wt%) showed significantly greater conductivity. Figure 3.4 shows the maximum current recorded at 2-volts as compare to SWCNT concentration.

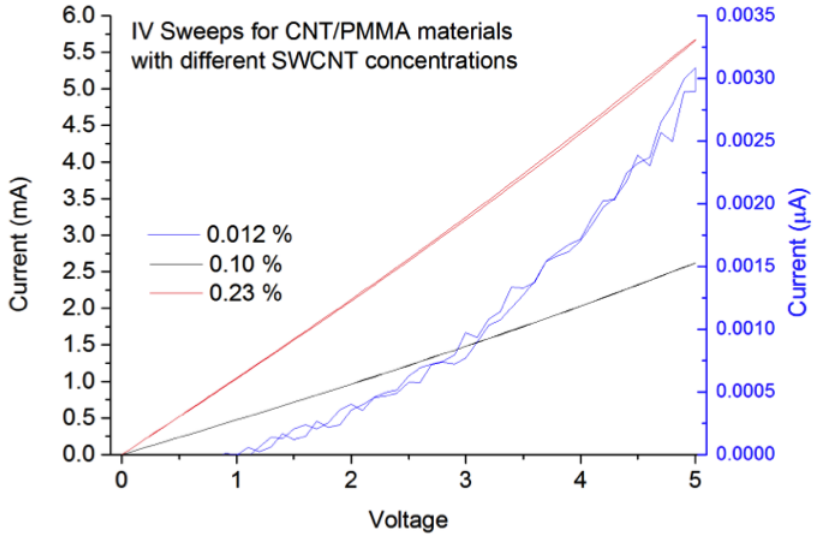


Figure 3.3: Current versus voltage graph

In order to increase the suspension stability of nanotubes, PMMA with increased viscosity was used. Figure 3.5 shows better electrode coverage with a PMMA wt% of ~ 25 and CNTs wt% of 0.1, as compare to lower viscosity of PMMA, shown in Figure 3.2. The main aim was to find which configurations of carbon nanotube composites have better computational capabilities with respect to different computational problems. The comparison of results for these material composites for the solution of simple logic gates is given in chapter 5.

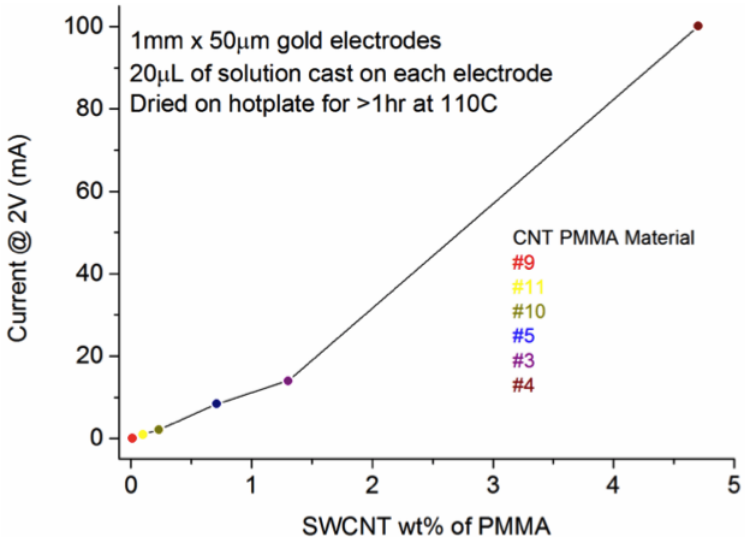


Figure 3.4: Current versus SWCNTs wt. % of PMMA graph

Figure 3.5: Electrode array with SWCNTs/PMMA (0.1% CNT) material (left) and an optical micrograph of the electrodes (right)

3.1.2 SWCNTs/PBMA [1]

This thesis reports on computational capabilities of another set of materials, where SWCNTs are embedded within another type of polymer called, poly butyl methacrylate (PBMA). The materials are prepared in same way as SWCNTs/PMMA composites using the techniques described in [146] and [147]. The composites were designed by mixing powders of SWCNTs with PBMA (Sigma Aldrich, M_w337000) and then dispersing these in Anisole (VWR, analytical reagent grade), using an Ultrasonic probe (Cole-Palmer 750W ultrasonic homogenizer) at a power of 20%. The SWCNTs/PBMA dispersions were visually uniform and remained

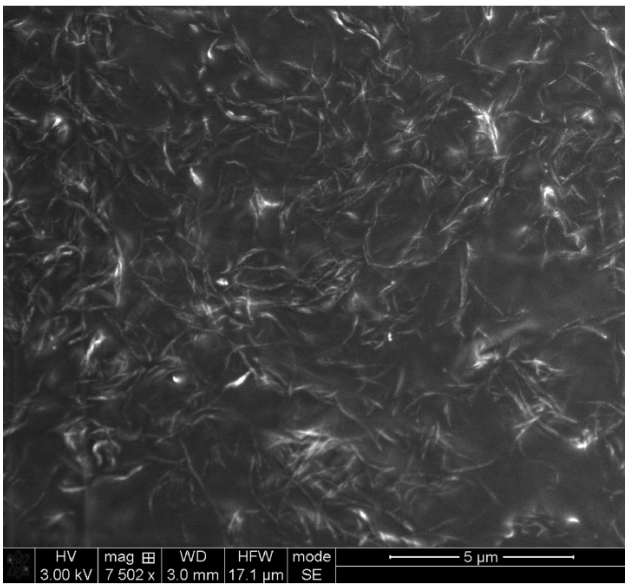


Figure 3.6: Scanning electron microscope image of a typical region of a spin coated SWCNT/PBMA composite

stable over time. The varying concentrations of SWCNTs were used with fixed polymer concentration to study the relationship between conductivity and its effect on computation. The SWCNTs/PBMA composited were spin coated on 16×16 electrode arrays (section 3.2, Figure 3.8) using with a final spin speed of 5000 rpm, these are then dried on a hotplate at $85°C$ for 10 minutes. The concentration of SWCNTs/PBMA determined the thickness of thin film and was about $1 - 5\mu m$. Figure 3.6 shows a typical region of SWCNTs/PBMA under an electron microscope where well dispersed bundles of carbon nanotubes can easily be seen. The different concentration of SWCNTs with PMMA used in experiments described in this thesis are listed in Table 3.2. It was observed that the concentration of SWCNTS in polymer affects viscosity of the composites. The viscosity increases significantly with the addition of SWCNTS to the polymer. This results in changing the electrical properties of the composite. However, electrical conduction was strongly dependent on the concentration of SWCNTs.

Table 3.2: List of different concentrations of SWCNT/PBMA composites used for initial experiments

SWCNT (wt.% fraction of PMBA	Solvent	PBMA wt%
0.11	Anisole	10
0.25	Anisole	10
0.51	Anisole	10
0.74	Anisole	10
0.99	Anisole	10
1.49	Anisole	10
2.39	Anisole	10

3.1.3 Reduced Graphene oxide composites

The reduced Graphene oxide sample were prepared by mixing reduced graphene oxide with 25 wt% of PBMA. The mixture was then dissolved in a solvent prepared with a mixture of Dimethyformide (DMF) and chloroform (CF) in the ratio of 7 : 3 respectively. Three reduced Graphene oxide samples with Graphene concentration of 0.475, 1.04, and 1.42 wt% of PBMA were used. The materials were placed on glass electrode arrays with 12 contact points, (section 3.2 Figure 3.7) using drop casting method.

3.2 Micro electrode arrays

The materials to be tested were placed on micro electrode arrays that were designed using conventional etch-back photo-lithographic techniques using chromium/gold on standard borosilicate glass slides. These electrodes were designed with either

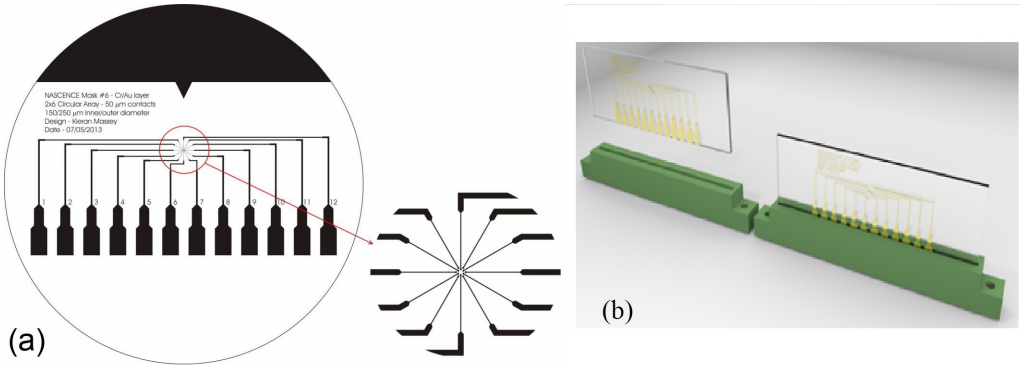


Figure 3.7: Electrode array layout showing (a) the mask used for photo-lithography and (b) the completed array in a PCB edge connector.

twelve contact points with an electrode spacing of $25\mu m$ and 16×16 contact points. The contact pads for the 16×16 electrodes had a diameter of $50\mu m$ and a pitch of $100\mu m$. These contacts can be assigned as an input, output, ground and configuration voltages. The material to be investigated is drop casted at the centre of 12 electrodes slides as shown in Figure 3.7 and spin coated on a 16×16 electrodes as in Figure 3.8.

A quick connect system (as shown in Figure 3.7(b)) using a Printed Circuit Board (PCB) edge connector (in green) was designed to easily replace the glass electrode containing the material with other glass electrode.

3.3 Signal generation device (SGD)

Once the materials were available, the next step was to apply voltage signals to the substrates and record the response. The experiments started with a very simple setup. A signal generation device, Figure 3.9, driven by a micro-controller, controls the serial communication to the PC and various Analogue-to-Digital (ADC) and Digital-to-Analogue (DAC) converters are required to apply and receive voltage signals.

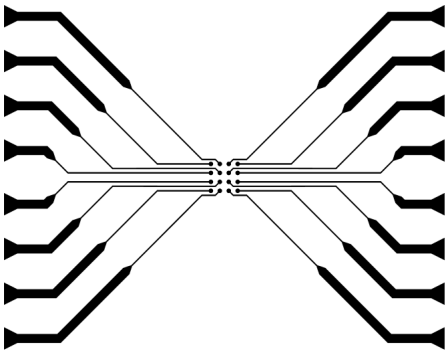


Figure 3.8: A 16×16 electrode array layout showing the mask used for spin coated SWCNTs/PBMA samples.

The mbed micro-controller (mbed.org) has the ability to apply different voltages and measure output voltages. The development environment of the micro-controller includes an on-line compiler, a version control system and 3rd party open source libraries. Upon successful compilation the binary file is downloaded on to the mbed's memory and executed when the reset button is pressed on the micro-controller. The device has following main features:

1. DAC (Digital to Analogue Converter) inputs (DC $0 - 4.095\ V$, 12 bit resolution).
2. A 2×16 cm LCD display.
3. A SD card interface: to store experimental results/data.
4. An ADC (Analogue to Digital converter) output ($0-3.3\ V$, 12 bit resolution); up to 6 available if required.
5. Serial communication with PC: This allows complex algorithms to be run on a desktop PC, with measurements being handled by the mbed micro-controller.

The complete set-up of experimental hardware is shown in Figure 3.10: The optimisation algorithm is run on PC and generated test inputs and configuration

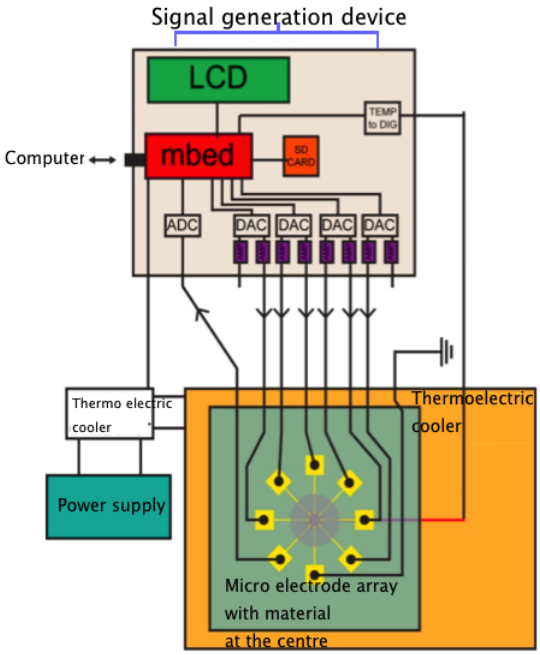


Figure 3.9: The schematic of signal generation device connected with the material

data, these test and configuration signals are sent to signal generation device (SGD) which applied signals on to the material. The response from the material is gathered by SGD and sent back to the optimisation/evolutionary algorithm, the loop continued until the fitness function is achieved.

3.4 Mecobo - a purpose built platform for EIM [2]

Evolutionary computation exploration requires the physical properties of the computational material to be manipulated in various ways. The effect of various signal properties such as, voltage/current levels, AC, DC, pulse or frequency, on material's computational capabilities are still unknown. Mecobo [148] was designed as an interface to handle

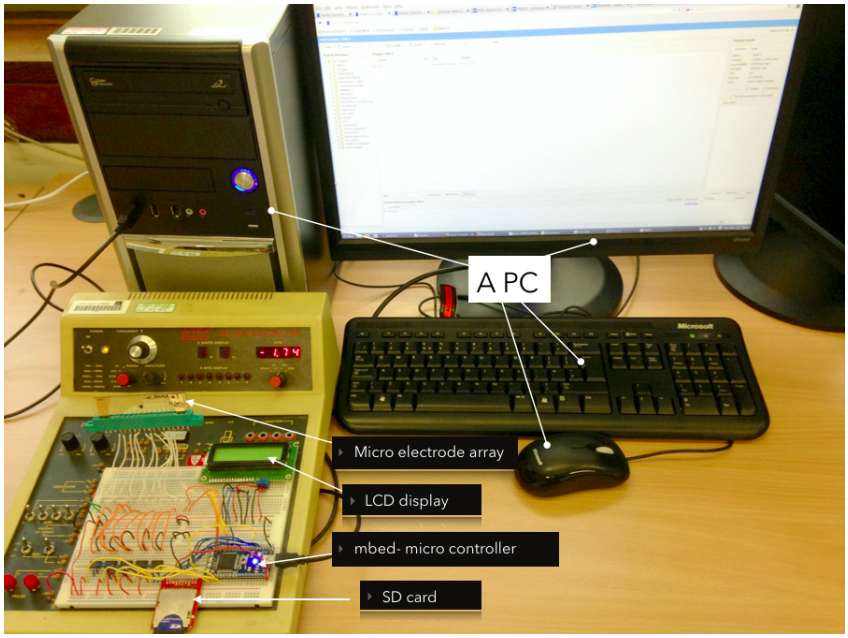


Figure 3.10: The complete setup to conduct EIM experiments.

various physical and electrical signal properties in order to facilitate unconstrained evolution of computation by the materials.

Figure 3.11 presents a systematic view of hardware interface (Mecobo) with software and the material sample. Mecobo can be connected to host computer over USB as a stand-alone interface. The host computer runs the system interface as well as the optimisation algorithm which provides the configuration information. Software interface configures the hardware interface by providing the configuration information for each connection between the nano material and hardware interface. Alternatively, it transforms the output response from the material into appropriate data format for the optimisation algorithm.

Figure 3.13 shows the main components of Mecobo. Mecobo was designed as PCB board with FPGA and micro-controller as main components. The micro-controller accepts the commands from the host computer and implements them on FPGA via shared memory. Whereas, FPGA establishes the physical and logical communication

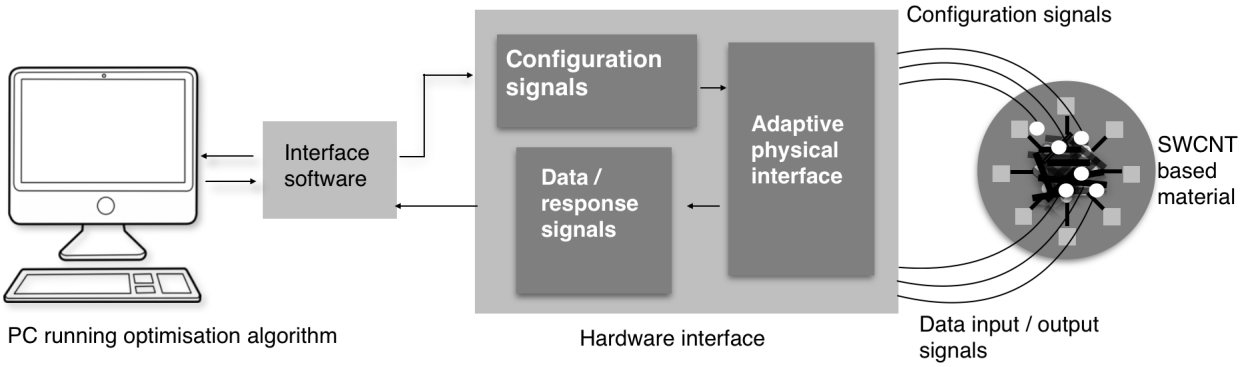


Figure 3.11: Mecobo hardware/software interface with material sample.

with the material substrates. Digital I/O can produce digital signals and sample responses. Where as, analogue output signals are produced by the DAC module. The DAC module can also produce static voltages or time dependent waveforms. The ADC modules perform the sampling of analogue waveforms received from the material.

There are different types of parameters that are associated with the digital and analogue signals that enable the variety of configuration signals applied to the material substrates. The list of these parameters with respect to different electrical signals is given in Table 3.3. Furthermore, a scheduler was implemented in the hardware that can schedule the time slots for different I/O or configuration signals or to compensate delays when materials need time to settle before any meaningful computation can be observed. A scheduler can be seen in Figure 3.14, that illustrates interface hardware implementation. After the time slots have been allocated for different input/output operations on different pins, these operations will be played back just like music or video editing applications. Figure 3.12, illustrates an example of this implementation, where PIN01,PIN02, PIN03 and PIN04 act as input pins and different voltage operations with different durations

Parameter	Description
Amplitude	Amplitude for static voltage signals Range:[0-255] 0=-5 V,255= +5 V
Frequency	Frequency of square wave signals
Cycle	Cycle of square wave signals
Phase	Phase of square wave signals
Start time	Start time of voltage application (in milli seconds)
End time	End time of voltage application (in milli seconds)

Table 3.3: Mecobo's adjustable parameters

are scheduled to be applied on to the material. PIN00 act as an output pin, which will receive the output signals from the material during that specific duration. Due to the fact that the SWCNTs/polymer samples are randomly distributed over the micro-electrode arrays and there are no specific input/output locations, the choice of input/output and configuration terminals should be left to optimisation control. In order to achieve this, a pin routing module was placed between signal generating modules and the sampling buffer. Hence, in contrast to previous hardware (mbed) where pin configuration was predetermined, the Mecobo implemented pin configuration under optimisation control.

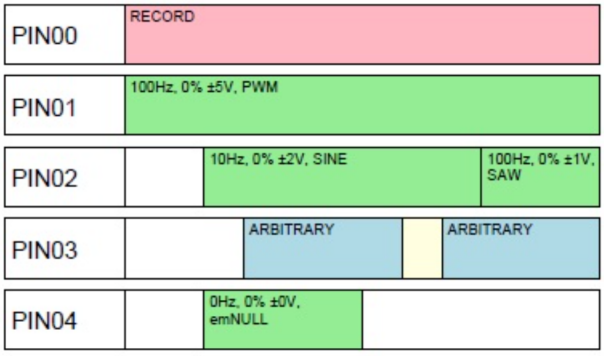


Figure 3.12: An example of implementation of track based model of scheduler.

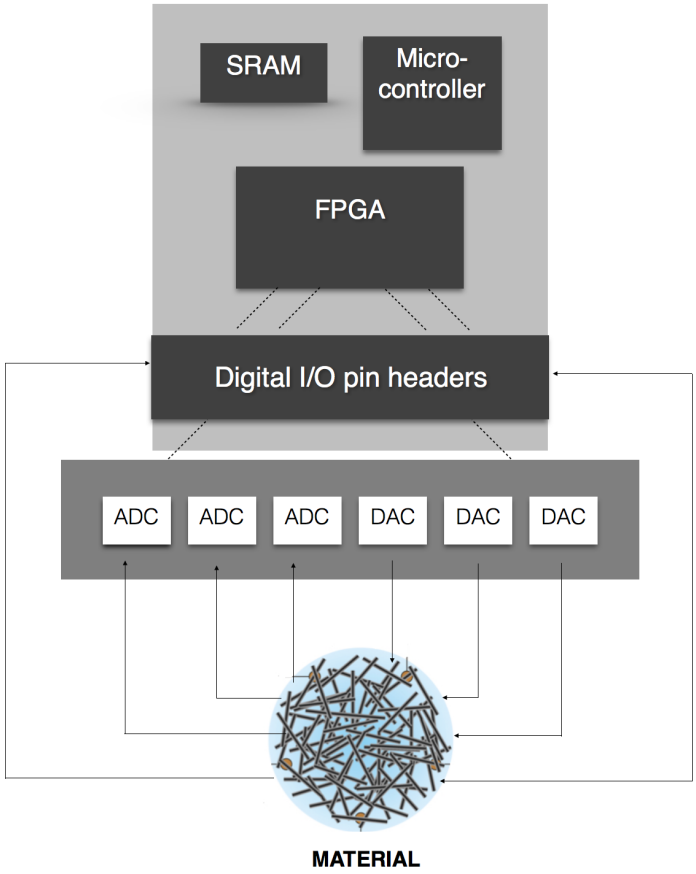


Figure 3.13: Mecobo block diagram

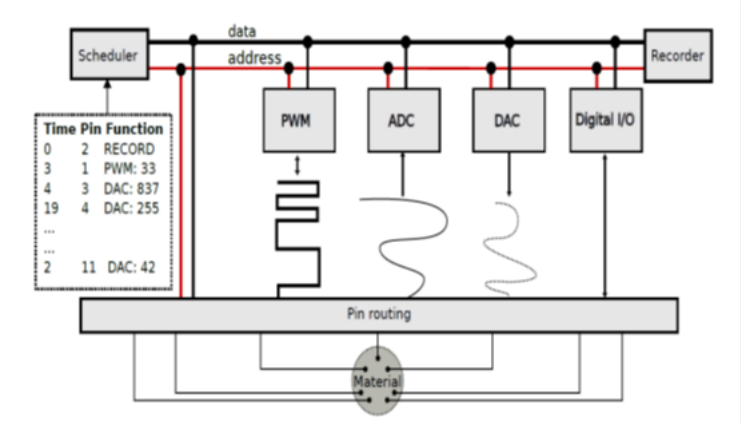


Figure 3.14: Mecobo overview- Hardware interface implementation.

Chapter 4

Optimisation algorithms and computational problems

This chapter discusses the optimisation and evolutionary algorithms that are used in the experiments described in following chapters. Section 2 discusses the Nelder-Mead algorithm, where as section 2 describes the Particle swarm algorithm. Section 3 discusses the threshold concept, that is used to implement threshold logic gates in different SWCNTs composites.

4.1 Introduction

Unconventional computing systems widely employ evolutionary algorithms in their design and implementation. An evolutionary algorithm is stochastic search technique that is inspired by different biological mechanisms to solve optimisation problems.

An important field of research called evolvable hardware uses evolutionary algorithms to design and create electronics. In [149] a genetic algorithm is used to evolve FPGA to produce circuits to calculate Boolean function. A lot of research on FPGAs and evolvable hardware and its implementation is reported in

literature, for example [150], [151], [111], [152], [62], [64]. A device called FPMA is described in [62] that uses genetic algorithms, to find solutions to the problems by exploiting the physical properties of FPMA. This technique includes hardware in the loop to calculate the objective function for every candidate solution. The FPMAallows *'evolution in materio'*.

Different stochastic search algorithms have the potential to be used with EIM. However, only genetic algorithms have been explored so far [153], [54], [38], [40], [41], [39]. An important direction of research in this thesis is to explore algorithms that can be appropriate to use with different types materials and different computational problems. This chapter describes three main algorithms that have been used for various experiments described in this thesis. In the start, a very simple, simplex based, heuristic search algorithm called Nelder-Mead algorithm is used for experiments described in chapter 5 and chapter 6. Later, Particle Swarm algorithm (PSO) and Differential evolution are used for experiments described in Chapter 5 and performance of Nelder-Mead algorithm and PSO is compared. Differential evolution algorithm showed better performance in solution of more complex logic circuits, such as Full adder, the results are discussed in detail in Chapter 5. Chapter 6 used variant of PSO called Shortest Position Value (SPV) rule for solving complex logic circuits on the purpose built platform, 'Mecobo'. The detailed description of these algorithms is given in following sections.

4.2 Nelder-Mead algorithm

Nelder-Mead algorithm is a popular direct search method that attempt to solve a problem using only the function value information. The experiments described in chapter 5, chapter 6 used Nelder-Mead algorithm for optimisation process. Nelder-Mead is a simplex based, direct search method, derivative free method

presented by Nelder and Mead [73] in 1965. If $f(x)$ is a non-linear function to be minimised, for $x \in \mathbb{R}^n$, then a simplex method constructs a changing pattern of $(n+1)$ vertices in $\mathbb{R}^n$. A simplex with two dimensions is regarded as a triangle and with four dimensions as a tetrahedron. The Nelder-Mead method is based on sequence of changing simplexes that are modified in such a way so that the simplex adapt itself towards the local landscape [73].

Various modifications and updates are available for this method, however, the very basic Nelder-Mead method described in [73], is chosen for solving logic gate problem discussed in chapter 4, and 5. The properties of Nelder-Mead method that made it appropriate as a first choice for the optimisation problem are its simplicity, robustness (tolerance to noise), ease of programming and low overhead in computation and storage.

4.2.1 The algorithm

At each iteration of algorithm a trial step is computed by constructing a simplex S of $n+1$ vertices ($n = 2$ is a triangle), where the vertices are denoted by $x_1, x_2, \cdots, x_{n+1}$. Each iteration begins with ordering and labelling these vertices according to their function values. For example at iteration k, the current vertices of simplex S^k are ordered as $x_1^k, x_2^k, \cdots, x_{n+1}^k$ with $f(x_1)^k \leq, f(x_2)^k \leq \cdots \leq f(x_{n+1})^k$. As the objective is to minimise the function f, x_1^k is referred to as the best point and x_{n+1}^k as the worst point.

After calculating the value of f at trial points, the worst trial point is removed and a new trial point is added. The new trial steps are generated by operations called, reflection, expansion, contraction and shrinkage and their coefficients are denoted by α,β,γ,δ respectively. According to [73] these coefficients should satisfy the condition $\alpha > 0$, $\beta > 1$, $0 < \gamma < 1$ and $0 < \delta < 1$. The standard value of these coefficients are used for the experiments described in this thesis, unless

stated otherwise. The most accepted standard values of these coefficients are: $\alpha = 1$, $\beta > 2$, $\gamma = 0.5$ and $\delta = 0.5$. The operations to produce new trial steps are as follows:

Reflection

A new reflected point x_r is created by reflecting the worst point, x_{n+1}. The reflected point x_r is created as follows:

$$x_r = \bar{x} + \alpha(\bar{x} - x_{n+1}) \tag{4.2.1}$$

Where $\bar{x}$ is the centroid of all n vertices of simplex except x_{n+1} and $\alpha = 1$. The centroid is computed as

$$\bar{x} = \frac{1}{n}\sum_{i=1}^{m} x_i \tag{4.2.2}$$

f_r is computed for the new reflected point x_r. The new point is only accepted

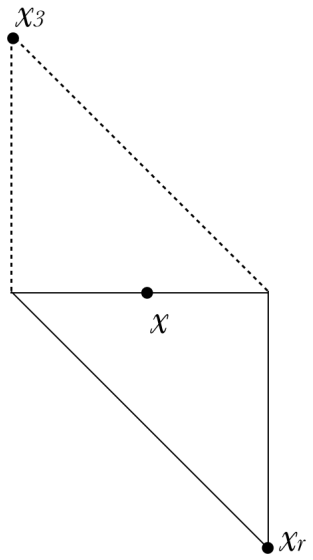


Figure 4.1: Reflection of simplex with two dimensions (a triangle). The original simplex is shown with dotted line.

if $f(x_1) \leq f(x_r) < f(x_n)$. The next iteration begins with the new simplex with

vertices defined by $x_1, x_2, \cdots, x_n, x_r$, although the new point x_r is not ordered with respect to other points in simplex. The reflection operation is shown in Figure 4.1.

If the value of function at x_r is less that the value of function at x_1, this implies that the trial step has produced a good vertex and the trial step will be expanded. The expansion operation is as follows

Expansion

The expansion process will be computed as follows:

$$x_e = \bar{x} + \beta(x_r - \bar{x}) \tag{4.2.3}$$

Where $\bar{x}$ is computed using equation 4.2.2 and $\beta = 2$.

The next step begins with computing the value of $f(x_e)$ and is compared with $f(x_1)$. If $f(x_e) < f(x_1)$, the expansion vertex is accepted, otherwise reflection vertex is accepted and next iteration will continue.

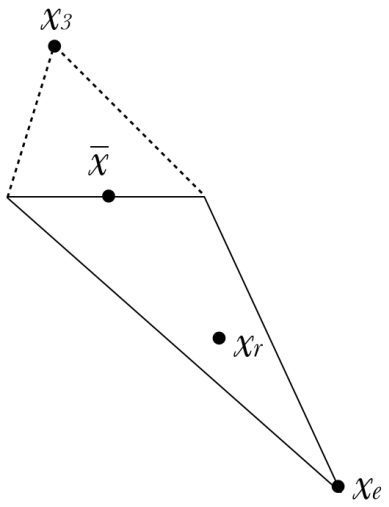


Figure 4.2: Expansion of a simplex. The original simplex is shown with dotted line

Contraction

Two types of contraction operations are performed if the reflected vertex is not better than x_n i.e. $f(x_r) \geq f(x_n)$.

- If the value of worst vertex $x_{(n+1)}$ is better than or equal to the reflected vertex then an internal contraction is performed

$$x_c = \beta x_{(n+1)} + (1+\gamma)\bar{x} \tag{4.2.4}$$

- otherwise an outside contraction is performed

$$\hat{x_c} = \beta x_r + (1-\gamma)\bar{x} \tag{4.2.5}$$

Where the value of beta is 0.5. the value of contraction vertex is accepted when its value lower than the value of x_n. In case, if both reflected and contraction vertices are not acceptable, then the shrink operation is performed.

Shrink

The shrink operation is performed by replacing all the vertices x_i with the new vertices, except x_1. The new vertices are computes as follows:

$$v_i = x_1 + \delta(x_i - x_1) \tag{4.2.6}$$

Finally, the function values $f(x_i)$ are calculated and sorted along with $f(x_1)$ and the new iteration commences with the simplex having vertices $x_1, v_2, \cdots, v_(n+1)$. Figures 4.1, 4.2, 4.3 represent the reflection expansion, contraction and shrink operations performed by the simplex, using the coefficients α, β, γ, δ. It can be easily seen that simplex undergoes noticeable changes during expansion and contraction stages.

Various termination criteria has been proposed for Nelder-Mead algorithm [154], [155]. For example termination criteria when the function values on two vertices become very close, or when the simplex values become very small [155]. The stopping criteria is based on the relative size of the simplex with respect

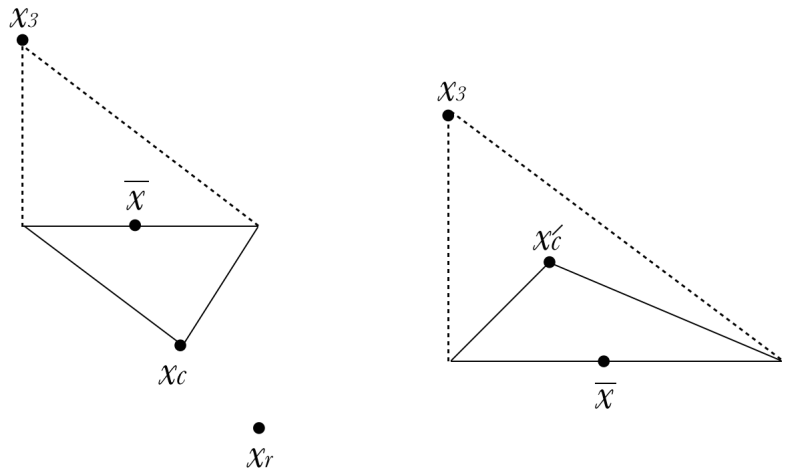


Figure 4.3: Inside, outside contractions and shrink operations of a Nelder- Mead simplex an The original simplex is shown with dotted line.

to the size of the initial simplex i.e. (tol=10^{-8}) and the maximum number of iterations. The peudocode is given below.

Pseudo code

for $i = 0$, $max - iterations$

Compute an initial simplex S_0

Sorts the vertices S_0 with increasing function values $S \leftarrow S_0$

while *tol* **do**

$\bar{x} \leftarrow \bar{x}(n+1)$

$x_r \leftarrow \bar{x} + \alpha(\bar{x} - x_{n+1})$ {Reflect}

$f_r \leftarrow f(x_r)$

if $f_r < f_1$**then**

$x_e \leftarrow \bar{x} + \beta(x_r - \bar{x})$**{**Expand**}**

$f_e \leftarrow f(x_e)$

if $f_e < f_r$**then**

Accept x_e

else

Accept x_r

end if

else if $f_1 \leq f_r < f_n$ **then**

Accept x_r

else if $f_n \leq f_r < f_{n+1}$ **then**

$x_c \leftarrow \beta x_r + (1-\beta)\bar{x}$ {Outside contraction}

$f_c \leftarrow f_{(x_c)}$

if $f_c < f_r$**then**

Accept x_c

else

Compute the points $x_i = x_1 + \delta(x_i - x_1),\ i = 2, n+1$ {Shrink}

Compute $f_i = f_{(v_i)}$ **for** $i = 2,\ n+1$

end if

else

$x_c \leftarrow \beta x_{(n+1)} + (1+\beta)\bar{x}$ {Inside contraction}

$f_c \leftarrow f_{(xc)}$

if $f_c < f_{n+1}$ **then**

Accept x_c

else

Compute the points$xi = x_1 + \delta(x_i - x_1), i = 2, n+1$ {Shrink}

Compute $f_i = f_{(v_i)}$ **for** $i = 2, n+1$

end if

end if

Sort the vertices of S with increasing function values

end while

4.3 Particle Swarm Optimisation

The Particle Swarm Optimization (PSO) algorithm is a stochastic optimization algorithm that is inspired by the social behaviour of animals. The interesting information sharing social behaviour of fish schools or animal flocks has been studied by several scientists [156] [157] and was believed that this evolutionary behaviour can be primarily suited for numerical optimization problems. The several examples from nature support this idea and it was the main inspiration for Particle Swarm algorithm. PSO is a member of wide category of swarm intelligent methods [158]. Although, initially proposed as a simulation of swarm behaviour it was used as an optimisation method in 1995 [159], [158].

PSO is simple to implement, it has high convergence rates like Genetic Algorithms (GAs) and requires few parameters to adjust. It has find its applications to a variety of problems, such as evolving the structure and weights for artificial neural networks, power system optimization, process control, dynamic optimization, adaptive control and electromagnetic optimization [160].

PSO is an evolutionary computing technique that begin their search with randomly generated population and utilize a fitness value to evaluate the population. A random search allows the algorithm to escape from local minima and explore flat regions but it comes with computational cost and slow convergence rate.

The experiments describe in chapter 5 used a simple PSO algorithm [161], whereas, Chapter 7 of this thesis used PSO with a variant of PSO called Shortest Position Value (SPV) rule [161].

PSO is selected for the experiments with materials for computation extraction because it offer various advantages over other evolutionary techniques. PSO is easy to implement, it does not require gradient information of the objective function being considered, but its values [159].

The general operation of PSO is as follows.

4.3.1 The algorithm

PSO is a population based optimisation algorithm that begins its search by undertaking a group of randomly selected individuals called 'Particles' (metaphor of birds in flocks). These particles scatter through the multidimensional problem search space. During flight each particle updates its velocity and position by using its own best experience *'pbest'* and the entire population's experience *'gbest'*. The updating mechanism drives the particle towards the desired objective function value and eventually whole population drives towards the region of desired objective function value. Step by step description of the algorithm is as follows.

1. **Initialisation:** The velocity and position of initial population are randomly set to pre-defined ranges.

2. **Velocity Update:** At each iteration a new velocity value V_i for each particle is evaluated according to its current velocity, personal best objective function value and the distance from the global best objective function value or local best position. It can be written as:

$$V_i = wV_i + c_1R_1(P_{i,best} - P_i) + c_2R_2(G_{i,best} - P_i) \tag{4.3.7}$$

Where P_i and V_i are position and velocity vectors of particle i respectively. $P_{i,best}$ is the best objective function value achieved by the particle and $G_{i,best}$ is the best objective function value achieved by the entire population. w is a scalar quantity that denotes inertia weight which controls the flying dynamics. A small inertia weight will facilitate a local search, where as large inertia weight will facilitate the global search, however a constant inertia weight (0.5) is used for the experiments in this thesis; $R1$ and $R2$

are used to maintain the diversity of the population and are normally in the range of $[0, 1]$; $c1$ and $c2$ are acceleration constants are kept as $c1 = c2 = 0.5$ for better results.

The inclusion of random variables allows PSO with stochastic searching. After each velocity update the constrains on velocity ensure that there is no random walking. The velocity update will be used to update the position of the particle in the search space.

3. **Particle update**

$$P_{i+1} = P_i + V_i \tag{4.3.8}$$

4. **Memory update** The algorithm keeps its memory keeping track of P_i and $G_{i,best}$ by following rule:

$$\begin{aligned} P_{i,best} &= P_i \qquad if \quad f(P_i) > f(P_{i,best}) \\ G_{i,best} &= G_i \qquad if \quad f(G_i) > f(G_{i,best}) \end{aligned} \tag{4.3.9}$$

where f is the objective function.

5. **Termination criteria** The algorithm repeats steps 2 to 4 until the termination criteria is met. For experiments described in this thesis the termination criteria is pre-defined number of iterations or a failure to make progress during the certain number of iterations. After termination the values of G_{best} and $f(G_{best})$ are returned as a solution.

Let S be the number of particles in the swarm with each having a position $x_i \in \Re_n$ in the search-space and a velocity $V_i \in \Re_n$ Let P_i be the best known position of

particle i and G be the best known position of the entire swarm. A Pseudo code of a general PSO algorithm is then: [159]

Pseudo code

Begin

For each particle $i = 1, \cdots, S$ **do**

Initialize the particle's position with a uniformly distributed random vector: $X_i \sim U(b_{low}, b_{up})$

Initialize the particle's best known position to its initial position: $P_i \leftarrow X_i$

If $f(P_i) < f(G)$**then**

Update the swarm's best known position: $G \leftarrow P_i$

Initialize the particle's velocity: $Vi \sim U(-|b_{up} - b_{lo}|, |b_{up} - b_{lo}|)$

While a termination criterion is not met **do**:

For each particle $i = 1, \cdots, S$ **do**

For each dimension $d = 1, ..., n$ **do**

Pick random numbers: $r_P, r_G \sim U(0, 1)$

Update the particle's velocity:

$V_{i,d} \leftarrow \omega V_{i,d} + \phi_P r_P (P_{i,d} - X_{i,d}) + \phi_G r_G (g_d - x_{i,d})$

Update the particle's position: $X_i \leftarrow X_i + V_i$

If $f(X_i) < f(P_i)$ **then**

Update the particle's best known position: $P_i \leftarrow X_i$

If $f(P_i) < f(G)$ **then**

Update the swarm's best known position: $G \leftarrow P_i$

End

4.3.2 Shortest position value rule (SPV)

The experiments discussed in Chapter 7 required a mixed integer problem formulation, with discrete integer values. The classic version of PSO generates continuous

values during particle initialisation and velocity update. Similarly, the discrete or binary versions of PSO generate discrete values only. Hence, in order to generate discrete integer variables the PSO with SPV rule [161] is implemented.

The SPV rule converts the continuous values generated by PSO into discrete values. If the particle in PSO is represented by the position vector $X_i = [x_1, x_2, x_3, \cdots, x_d]$, where d is the dimension and i is the individual. SPV rule generates a new sequence vector, where first element is based on the dimension index of smallest value represented in the position vector. For example if the individual generated by PSO with dimension 5 is:

$$X_{id} = \{4.83, -0.55, 1.90, 4.46, 1.05\}$$

A new vector F_{id} is generated according to SPV rule where dimension value of the smallest element becomes the first element in the new sequence and same rule applies for other elements. i.e.

$$F_{id} = \{2, 5, 3, 4, 1\}$$

The rule is applied at every iteration during particle generation and at velocity update.

4.4 Differential evolution (DE)

Differential evolution is a direct search, population based stochastic search optimisation algorithm introduced by Rainer Storn and Kenneth Price in 1996 [162]. It was developed to solve optimisation problems in continuous domain and can easily be applicable to practical applications. The algorithm found a wider acceptance in science and engineering, where problems may have objective functions that are non-differentiable, non-continuous, non-linear, constrained and have many

local minima [162]. It can be used in such applications to find approximate solutions to optimisation problems [163] [164]. DE is an evolutionary algorithm that optimises a problem by maintaining a population of candidate solutions. Like an evolutionary algorithm the optimisation process continues by creating new solutions using the mutation, recombination and selection and keeping the candidate solutions with the best fitness score. Hence, new candidate solutions are only a measure of quality of underlying optimisation problem and therefore, gradient information is not needed.

DE offers various advantages over many other optimisation algorithms. Being a direct search method it can easily be used in optimisation processes that involve experimental data rather than a computer simulation. It requires few variables during the optimisation process, and stochastic perturbation of candidate solutions requires less user input. The general operation of DE is given below:

4.4.1 The algorithm

DE is an evolutionary algorithm and follows the general process of initialisation, mutation,recombination and selection.

Initialisation:

The DE works by having a population of candidate solutions called agents. The agents are randomly selected with no information about the problem. In order to optimise a function $f : \mathbb{R}^n \to \mathbb{R}$ with D real parameters, the size of population N is selected. The function $f : \mathbb{R}$ takes a candidate solution (agent) as an argument and produces a real number as an output which is the fitness value of the current agent. The candidate solution is a parameter vector of following form:

$$x_{i,G} = [x_{1,i,G}, x_{1,i,G}, \cdots, x_{D,i,G}], i = 1, 2, \cdots, N \tag{4.4.10}$$

Where G, denotes the generation number. The upper and lower bounds of parameters are:

$$x_j^L \leq x_{j,i,1} \leq x_j^U \tag{4.4.11}$$

The initial values for parameters are randomly selected between the interval $[x_j^L, x_j^U]$. Each agent in population N, then goes through the process of mutation, recombination and selection.

Mutation

During mutation, for a given agent $x_{i,G}$, three distant agents, $x_{r1,G}, x_{r2,G}$ and $x_{r3,G}$ are randomly selected from the population. The weighted difference of two agents is added to the third agent as follows:

$$v_{i,G+1} = x_{r1,G} + F(x_{r2,G} - x_{r3,G}) \tag{4.4.12}$$

Where F is a mutation constant with value $[0, 2]$ and $v_{i,G+1}$ can be named as donor agent.

Recombination

Where mutation expands the search space, recombination incorporates successful solutions from previous generations. During recombination phase a new trail agent $u_{i,G+1}$ is created from the elements of target agent $x_{i,G}$ and elements of donor agent $v_{i,G+1}$. Elements of donor agent enter the trial agent with probability CR (where CR is the cross over rate and $CR \in [0, 1]$).

$$u_{i,G+1} = \begin{cases} v_{j,i,G+1}, & \text{if } r_{j,i} \leq CR \text{ or } j = I_{rand} \\ x_{j,i,G+1}, & \text{if } r_{j,i} \leq CR \text{ or } j \neq I_{rand} \end{cases} \qquad \begin{aligned} i &= 1, 2, \cdots, N \\ j &= 1, 2, \cdots, D \end{aligned} \tag{4.4.13}$$

Where $r_{j,i} \sim \mathrm{U}[0, 1]$ is a uniformly distributed number and I_{rand} is a random integer between $[1, 2, \cdots, D]$ and it ensures that $v_{j,i,G+1} \neq x_{j,i,G+1}$.

Selection

The next step is the selection between trial agent and target agent. The agent with the lowest fitness function value is selected for the next generation.

$$x_{i,G+1} = \begin{cases} u_{i,G+1}, & \text{if } f(u_{i,G+1)} \leq f(x_{i,G}) \\ x_{i,G}, & otherwise \end{cases} \tag{4.4.14}$$

$$i = 1, 2, \cdots, N$$

The process of mutation, recombination and selection continues until the termination criteria is met.

4.4.2 Parameter selection

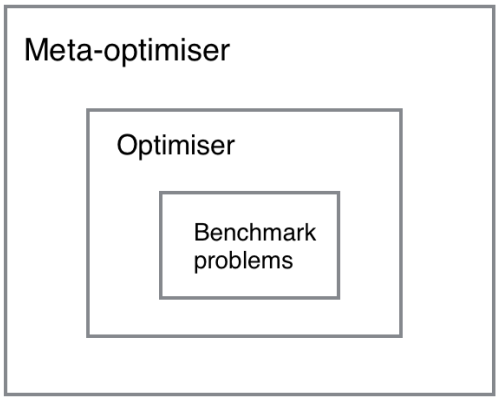


Figure 4.4

The parameters named: population size(N), cross over rate (CR), and mutation constant$F$, described in DE are generally selected by the User. These parameters determine the efficiency and behaviour of the algorithm for optimising a given problem. Therefore, selection of good parameters to yield good optimisation performance is important. Different approaches have been suggested by [165], [166], [167], [168], [169] to select these parameters. The algorithm used for experiments

describe in this thesis used the approach describe in [170]. The process is simple and is depicted in Figure 4.4. It uses an overlying optimiser called 'meta-optimiser' to select DE parameters for different scenarios, which in turn solves the optimisation problem. The DE parameters can be selected from the list [170] according to dimensionality of the problem and allowed number of fitness evaluations. The Pseudocode of the differential Evolution algorithm is given below.

Pseudocode

Begin

Choose the population number N and the parameter number D to be optimised

Randomly initialise CR in an interval [0.8,1]

Create a random initial population $x_{i,G-1},\ i = 1, 2, \cdots, N$

Do

Randomly initialise F in an interval $[0, 2]$

For i=1 to N

Select 3 different individuals randomly from the population G_i

Select a random integer number j_{rand} between $[1, D]$

For $j = 1\ to\ D$

Uniform crossover:

If $((rand(0, 1) \leq CR) or (j = j_{rand}))$

Mutation

$u_{i,G+1} = x_{r1,G} + F(x_{r2,G} - x_{r3,G})$

Else

$u_{i,G+1} = x_{j,i,G+1}$

End if

End For

While the termination condition is achieved

End

4.5 Computational problems

4.5.1 Threshold logic gates

Threshold logic gates are based on majority or threshold decision principle. This means that the output value depends upon the arithmetic sum of inputs and the threshold. The problem is studied in detail with different SWCNTs based materials. A threshold logic gate is defined as a logic gate with n binary input variables, x_i, $i = 1, \cdots, n$, for which there exists $(n+1)$ weights and a threshold θ, such that the output of a logic gate is:

$$y = \begin{cases} 1 & if \sum_{i=0}^{n} w_i x_i \geq \theta \\ 0 & if \sum_{i=0}^{n} w_i x_i < \theta \end{cases} \tag{4.5.15}$$

The experiments described in chapter 5, chapter 6, and chapter 7 present computations performed by different SWCNTs based materials using the threshold concept to find the solutions for various Boolean functions. It includes solutions of some simple logic gates gates using one threshold, such as, AND, OR and XOR and multiple thresholds to find solutions of some complex circuits such as, Half adder and Full adder.

4.5.2 Classification

Machine learning studies how to automatically learn to make accurate predictions based on past observations. In machine learning, classification problems classify new instances (observations) into given set of sub-categories. The identification of categories is based on a data set called 'training data set', which contains instances (observations) where the categories of instances are accurately known. The algorithm that implements the classification is generally known as the classifier.

The general construction of a classification procedure is shown in Figure 4.5. The

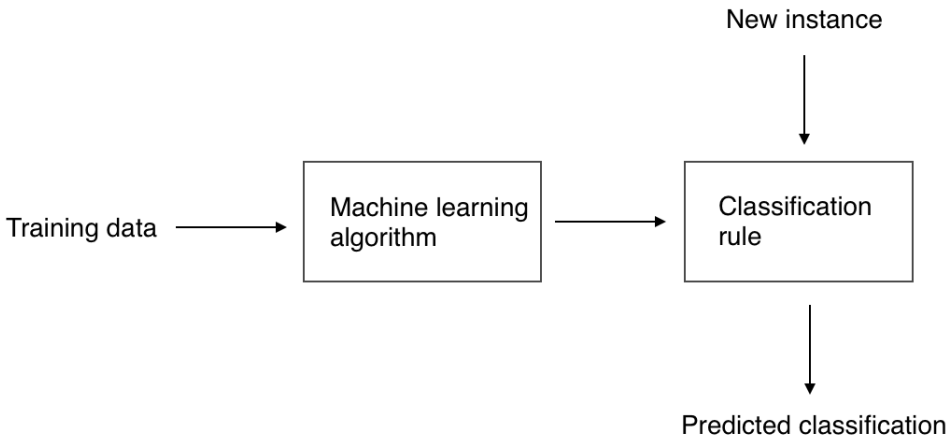


Figure 4.5: The general construction of a classification procedure

work in this thesis studies the implementation of classification problem with the SWCNTs based materials. The work is described in detail Chapter 9, where the results of implementing binary classification as well as multiple class data classification have been discussed.

4.5.3 Tone discrimination

Tone discriminator is defined as a device that can discriminate between two signals and returns a different response for each signal [39]. The problem is studied in detail with different SWCNTs based materials and different frequency signals and results are presented in Chapter 8.

4.6 Summary

One of the major objective of this study is to find suitable methods of optimisation for implementation of different computational problems with variety of SWCNTs

based materials. Three different optimisation algorithms and methods used to implement experiments described in this thesis are presented in this chapter. Three algorithms, i.e Nelder-Mead and Particle Swarm Optimisation and Differential Evolution have different methods of implementation. These are chosen because of their simplicity of implementation and use. These algorithms found wider acceptance for the solution of variety of optimisation problems in different fields. The performance of these algorithms will be compared in the experiments for their suitability with different computational problems and materials.

The threshold method is used to implement various Boolean functions such as AND, OR, Half adder and Full adder. The detailed discussion about the implementation of this method for different logic gates and circuits is given in following chapters.

Chapter 5

Logic gates/circuit training in SWCNTs/PMMA composites using mbed

This chapter presents the results of initial experiments of computations performed by SWCNTs based composites using the very basic hardware set up (mbed) 3.3. The hardware allows the application of static voltage signals to the material composites and the analogue voltage signals to be collected from it. The computation is based on threshold logic concept 4.5.1. Electrical conductivity is the property of the material, that is manipulated to evolve Boolean logic gates.

These experiments reported the computational capabilities of SWCNTs/PMMA based materials for the first time. Previously, Liquid Crystals were reported to solve some basic computational problems [40], [41], [39], however the need for a more robust material was identified which can implement EIM more efficiently. After having initial satisfactory results with one mixture, various other mixtures of SWCNTs/Polymer were used to study the effect of SWCNTs/polymer concentration on computation. The results of these experiments provided an indication, that

the SWCNTs based composites are suitable substrates that can solve a basic computational task (Boolean logic gates) and makes them an ideal candidate for further studying of other computational problems. Three different optimisation algorithms have been used in different experiments and their performances are compared.

It is also reported in this chapter, that the choice of contact points, for the application of input signals and the collection of output signals have a significant effect on the computation performed by the SWCNTs/PMMA composites.

Later, three different compositions of Graphene/PMMA composites are also studied for their basic computational capabilities. The detail description of problem formulation for threshold based logic gates/circuits and analysis of results are reported in this chapter.

5.1 Introduction

The candidate materials are complex in nature with complex physical properties. Various computational problems are reported in [54] that can be implemented with the materials suitable for computation. One such problem named, Boolean logic gates, is the simple and the most studied problems in the field of evolutionary computing. For example, one/two bit adder problems serve as the standard benchmark for testing genetic algorithms efficiency [171]. Hence, in order to test the SWCNTs based material systems for their computational capabilities, the initial experiments implemented very basic Boolean logic gates such as AND and OR. Later, more complicated logic circuits named, half adder and full adder are also implemented. The computational problem is implemented with varying concentrations of SWCNTs and polymer (PMMA) and Graphene/PMMA. In order to train the material as boolean logic gate/circuit, an optimisation problem

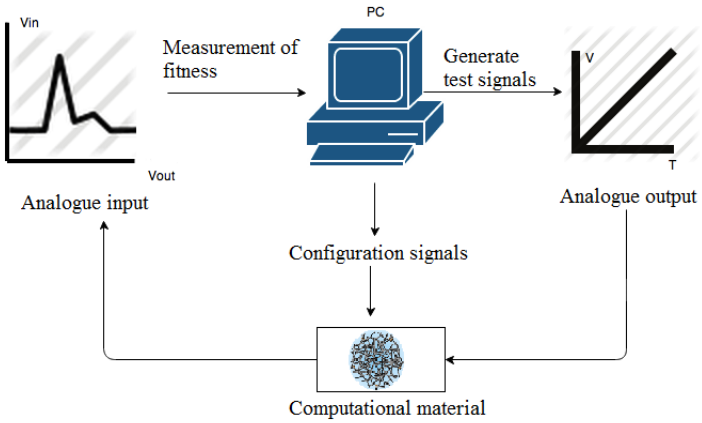


Figure 5.1: The general idea to train the material to solve a computational problem

is formulated with continuous and binary constraints.

For the experiments reported in this chapter, an optimisation problem has been formulated to evolve Boolean logic gates/ circuits in SWCNTs based materials using EIM. The general idea to train the material to solve a computational problem is shown in Figure 5.1. The optimisation algorithm provides the values for the inputs and configuration signals that are applied as stimuli on to the material. The response is then gathered from the material and the fitness function is evaluated, if the desire fitness value is not achieved, new data for configuration signals along with other decision variables are generated by the optimisation algorithm. The input/configuration signals are applied on to the material and response is gathered again. This loop continues until the desired fitness value is achieved. In order to start the experiments and keep things simple, a very basic optimisation algorithm called, Nelder-Mead algorithm (Section: 4.2) is used. The methodology and problem formulation for evolving boolean logic gates is given in following sections.

5.2 Optimisation procedure

The experiments used SWCNTs/PMMA composites that are placed on a 12 electrode arrays. The detailed description about SWCNTs/PMMA and electrodes is given in chapter 4. The material has randomly dispersed network of SWCNTs that act as a network of resistors and their output voltage increases monotonically with the current as shown in Figure 5.3. These output measurements can be put into non overlapping, staggered bands. Hence, threshold voltages for the output (based on a sum of the inputs) can be defined. This way it is easy to achieve an AND and OR gate and other Boolean functions.

In order to train the material to behave as Boolean logic gates/circuits, the optimisation problem is formulated with hardware in the loop, i.e. the objective function J is evaluated directly from the material. Since, every optimisation problem is represented in the form of parameters and decision variables, where the parameter value remains constant and decision variable values are changed by the optimisation algorithm. The objective function and constraints are expressed in terms of these parameters and decision variables. The parameter for current problem formulation are:

- The number of binary inputs p.
- The number of configuration voltages r.
- The number of binary outputs m.
- The number of thresholds L.
- The lower and upper bounds of the voltages applied at the electrodes, G_{min} and G_{max}, respectively.
- $G_0 = 0$; ground voltage.

- The truth table of the desired logical circuit $T(A) \in (0,1)^m$, where $A = [A_1 ... A_p]^T \in (0,1)^p$ is the binary input vector.
- The number of examples K used for training the material. A training example with index $k, k = 1, \cdots, K$ is a pair in the form of $(A^{(k)}, T^{A(k)})$. For example for an AND gate an instant of a training example with index k is: $((1,1)^{(k)}, 1^{(k)})$. The inputs were sent in an arbitrary order.
- B_{max} an upper bound of the scaling factor β.

The decision variables are as follows:

- G_{b_0}, G_{b_1} the input voltages that signify a binary 1 and 0 respectively;
- $G_1, \cdots, G_r$ the configuration voltages which can be set between $[G_{min}, G_{max}]$, these affect the measurements at the materials output locations;
- The scaling factor $\beta \in [0, B_{max}]$, without units for calculating threshold values.

Hence, a candidate solution is a vector of the following form:

$$x = [G_{b_0},\ G_{b_1},\ G_1,\ \cdots G_r,\ \beta]^T \tag{5.2.1}$$

Depending upon the type of logic circuit the electrode pins on glass slide are assigned as input pins, output pins and the remaining are dedicated as configuration pins. For example, on a 12 electrode array, a 2-input and 1-output logic gate will have 9 configuration pins. The arrangement of input, output and configuration terminals is predefined, as the hardware set up does not allow to put the terminal selection under optimisation control. Figure 5.2 shows an example of arrangement of input, output and configuration electrodes. For the system in use, the voltages

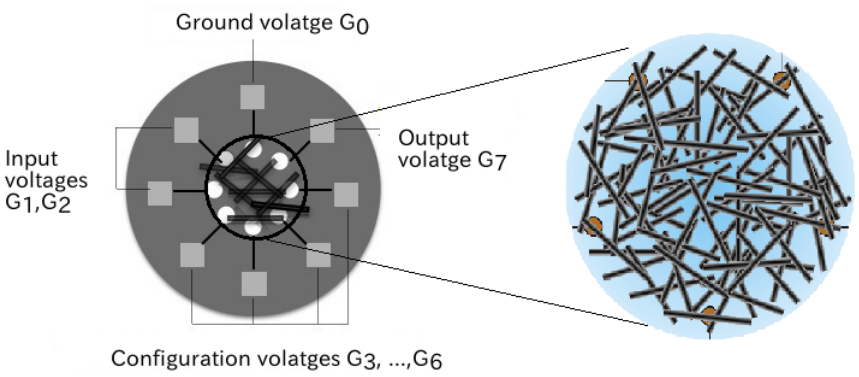


Figure 5.2: Randomly dispersed network of SWCNTs over electrodes

measurements are used. For example, in Figure 5.2, two locations G_1 and G_2 are charged with binary input voltages (e.g. G_{b_0} represents a binary 0 and G_{b_1} represents a binary 1) and an output voltage value (M) is collected at G_7. The rest of the electrodes are charged with the configuration voltages G_q (where $q = 1 \cdots r$). The input and configuration voltages control the conductive properties of the material and effect the output $M_j, j = 1, \cdots, m$ measured at j different electrodes. In other words, M_j is a measure of material's particular physical quantity (in this case conductance). The output gathered from the material can be arranged in to staggered bands and threshold logic is implemented using equation (5.2.2). This idea is shown in Figure 5.3.

The operation of a Threshold logic circuit with m binary outputs and $L_j + 1$, $j = 1, \cdots, m$ thresholds can be generalised as:

$$y_j = Y_{j,c} \quad \text{if} \quad \theta_{j,c-1} \leq \sum_{i=1}^{n} M_j < \theta_{j,c} \tag{5.2.2}$$

Where $Y_{j,c} \in 0, 1$ and θ_c, $c = 0, \cdots, L$ are the known thresholds for each output.

The outcome of this operation must be equivalent to particular gate's truth table $Y(A) \in \{0, 1\}^m$. Where $A \in \{0, 1\}$ is the circuit's binary input and $Y_j(A)$

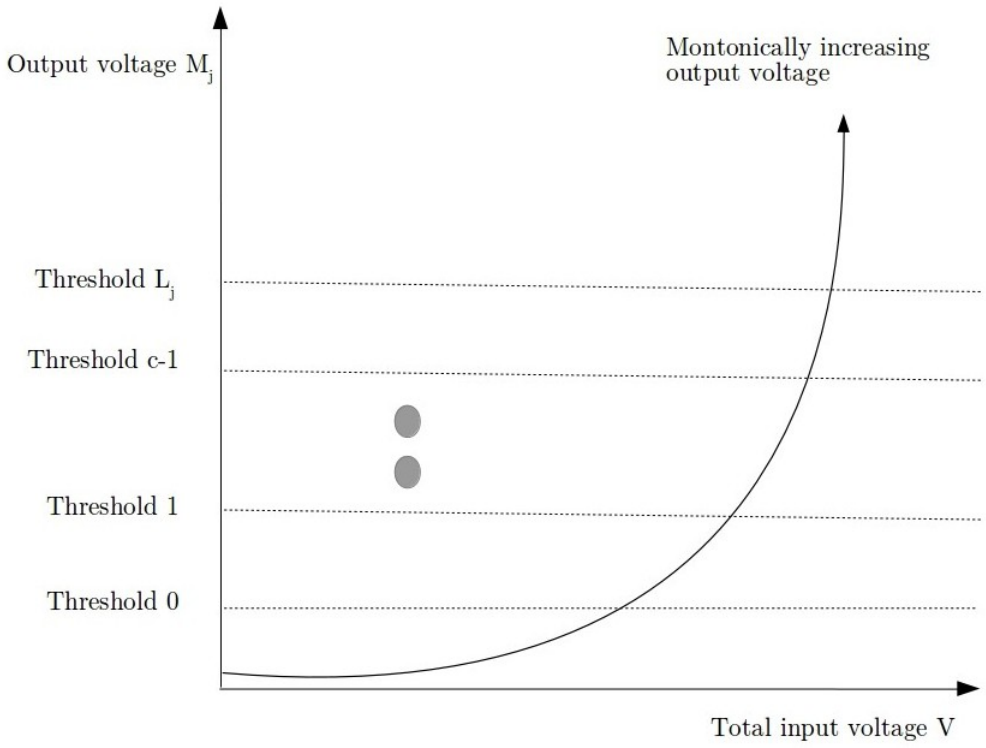


Figure 5.3: The division of range of output M into staggered bands to be assigned to input pairs, implementing the equation (5.2.2)

is the respective binary output j. The computation performed by the material is embedded in the mapping of inputs into particular bands of outputs, as defined by the equation (5.2.2). The general form of this mapping is given by

$$H_j = F_j[M_j, Y(A), \theta_{j,0}, \cdots, \theta_{j,L_j}] \text{ where } j = 1, ..., m \tag{5.2.3}$$

The function F_j maps the measured output M at jth location to a band assigned according to (5.2.2). The mapping may be different for the measurements collected at different locations i.e $F_{j1} \neq F_{j2}$.

The threshold voltage values are set as a function of some decision variable voltage value. Hence, it is the optimisation that decides threshold values, instead of user-defined values which can be time-consuming and may be irrelevant to

the material. In some experiments, the threshold value is then multiplied by a scaling factor β which is also decided by the optimisation. A scaling factor can help to keep the values within the acceptable range of the optimisation algorithm. Otherwise it may be possible that these values can exceed maximal values.

In other words, the threshold values $\theta_{j,c}$ used for the measurement collected at output location j are calculated from

$$\theta_{j,c} = f_{j,c}((G_{b_0},\, G_{b_1},\, G_1,\, \cdots,\, G_r) \times \beta), j = 1, \cdots, m, c = 0,\, \cdots,\, L_j \quad (5.2.4)$$

For a given circuit the threshold values for each output j are organised in the vector θ_j.

The objective function for the current optimisation problem is selected as a quadratic expression of the error when a potential solution x (equation:(5.2.1)) is applied to the material. $M_j^{(k)}(x, A^{(k)})$ is the output measured at electrode j when binary input(s) $A^{(k)}$ is applied. The corresponding binary outcome $H_j(x, A^{(k)})$ is calculated according to the threshold rules described in equation (5.2.2) and is given by

$$H(x, A^{(k)}) = \begin{cases} 1 & \text{if } M(x, A^{(k)}) \geq \theta \\ 0 & \text{if } M(x, A^{(k)}) < \theta \end{cases} \quad (5.2.5)$$

The mean total error from translating the material response according to equation (5.2.5)is

$$J_e(x) = \sum_{k=1}^{K} \sum_{j=1}^{m} [H_j(x, A^{(k)}) - T_j(A^{(k)})] \quad (5.2.6)$$

Although a limit is set on maximum voltages applied on material but in some experiments a penalty term is added to (5.2.6) in order to penalise large voltages used for reference inputs and is given by

$$\rho(x) = \frac{G_{b_1}^2}{G_{max}^2} \quad (5.2.7)$$

The rationale behind this is that high voltages can possibly destroy the material structures that may be unfavourable to the problem and eventually to the process of evolution. Hence, low level of voltages are preferable and limits on voltages and penalisation are double precautionary measures to control the level of reference input voltages. The optimisation aims at minimising the following objective function J

$$\min_x J = J_e(x) + \rho(x) \tag{5.2.8}$$

subject to

$$b_l \leq x \leq b_u \tag{5.2.9}$$

$$G_i \in [G_{min}, G_{max}], i = 1, \cdots, n \tag{5.2.10}$$

$$\beta \in [0, B_{max}] \tag{5.2.11}$$

Also, $b_l = [G_{min}, \cdots B_{min}]^T$ and $b_u = [G_{max}, \cdots B_{max}]^T$. Three different optimisation and evolutionary algorithms are used for solving this problem. The Nelder-Mead with random restart as describe in [73], the Particle Swarm Optimisation algorithm [172] and the Differential Evolution algorithm [162] using the parameters discussed in [170].

5.2.1 Effect of changing connections

The SWCNTs based composites are placed on glass electrode arrays, where the electrodes are arranged in a circular fashion. Each electrode is connected to a micro-controller pin (MCP) on the mbed. The arrangement of input, output and configuration terminals can be changed manually by changing the connections to the pins on mbed micro-controller. In other words, any of the electrodes can be assigned as an input, output, ground or a configuration. A basic arrangement of the electrodes is illustrated in Figure 5.2. In order to study the effect of distance between the input and output electrodes on computation performed by

the material, the experiments are carried out by using four different arrangements of electrodes. The four different arrangements are illustrated diagrammatically in Figure 5.4. Where as, Figure 5.5 illustrates how these configurations are achieved by changing wires on micro-controller manually. For instance, in Figure 5.5(a) Micro-controller pin (MCP1) is used as an input pin, and diagrammatically this arrangement is equivalent to configuration 1 in 5.4. This is the arrangement where the input terminal can be put closest to an output terminal. Similarly, the distance between input and output terminal was increased by changing wires connections manually on micro-controller board and material's computational capabilities are studied for the solution of logic gates problem. SWCNTs/PMMA based composites with varying concentrations are used in this experiment. The detailed description of the results of these experiments is given in section 5.3.

5.3 Results and discussion

This section discusses the results of implementation of threshold concept to evolve Boolean logic gates in SWCNTs/PMMA material composites. It reports results of boolean gates/circuit with varying concentrations of SWCNTs/polymer(PMMA). The detail of these material composites is given in Table 3.1.1. The method is used to implement 2-input, 1-output logic gates (AND, OR) and then more complicated circuits such as half adder and full adder. The section report on results using Nelder-Mead algorithm with periodic restart, Particle swarm algorithm and differential evolution algorithm.

5.3.1 Logic gates/circuits using Nelder-Mead algorithm

During training phase the NM algorithm solved the optimisation problem describe in section 5.2 using K number of training data for different Boolean logic gates/circuits.

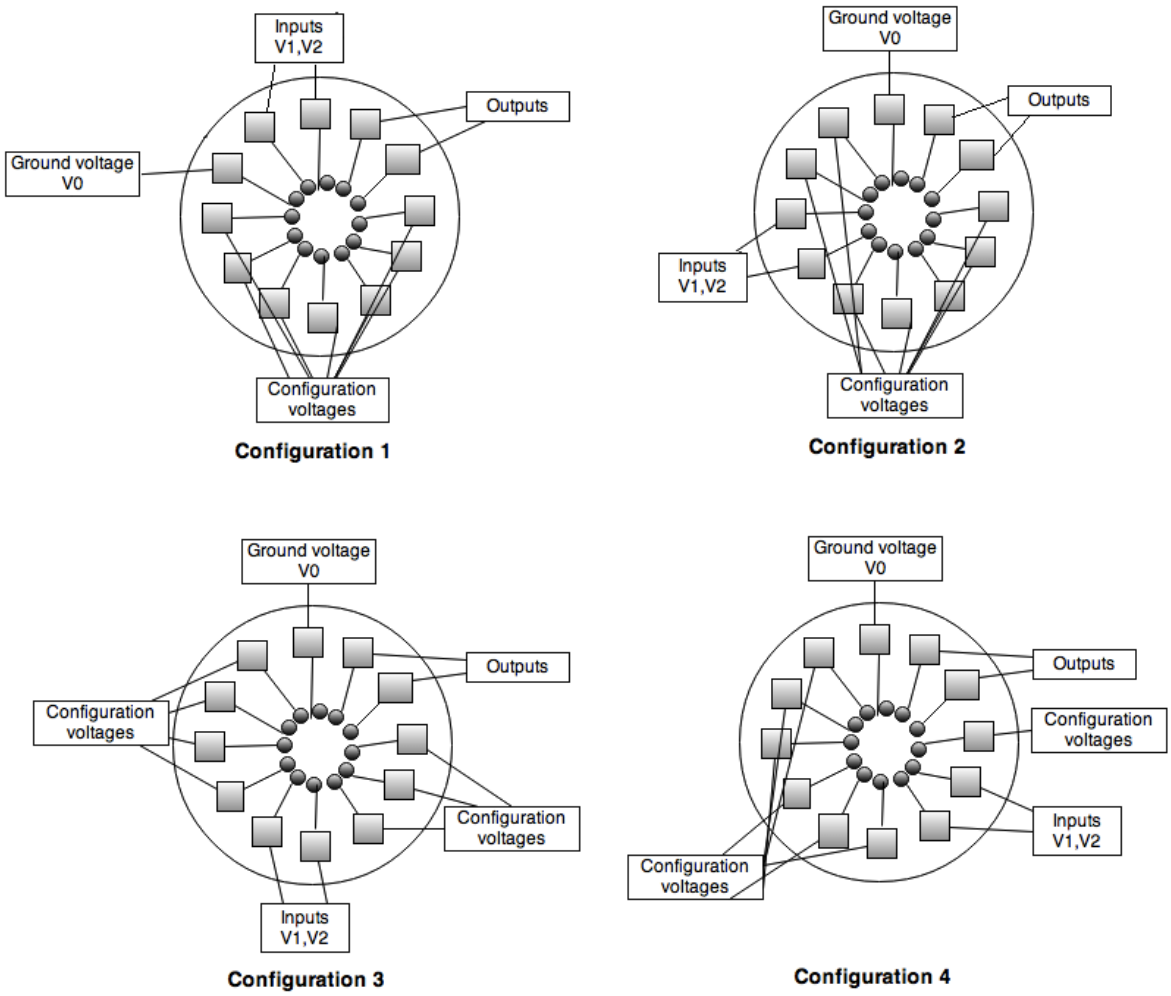


Figure 5.4: Diagrammatic representation of different configurations of micro-controller pins

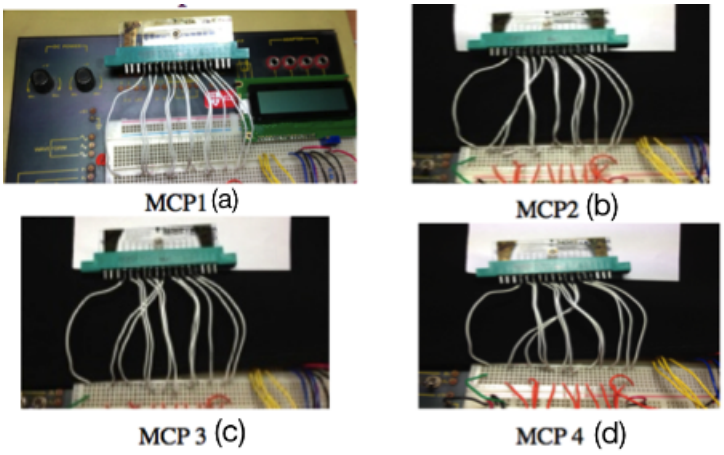


Figure 5.5: Configuration of micro-controller pins as input

The termination criteria is set to be number of maximum iterations or the desired value of objective function value.

Once the training phase is over, the verification phase begins by applying back the optimal configuration voltages and the inputs from K verification input samples of data on to the material. The verification samples are different from training samples and the verification is performed 10 times and each time a mean total error is calculated. Since, the material's internal structure is fixed, the evolution finds the best conductive pathways to solve the particular problem. The following sections discuss the results of different logic gates/circuits in detail.

AND gate

The AND gate is a 2-input, 1-output logic gate that implements the Boolean logic according to the truth table shown in Table 5.1 and are represented symbolically as in Figure 5.6. AND gate requires one threshold to calculate the output of the circuit. If $M(x, A^{(k)})$ is the measured output at the output terminal and θ is the

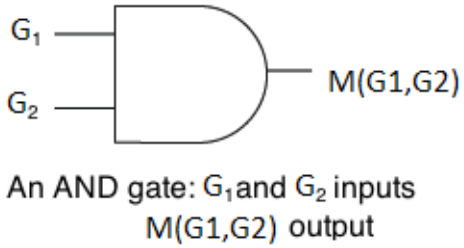


Figure 5.6: The AND gate

G_1	G_2	$M_{AND}(G_1, G_2)$
0	0	0
0	1	0
1	0	0
1	1	1

Table 5.1: Truth table for AND gate

threshold value, then the operation of AND gate is summarised as:

$$Output = \begin{cases} 1 & \text{if} \quad M_1(\mathbf{x}, \mathbf{A^{(k)}}) \geq \theta \\ 0 & \text{otherwise.} \end{cases} \tag{5.3.12}$$

This means that the measured output is equal to logical 1 if it is greater than the threshold value θ, otherwise it will be equal to logical 0. Nelder-Mead and Particle swarm algorithms are used to find the optimal solution.

The optimal solution achieved by the Nelder-Mead algorithm for the material with SWCNTs 1.3 wt% fraction of PMMA (5.0%) is given in Table 5.2. The solution incorporates the value of G_{b_0} which is used to calculate logical 1 at the input, β the scaling factor, which is multiplied by the the G_{b_1} to calculate the threshold value, i.e. $\theta = G_{b_1} \times \beta = 0.51G$ and the 7 configuration voltages

The output measurements at the output terminal after the optimisation found

β	1.655
G_{b_1}	0.31
Configuration voltage values	$G_1 = 0.808,\ G_2 = 1.17,\ G_3 = 0.10,\ G_4 = 0.84$
	$G_5 = 0.64,\ G_6 = 0.31,\ G_7 = 0.88$

Table 5.2: Optimal solution for AND gate, (SWCNTs 1.3wt% fraction of PMMA (5.0%))

a solution are shown in Figure 5.7. The x-axis represents the order over time two binary inputs applied at input terminals. Where as, the y-axis depicts the voltage level measured at the output location. The solid red line running through the figure represents the threshold $\theta = 0.51$. It can be seen in Figure 5.7 that the threshold value to calculate logical 1 is kept at a higher level for binary input pair $(1, 1)$. The output measurements for binary input pair $(0, 0)$ is $0.136\,V$, for $(0, 1)$ is $0.47\,V$, for $(1, 0)$ is $0.278\,V$ and for $(1, 1)$ is $0.57\,V$, which is above the threshold value, found by the optimisation. The spread of outputs along with threshold value is a result of optimisation process and the material's internal properties which are result of interconnected complex conductive network of SWCNTs.

The different material composites were tested for training as logical AND gate, The results are presented in Table 5.3. Most of material composites with different SWCNTs concentrations were successfully trained as an AND gate. However, very low concentrations of SWCNTS i.e. 0.05 wt % of PMMA and 0.02 wt % were not successfully trained, the minimum obtained for these composites was high and the verification accuracy was equal to 50%. The material with 0.1% SWNCTs was successfully trained in a much lower number of iterations as compare to other concentrations of SWCNTs/PMMA.

The results of various materials with varying concentrations of SWCNTs and PMMA are given in Table 5.3. The fitness values include the penalty term added

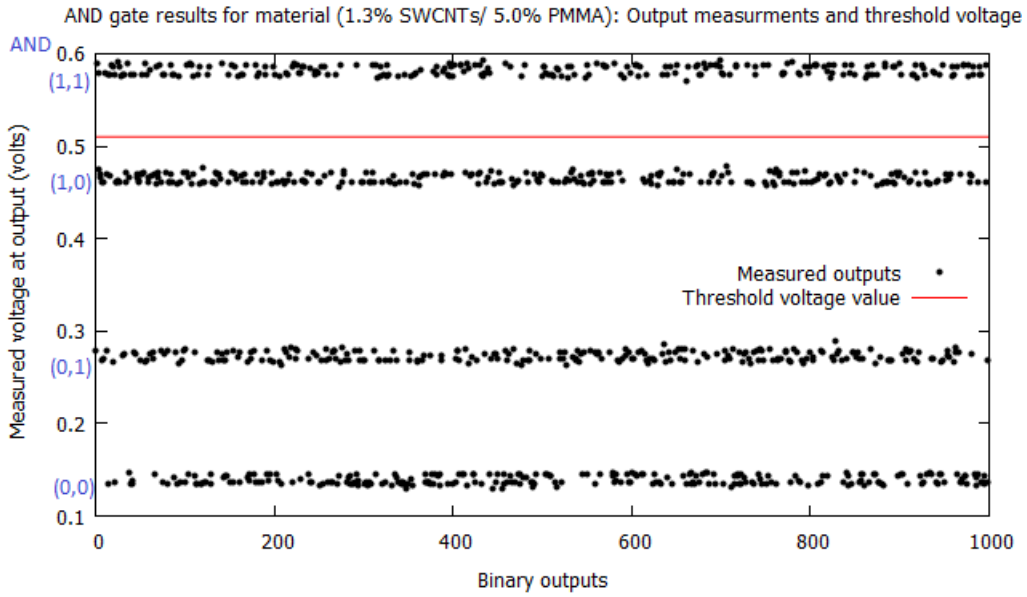


Figure 5.7: Output voltage measured for random binary inputs for AND gate: outputs spread across a threshold value (red line), that is kept high to measure an output when both inputs are high (1,1)

AND gate results with varying SWCNTs/PMMA concentrations					
%SWCNTs	% PMMA	$\oint_{min}$	$\oint_{max}$	$\oint_{avg}$	FE_{avg}
0.012	16.8	6.00	22.00	17.71	1305
0.05	24.5	12.00	23.00	16.00	1056
0.1	14.8	0.02	0.05	0.04	82
0.23	16.5	0.005	0.02	0.01	73
0.71	4.9	0.02	0.03	0.04	52
1.3	5.0	0.01	0.04	0.05	150
407	5.2	0.00	0.004	0.002	123

Table 5.3: AND gate results with varying SWCNTs/PMMA concentrations

to penalise large input voltage values i.e. x_{b_1}. The minimum, maximum and average fitness values from 5 experiments for each material sample is listed as $\oint_{min}$, $\oint_{max}$, $\oint_{avg}$, where FE_{avg} represents the average of function evaluations from these experiments.

5.3.2 OR gate

The OR gate is a 2-input, 1-output logic gate that implements the Boolean logic according to the truth table shown in Table 5.4 and are represented symbolically as in Figure 5.8. This logic gate also requires one threshold to calculate the output of the circuit. However, the threshold require to calculate a boolean 1 is kept very low. If $M(x, A^{(k)})$ is the measured output at the output terminal and θ is the threshold value, then the operation of OR gate is summarised as:

$$Output = \begin{cases} 1 & \text{if} \quad M_1(\mathbf{x}, \mathbf{A^{(k)}}) \geq \theta \\ 0 & \text{otherwise.} \end{cases} \tag{5.3.13}$$

The equation implies that the measured output is equal to logical 1 if it is

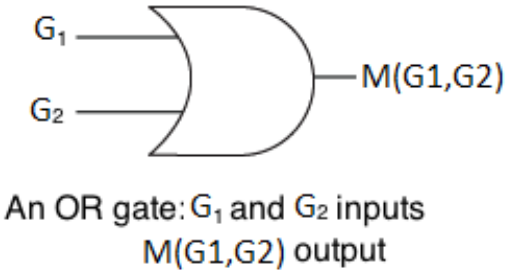


Figure 5.8: The OR gate

G_1	G_2	$M_{OR}(G_1, G_2)$
0	0	0
0	1	1
1	0	1
1	1	1

Table 5.4: Truth table for OR gate

greater than the threshold value θ, otherwise it will be equal to logical 0. The PSO and Nelder-Mead algorithm are used for training the material. The optimal

β	1.66
G_{b_1}	0.31
Configuration voltage values	$G_1 = 0.59, G_2 = 2.47, G_3 = 3.6$
	$G_4 = 1.42, G_5 = 1.00, G_6 = 0.42, G_7 = 0.31$

Table 5.5: Optimal solution for OR gate, (SWCNTs 1.3 wt% fraction of PMMA(5.0%))

solution achieved by the optimisation algorithm for the material with SWCNTs $1.3 wt\%$ fraction of PMMA (5.0%) is given in Table 5.5. The solution provides the value of voltage G_{b_1} which is used to calculate logical 1 at the input and the scaling

factor $\beta = 1.66$, both of these values are used to calculate the threshold value for classifying the output as logical 0 or 1. The output measurements collected at the terminal after optimisation process finished are shown in Figure 5.9. The threshold value is represented by the horizontal red line across the graph. It can be seen that the threshold value is kept at lower level in compression to output measurements to calculate a logical 1 for binary input pairs $(0, 1), (1, 0)$ and $(1, 1)$.

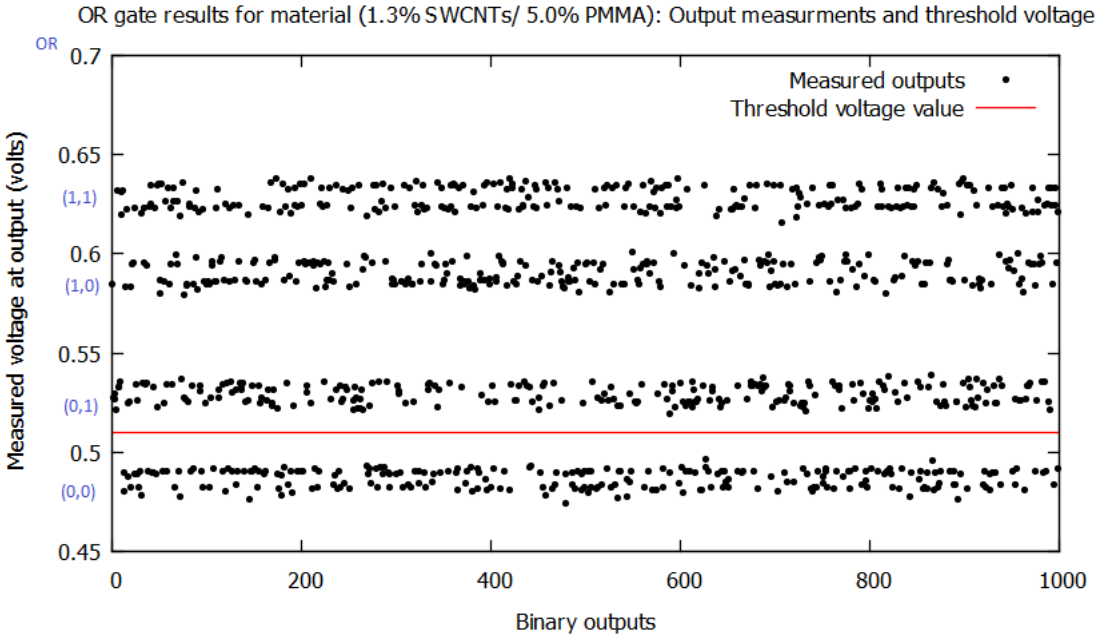


Figure 5.9: Output voltage measured for random binary inputs for OR gate: outputs spread across a threshold value (red line), that is kept low to measure a high output when any of input is high i.e. $(0, 1)(1, 0)(1, 1)$

Similar to AND gate training, the materials with varying SWCNTs/PMMA concentrations are used to train as an OR gate. The results of these experiments using Nelder-Mead algorithm are summarised in Table 5.6. The fitness values incorporate the penalty term, used to penalise large input voltages. It can be

Half Adder results with varying SWCNTs/PMMA concentrations					
%SWCNTs	% PMMA	$\oint_{min}$	$\oint_{max}$	$\oint_{avg}$	FE_{avg}
0.012	16.8	32.00	52.00	47	1314
0.05	24.5	72.00	82.4	51.6	1055
0.1	14.8	0.02	0.05	0.03	29
0.23	16.5	0.00	0.002	0.00	23
0.71	4.9	0.02	0.03	0.03	24
1.3	5.0	0.01	0.04	0.05	22
407	5.2	0.00	0.09	0.05	123

Table 5.6: OR gate results with varying SWCNTs/PMMA concentrations

seen in the table that the most of material composites successfully achieved the local minimum. However, similar to AND gate results the materials with very low SWCNTs concentrations i.e. 0.05% and 0.012% were not successfully trained, which in the end resulted in 50% accuracy during verification phase.

In contrast to AND gate results, presented in Table 5.3, the algorithm achieved the desired minimum for an OR gate in significantly less number of iterations. For instance, it took 130 iterations for the material with SWCNTs $1.3\,wt\%$ fraction of PMMA (5.0%) to train as an AND gate, where as, the OR gate was achieved for the same material in only 9 iterations. The reason is the due to the complexity of the problem.

The results of various materials with varying concentrations of SWCNTs and PMMA are given in Table 5.6. The fitness values include the penalty term added to penalise large input voltage values i.e. x_{b_1}. The minimum, maximum and average fitness values from 5 experiments for each material sample is listed as $\oint_{min}$, $\oint_{max}$, $\oint_{avg}$ respectively, whereas, FE_{avg} represents the average of function evaluations from these 5 experiments.

G_1	G_2	$M_{(XOR)}(G_1, G_2)$	$M_{(AND)}(G_1, G_2)$
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

Table 5.7: Truth table for half adder circuit

5.3.3 Half adder

The half adder is a 2-input, 2-output logic circuit that implements the Boolean logic according to the truth table shown in Table 5.7 and is represented symbolically as in Figure 5.10.

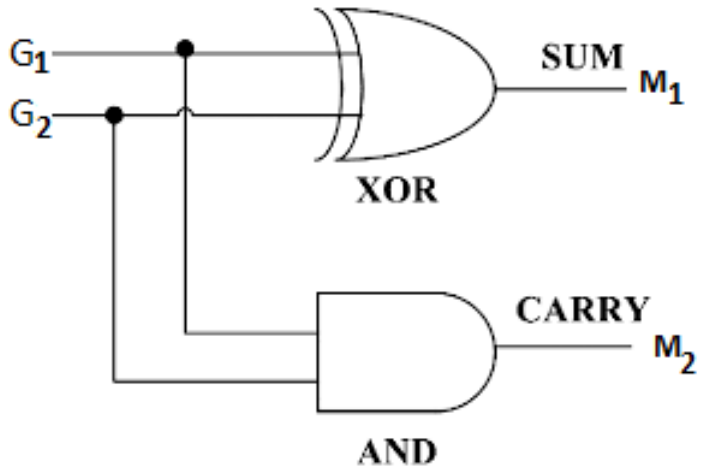


Figure 5.10: The half adder circuit

It is a more complicated circuit than simple logic gates as it implements two boolean gates, i.e. an AND gate and a XOR gate to implement its Boolean function. The XOR gate imposes a requirement to translate the output voltage $M_{(XOR)}$ into a 0 for the binary input $(1, 1)$, where as, the output $M_{(AND)}$ requires one threshold which corresponds to an AND gate.

Optimal solution for half adder circuit $AB, A \oplus B)$			
$\beta = 0.58$			
$G_{b_1} = 1.214$			
$G_1 = 0.25$	$G_2 = 0.031$	$G_3 = 0.637$	
$G_4 = 0.84$	$G_5 = 1.76$	$G_6 = 1.34$	$G_7 = 0.92$

Table 5.8: Optimal solution for half adder circuit $AB, A \oplus B)$

The carry operation for an AND gate is summarised as:

$$Output\ Carry = \begin{cases} 1 & \text{if} \quad M_{(AND)}(\mathbf{x}, \mathbf{G}) \geq \theta_1 \\ 0 & \text{otherwise.} \end{cases} \tag{5.3.14}$$

The equation 7.3.8 implies that $M_{(AND)}$ should be greater that θ_1 to calculate a logical 1, otherwise it will be 0. Where as, the sum operation is as follows:

$$Output\ Sum = \begin{cases} 1 & \text{if} \quad \theta_2 \leq M_{(XOR)}(\mathbf{x}, \mathbf{G}) \leq \theta_3 \\ 0 & \text{otherwise.} \end{cases} \tag{5.3.15}$$

The equation 7.3.9 implies that the measured output $M_{(XOR)}$ will be interpreted as logical 1 when its value lies between two thresholds, θ_2 and θ_3. If $M_{(XOR)}$ lies below θ_2 and above θ_3 it will be interpreted as logical 0. The results of training the material (SWCNTs 0.1 wt% fraction of PMMA(14.8%) as a half adder circuit using equation 7.3.8 and equation 7.3.9 are shown in Table 5.8. The solution includes the value of scaling factor β, the value of voltage used as logical 1, and 7 configuration voltage values. The three thresholds (θ_1,θ_2,θ_3) used for calculating carry and sum are function of three configuration voltages G_1, G_2 and G_3 respectively, which are then multiplied with the scaling factor β. Hence, the threshold value θ_1 to calculate carry output is 0.25 and the two thresholds θ_2 and θ_3 for calculation of sum outputs are 0.018 and 0.37 respectively. The output measurements on two electrodes are shown in Figure 5.11. The three thresholds

OR gate results with varying SWCNTs/PMMA concentrations					
%SWCNTs	% PMMA	$\oint_{min}$	$\oint_{max}$	$\oint_{avg}$	FE_{avg}
0.012	16.8	22.00	23.00	22.0	1314
0.05	24.5	21.00	24.00	22.00	1055
0.1	14.8	0.00	0.02	0.03	182
0.23	16.5	22.00	25.00	23.00	1354
0.71	4.9	5.0	8.00	6.10	1296
1.3	5.0	11.0	15.55	12.5	1050
407	5.2	17.00	20.0	18.4	1120

Table 5.9: Half adder circuit results with varying SWCNTs/PMMA concentrations

can be seen very clearly. The threshold to calculate logical 1 for AND gate inputs $(1, 1)$ is placed at a distinctive location as compare to calculate logical 0 for the rest of possible inputs. On the contrary, for XOR outputs all possible inputs and the two thresholds to calculate XOR outputs are also clearly distinguishable.

The experiments to solve half adder problem are repeated with varying concentrations of SWCNTs and PMMA. The results of these experiments are generalised in Table 5.9. The results presented in this table are from 5 different experiments, where minimum, maximum and average fitness values are represented by $\oint_{min}$, $\oint_{max}$, $\oint_{avg}$, respectively and average number of function evaluations are represented by FE_{avg}. In similarity to AND and OR gates experiments, the fitness value incorporates the penalty term used to penalise large input values. In contrast to simple logic gates, AND and OR, only one material with SWCNTs concentration 0.1 $wt\%$ fraction of PMMA (14.8%), was trained as a half adder circuit.

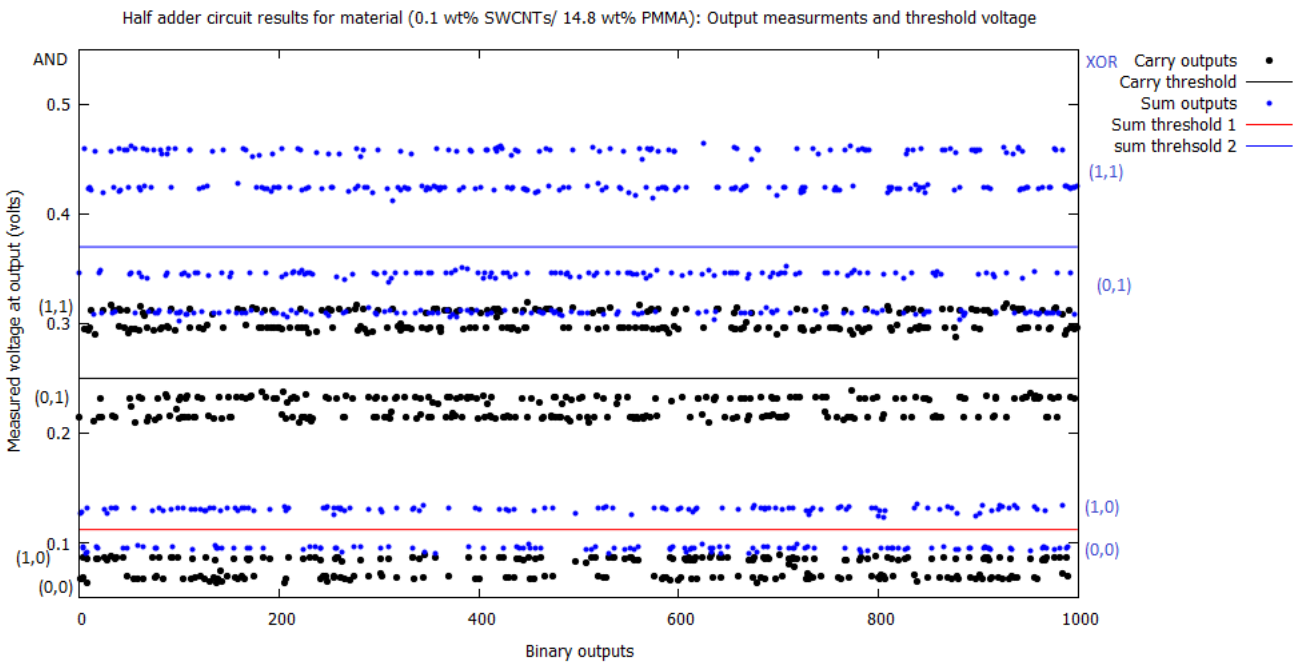


Figure 5.11: Output voltage measured for random binary inputs for half adder: outputs spread across three thresholds.

5.3.4 Logic circuits using Differential Evolution algorithm

The method describe in above sections to train the SWCNTs based materials to behave as Logic gates was successful. However, the Nelder-Mead algorithm was able to solve basic logic gates and circuits but it failed to solve complex logic circuits in some cases. In separate experiments, the DE algorithm was used to train the material composites as Threshold logic gates and Threshold logic circuits. In contrast to Nelder-Mead algorithm, DE algorithm solved all of the considered cases of Threshold logic circuits. The results of these experiments with 0.53% SWCNTs concentrations with fixed PMMA are discussed in the following sections. The values of $G_{min} = -4$ and $G_{max} = +4$ are used, unless stated other wise. Also, no penalty term is added to the objective function J as used in equation and instead it is calculated using (5.2.6). In other words least square

optimisation problem is as follows

$$\min_{x} J = \sum_{k=1}^{K} \sum_{j=1}^{m} [H_j(x, A^{(k)}) - T_j(A^{(k)})] \tag{5.3.16}$$

Logic circuit $(AB, A + B)$

The first circuit considered is a 2-input, 2-output logic circuit that is represented symbolically as in Figure 5.12. It can be seen that the circuit consist of an AND and an OR gate. The principle on which material operates is shown in Figure 5.3. The two connections are charged with input voltages G_1 and G_2, which also take

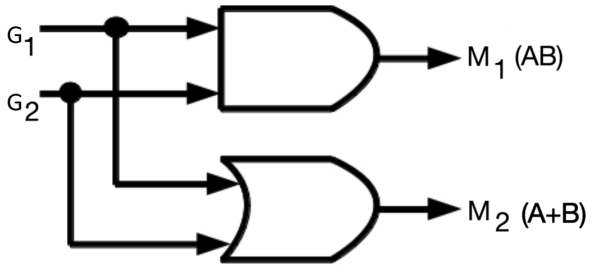


Figure 5.12: The Logic circuit (AB,A+B)

the values of logical 1 i.e. G_{B_0} and logical 0 i.e. G_{B_1} respectively. The two outputs are measured at two distinct locations on the material i.e. M_1 and M_2. The circuit considered requires a single threshold to measure the outcome. The solution found by optimisation for the circuit is shown in Table 5.10. It includes the value of G_{B_0} which is used as logical 0 and G_{B_1} which is used as logical 1. The value of G_{B_1} is also used as threshold θ i.e. $\theta = f(x) = G_{B_1} = 3.279$ V. The output measurements at two different locations are shown in Figure 5.13. The x-axis represents the order over time the binary inputs (G_1, G_2) were applied and y-axis represents the voltage measured at two output locations. The left side represents the binary inputs corresponding to the AND gate and right side

Optimal solution for the circuit: (AB, A+B) for the material (0.53% SWCNTs/PMMA	
$G_{B_0} = 3.279$	$G_{B_1} = 11.200$
$G_2 = 2.447$	$G_3 = 3.200$
$G_4 = 11.200$	$G_5 = 5.054$
$G_6 = 7.588$	$G_7 = 9.556$

Table 5.10: Optimal solution for the circuit: (AB, A+B)

represents the binary input corresponding to the OR gate. The solid line running through the diagram represents the threshold $\theta = 3.279$ V. It can be seen in Figure 5.13 that the voltages measured at two different locations take specific band and are different for each output i.e. M_1 and M_2. For example, for the first test binary inputs $(0, 0)$, the measurement for AND gate us 3.055 V and for OR gate us 3.248 V. Both output measurements are less than the threshold value $\theta = 3.279$, the corresponding output for both is a 0. The output measurements for binary input pair $(0, 1)$ are 3.138 V for AND and 3.338 V for the OR gate. The value of AND gate is below the threshold value, where as, the value of OR gate is above the threshold value, hence a $(0, 1)$ output is registered. For $(1, 1)$ binary inputs the output for an AND gate is 3.318 V and the output for an OR gate is 3.442 V, both the values are larger then the threshold value, hence an outcome of $(1, 1)$ is registered. Both the outputs of the circuit were consistent for different input pairs and stayed within specific bands. For instance, it can be seen in Figure 5.13 that for an AND gate the output voltages for the input pairs $(0, 1)$ and $(1, 0)$ are at a good distance, as compare to an OR gate, where these values are closer. This kind of spreading and the threshold values are result of optimisation process and material's physical properties, which in this case are due to conductive network of carbon nanotubes).

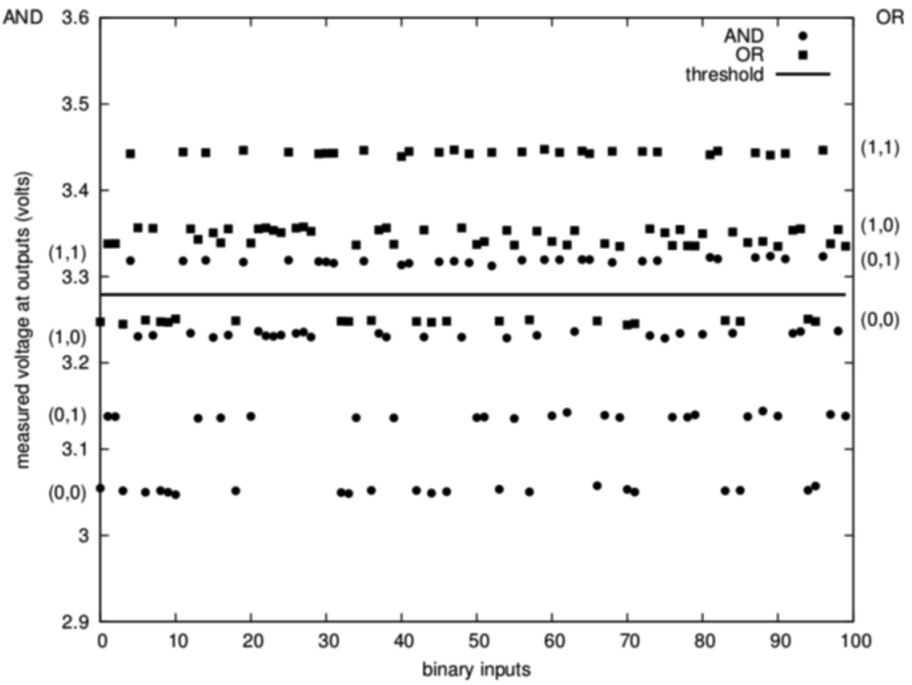


Figure 5.13: Output voltages measured for the random binary input pairs for the circuit (AB,A+B)

Logic circuit $(AB + BC)$

The logic circuit $(AB + BC)$ is a three input and a single output logic circuit that requires a single threshold to calculate the outcome of the circuit. In this case, the threshold was set as a function of $G_1\beta$ and the values of $B_{max} = 4$ and $G_{max} = 11.2V$ were set. The optimal solution found by DE is given is Table 5.11. The corresponding output measurements for random binary input triplets when the optimal solution is applied are shown in Figure 5.15. The circuit is difficult in terms of optimisation as the outcome of input triplet $(1, 0, 1)$ with two 1's is a 0. Where as, the outcome of the inputs triplets $(0, 1, 1)$ and $(1, 1, 0)$, also with two 1's result to a 1. Therefore, the solution must be able to differentiate between these input signals. Optimisation found the solution by differentiating the $(1, 0, 1)$ from the other two input triplets by a very small margin. The voltage

Optimal solution for $(AB + BC)$ circuit: Material 0.53% SWCNTs/PMMA		
$G_1 = 8.859$	$G_2 = 3.693$	$G_3 = 0.000$
$G_4 = 0.172$	$G_5 = 7.462$	$G_6 = 7.577$
$G_{b_1} = 8.672$	$\beta = 0.406$	$\theta = 3.597$

Table 5.11: Optimal solution for $(AB + BC)$ circuit in Volts except for β

value recorded for the $(1, 1, 0)$ is 3.650, for $(0, 1, 1)$ is 3.598 and for $(1, 0, 1)$ is 3.571. Both the Nelder-Mead and DE were able to find the solution, however the results are presented for DE algorithm.

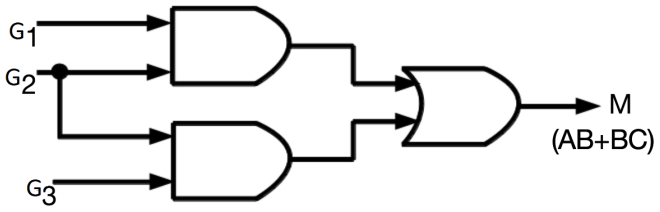


Figure 5.14: The Logic circuit $(AB + BC)$

Logic circuit: Full adder

The full adder is a 3-input, 2-output logic circuit, that implements the Boolean logic according to truth table given in Table 5.12. It can be seen in the table that the circuit requires a single threshold to calculate carry. In this case the threshold for carry is calculate as follows.

$$\theta_c = G_4 \tag{5.3.17}$$

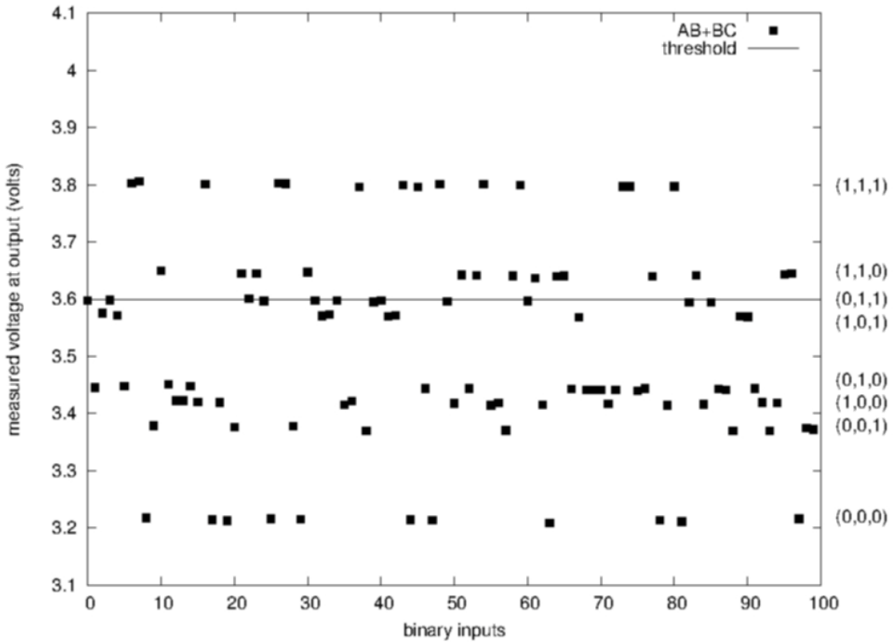


Figure 5.15: Output voltages measured for the random binary input triplets for the circuit $(AB + BC)$

The threshold function to map the measured voltages to binary outputs when binary triplet $A = (x_1, x_2, x_3)$ is given as input is as follows:

$$H_1(A) = \begin{cases} 1 & \text{if} \quad M_1 \geq \theta_c \\ 0 & \text{otherwise.} \end{cases} \tag{5.3.18}$$

The calculation of sum output of the circuit is different as compare to carry. The outcome of sum is a 1, when there is single 1 in input triplet, the outcome is a 0 when there are two 1s in the input triplet and the outcome is again 1, when there are three 1s in the input triplet. Hence, it requires three thresholds to calculate the outcome of sum i.e. to classify the measured output voltage. In current case, the three thresholds to calculate the outputs for sum are as follows:

$$\theta_{s,1} = G_1\beta \tag{5.3.19}$$

$$\theta_{s,2} = G_2 \tag{5.3.20}$$

Inputs			Outputs	
A	B	C	$AB + C(A \oplus B)$ (carry)	$A \oplus B \oplus C$ (sum)
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

Table 5.12: Full adder's truth table

$$\theta_{s,2} = G_3 \tag{5.3.21}$$

The threshold mapping for measured output voltages to binary outputs when binary input triplets are given is as follows:

$$H_2(A) = \begin{cases} 1 & \text{if} \quad M_1 \geq \theta_c \\ 0 & \text{if} \quad \theta_{s,2} \leq M_2 < \theta_{s,1} \\ 1 & \text{if} \quad \theta_{s,3} \leq M_2 < \theta_{s,2} \\ 0 & \text{if} \quad M_1 \leq \theta_{s,3} \end{cases} \tag{5.3.22}$$

In case of full adder circuit the Nelder-Mead algorithm failed to find the solution, however DE was successfully found the solution and is given in Table 5.13.

Figure 5.17, shows the spreading of measured output voltages when the respective binary input triplets are applied. It can be seen that the thresholds are well separated for the input triplets. There is significant variance for carry and sum outputs. The carry output has less variance as it requires one threshold to

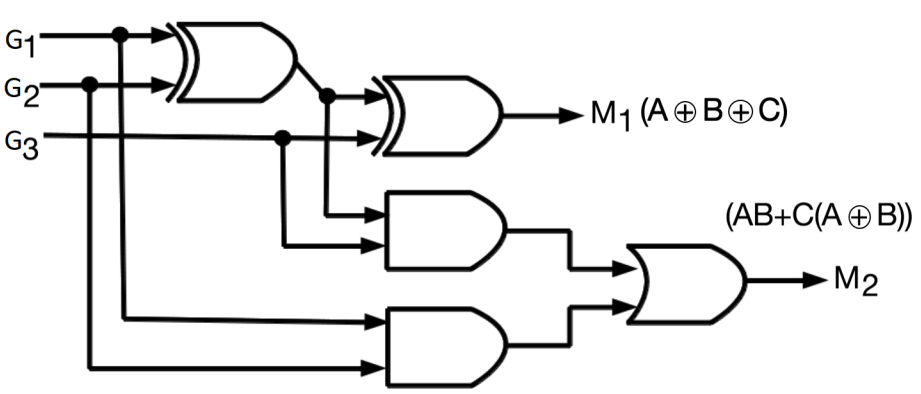


Figure 5.16: The Full adder

Optimal solution for full adder circuit: Material 0.53% SWCNTs/PMMA		
$G_1 = 2.447$	$G_2 = 3.710$	$G_3 = 3.161$
$G_4 = 2.968$	$G_5 = 10.718$	$G_6 = 5.060$
$G_{b_1} = 7.717$	$\beta = 1.7$	$\theta_c = 2.968$
$\theta_{s,1} = 4.160$	$\theta_{s,2} = 3.710$	$\theta_{s,3} = 3.161$

Table 5.13: Optimal solution for full adder circuit in Volts except for β

differentiate the inputs that has at least two 1's, where as the outputs for sum are widely spread in order to differentiate the inputs with one 1's from two and three 1's.

5.3.5 Effect of changing connections

The micro-controller pins are arranged in a circular manner on glass slide as shown in Figure 3.7. The detailed arrangement of these pins as inputs and outputs, used for the current the experiments is illustrated in Figure 5.4. Figure 5.5, illustrates how this arrangement can be altered. For the first experiment,

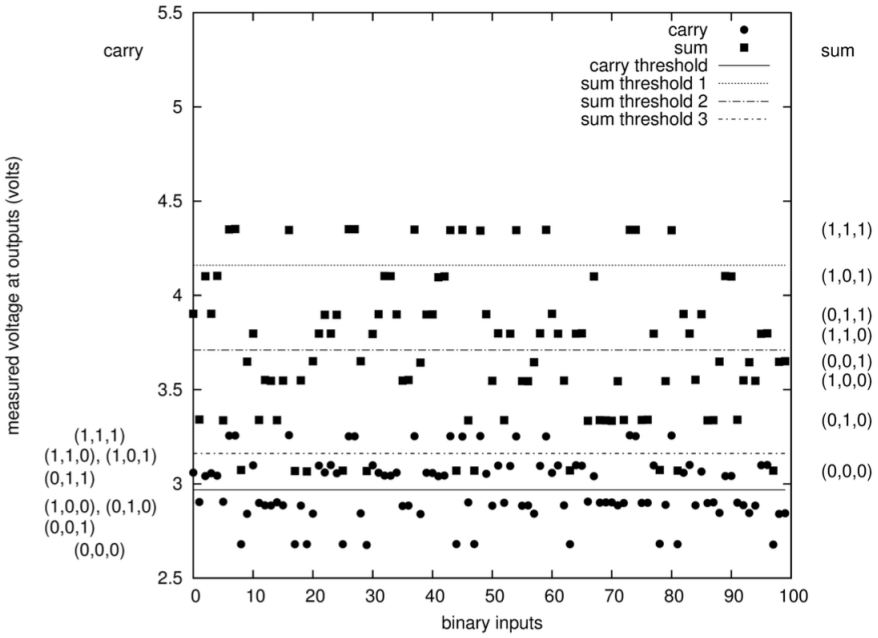


Figure 5.17: Output voltages measured for the random binary input triplets for the full adder circuit

micro-controller pin 1 (Figure 5.4: MCP1) was used as an input terminal. In this way, input pin is closest to the output pin on the glass slide, as shown in Figure 5.4, (configuration 1). Then MCP2, MCP3 and MCP4 were used as input application points respectively. This way the distance between the input pins and output pins on glass slides was increased. The distance was maximum when MCP4 is used as input pin. The results of these experiments are presented in Table 5.14, which shows the success of convergence to a minimum when the pin distance is varied. The best convergence was achieved when the input pins were closest to output pins, all other pin configurations failed to achieved the desired minimum during total number of iterations. The results of these experiments clearly highlighted that distance between input and output pins has a direct effect on convergence. This is due to the fact that the concentration of SWCNTs can be different on each contact electrode. It can also be concluded that if the pin selection is added as

a parameter to the optimisation process, optimisation may be able to find better interconnected conductive pathways which may lead to better solutions for the chosen computational problem.

Wires	Fitness value achieved	Number of Function evaluations	Number of Iterations
MCP1	0.062000	117	69
MCP2	No convergence	1160	901
MCP3	No convergence	996	901
MCP4	No convergence	1104	901

Table 5.14: Fitness values at different micro-controller pins

5.3.6 Particle Swarm algorithm vs Nelder-Mead algorithm

The initial experiments for solving logic gate problem are performed with Nelder-Mead algorithm. Later, Particle swarm algorithm was used to compare its performance with Nelder- Mead algorithm. Particle swarm algorithm with swarm size 16 is used for experiments described in this chapter. Comparison of these two algorithms with two different materials to solve AND and OR gate is given in Table 5.15 and Table 5.16. As shown in both PSO required less number of function evaluations to reach minimum, as compared to Nelder-Mead algorithm.

5.4 Resistors vs SWCNTs?

SWCNTs/ PMMA composites have a random arrangement of carbon nanotubes that act as a network resistors. SWCNTs/ PMMA composites are in dry state and any physical reorganisation of SWCNTs was unlikely after an optimal solution.

No. of function evaluations performed for an AND gate		
Material	Nelder-Mead algorithm	Particle Swarm algorithm
1.3 CNT (wt% fraction of PMMA(5.0))	190	27
0.10 CNT(wt% fraction of PMMA(14.8))	116	68

Table 5.15: Performance of PSO vs Nelder-Mead algorithm for solving an AND

No. of function evaluations performed for an OR gate		
Material	Nelder-Mead algorithm	Particle Swarm algorithm
1.3 CNT (wt% fraction of PMMA(5.0))	9	5
0.10 CNT(wt% fraction of PMMA(14.8))	116	68

Table 5.16: Performance of PSO vs Nelder-Mead algorithm for solving an OR gate

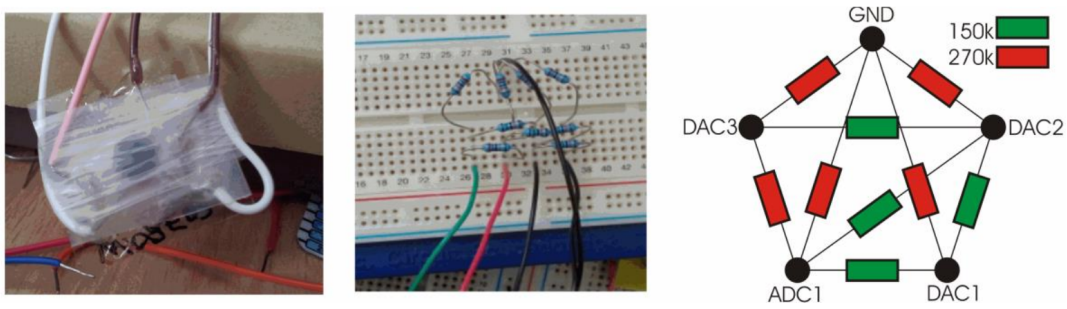


Figure 5.18

However, there was a possibility that the conductivity of SWCNTs based composites would have been altered by the application of voltage signals. After having promising results of training SWCNTs/ PMMA composites as basic logic gates and a half adder circuit, experiments have been performed to verify if the similar effect can be achieved by using the network of resistors. An experiment aimed to find if a non linear response can be obtained from the network of resistors by applying the same optimisation procedure as performed with SWCNTs/ PMMA composites to behave as logic gates (AND and OR). Figure 5.18 shows the experimental set-up that is used to implement this idea. The optimisation was performed using resistor network and it showed a linear dependence on input voltages and was not successful to train as a logic gate (AND and OR). Where as, the same experiment was implemented with SWCNTs/ PMMA composites and the output response showed a non-linear dependence on the input voltages. Hence, it was concluded that application of optimised input signals on to SWCNTs/ PMMA network produced the logic gate behaviour and the non-linear response from the material composites is key to successful implementation of the optimisation process.

These experiments are different from SPICE simulation modelling of the SWCNT based material, where the material's model is produced by observing the material's behaviour under different inputs [120]. The model produced is a circuit consisting

of common elements like capacitors, inductors and resistors. It was shown that the model thus produced can solve a travelling salesman problem, however it cannot be trained to solve a NOT gate problem. Its is the material's non-linear behaviour which is required to solve a computational problem.

5.5 Stability of results

In order to observe the stability of the results, different approaches were used to observe the behaviour of the material. This is important, so that the same optimal solution can be reapplied to the same material composites for the solution of a computational problem. The following approaches were used to test the stability of the results:

- The micro-controller was switched off for some time and then switched on again and the optimal signals are applied on to material composites.
- Some random signals were applied on to the material composites and then the optimal solution was applied.
- The optimal solution was applied after some days and then two and three months duration.

The outcome of these tests showed that the optimal solutions obtained are reproducible over time, and as such, no significant degradation was observed.

5.6 Conclusions

This chapter has presented the results that SWCNTs/ PMMA can be trained to implement threshold logic gates using the principal of evolution in materio and material's conductive property. In order to train the material an optimisation

problem as been formulated with continuous and binary constraints. The procedure was successful to evolve logic gates and circuits by selectively applying configuration voltages on to different material locations. A one-to-one mapping has been established between inputs and outputs. The experiments were conducted with the different SWCNTs/ PMMA composites using the Nelder-Mead algorithm, Particle Swarm Optimisation algorithm and Differential Evolution algorithm. It was the first time that the EIM concept is used to train the SWCNTs/ PMMA composites as threshold logic gates and threshold logic circuits. Previously, Liquid crystal based material was used for logic gates using genetic algorithms [171].

Different SWCNTs/ PMMA based composites with varying concentration of SWCNTs and polymer (PMMA) are explored for their computational capabilities. A link between SWCNTs concentration and ability to optimise for the logic gate/circuit was observed. The composites with very low concentrations of SWCNTs, i.e. (SWCNTs 0.012%) and (SWCNTs 0.05%) failed to converge for OR, AND and half adder circuit. Where as, the composite with SWCNTs 0.13% concentration was successfully trained for AND, OR as well as the half adder circuit. Hence, a conclusion can be drawn that a very specific SWCNTs concentration is required for simple and complex logic circuit.

Table 5.17 describes the effort with which the logical gates have been evolved in this SWCNTs/ PMMA composite. The results show that as the complexity of the problem is increased, the number of function evaluations required to train the materials as a half adder circuit also increased as compare to an AND and OR gate.

It is can also be concluded that although the networks of SWCNTs in the polymer are not mobile to allow any physical manipulation of electrical pathways, the combination of SWCNTs and PMMA (polymer) produced a complex physical architecture that can be used to train the material to a solve basic computational

Gate	Fitness value achieved	No. of Function evaluations	No. of iterations
AND	0	116	56
OR	0	113	56
half adder	0	174	110

Table 5.17: Results showing optimal solution for AND, OR and half adder circuit with the 1.3 CNT wt% fraction of PMMA (5.0)

problem.

There were no physical or structural changes within the composites after the optimisation process was completed and as a result, the composites were very stable in terms of their internal structure.

It is also important to make the distinction that the physical evolution is not happening inside the material composites as the materials are fixed. Instead, the optimisation algorithms find a set of optimal voltage levels to produce a desired response. In other words, it is the electrical conductance that is changed and manipulated in order to obtain responses whose interpretation using the pre-selected threshold(s) can be used for simple logic gates and logic circuits.

In separate experiments, two different optimisation algorithms, i.e Nelder-Mead algorithm and Particle Swarm algorithm, were compared for their performance to solve basic logic gates. PSO was quick to converge to the solution as compare to Nelder- Mead algorithm. Later, in separate experiments, differential evolution algorithm and Nelder-Mead algorithm were used to train the two different concentrations of SWCNTs/ PMMA i.e 0.53% and 0.23% as complex threshold-based logic circuits. The Nelder-Mead algorithm failed to address more complex Boolean functions, where as DE was consistently successful to solve considered complex logic circuits. Similarly, out of two concentrations of material composites used, only the concentration of 0.53% SWCNTs/ PMMA succeeded in behaving as a logic circuit for all the

considered cases. The results of this study have been published in [173].

As a result of these experiments, a three-dimensional research space has been highlighted that consist of (a) computational material, (b) computational task, (c) optimisation algorithm used to train the material. The exploration of these research spaces can provide insights into new paradigms of computation.

It can be concluded from these experiment that SWCNTs/ PMMA composites are suited for approximating simple digital logic. However, the clear link between SWCNTs concentrations and ability to train them as logic gates cannot be established from these experiments. This is due to the fact that every material composite used, have different concentration of SWCNTs as well as the polymer (PMMA), both of which effect material's conductive properties. In order to explore this observation further, the next chapter investigates the another set SWCNTs based composites for the solution of logic gates problem. These experiments helped in identifying the link between SWCNTs concentration and its effect on the solution of a simple computational task.

Chapter 6

Studying the correlation between SWCNTs concentration and computing

This chapter discusses the results of studying the correlation between the concentration of SWCNTs and solving a simple computational problem of threshold based logic gates/circuits.

6.1 Introduction

The outline for an suitable materials for use with EIM is presented in [54] and is described in Chapter 3, section 3.1. It was observed in experiments describe in Chapter 5, that the suitable materials should have some non-linearity in their current versus voltage response because otherwise, a network of standard resistors and capacitors could be enough [174]. Also, one material system should be benchmarked against another and it therefore, requires a simple method or experiment where different material systems can be compared to each other for computational

problems.

The previous chapter reported the results of computing with different SWCNTs concentrations with different percentage of polymer (PMMA). However, a clear indication of SWCNTs concentration and computational capability cannot be outlined as the polymer concentration was different for every sample. The concentration of polymer as well as SWCNTs affects the electrical behaviour of the composite and the experiments were intended to use electrical signals to manipulate the material composite properties. Hence, it was important to study the correlation between SWCNTs concentrations and their computational capabilities.

In order to do so, the next batch of material composites was used produced that used varied SWCNTs concentration in a fixed polymer called Poly Butyl Methacrylate (PBMA). Different concentrations of SWCNTs are used with a fixed polymer concentration, in order to compare the conductivities of different samples and its effect on their computational capabilities. The material composites are then tested for solving simple logic gates as well as a complex logic circuit Half adder, using the threshold concept described in Chapter 4, Section 4.5.1. The conclusions are drawn on a correlation between the concentrations of SWCNTs and their effect on solving computational problems.

The concentration of SWCNTs affect the material's morphological and electrical behaviour therefore, material viscosity and electrical characteristics are discussed in following section. The later sections present the detailed analysis of the results of these experiments.

6.2 Viscosity and electrical characteristics of SWCNTs/PBMA composites

The detailed description of SWCNTs/PBMA preparation is outlined in chapter 3, section 3.1.2. In order to produce stable substrates, the composites are deposited using spin-coating technique, discussed in chapter 3, section 3.1.2. This provides more even coverage over the electrodes and was more stable than drop-cast material composites (used in experiments described in Chapter 4). An electron micrograph in Figure 3.6 shows well dispersed bundles of SWCNTs over the electrodes. The 4×4 grid electrode slides is used for these composites, the detailed description of these electrode slides is also given in chapter 3, section 3.2. The mask of these electrodes is shown in Figure 3.8.

The concentration of SWCNTs in polymer affects the viscosity and electrical properties of the composites. Before studying the material composites for their computational capabilities the viscosity and electrical properties were studied in detail as they may have an impact computational capabilities of the materials system. The detail description of these two observations is given in following sections.

6.2.1 Viscosity measurements

[3] In general, the relationship among viscosity, sheer stress and shear rate for liquid material is express as

$$\tau = \eta\gamma \tag{6.2.1}$$

However, SWCNTs/polymer composites follow a non-Newtonian liquid behaviour of shear thinning, which means when the viscosity of the material is decreased and rate of shear is increased. A common power law describes the shear thinning

as;

$$\tau = A_c \gamma^n \tag{6.2.2}$$

Where A_c is the consistency parameter and n is the power law index. At $n = 1$, Newtonian behaviour is observed where the shear stress is proportional to shear rate. The comparison of shear stress and the shear rate is expressed solid lines for different SWCNTs/ PBMA composites using equation 6.2.2 in plotted in Figure 6.1. Along with the experimental data in Table 6.1 and the plotted lines in Figure 6.1, it can be seen that the experimental data fit reasonably well with these lines. The values of different parameters in equation 6.2.2 obtained from

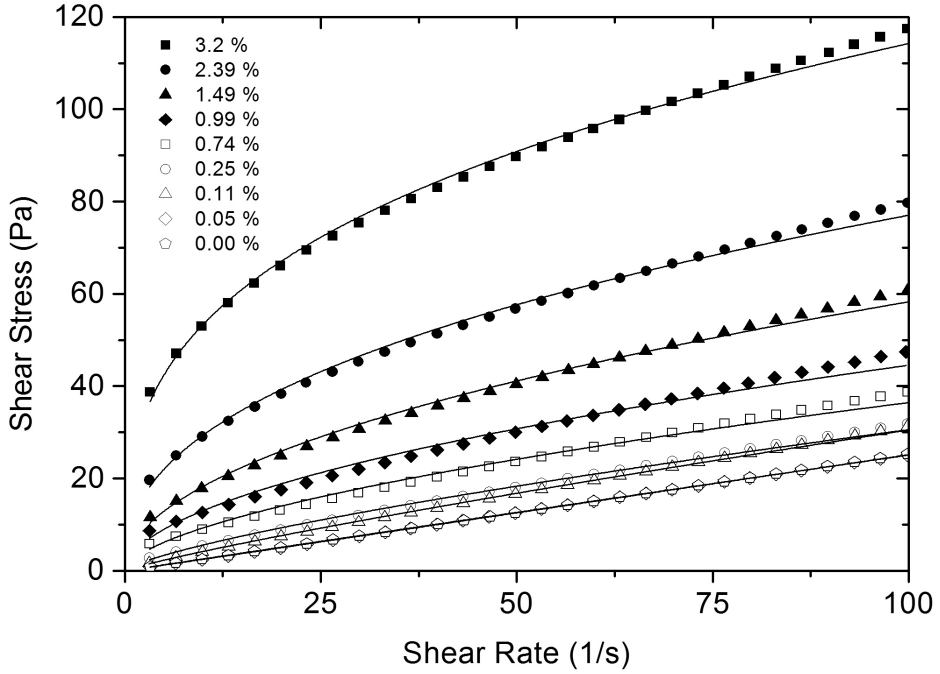


Figure 6.1: Comparison of shear stress and shear rate in various SWCNTs/ PBMA composites

the experimental data are given in Table 6.1.

The pure polymer viscosity, without addition of SWCNTs is 0.3 Pa.S and power law index is approximately equal to 1. Gradually, different concentrations

Concentration of SWCNTs (wt%)	A_c(Pa.S)	n
2.39	11.2	0.42
1.49	5.8	0.50
0.99	3.8	0.53
0.74	2.4	0.59
0.25	1.0	0.74
0.11	0.58	0.86
0.05	0.3	1.0
0.00	0.3	0.99

Table 6.1: Viscosity and power law index values for curve fitting experimental data in Figure 6.1 [3]

of SWCNTs are added to the polymer matrix and their viscosity and power law indexes are compared. At 0.05 SWCNTs concentration, the changes in viscosity and power law index are almost negligible, however with the increased concentration of SWCNTs to the polymer, the viscosity of the matrix is increased, but the power law index of the matrix is decreased. This is an indication that the material is going through shear thinning. The increase in the rate of shear thinning may be attributed to the fact that the bonds between the long threads of SWCNTs can easily being broken by the shear force. At higher viscosities small shear force will be needed to break the bonds between the SWCNTs threads. Figure 6.2 shows the comparison of different SWCNTs concentrations and their effect on viscosity. It can be seen that increased concentration of SWCNTs increased the value of viscosity. However, after the 1% concentration of SWCNTs the rate of increase in viscosity becomes gradual and it increases logarithmically. This is attributed to the fact that above 1%, large bundles of SWCNTs are already present within the polymer matrix and increased addition of SWCNTs has a minimal effect on

the viscosity of the matrix.

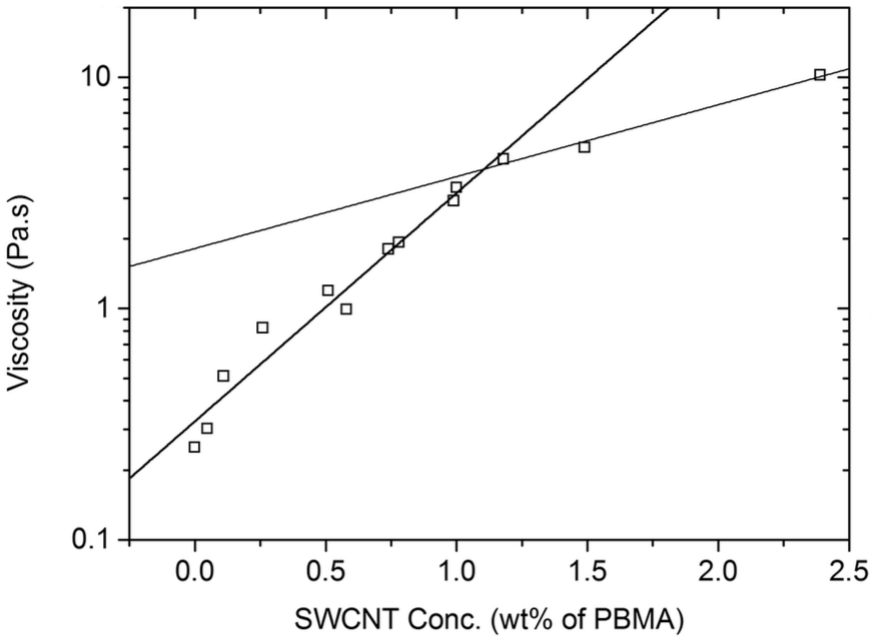


Figure 6.2: Comparison of viscosity versus SWCNTs [3]

6.2.2 Electrical measurements [3]

The different concentrations of SWCNTs in polymer matrix are studied for their current versus voltage properties using electrode slides with an electrode gap of $50\mu m$ (chapter 3, section 3.2). All the material composites showed a non-linear behaviour in terms of current versus voltage measurements. At lower concentrations, such as 0.1%, the current was measured in nA, but as the concentration of SWCNTs was increased in the polymer matrix, the current measurements were recoded in mA. This behaviour can be attributed to the fact that, the increased concentration of SWCNTs led to increased interconnections between SWCNTs bundles. This behaviour is also noted in viscosity measurements of these composites, as described in the previous section. The comparison of current versus SWCNTs

concentration is shown in Figure 6.3. The log of current versus SWCNTs concentration is linear up to 1.0% SWCNTs concentration, but this gradient reduces after 1.0% concentration. This behaviour can be attributed to the electrical percolation threshold of the SWCNTs network, where the voltage will become less affected by the increased concentration of SWCNTs. The electrical percolation threshold for SWCNTs network, as reported in literature [175], is between $0.17 - 0.70\%$ and several other factors such as a polymer, the source of SWCNTs and additional purification also affect the percolation threshold of SWCNTs networks. This percolation threshold for current experiments is in accordance to what is reported in literature [175], however, the polymer used for current experiments is PBMA (butyl group instead of methyl group), which has longer chains length and aids in better suspension and formation of thin films. In order to understand the

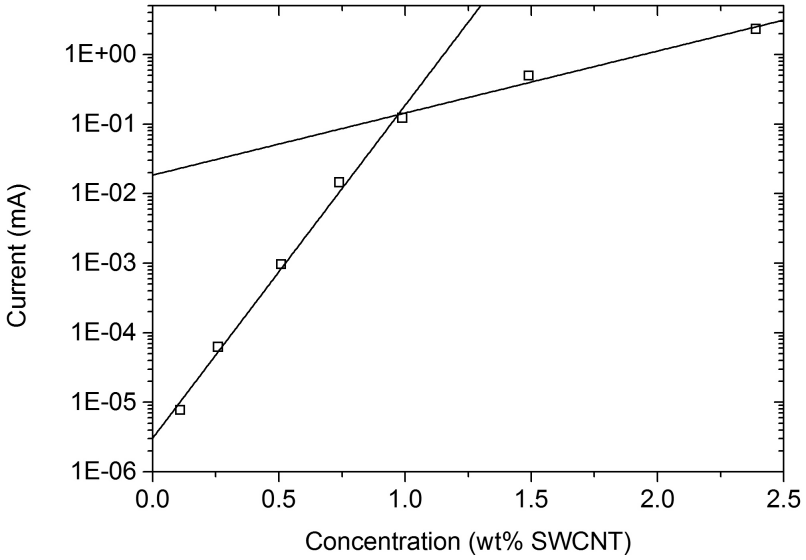


Figure 6.3: Comparison of current and voltage for various SWCNTs concentrations [3]

conduction mechanism of SWCNTs/polymer composites, the logarithm of current versus voltage were plotted, which indicated that at higher concentrations the

current was directly proportional to voltage with a slope 1. However, at lower concentration of SWCNTs the slope was recoded as 1.5, which means that there are other mechanisms at work with the SWCNTs network. It was suggested that this could be the Poole-Frankel (PF) effect [176], which is widely reported in literature as a conduction mechanism in Carbon nanotubes network. Hence, the current versus voltage relationship can be written as:

$$I \propto Vexp\left(- \frac{E_d - \beta_{PF} V^{\frac{1}{2}}}{kT} \right) \tag{6.2.3}$$

where

$$\beta_{PF} = \left(\frac{e^3}{\pi \epsilon_0 \epsilon_r d} \right)^{\frac{1}{2}} \tag{6.2.4}$$

The Poole-Frankel fit for two concentrations of SWCNTs (i.e. very low: 0.11% and very high: 3.20%) is plotted in Figure 6.4. A linear response with gradient 0.46 is observed in lower concentration of SWCNTs. However, higher concentration of SWCNTs has not showed a linear response between current and voltage relationship which suggested that Poole-Frankel is not the dominant conduction mechanism in these networks of films.

6.3 Results and discussion

The SWCNTs/PBMA based material posses complex electrical characteristics, with a field dependent conductivity when the concentration of SWCNTs is below a certain threshold. Using this data various concentrations listed in Table 3.2 in chapter 3, are studied for computer controlled optimisation of logic gates/circuits using the hardware described in Chapter 3, Section 3.3. The problem formulation is same as described in chapter 5, section 5.2 and AND, OR and half adder circuits are solved.

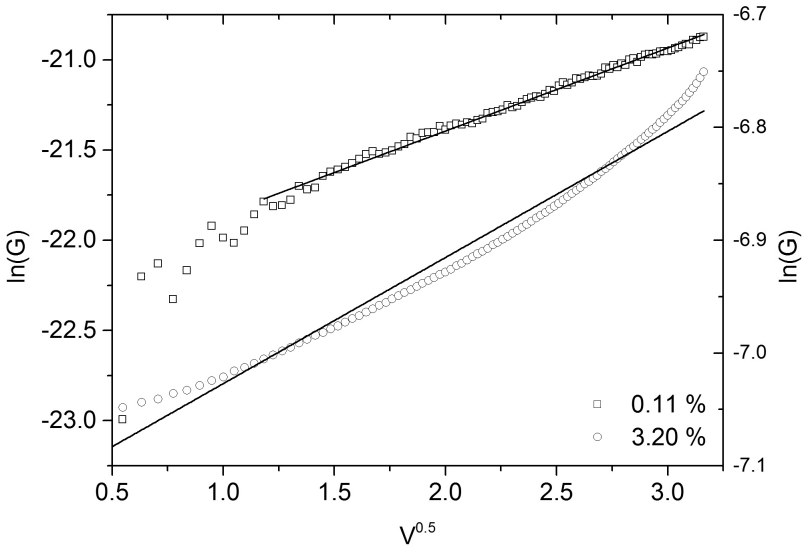


Figure 6.4: Poole-Frankel fit for low (0.11%) and high (3.20%) concentrations of SWCNTs [3]

The two parameters, i.e. fitness function value (0 as best value) and the number of function evaluations to achieve the fitness value are used to judge the suitability of the material for the computational problem. The number of function evaluations to reach the desired fitness value provides an indication that how well the material's properties are used for performing the target computation. The less number of function evaluations to reach an optimal solution indicates a more flexible material and an efficient hardware-in-loop optimisation. These investigations have been performed by keeping all the parameter values of Nelder-Mead algorithm the same for all the material composite. Table 6.2 provides the values of the number of average number of function evaluations achieved during 5 different optimisation runs by each material composite to reach the solution. It can be seen that the material with SWCNTs concentration above 0.1% require less number of function evaluations to converge to optimal solution, whereas, the concentration less that 0.1% required a larger number of function evaluations and reached the

Concentration of SWCNTs (wt%)/PBMA	Average no. of Function Evaluations (FE_{avg}) using NM		
	OR	AND	Half Adder
	FE_{avg}	FE_{avg}	FE_{avg}
0.11	1881	1889	1788
0.25	1115	1881	1998
0.51	1228	1258	1547
0.74	1228	1717	1587
0.99	22	45	142
1.49	39	32	285
2.39	21	45	118

Table 6.2: Average number of function evaluations during 5 different runs to train various concentrations of SWCNTs/PBMA composites for AND, OR and Half Adder circuit

termination criteria (maximum number of iterations allowed) without achieving the desired fitness function value. The number of function evaluations performed by the Nelder-Mead algorithm depends upon the simplex's expansion, contraction or shrunk rate during each iteration.

The fitness function value indicates the success of reaching the optimal solution for approximating the logic gate/circuits, with 0 being the desired value. The minimum ($\oint_{min}$), maximum ($\oint_{max}$) and average ($\oint_{avg}$) of fitness values achieved during optimisation process and the test accuracy (Φ_{avg}) achieved by various concentrations of SWCNTs/PBMA composites for three different logic gates/circuits are provided in Table 6.3. These values are achieved during 5 different runs for every material sample. In a similar manner to number of function evaluations,

SWCNTs (wt%)/ PBMA	Nelder Mead Optimisation results from 5 different runs											
	OR				AND				Half Adder			
	$\oint_{min}$	$\oint_{max}$	$\oint_{avg}$	Φ_{avg}	$\oint_{min}$	$\oint_{max}$	$\oint_{avg}$	Φ_{avg}	$\oint_{min}$	$\oint_{max}$	$\oint_{avg}$	Φ_{avg}
0.11	12	23	22	25%	4	6	6	22%	14	15	14	15%
0.25	9	11	9	28%	3	4	4	35%	17	14	17	15%
0.51	9	12	9	23%	6	7	8	48%	9	11	9	12%
0.74	4	5	4	48%	1	3	3	45%	1	3	3	55%
0.99	0	0	0	100%	0	0	0	100%	0	0	0	100%
1.49	0	0	0	100%	0	0	0	100%	0	0	0	100%
2.39	0	0	0	100%	0	0	0	100%	0	0	0	100%

Table 6.3: Fitness function values for various concentrations of SWCNTs/PBMA composites for AND, OR and Half adder circuit

the material composites having concentration of 1.0% and above achieved a fitness value of 0 for all three types of logic gates/circuits. Where as, at lower concentrations the fitness values are greater than zero indicating that the algorithm failed to converge to an optimal solution for all three types of logic gates/circuits.

It can be seen in Table 6.2 and Table 6.3 that AND and OR gate require less number of function evaluations for the material composites having SWCNT concentration above 1.0%. Whereas, the solution to half adder requires more iterations as compare to AND and OR gate, for the material composites having a concentration above 1.0%. This is attributed to the fact that in the half-adder circuit the optimisation algorithm is training the material to behave as two logic gates at the same time (an AND and XOR), for the same inputs. This increases the complexity and hence require more function evaluation to reach an optimal solution. It should be noted that threshold of 1.0% concentration of SWCNTs is

retained for the solution of OR, AND and a Half adder circuit. Hence, a denser and randomly dispersed network of SWCNTs in the polymer matrix (PBMA) is more favourable to training logic gates/ circuits, following the threshold scheme. The concentration of SWCNTs is directly proportional to the point where the conductivity starts increasing. Below this threshold concentrations, there are not enough connections within the SWCNTs/polymer matrix from which a meaningful computation can be extracted.

In some experiments performed by Mohid et al. [112] it was reported that the material sample with (1.0% SWCNT/PBMA) was successfully trained to solve even parity [116], tone discrimination [114], Robot controller [117] and data classification problems [115]. These investigations have also reported that the material samples with concentration lower than 0.02% in polymer (PMMA) cannot be evolved to solve the computational problem, however the ratio of PMMA varied. It was also observed that no evolution is possible when there was no material was present on board. These experiments also supported that the 1.0% SWCNT concentration in fixed polymer ratio (PBMA) are suitable to studied further for EIM and also other materials with similar physical and conductive properties can also be investigated for this study.

6.4 Conclusions

The work discussed in this chapter described the relationship between the concentration of SWCNTs in PBMA matrix and its effect on training the composite for solution of a computational problem. The SWCNTs/PBMA composites possess complex electrical and mechanical properties that can be used for unconventional computing. The study of electrical and mechanical properties of these composites showed a percolation threshold of 1.0%, where the electrical and mechanical properties of

the composites changed, which resulted in sheer thinning behaviour and change in electrical conductivity of the composites. Above the percolation threshold the rate of increase of conductivity is reduced and at lower concentrations, there are less number of interconnections between SWCNTs network and Poole-Frankel provided a good fit to data.

The various concentrations of SWCNTs/PBMA composites have been used to solve logic gate/circuit problem, using a specially designed hardware platform. A certain threshold in terms of SWCNTs concentration is observed for the solution of Logic gates/circuits. The threshold logic concept to interpret the outputs from the material as logic gate was used in conjunction with Nelder-Mead algorithm, which provided the values for optimal voltages. The results of these experiments showed that a threshold of 1.0 wt% SWCNTs is most suitable for solving logic gate/circuit problem. This is similar to what has been observed during the study of electrical and mechanical properties of these material composites.

These experiments are different from data classification and frequency classification problems [115], [177], where varying concentrations of SWCNTs as well as varying concentrations of PMMA were used to solve these problems. No conclusion was drawn from these experiments as to show if the concentration of SWCNTs matters in the polymer ratio to solve these computational problems.

These results provided a clear indication of the link between the SWCNTs concentration and the ability to solve a simple computational problem. The further investigation of these material composites to solve other computational problems will provide more insight into this relationship.

Chapter 7

Training SWCNTs/PMMA composites to solve complex logic circuits using Particle Swarm algorithm on Mecobo

This chapter presents the results of using a particle swarm optimisation (PSO) algorithm for evolving complex logic circuits in SWCNTs based composites on a purpose-built platform, Mecobo (version 4.1) (Chapter 3, Section 3.4). The material used is a composite of SWCNTs, dispersed randomly in a polymer(PMMA) forming a complex conductive network. Following the EIM methodology, the conductance of the material is manipulated for evolving complex threshold-based logic circuits.

The results of experiments in chapter 5, section 5.3.5 showed that the choice of contact points for the application of incident signals (input and configuration signals) and the collection of output signals have a significant effect on the computation performed by the SWCNTs/PMMA composites. The Mecobo platform is flexible

enough to put the choice of input and output signals under optimisation control. Hence, the experiments described in this chapter use a different approach from previous experiments for solving logic gates problem describe in chapter 5 and chapter 6. The material training problem is formulated as a constrained, mixed integer optimisation problem. The problem is solved using PSO in conjunction with the shortest position value rule. The results showed that the conductive properties of SWCNTs can be used to configure these materials to evolve multiple input/ output logic circuits using a more flexible hardware.

7.1 Introduction

The EIM methodology requires a platform that provides access to physical properties of the material in use and can bridge the gap between the analogue nature of the material and the digital nature of a computer supervising an evolutionary search. The experiments described in chapter 5, section 5.3.5, [178], and chapter 6, an *mbed* platform was used for evolving threshold logic circuits using population based optimisation.

The hardware used for those studies was relatively inflexible and did not allow the use of algorithms with extended vectors of decision variables regarding the selection of an incident signal's location of application on the material body. In comparison, Mecobo is a more powerful and versatile platform which allows for a more flexible problem formulation to be realised. The Mecobo board can interface with a large variety of materials and also has the flexibility to control and map the variety of input, output and configuration signals and their properties. In addition to the different hardware used, this chapter extends the work reported in [178], [3] and [179] by using a Particle Swarm Optimisation (PSO) instead of the Nelder-Mead and Differential Evolution algorithms.

The material in use has SWCNTs that are randomly distributed forming an inhomogeneous random network of nanotube bundles. This material is spread over the micro-electrode arrays and the input/output locations can be arbitrary. The choice of input/output and configuration electrode terminals are left to be decided by the optimisation algorithm.

In order to achieve this, a pin routing module is placed between signal generating modules and the sampling buffer. Hence, in contrast to previous experiments, where pin configuration was predetermined, the experiments presented in this chapter implement variable pin configuration that is under the control of the optimisation algorithm.

Different computational problems have been suggested for such a system but for initial experiments, the calculation of Boolean functions based on threshold logic is considered here.

7.2 Material training

As stated earlier, the threshold logic gates operate on the principle of comparing the output of a device to a pre-specified threshold for deciding if it is at a logical 0 or 1 state. Instead of following this detailed approach where a complete determination of weights and thresholds is required, a different operational principle is adopted. The device in consideration is a piece of inhomogeneous conducting material where a number of inputs are applied directly to its body and a number of output measurements are collected from it.

The material training problem is slightly different for the experiments describe in this chapter 5 and 6, although it also involves the training of the materials to behave as threshold logic gate/circuit.

There are two types of inputs that are applied in this device, configuration

voltages V_z, $z = 1, \ldots, q$, and the computation arguments x_i, $i = 1, \ldots, n$. The configuration inputs are used for changing the properties of the material and bring it to a computation inducing state. A material state is characterised as such when the response of the material can be uniquely translated to the correct computation when an arbitrary set of input arguments are applied to its body. The detailed description is as follows.

Let q denote the number of configuration inputs V_z, $z = 1, \ldots, q$, organised into vector $\mathbf{V}$ and $\mathbf{x} \in X \subset \mathbb{R}^n$ the space of possible inputs used for representing a binary vector $\mathbf{A} \in \{0,1\}^n$, i.e. there is a one-to-one mapping $\mathbf{R} : X \to \{0,1\}^n$. $\mathbf{R}(\mathbf{x}) = \mathbf{A}$ means that $\mathbf{x} \in X$ uniquely represents binary input $\mathbf{A} \in \{0,1\}^n$. The measured response of the material when q configuration and n computational argument inputs are applied, at output location j is $M_j(\mathbf{x}, \mathbf{V})$. A threshold logic gate based on M_j can be obtained as follows.

$$y_j = Y_{j,p} \text{ if } \theta_{j,p-1} \leq M_j(\mathbf{x}, \mathbf{V}) < \theta_{j,p}, \qquad (7.2.1)$$
$$j = 1, \cdots, m, \ p = 1, \ldots, L_j.$$

The detailed design effort required for obtaining the weights in threshold logic is replaced by a search process that aims at identifying a suitable $\mathbf{R}$ and finding the values of x_i, V_z and $\theta_{j,p}$. Making the material respond in a unique way to a given combination of configuration and (arbitrary in the domain of definition) computation argument inputs, is the task of EIM.

For the particular type of material used here, the output M_j is an increasing function of the total input, $\mathbf{x}$ and $\mathbf{V}$. The principle of operation is based on the implementation of rule (7.2.1) by dividing the output from location j into bands defined by the thresholds $\theta_{j,p}$. The computation is then performed by mapping a particular input to the corresponding output band, which indicates a logical 0 or 1 consistent with the Boolean function truth table. The truth table $\mathbf{Y}(\mathbf{A}) \in \{0,1\}^m$, $A \in \{0,1\}^n$, dictates the value of Y_j when a vector $\mathbf{x}$ representing $\mathbf{A}$ is applied to

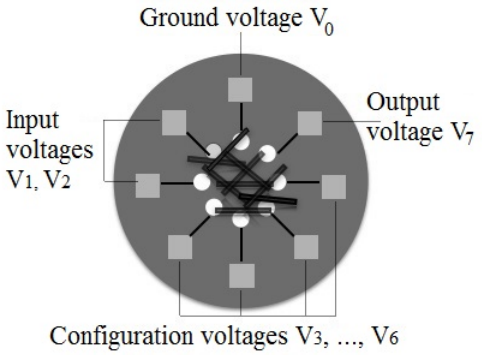


Figure 7.1: An example of arrangement of input, output and configuration electrodes

the material while it is in the computation inducing state.

Mecobo allows the parametrisation of the pins used for applying voltage inputs and those used for output measurements. Hence, the problem formulation is an extension of the one proposed in chapter 5 and 6, in the sense that the optimisation algorithm is allowed to select the pin assignment. Figure 7.1 depicts an instant of a particular assignment for a two-input single-output gate. There are eight locations where electrodes are connected with the material; each one can be selected to be an input or an output (location 0 is always the ground). In the instant shown in Figure 7.1, locations 1 and 2, where voltages V_1 and V_2 are applied, correspond to the gate's input terminals. The configuration voltages are applied at electrode terminals 3–6; these are used for manipulating the conductivity of the material so that the measured output can be interpreted according to threshold rule (7.2.1). The output is measured at location 7 where voltage V_7 is collected, i.e. $M_1(\mathbf{x}, \mathbf{V}) = V_7$.

Let $\mathbf{P} = [P_1 \dots P_{n+q+m}]^T$ denote the vector describing the pin assignment. $P_\ell \in \{1, \dots, q+n+m\}$ is the pin number at position ℓ. The first n positions in $\mathbf{P}$ correspond to the argument inputs, the next q to the configuration voltages and

the last m to the outputs. For the example of Figure 7.1, $\mathbf{P} = [1, 2, 3, 4, 5, 6, 7]^T$, with the pins' numbering starting where V_1 is applied and increasing in the counter-clockwise direction.

The material training problem at hand aims at identifying the optimal pin assignment $\mathbf{P}^*$, configuration voltages $\mathbf{V}^*$ and vector of thresholds θ^* for a given mapping $\mathbf{R}$ such that when the $\mathbf{V}^*$ are concurrently applied, the application of any input voltage $\mathbf{x}$ results in measurements M_j that when rule (7.2.1) is applied using using θ^* the equation $\mathbf{y}\left(\mathbf{P}^*, \mathbf{R}(\mathbf{x}), \mathbf{V}^*, \theta^*\right) = \mathbf{Y}\left(\mathbf{R}(\mathbf{x})\right)$ is true, ($\mathbf{y}$ is the vector of y_j).

A set of K training data is $\left(\mathbf{A}^{(k)}, \mathbf{Y}(\mathbf{A}^{(k)})\right)$, $k = 1, \ldots, K$ covering all possible combinations of binary inputs–outputs is generated. The objective function $J$of the optimisation problem is

$$J = \sum_{k=1}^{K} \left[\mathbf{y}\left(\mathbf{R}\left(\mathbf{x}^{(k)}\right), \mathbf{V}, \theta\right) - \mathbf{Y}\left(\mathbf{A}^{(k)}\right)\right]^2 \tag{7.2.2}$$

where

$$\mathbf{x}^{(k)} = R^{-1}\left(\mathbf{A}^{(k)}\right). \tag{7.2.3}$$

J is calculated by interacting with the material based on a given pin assignment. In the most general case, the vector of decision variables $\mathbf{B}$ has the form

$$\mathbf{B} = \left[\mathbf{P}^T\ \mathbf{x}^T\ \mathbf{V}^T\ \theta^T\right]^T. \tag{7.2.4}$$

A simple mapping $\mathbf{R}$ is obtained by setting $X = \{0, 1\}$ and $A_i = 0$ is represented by $x_i = 0$ V and $A_i = 1$ by $x_i = 1$ V. In this case $\mathbf{x}$ is fixed and therefore not part of $\mathbf{B}$ in (7.2.4). The case where $\mathbf{x}$ a function of $\mathbf{V}$ is discussed in [178].

The training optimisation problem is that of minimising (7.2.2) subject to (7.2.3), (7.2.1) and the following simple bound constraints on the configuration voltages $\mathbf{V}$.

$$V_{z,\min} \leq V_z \leq V_{z,\max},\ z = 1, \ldots, q. \tag{7.2.5}$$

Notice that there are no explicit constraints for the thresholds θ. The optimisation algorithm discards those solutions where the ordering necessary for (7.2.1) to work is not held.

This optimisation problem is solved using a classical particle swarm optimisation (PSO) algorithm, [180]. However, the $\mathbf{P}$ part of the vector of decision variables poses a permutation problem as a pin can only be used for a single input or output. This is dealt with by using the shortest position value (SPV) rule as proposed in [172] and [161]. The SPV rule converts the continuous position generated by PSO to a discrete value. The SPV rule is applied in two phases, one when the particles are generated and the other when the particles are updated. The rest of decision variables are considered as continuous.

Although $\mathbf{V}$ is continuous for the PSO algorithm, the implementation of a particular value goes through the Mecobo board, which can apply discrete levels of voltage. For all pins, the minimum voltage is $V_{\text{min}} = -5.0$ V and the maximum is $V_{\text{max}} = +5.0$ V. This voltage range is divided into 255 discrete equidistant levels with 0 corresponding to -5.0 V and 255 to $+5.0$ V.

7.3 Results and discussion

A number of Boolean functions have been used for making the material yield a threshold logic circuit in the sense described in section 4.5.1. The material used had a concentration of 0.1% SWCNT/14.8% PMMA, see section 3.1.1. Simple AND and OR gates were easily obtained. Here, the results are discussed for a 3-input 1-output logic circuit, the half-adder and the full-adder.

An initial trial of applying random configuration voltages resulted in outputs not being able to be translated into computational outcomes when each circuit's threshold rule was applied. It is the PSO algorithm that identifies suitable values

for those configuration inputs and thresholds. In other words, the material in its initial state cannot perform the required calculation.

7.3.1 Logic circuit $(A_1 + A_2 + A_3) \oplus (A_1A_2A_3)$

This is a three input, one output logic circuit based on AND, OR and XOR. Its truth table is shown in Table 7.1. At least one and at most two inputs need to be true in order for the output to be true. If all inputs are the same, either true or false, the output is false.

There are 16 electrodes on the glass slide containing the material. Hence, referring to the problem formulation of section 7.2, $n = 3$, $m = 1$ and $q = 8$. Two thresholds, θ_1 and θ_2 are required to separate between the output true and false states, one for distinguishing an all false inputs state and another for the all true inputs. The rule implemented is

$$Output = \begin{cases} 1 & \text{if} \quad \theta_1 < M_1(\mathbf{x}, \mathbf{V}) < \theta_2 \\ 0 & \text{otherwise.} \end{cases} \tag{7.3.6}$$

The optimal solution the PSO algorithm converged to is given in Table 7.2. It provides the optimal configuration voltages as well as the threshold values and pin assignment. Figure 7.2 shows the output measurements and optimal thresholds for all possible binary inputs. These were obtained from measuring the voltage at pin 10, while the material was constantly charged with the optimal configuration voltages at pins 2, 11, 8, 0, 7, 9, 1, 4, and a random sequence of binary triplets was applied to pins 5, 6 and 3. The output measurements between the two thresholds are interpreted as logic 1 and outside of it as 0.

Inputs			Output
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	0

Table 7.1: Truth table for $(A_1 + A_2 + A_3) \oplus (A_1A_2A_3)$.

Thresholds	$\theta_1 = -3.64$, $\theta_2 = -0.86$
Configuration voltages	$V_1 = -1.64$,$V_2 = -2.68$,$V_3 = -2.87$ $V_4 = -3.54$,$V_5 = -2.34$, $V_6 = -1.81$ $V_7 = -1.89$, $V_8 = -2.19$
Pin assignment (signal) $\rightarrow$) (pin no.)	$x_1 \rightarrow 5$, $x_2 \rightarrow 6$, $x_3 \rightarrow 3$ $M_1 \rightarrow 10$, $V_1 \rightarrow 2$, $V_2 \rightarrow 11$ $V_3 \rightarrow 8$, $V_4 \rightarrow 0$,$V_5 \rightarrow 7$ $V_6 \rightarrow 9$, $V_7 \rightarrow 1$, $V_8 \rightarrow 4$

Table 7.2: Optimal solution for logic circuit $(A_1 + A_2 + A_3) \oplus (A_1A_2A_3)$

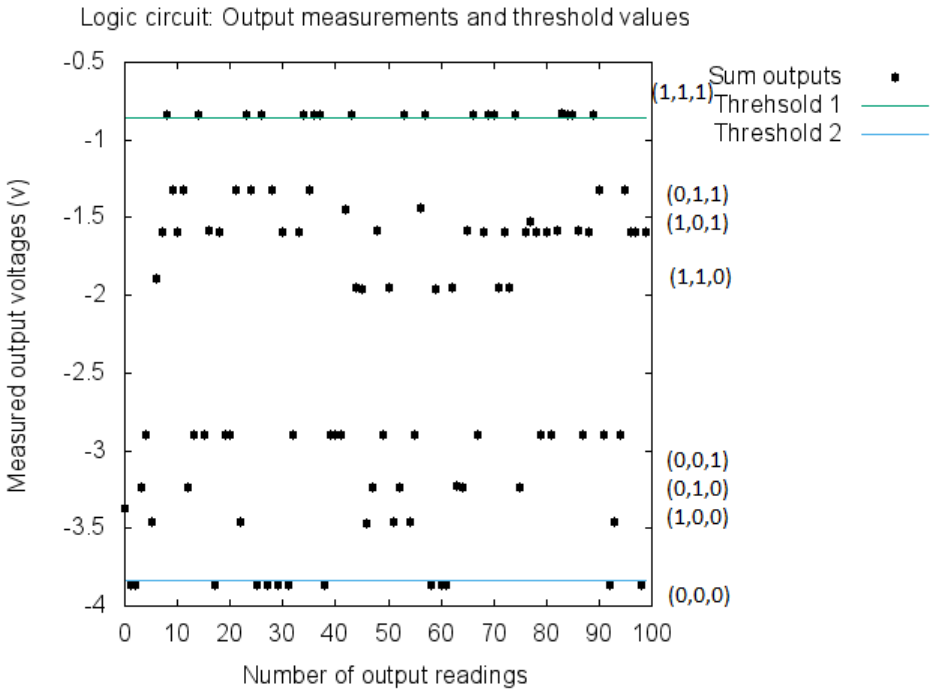


Figure 7.2: Material response for the $(A_1 + A_2 + A_3) \oplus (A_1A_2A_3)$ circuit; output measurements and thresholds.

It has also been observed from the optimal solutions of various runs that the output pin lies almost very close to the input pins. The 5 optimal solutions from 5 runs for solving the same logic circuit are given in Table 7.3. The pin assignment of these 5 solutions is also represented diagrammatically in Figure 7.3.

Optimal solution	Input 1 (x_1)	Input 2 (x_2)	Input 3(x_3)	Output 1$(M_1)	Configuration Pins
a	6	8	3	5	0,4,2,7,1,9,11,10
b	5	7	8	6	10,3,2,4,1,9,11
c	7	5	4	6	3,8,10,9,0,1,11,3
d	6	7	5	3	4,8,10,11,1,9,2,0
e	8	6	3	7	1,2,4,6,10,9,0,11

Table 7.3: Pin assignment for various optimal solutions for logic circuit $(A_1 + A_2 + A_3) \oplus (A_1A_2A_3)$

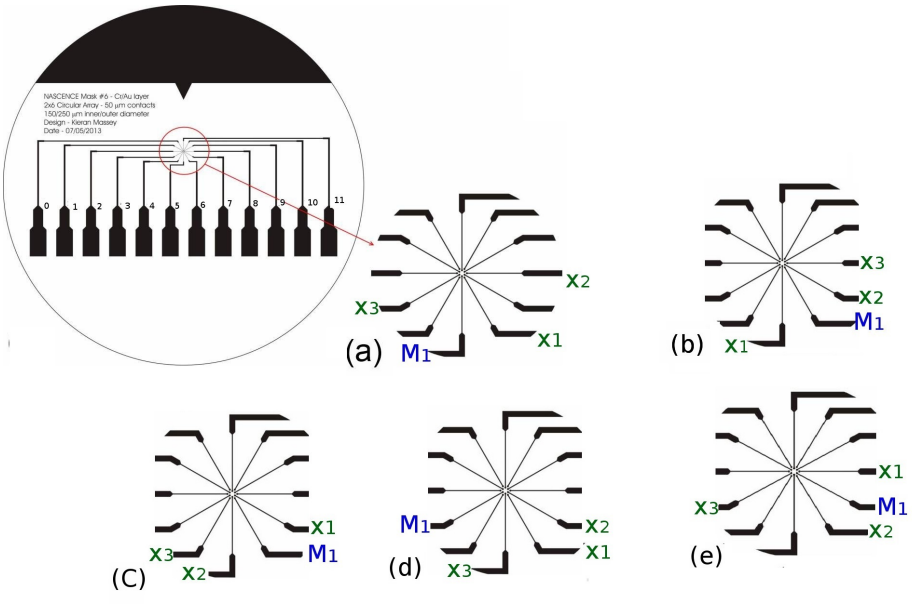


Figure 7.3: Pin assignment in 5 different optimal solutions achieved for the $(A_1 + A_2 + A_3) \oplus (A_1A_2A_3)$ circuit

7.3.2 Half-adder

The half-adder is a two-input two-output circuit using an XOR and an AND output. Its truth table is shown in Table 7.4.

Inputs		Output sum (XOR)	Output carry (AND)
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

Table 7.4: Truth table for half adder

For the training optimisation problem, $n = 2$, $m = 2$ and $q = 8$. The two output measurements M_1 and M_2 are collected for the XOR and AND,

respectively, for any binary pair (A_1, A_2). In order to make the material behave as a half-adder based on the use of thresholds, the output corresponding to AND, M_1, requires the use of a single threshold and the XOR measurement, M_2, needs two. For M_1 and the AND carry output,

$$Output\ Carry = \begin{cases} 1 & \text{if} \quad M_1(\mathbf{x}, \mathbf{V}) \geq \theta_1 \\ 0 & \text{otherwise.} \end{cases} \tag{7.3.7}$$

Hence, a measured M_1 larger than θ_1 is interpreted as a logical 1, otherwise as a 0.

For M_2 and the XOR sum output,

$$Output\ Sum = \begin{cases} 1 & \text{if} \quad \theta_2 \leq M_2(\mathbf{x}, \mathbf{V}) \leq \theta_3 \\ 0 & \text{otherwise.} \end{cases} \tag{7.3.8}$$

Hence, a measured M_2 between the two thresholds θ_2 and θ_3 is interpreted as a logical 1 at the sum output, otherwise if it is below θ_2 or above θ_3 as a logical 0.

Training the material based on equations. (7.3.7) and (7.3.8) for interpreting the measured outputs as half-adder circuit computation results to the solution shown in Table 7.5.

The output measurements at two electrodes are shown in Figure 7.4. The three thresholds can be differentiated clearly. One is used for calculating the carry and two for the sum. The electrode pins 7, and 6 are chosen as input terminals and 1 and 3 as output; the rest of the pins are configuration voltage terminals. It should be noted that the measurements M_1 and M_2 are taken concurrently, while the configuration voltages are constantly been applied.

The 5 different optimal solutions from multiple runs are observed for the pin assignment and the distance of input pins from output pins. The data is presented in table and drawn in Figure 7.5. As observed for previous logic circuit, the input pins lie closely with the output pins.

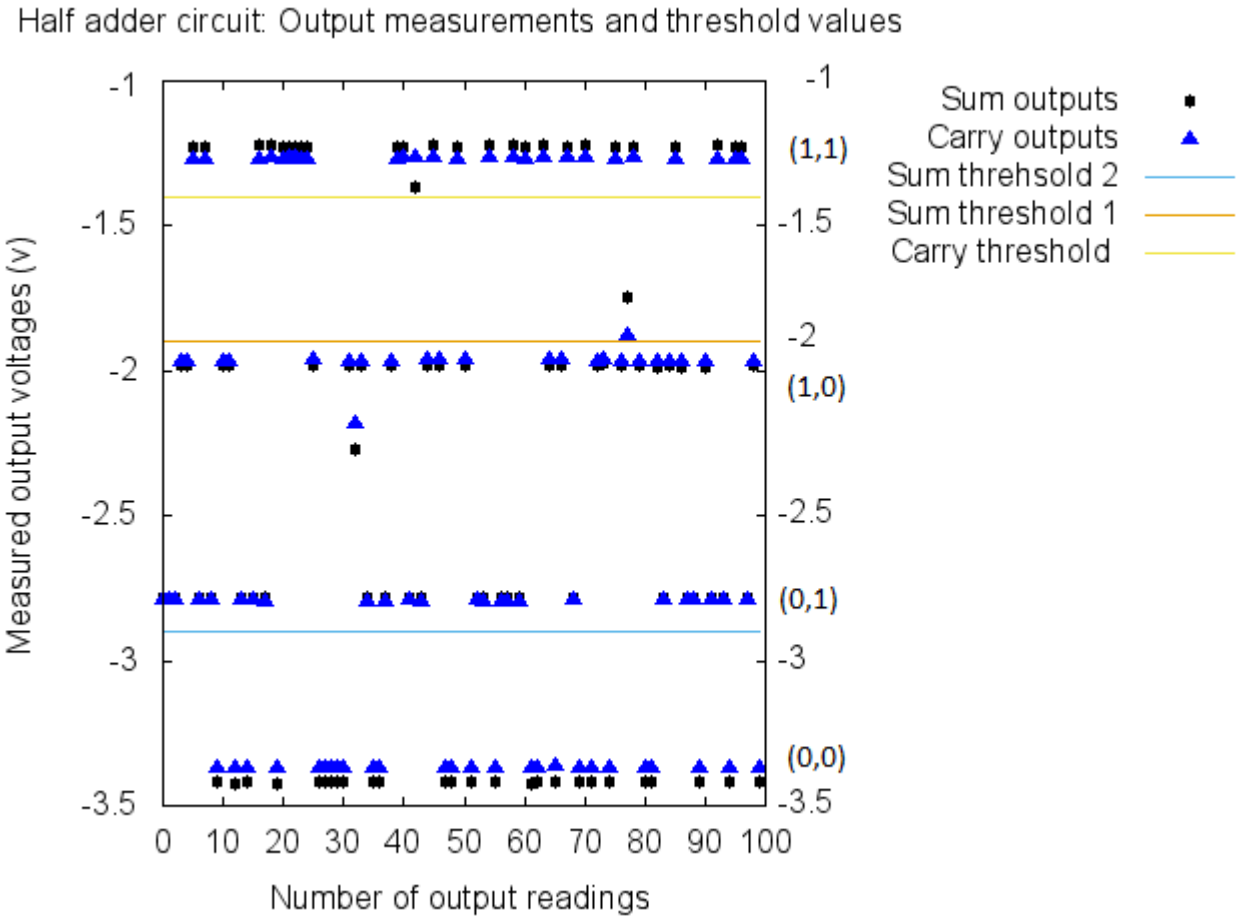


Figure 7.4: Material response for the half-adder circuit; output measurements and thresholds.

Threshold for carry	$\theta_1 = -1.4$
Thresholds for sum	$\theta_2 = -2.9$, $\theta_3 = -1.9$
Configuration voltages	$V_1 = 0.03$, $V_2 = 1.45$, $V_3 = 3.19$ $V_4 = -0.21$, $V_5 = -0.75$, $V_6 = -0.68$ $V_7 = -0.53$, $V_8 = -1.75$
Pin assignment (signal) $\rightarrow$) (pin no.)	$x_1 \rightarrow 7$, $x_2 \rightarrow 6$, $M_1 \rightarrow 1$ $M_2 \rightarrow 3$, $V_1 \rightarrow 9$, $V_2 \rightarrow 4$ $V_3 \rightarrow 2$, $V_4 \rightarrow 10$, $V_5 \rightarrow 5$ $V_6 \rightarrow 0$, $V_7 \rightarrow 11$, $V_8 \rightarrow 8$

Table 7.5: Optimal solution for half-adder circuit.

Optimal solution	Input 1 (x_1)	Input 2 (x_2)	Output 1(M_1)	Output 2 (M_2)	Configuration Pins
a	8	6	5	7	0,4,2,7,1,9,11,10
b	6	5	8	3	10,3,2,4,1,9,11
c	5	7	8	4	3,8,10,9,0,1,11,3
d	5	6	8	7	4,8,10,11,1,9,2,0
e	5	6	4	7	1,2,4,6,10,9,0,11

Table 7.6: Pin assignment for various optimal solutions for Half adder logic circuit

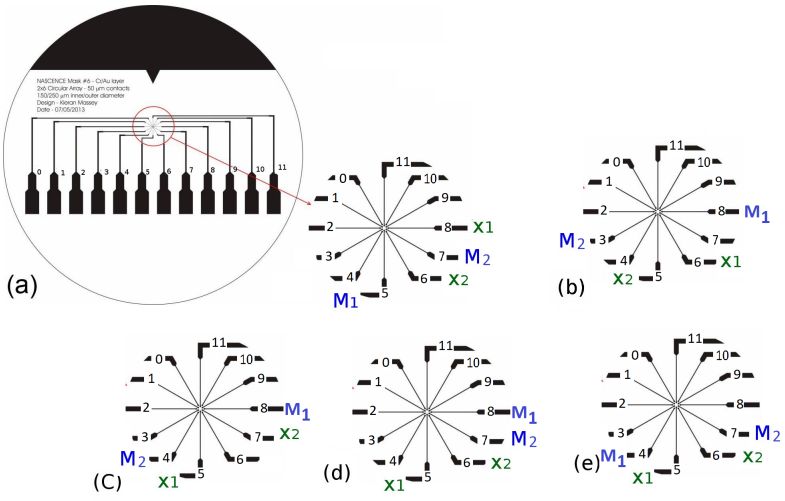


Figure 7.5: Pin assignment, with inputs x_1, x_2 and outputs M_1, M_2 in 5 different optimal solutions achieved for the Half adder circuit

7.3.3 Full-adder

The full-adder circuit has three inputs and two outputs, the sum and carry, hence $n = 3$ and $m = 2$ leaving $q = 7$ configuration voltages available for obtaining a solution. Again, two measurement pins are used, M_1 for the carry and M_2 for the sum. The circuit's truth table is given in Table 7.7.

A single threshold is used for the carry leading to the following interpretation of measurement M_1.

$$Output\ carry = \begin{cases} 1 & \text{if } M_1(\mathbf{x}, \mathbf{V}) \geq \theta_1 \\ 0 & \text{otherwise.} \end{cases} \tag{7.3.9}$$

The sum operation is more complicated and requires three thresholds to calculate it at the output terminal where M_2 is measured. This leads to the following

Inputs			Output sum	Output carry
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

Table 7.7: Truth table for the full-adder circuit.

threshold rule.

$$Output\ sum = \begin{cases} 0 & \text{if } M_2(\mathbf{x}, \mathbf{V}) < \theta_2 \\ 1 & \text{if } \theta_2 \leq M_2(\mathbf{x}, \mathbf{V}) < \theta_3 \\ 0 & \text{if } \theta_3 \leq M_2(\mathbf{x}, \mathbf{V}) < \theta_4 \\ 1 & \text{if } M_2(\mathbf{x}, \mathbf{V}) \geq \theta_4 \end{cases} \tag{7.3.10}$$

The optimal solution is given in Table 7.8. Seven configuration voltages were used with three inputs and two outputs. The three inputs are applied at pins 4, 1 and 2, whereas the outputs are collected from pins 7 and 5. The response of the material to each of the specific 16 possible binary inputs is shown in Figure 7.6, along with the thresholds used. Compared to the half-adder response shown in Figure 7.4, the full-adder measured output is more complicated because of the three, rather than two, binary inputs. The optimisation is able to find a solution, where the response of the material is organised so that the interpretation scheme followed results to the full-adder outcome.

In order to observe the distance between input and output pins for Full Adder circuit, the pin assignment form 5 different optimal solutions is presented in table

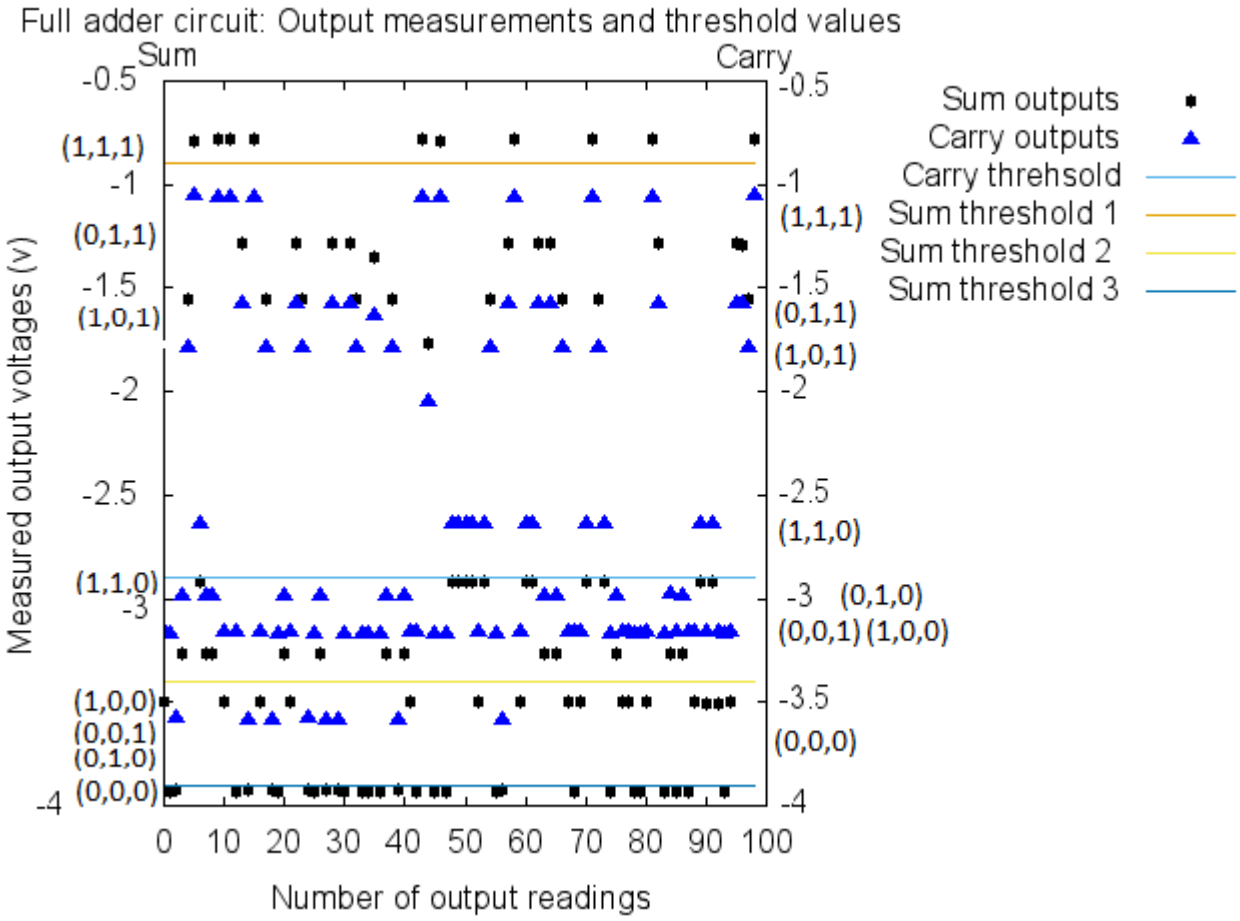


Figure 7.6: Material response for the full-adder circuit; output measurements and thresholds.

Threshold for carry	$\theta_1 = -2.9$
Thresholds for sum	$\theta_2 = -3.9$, $\theta_3 = -3.4$, $\theta_4 = -0.9$
Configuration voltages	$V_1 = 0.03$, $V_2 = -0.62$, $V_3 = -1.99$ $V_4 = -2.17$, $V_5 = -2.82$, $V_6 = -1.97$ $V_7 = -1.74$
Pin assignment (signal) $\rightarrow$) (pin#)	$x_1 \rightarrow 4$, $x_2 \rightarrow 1$, $x_3 \rightarrow 2$ $M_1 \rightarrow 7$, $M_2 \rightarrow 5$, $V_1 \rightarrow 10$ $V_2 \rightarrow 6$, $V_3 \rightarrow 3$, $V_4 \rightarrow 9$ $V_5 \rightarrow 0$, $V_6 \rightarrow 11$, $V_7 \rightarrow 8$

Table 7.8: Optimal solution for the full-adder circuit

7.9 and drawn diagrammatically in Figure 7.7. It can be seen in the diagram the optimisation has chosen the input and output pins where the distance between these was minimal.

Optimal solution	Input 1 (x_1)	Input 2 (x_2)	Input 3(x_3)	Output 1 (M_1)	Output 2 (M_2)	Configuration Pins
a	6	8	3	4	7	5,1,0,2,9,11,10
b	8	5	6	4	3	10,0,1,7,2,9,11
c	5	6	4	7	8	3,10,9,0,1,11,2
d	7	5	4	6	3	0,1,2,8,10,11,9
e	5	6	8	4	7	0,1,3,2,10,9,11

Table 7.9: Pin assignment for various optimal solutions for Full adder logic circuit

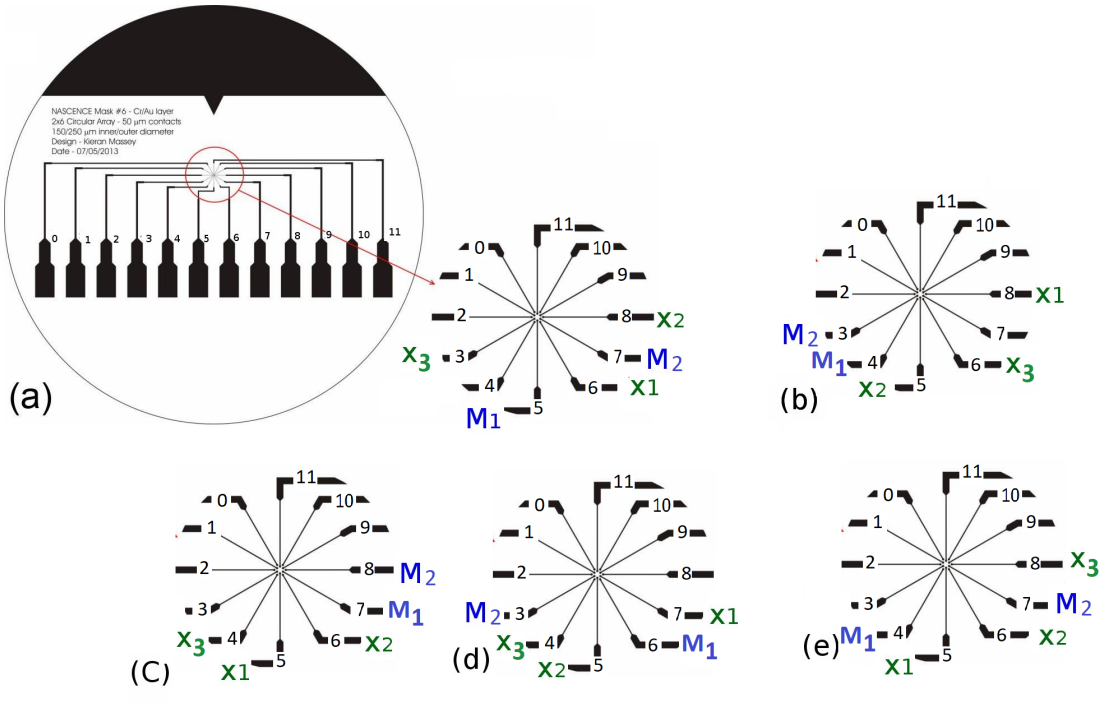


Figure 7.7: Pin assignment in 5 different optimal solutions achieved for the Full adder circuit

The generalised results for three different logic circuits from 5 different experiments for each logic circuit are presented in Table 7.10. The first four columns represent the minimum, maximum, average error, standard deviation of training errors. The average number of iterations required to achieved the optimal solution during the training phase are given in last column. Once the training is terminated the test phase is performed on the trained material by applying back the optimal solution along with new randomly generated test data. The verification (test) procedure is repeated ten times and the average verification accuracy is given as (Φ_{avg}). It can be seen that PSO algorithm solved all the logic gate cases during the training and testing phases. However, as the complexity of the circuit is increased i.e. increased number of inputs, outputs and the thresholds, the number of iterations required to find the optimal solution are also increased.

Particle Swarm Optimisation results from 5 different runs for three different logic circuits						
Logic circuit	$\oint min$	$\oint max$	$\oint avg$	$stedv$	Φ_{avg}	It_{avg}
$(A_1 + A_2 + A_3) \oplus (A_1A_2A_3)$	0.00	0.02	0.0	0.0	100%	95
Half adder	0.00	0.03	0.0	0.01	100%	128
Full adder	0.00	0.03	0.00	0.01	100%	196

Table 7.10: Training errors and verification accuracy from 5 different runs for three logic circuits

7.4 Conclusion

This chapter presents the results of applying PSO for EIM where the material consists of a SWCNTs/PMMA mixture. By manipulating the conductance of the material, it is possible to perform calculations of Boolean functions based on a threshold logic interpretation scheme. The Mecobo platform allowed the flexibility of selecting the input, output and configuration terminals and put them under the control of optimisation algorithm. The experiments demonstrated Mecobo's functionality and suitability as an interface for evolvable material. For the experiments reported here, material with 0.1% SWCNTs concentration was used. The material used does not change its state and the results are reproducible with same configuration voltages without requiring new evolutionary search.In future work, the material with varied concentrations will be tested for similar computation problems.

Chapter 8

Training SWCNTs/Polymer composites as a tone discriminator

This chapter presents the results of implementation of tone discrimination problem with various SWCNTs/Polymer mixtures using the purpose-built platform, Mecobo. The platform allows the application of square wave signals, hence, the materials are trained to discriminate between a low and a high frequency signal using an interpretation scheme. In order to train the given material to generate a different response for a low and high frequency signal a typical training and verification scheme is followed. The problem is formulated as a mixed integer optimisation problem which is solved using PSO algorithm in conjunction with SPV. The results showed that SWCNTs/polymer based materials can be successfully trained as tone discrimination devices.

The detail description of the implementation of the problem and results are given below.

8.1 Introduction

A tone discrimination is a device which, when presented with two signals, generates a different response for each type of signal. The problem was first solved using the evolutionary algorithms by Thompson [181], which resulted in a device, where the FPGA can generate a low output for low frequency signal and a high output for a high frequency signal. Later, [39] used Liquid Crystals and FPGAs to solve the same problem. Also, while this thesis was being written, [177] reported on using SWCNT based materials and Genetic Algorithms to evolve a device that when presented with one of the two frequency signals, generated a low voltage signal ($< 0.1V$) for low frequency and a high voltage signal ($> 0.1V$) for high frequency.

The experiments describe in this chapter are loosely based on these experiments. However, the problem formulation is different and it is implemented with the Particle Swarm Optimisation algorithm for the selection of configuration data. A variety of SWCNTs concentrations and polymers based materials are used for this work. The problem is implemented using purpose-built platform, Mecobo, which allows the application of square wave signals and is also flexible enough to implement the extended vectors of decision variables with regards to the selection of application of incident signals on the material. In general, the SWCNT materials are trained to distinguish between high and low frequency signals. The detailed methodology of these experiments is discussed in following section.

8.2 Material training

The experiments are performed with different SWCNTs concentrations and polymer (PMMA and PBMA) combinations. Each materials is placed on an electrode slide, (12 electrodes in case of SWCNTs/PMMA material samples and 16 electrodes in case of SWCNTs/PBMA material samples).

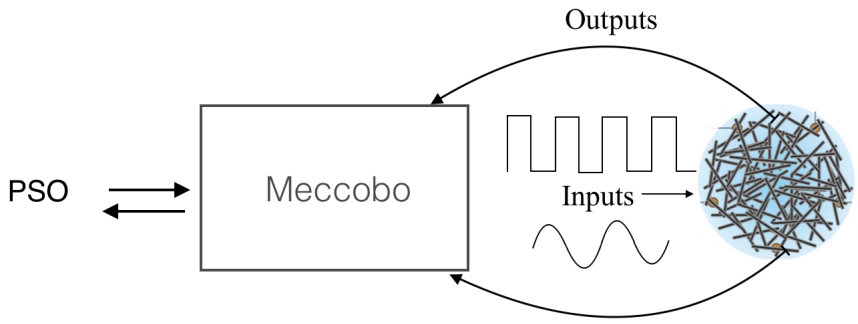


Figure 8.1: General overview of the system of training the material as a tone discriminator

A general illustration of the material training is given in Figure 8.1. There are two types of incident signals that are applied on to the material. One is the computer generated configuration signals, which are in the form of static voltages and are denoted by $V_z, z = 1, \cdots, q$, (where q is the number of configuration signals) with range $-5V$ to $+5V$. These signals are selected by the optimisation algorithm (PSO). The others are the input signals, which are denoted by x_i (where i is the number of input signals).

Each input signal is in the form of a square waves ($\pm 3.5V$, duty cycle 50%) with a randomly generated frequency between the range of $10\,kHz$ and $300\,kHz$. These two different types of signals are applied at different locations on to the material samples and the outputs, denoted by O_m, (where m is the number of outputs) are collected at different locations from the material. The choice of input, output and configuration signals is dictated by the PSO algorithm which is implemented with SPV rule.

The configuration signals change the material internal properties and bring it to a computing inducing state. The outputs are a measure of configuration voltages V_z and the square wave signals x_i.

The overall material training problem is formulated as an optimisation problem

which is solved iteratively using the PSO algorithm with swarm size 7. The optimisation algorithm search the space for possible configuration inputs values. The desired fitness value is 0 which is when the computational task for the selected arguments is performed correctly by the material. The fitness value and the number of iterations (1000) are set as the termination criteria for this iterative process.

The pins are assigned according to vector $P = [P_1, \cdots, P_{i+q+m}]^T$. P_l is the pin number at position l on the electrode, where $P_l \in 1, \cdots, i+q+m$. The first i positions correspond to inputs, next q correspond to configuration signals and last m correspond to outputs. The configuration signals values, denoted by V, are supplied as continuous amplitudes values by the optimisation algorithm. The Mecobo board translates these amplitude values to discrete static voltage values. The voltage range is divided between 255 equidistant levels from $0 - 255$, where 0 corresponds to $V_{min} = -5.0V$ and 255 corresponds to $V_{max} = +5.0V$.

Out of 8 input and 8 output locations on Mecobo board, one electrode is used as an input electrode, where a square wave signal with randomly selected frequency is applied, 6 electrodes are used for application of configuration voltages and 2 electrodes are used as output electrodes (i.e $i = 1$, $q = 6$, $m = 2$). An instance of particular assignment of input, outputs and configuration electrodes with 12 electrode slide is shown in Figure 8.2, where I_1 is the input square wave signal with some random frequency, $V_2, \cdots, V_8$ are the configuration signals O_1 and O_2 are the outputs.

The material training problem aims at identifying the optimal pin assignment P and configuration voltages V, such that when square waves signals with different frequencies are applied on to material, it can differentiate between high ($\geq 100kHZ$) and low ($< 100\,kHZ$) frequency signal. A decision variable vector B generated

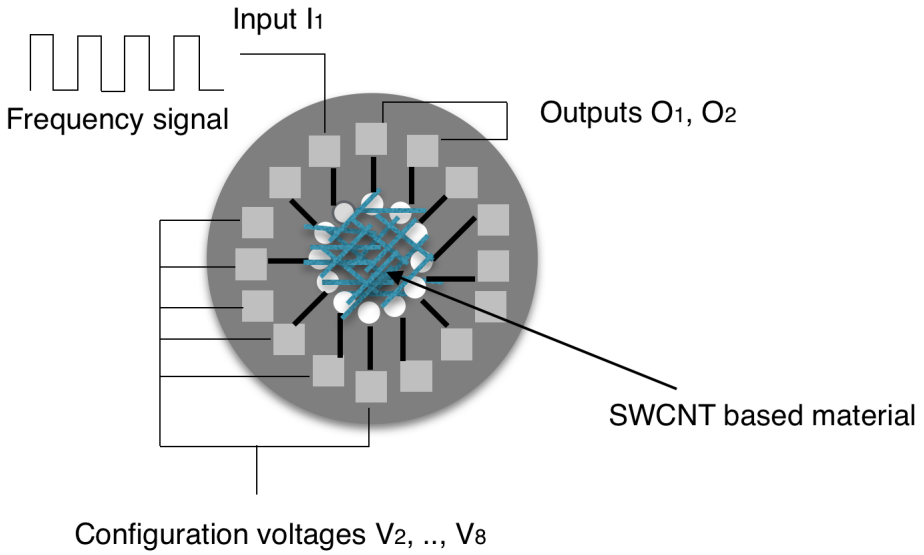


Figure 8.2: An example of arrangement of input, output and configuration electrodes

by PSO algorithm at each iteration, can be represented as follows:

$$B = [P^T x^T V^T]^T \tag{8.2.1}$$

Let S be the vector containing 100 randomly selected frequencies between the available range on the platform $10kHz - 300kHz$. Each frequency is applied at the input electrode x_1 for 20 milli-seconds in the presence of configuration signals (20 milli-seconds) and the outputs O_1, O_2 are recorded within this time frame. The output are voltage values which were recorded from the material at 5kHz in the output buffer, these values are then averaged and sent as a real number (a voltage value). A vector H is decided by the following equation.

$$H(i) = \begin{cases} 0 & \text{if } O_1 < O_2 \; and \; S[i] = LOW \\ 0 & \text{if } O_1 > O_2 \; and \; S[i] = HIGH \\ 1 & otherwise \end{cases} \tag{8.2.2}$$

The fitness is calculated as follows:

$$\min J = \sum_{i=1}^{K} \sum_{i=1}^{O} [(H_i)/K] \tag{8.2.3}$$

Where $K = 100$ are the training samples.

The training problem aims to minimise the Equation 8.2.3 for a population of S subject to simple bound constraints on configuration voltages, i.e. $V_{z,min} \leq V_z \leq V_{z,max}, z = 1, \cdots, q$, electrode assignment P and the rule 8.2.2.

The general overview of training the material as a tone discriminator device is summarised in Figure 8.1. The PSO sends the values of decision variable to Mecobo. The Mecobo board interfaces with the material and applies the values of input and configuration signals on the material. The response (outputs) is gathered from the material and it is sent back to the Mecobo board, which then sends it back to the computer. The fitness value is calculated using equations 8.2.2 and 8.2.3 and sent back to PSO. The process is repeated until the termination criteria are met.

8.3 Results and discussion

After the termination of training phase, the verification phase starts. During this the verification data (a new set of randomly generated frequencies between 10 $kHz - 300$ kHz) is used and the optimal solution (configuration voltages and pin scheme) are applied to the material, the outputs O_1, O_2 are measured at the two terminals and the accuracy is calculated according to Equation 8.2.2. For example, if the first sample from verification data is a 100 kHz frequency signal and is applied on the material along with the optimal configuration signals and the resulting $O_1 < O_2$, then it is an accurate response. The same response is calculated for all the samples in the verification data set according to rule 8.2.2 and the percentage accuracy for all of the verification data set is reported.

Optimal solution for discriminating frequencies below and above 100 kHz	
Configuration voltages	$V_1 = -0.56$,$V_2 = 2.09$,$V_3 = -0.76$ $V_4 = 5.0$,$V_5 = -3.74$, $V_6 = -5.0$
Pin assignment	$x_1 \rightarrow 2$, $O_1 \rightarrow 5$, $O_1 \rightarrow 7$ $V_1 \rightarrow 0$, $V_2 \rightarrow 3$ $V_3 \rightarrow 4$, $V_4 \rightarrow 0$,$V_5 \rightarrow 8$ $V_6 \rightarrow 1$

Table 8.1: Optimal solution for discriminating frequencies below and above 100 kHz using material SWCNTss(0.75%)/ PMMA

In the first set of experiments, the material used is SWCNTs(0.75%)/PMMA on a 12 electrode slide and the verification data set consists of 100 rows of data. The desired fitness value (0) is achieved after approximately 108 function evaluations. The optimal solution to achieve this computation is given in the Table 8.1.

Figure8.3 shows the configuration voltages trajectories during the training phase. It can be seen that the search process started randomly and settles over the number of iterations as the optimisation process tried to find the optimal voltage values. The objective function trajectory during the training process is shown in Figure,8.4.

The optimal solution states that the outputs are recorded by measuring the voltages at pins, (5 and 7), when the optimal configuration voltages are applied at pins, $(0, 3, 4, 8, 6, 1)$, and the square waves with varying frequencies (from verification data set) are applied at the input pin (2). The outputs are collected concurrently, while the configuration pins are being charged with the optimal configuration voltages. The first 20 rows of verification data are shown in Figure 8.5, which shows the output measurements collected for the respective input frequencies (shown as a number right to the output 1) during the verification

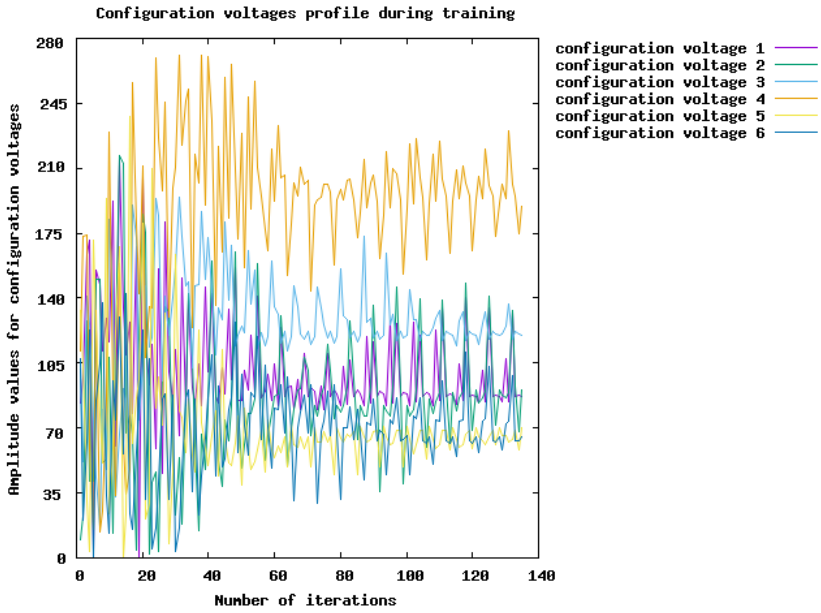


Figure 8.3: Configuration voltages trajectories

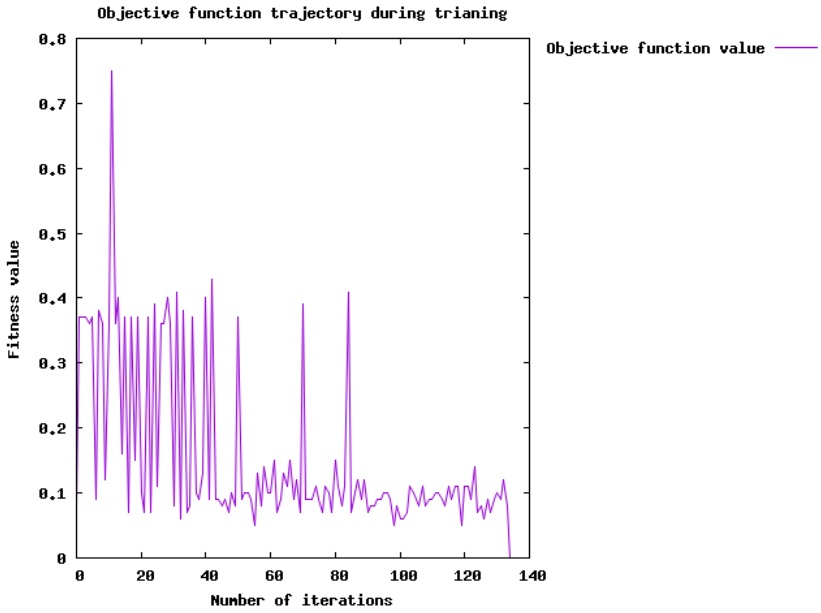


Figure 8.4: Objective function's convergence trajectory

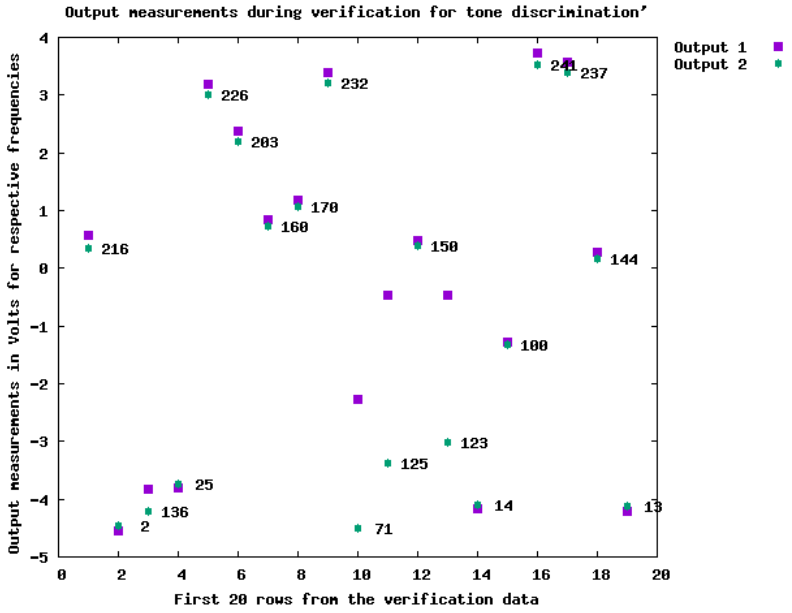


Figure 8.5: Output measurements with their respective frequencies during verification phase

phase. The implementation of the rule 8.2.2 can be seen in the Figure 8.5. The only first 20 results are used in the Figure for the sake of clarity. The two outputs are shown in different colours and shapes, i.e. output 1 is a purple square and output 2 is a green circle. The number next to the two outputs is the frequency value for which the outputs were recorded. For example, the first value of the frequency is 216 in the Figure 8.5, the two recorded outputs are shown next to it. According to the rule, for frequency value ≥ 100, output 1 should be $>$ than output 2, which is same as shown for the frequency value 216, i.e output 1 (in purple square) is greater that output 2 (in green circle). The same can seen for the rest of values in the Figure 8.5. For the frequency values: ≤ 100, output 1 is $<$ output 2 and for the frequency values: ≥ 100, output 1 is $>$ output 2.

In order to observe if the computation is actually performed by the material a separate experiment is performed. Some random configuration signals are applied

on the material along with inputs and outputs are recorded. It is observed that the recorded outputs are not translated into computational outcome when the rule in equation 8.2.2 is applied. Same was observed when no configurations are applied on the material. Hence, it is concluded, that it is the PSO algorithm that identifies the suitable configuration signals, in order to make the material perform the desired computation. The material on its own cannot act as a tone discriminator.

8.3.1 Comparison of different concentrations of SWCNTs in fixed polymer for tone discriminator problem

In the next sets of experiments, three different concentrations of SWCNTs in a fixed polymer (PBMA) ratio are used for the solution of tone discrimination problem. The detailed description of these material samples and thin film formation is given in Chapter 3. As discussed earlier, these materials composites are deposited on electrodes as thin films. These thin films possess complex electrical properties with a field conductivity if the concentration of SWCNTs is below a certain threshold. These Polymer/SWCNTs mixtures have dielectric properties of the PBMA and mixed differing electronic properties of SWCNTs (both metallic and semiconducting).

The material training problem formulation is same as discussed in the previous section. However, the materials samples are placed on a 16 electrode slide but Mecobo allows 8 inputs and 8 outputs only. Due to this limitation, 1 input, 7 configuration voltages (total 8 input signals) and 2 output scheme is used for these experiments. An example of this scheme can be seen in Figure 8.2.

The optimal solutions for the tone discrimination problem using three different concentrations of SWCNTs in fixed polymer (PBMA) are given in Table 8.2. Where as, the verification accuracies are shown in Table 8.3. Figures, 8.6, 8.7,

8.8 show the responses of the three materials (three different concentrations of SWCNTs) when low and high frequency signal are applied under the effect of configuration voltages. The first 100 rows of verification data are shown in each of the Figures, 8.6, 8.7, 8.8. The implementation of the rule 8.2.2 can be seen in these figures. For instance, Figure 8.6 shows the output measurements collected for the respective input frequencies during the verification phase. The two outputs are shown in different colours and shapes, i.e. output 1 is a green cross and output 2 is a purple plus sign. The number next to the two outputs is the frequency value for which the outputs were recorded. For example, the first value of the frequency is 154 kHz in the Figure 8.6, the two recorded outputs are shown next to it. According to the rule, for frequency value $\geq$ 100, output 1 should be $>$ than output 2, which is same as shown for the frequency value 154, i.e output 1 (green cross) is greater that output 2 (purple plus sign). The same can seen for the rest of the values in the Figure 8.5. For the frequency values: $\leq$ 100, output 1 is $<$ output 2 and for the frequency values: $\geq$ 100, output 1 is $>$ output 2. It can be seen in all three figures that the response output 1 is $<$ output 2 for the frequency values: $\leq$ 100, and for the frequency values: $\geq$ 100, output 1 is $>$ output 2.

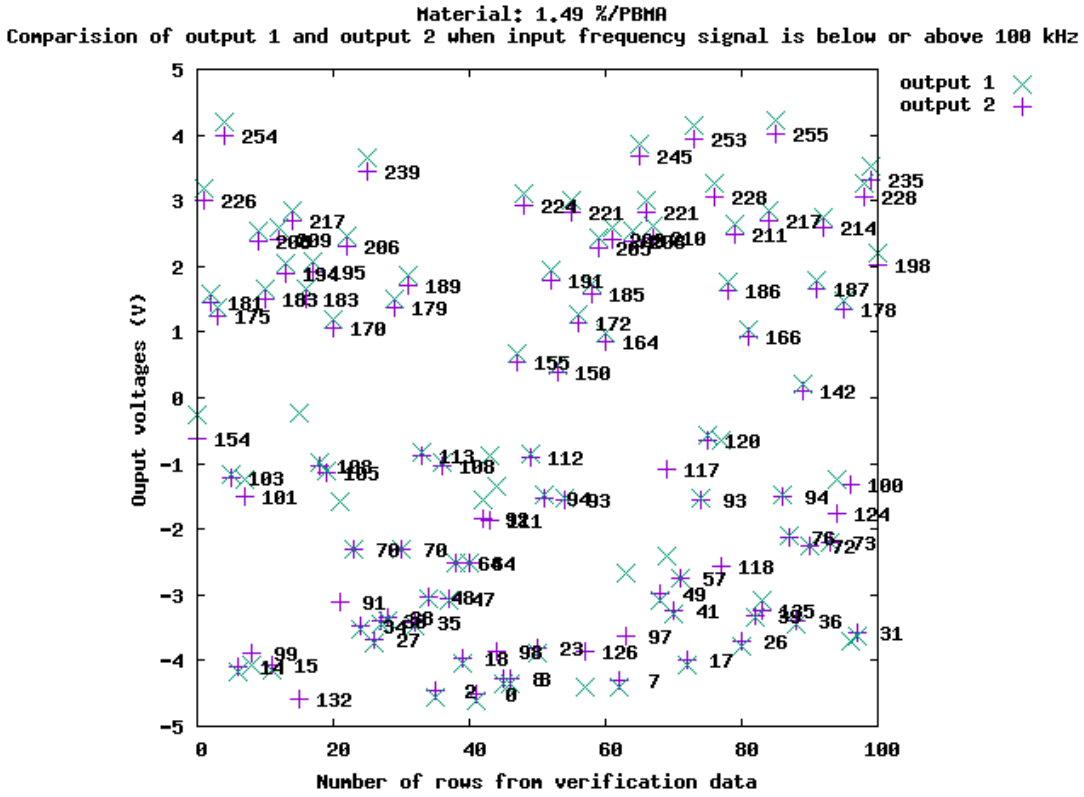


Figure 8.6: Comparison of two outputs at frequencies below and above 100 kHz for material 1.49% SWCNTs/PBMA

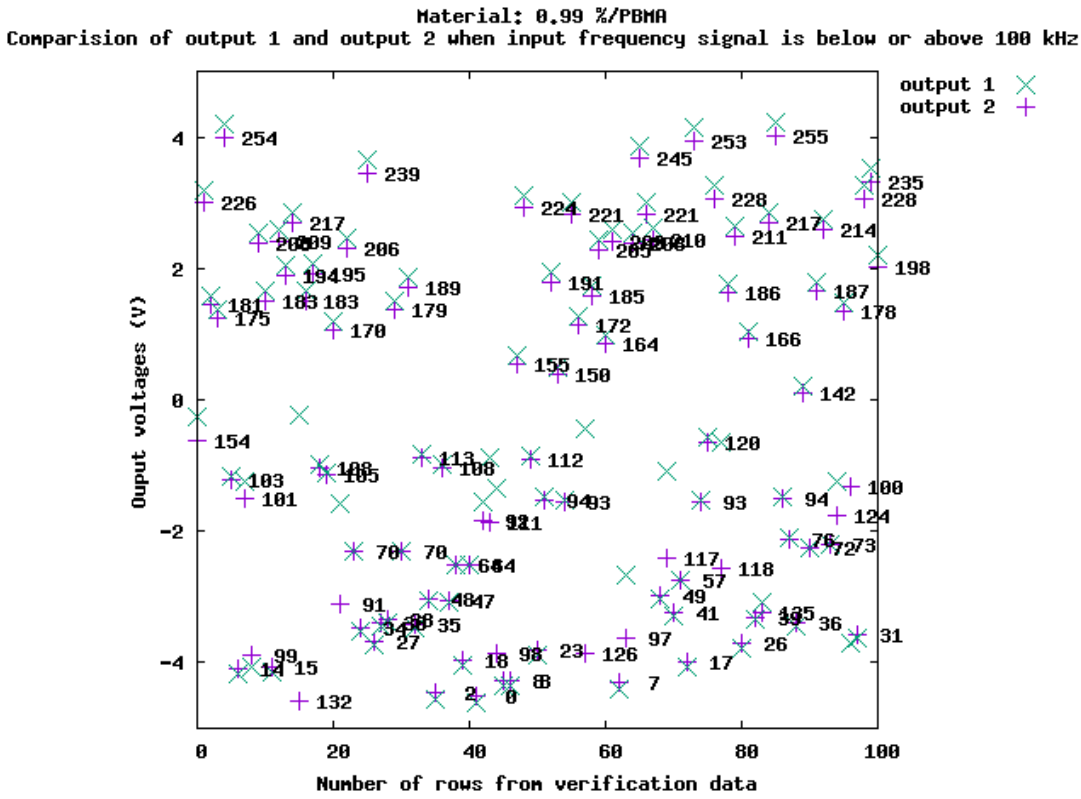


Figure 8.7: Comparison of two outputs at frequencies below and above 100 kHz for material 0.99% SWCNTs/PBMA

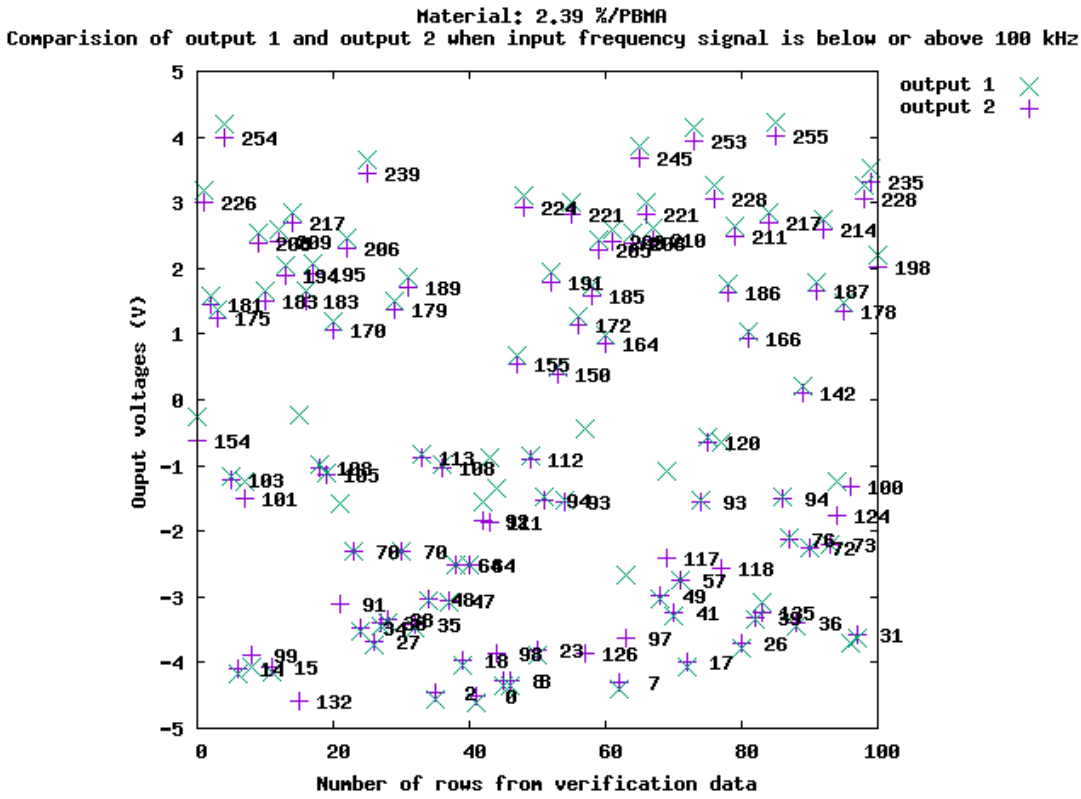


Figure 8.8: Comparison of two outputs at frequencies below and above 100 kHz for material 2.37% SWCNTs/ PBMA

It can be seen that all the three concentrations of SWCNTs, are successfully trained as a tone discriminator. These three materials listed in Table 8.3 with varying concentrations of SWCNTs in PBMA were successfully trained for AND, OR and Half Adder logic circuits and are discussed in detail in Chapter 6, Table 8.3. Hence, it can be concluded that a dense material with the random dispersion of SWCNTs is more flexible where square wave signals can be correctly mapped to the possible outputs, following the outlined interpretative scheme (equation 8.2.2).

These experiments also support the fact discussed in [3] that a certain concentration level of conducting elements (SWCNTs) is required in order to make these materials

wt% SWCNTs/PBMA	Optimal solution: Amplitude for configuration voltages	Pin assignment
0.99%	$V_1 = 127$, $V_2 = 120$, $V_3 = 195$, $V_4 = 329$, $V_5 = 89$, $V_6 = 105$	$x_1 \rightarrow 5$, $O_1 \rightarrow 6$, $O_2 \rightarrow 7$ $V_1 \rightarrow 8$, $V_2 \rightarrow 2$,$V_3 \rightarrow 0$,$V_4 \rightarrow 1$,$V_5 \rightarrow 3$,$V_6 \rightarrow 4$
1.49%	$V_1 = 125$, $V_2 = 97$, $V_3 = 172$, $V_4 = 250$, $V_5 = 83$, $V_6 = 90$	$x_1 \rightarrow 8$, $O_1 \rightarrow 5$, $O_2 \rightarrow 6$ $V_1 \rightarrow 7$, $V_2 \rightarrow 4$,$V_3 \rightarrow 0$,$V_4 \rightarrow 2$,$V_5 \rightarrow 1$,$V_6 \rightarrow 3$
2.39%	$V_1 = 44$, $V_2 = 144$, $V_3 = 79$, $V_4 = 5$, $V_5 = 50$, $V_6 = 16$	$x_1 \rightarrow 4$, $O_1 \rightarrow 3$, $O_2 \rightarrow 5$ $V_1 \rightarrow 0$, $V_2 \rightarrow 1$,$V_3 \rightarrow 2$,$V_4 \rightarrow 6$,$V_5 \rightarrow 8$,$V_6 \rightarrow 7$

Table 8.2: Optimal solutions for different concentrations of SWCNTs in fixed polymer (PBMA) for the solution of tone discrimination problem

wt% SWCNTs/PBMA	Average training accuracy	Average testing accuracy
0.99%	100%	100%
1.49%	100%	100%
2.39%	100%	100%

Table 8.3: Average Training and testing accuracies for tone discriminator problem using different concentrations of SWCNTs from 6 different runs.

behave as a simple computing device. The concentration of SWCNTs in the polymer is directly related to the point where the conductivity starts increasing rapidly, which is the percolation threshold of SWCNTs in the polymer [3]. Below this percolation threshold there are not enough conductive pathways among the SWCNTs network. It should also be stressed that the material is fixed, there is no physical change in the structure of the material. The application of square wave signal affect the conductivity of the material and this change in conductivity is the mechanism for optimising the material to behave as atone discriminator.

8.4 Conclusions

The work presented in this chapter supported the fact that the complex electrical and mechanical nature of SWCNTs/PMMA and SWCNTs/PBMA materials can be used for the solution of a simple computational problem, in this case, a tone discriminator. When trained using the Particle Swarm optimisation algorithm and the purpose-built platform, these material samples can correctly mapped the square wave signals to their possible outputs following the given interpretation scheme.

A typical training and verification process is followed to train the material and the results are reported. It is shown that the method successfully trained

all the considered material samples as the tone discriminator, without having the internal knowledge of the material samples.

Three different concentrations of SWCNTs in a fixed polymer ratio are used to study the effect of concentration of SWCNTs for the solution of tone discrimination problem. All three concentrations were successfully trained for the given problem. This supported the fact, that a dense network of SWCNTs is required for computation like tone discrimination. This type of network provides sufficient conductive pathways between the SWCNTs network, in order to train these networks for simple computational tasks.

The work provided an insight that the SWCNTs/polymer composites can be explored further for other computational problems. Similarly more conductive networks, based on liquid host material may offer more flexibility.

Chapter 9

Training SWCNTs/Polymer composites as a data classifier

9.1 Introduction

The experiments presented in the previous chapters showed that the EIM methodology has a broad scope which can be explored in four dimensions: (a) different types of materials (b) physical properties of materials (c) the computational problem addressed and (d) the optimisation algorithm used for solving the corresponding training problem. The experiments presented in this chapter explored the potential of EIM technique for the solution of machine learning problems. SWCNTs with two polymers are used for solution binary data classification and multiple-class data classification. In separate experiments, different concentrations of SWCNTs in fixed polymer ratio are used to study the complex nature of voltage/current relationship of the material and its effect on the solution of the classification problem. Previously, Genetic Algorithms have been used to successfully solve data classification problems using EIM [115]. The approach was evaluated only on Iris and Lenses datasets [182] using material 0.71% (weight % fraction of PMMA).

The data classification problem is solved using Particle swarm optimisation algorithm using the purpose-built platform, Mecobo. The experiments started with training the SWCNTs based materials as binary data classifiers using different binary data sets with an error minimisation objective for PSO. Subsequently, the solution, which is a combination of SWCNTs material and the optimal inputs is verified and the results are reported.

The experiments also reported the results of different classifiers, which showed the success of kNN method and kNN ball tree method as an effective method of classification. Later, the SWCNT based materials were trained with multi-class data sets using kNN ball tree algorithm. This chapter also reports the comparison of SWCNTs concentration in fixed polymer and its effect on solution of different classification problems.

The results of these experiments demonstrated that EIM methodology can be successfully used to train the SWCNTs based materials to a state where data classification can be performed and is verified by several tests.

The following sections discuss the different classification rules, training methodology and the optimisation problem formulation and the results of these experiments in detail.

9.2 Classification rule 1: Comparison of output with a threshold value

The EIM methodology is used in order to train the SWCNTs based materials as data classifiers. The methodology follows that various inputs signals V_{in} are applied on to the material body and an output response is gathered from the material. In order to classify the new instance (output from the material) a classification rule is implemented. A classification rule assigns the new data

items to the set of predefined classes [183]. The experiments described in this chapter implemented different classification rules for data classification which are as follows.

The first rule used in experiments describe in this chapter compares the output V_{out} recorded from the material with a fixed threshold value θ and the class of input value V_{in}, i.e. $C(V_{in})$ is decided according to the following rule.

$$C(V_{in}) = \begin{cases} C_1 & \text{if } V_{in} < \theta \\ C_2 & \text{otherwise} \end{cases} \tag{9.2.1}$$

9.3 Classification rule 2: Comparison of two outputs

The second rule compares the values of two outputs V_{out1}, V_{out2} recorded from the material, when the inputs signal V_{in} is applied on it. The class of the input signal $(C(V_{in}))$ is decided according to following rule.

$$C(V_{in}) = \begin{cases} C_1 & \text{if } V_{out1} < V_{out2} \\ C_2 & \text{otherwise} \end{cases} \tag{9.3.2}$$

9.4 Classification rule 3: kNN algorithm

The Nearest Neighbour (NN) technique is a simple and highly efficient in classification. It is efficiently and effectively used in text categorisation, pattern recognition and object recognition. The kNN classifier makes prediction by searching the training data set for the k nearest neighbours (where k usually is a small positive integer). The ties in nearest neighbours are usually resolved either randomly or choosing only one nearest neighbour. The latter approach is adapted for the experiments describe in this chapter.

The algorithm starts by storing the training instances and their respective class labels. The next step is the classification, where an unlabelled instance (which

in this case are the outputs from the material) is classified by assigning a class label which is the most frequent among the k training instances nearest to the unlabelled instance. The nearest neighbours are determined by calculating the distance between the query instance from all the instances in the data set. The most commonly used distance metric, Euclidean distance is used here to determine the distances. It calculates the distance between two points $p = (p_1, p_2, \cdots, p_n)$ and $q = (q_1, q_2, \cdots, q_n)$, according to the following equation.

$$d(p,q) = d(q,p) = \sqrt{(q_1 - p_1)^2 + q_2 - p_2)^2 + \cdots q_n - p_n)^2} = \sqrt{\sum_{i=1}^{n}(q_i - p_i)^2} \tag{9.4.3}$$

9.5 Classification rule 4: kNN ball tree algorithm

Although the kNN is an effective and simple technique, however, with the simplicity being the major advantage, this technique comes at the cost of computer complexity and memory. In order to overcome these limitations many other techniques are developed which are broadly classified in to structureless and structure based techniques.

For example, given a query point q, a common task is to search for the k closest points to q among all points in a dataset. Similarly, one might want to get all points whose distances to q are less than the radius r (i.e. range queries). Such queries can be answered using a space-partitioning data structure, such as KD-tree and Ball tree [5]. The experiments describe in this chapter implement kNN Ball tree algorithm. The detail description of this rule is as follows.

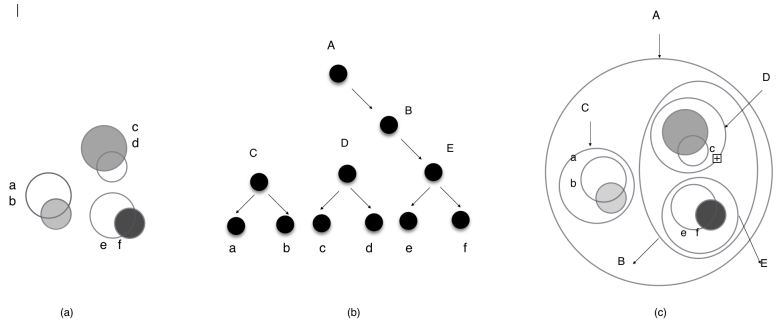


Figure 9.1: (a) Set of balls in a plane (b) a corresponding binary tree for these ball (c) Subsets of balls in a ball tree

9.5.1 Ball tree

In general, a ball-tree [184] is a binary tree in which every node defines a D-dimensional hypersphere or a ball, containing a subset of the points to be searched. Each node of the tree represents a ball, that is a hyper-spherical partition (e.g. a circle in 2D space). While the balls themselves may intersect, each point is assigned to one or the other ball in the partition according to its distance from the centre of the ball. Figure 9.1 illustrates the concept of a 2-dimensional ball tree.

Each node in ball tree represents a set of points named as Points (Node). The root node in any given data set represents all the set of points in the dataset. A node can be a leaf node or a non-leaf node. A leaf node contains a list of the point represented by the node. A non-leaf node does not have explicit set of points and have two child nodes: *Node.lc and Node.rc*, where

$$Points(Node.lc) \cap Points(Node.rc) = \phi \tag{9.5.4}$$

$$Points(Node.lc) \cup Points(Node.rc) = Points(Node) \tag{9.5.5}$$

9.5.2 Ball tree partitioning

In order to build a hierarchical ball tree the key point is to partition each node. A typical way is to choose a pivot among the data points of the node. A pivot may be one of the data points of the node or it can be centroid of $Points(Node)$ Suppose x.lpvt and x.rpvt are two chosen to be pivot points in Points(Node). Ideally, the distance between these two points is the largest of all pair distances in Points(Node). i.e

$$||x.lpvt - x.rpvt|| = max_{p1,p2\epsilon Points(Node)}||p1 - p2|| \tag{9.5.6}$$

After x.lpvt and x.rpvt are found, the *Node* will be partitioned. A common strategy is to project all the points to a vector $\vec{u} = x.\vec{l}pvt - x.\vec{r}pvt$ and find a median point N. In the next step all the points projected to the left of N are assigned to left child node $Node.lc$ and the points projected to the right of N will be assigned to right child node $Node.rc$.

A $d-1$ dimensional plane, denoted by L, is orthogonal to the vector $\vec{u}$ and passes through median N act as a decision boundary. It separates all the points to its left as $Node.lc$ and all the points to its right as $Node.rc$, as seen in Figure 9.2. A median point ensures the depth of tree remains $O(log\ n)$. The python implementation of Ball tree [185] is used in experiments described in this chapter. It implements $1/2(x.\vec{l}pvt - x.\vec{r}pvt)$, which practically ensures the depth of tree as $O(log\ n)$. Every node $Node$ in ball tree has a hypersphere H with radius $Node.radius$ which encapsulates all the points in $Points(Node)$. The points are centred at the hypersphere's centre $Node.center$, such that:

$$Points(Node) \subseteq H(Node.center, Node.radius)$$

The hypersphere of two child nodes may overlap.

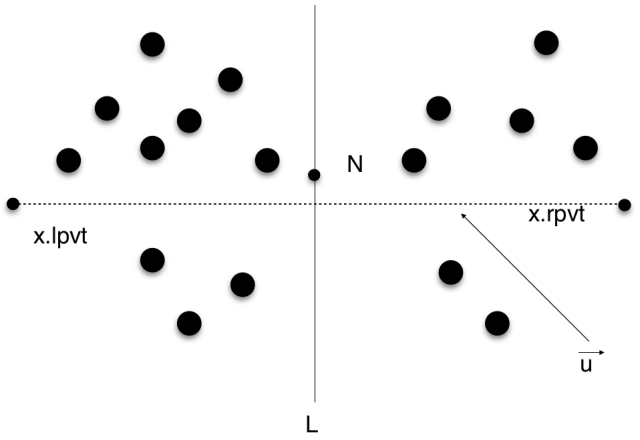


Figure 9.2: Partitioning in ball tree

9.5.3 Search in ball tree

The objective of kNN search in ball tree is to find 'k' closest points to a query point 'q' by using a distance metric. The distance metric used in kNN Ball tree method for current experiments is 'Minkowski distance d', which is calculated between the two points $p = [p_1, p_2, \cdots, p_n]$ and $q = [q_1, q_2, \cdots, q_n]$ as follows:

$$d(p, q) = \Big(\sum_{i=1}^{n} |p_i - q_i|^c \Big)^{1/c}$$

. The kNN ball tree algorithm starts its search by examining the root node first. The decision boundary 'L' is used to decide which child node to search first. If the query point 'q' lies to the left of 'L' then left child node will be searched otherwise right child node will be searched. The experiments in this chapter implemented the simplest search algorithm known as KNS1 for searching. The KNS1 keeps a queue 'Q' which keeps k nearest points visited so far. At each node 'Node' following three operations are performed and the 'Q' is updated accordingly.

- If the distance of the query point 'q' to currently visited node 'Node' is greater than the maximum distance kept in 'Q', then the node will be ignored

and 'Q' will not be updated.

- if the node visited is the leaf node, then calculate the distance of query point 'q' to every point in leaf node. Update 'Q' and return an updated 'Q'
- If the node visited is an internal node then the kNN search algorithm (see pseudo code below) will be called recursively on two children, it first searches the child who is closest to query point 'q' and return the updated 'Q'.

The pseudo code for the KNN ball tree algorithm is as follows:

Pseudo code

Function Knn-Ball-tree

Input:

Query point: **'q'**

Number of nearest neighbours: **'k'**

Maximum first queue containing the **'k'** closest points: **Q**

The node, a hypersphere in the ball tree: **(Node)**

Output:

Q, after visiting the Node **(Node)**

```
if distance(q, Node.pivot) ≥ distance(q, Q.first) then
    return Q unmodified
else if Node is a leaf node then:
    for each point p in Node do:
        if distance(t, p) < distance(t, Q.first) then
            add p to Q
            if size(Q)>k then
                remove the furthest neighbour from Q
            end if
        end if
```

repeat

else

let **child1** be the child node closest to **q**

let **child2** be the child node furthest from **q**

Knn-Ball-tree(**q, k, Q, child1**)

Knn-Ball-tree(**q, k, Q, child2**)

end if

end **Function** [186]

9.5.4 Training problem formulation

In order to train the material as a binary classifier an optimisation problem is formulated with the hardware in the loop. The training determines the class of the input data V^{in} which is in the form of voltage signals. The number of inputs, outputs and configuration signals depends on the data set considered for training/testing. As each optimisation problem is expressed in terms of its parameters and decision variables, therefore, for the current classification problem these are listed as follows:

Parameters:

- Number of inputs n (number of attributes used from the data set)
- Number of outputs m, where $n = m$
- Number of configuration voltages j
- Upper and lower bounds for configuration voltages i.e $V_{q_{min}}, V_{q_{max}}$. For current experiments ($V_{q_{min}} = 0$, $V_{q_{max}} + 5V$)
- Training data set T_r, where a training vector is of the form $A = [A_1, \cdots, A_n]$

- Number of training instances 'K' used for training the material.

Decision variables:

- Configuration voltage values $V_q, q = 1, 2, \cdots, j$ used for affecting the measurements at the material's output locations.
- Pin assignment, $[P_1, \cdots, P_{n+q+m}]^T$

9.5.5 Percentage classification error (PCE)

The contact locations of inputs, outputs and configurations signals are decided by the optimisation algorithm before their application and evaluation. The optimisation's decision vector is defined as follows.

$$X = [V_1, V_2, \cdots, V_{n+q+m}, P_1, \cdots, P_{n+q+m}]_r^T \tag{9.5.7}$$

where $P \in \mathbb{N}$ represents the set of possible pin assignment. For the specific configuration voltages V_q and the pin assignment, the material response is recorded at m output locations. This response is then used in comparison scheme for deciding the class of V_{in}.

The output response is a voltage value which is passed on to the classification rule to predict the class of given instance V^{in}. For example, the kNN ball tree finds the k nearest neighbour of the outputs and returns the class that is the class of majority k instances.

During the training and testing phase the Percentage Classification Error (PCE) is calculated, which is the percentage of incorrectly classified patterns of train or test data sets. Each predicted outcome is compared with the actual class by using the classification rule and if they are not exactly the same, the pattern is said to be incorrectly classified. It is calculated for all the train and test data set separately and the total incorrectly classified pattern number is the

percentage with respect to the size of train and (or) test data. For example, the PCE calculated during the training phase is given by following equation.

$$PCE_{(tr)} = 100 \times \frac{\text{Number of misclassifies samples}}{\text{size of training data}} \tag{9.5.8}$$

And the optimisation aims at finding an optimal value of 9.5.5 AND minimising the following

$$\min J = \sum_{i=1}^{K} \frac{\text{Number of misclassifies samples}}{\text{size of training data}} \tag{9.5.9}$$

subject to voltage constraints $V_{q_{min}}, V_{q_{max}}$, pin assignment and the classification rule. The experiments described in this chapter implemented the PSO algorithm with swarm size 7 and maximum number of iterations as termination criteria are used to solve the optimisation problem.

9.6 Testing Phase

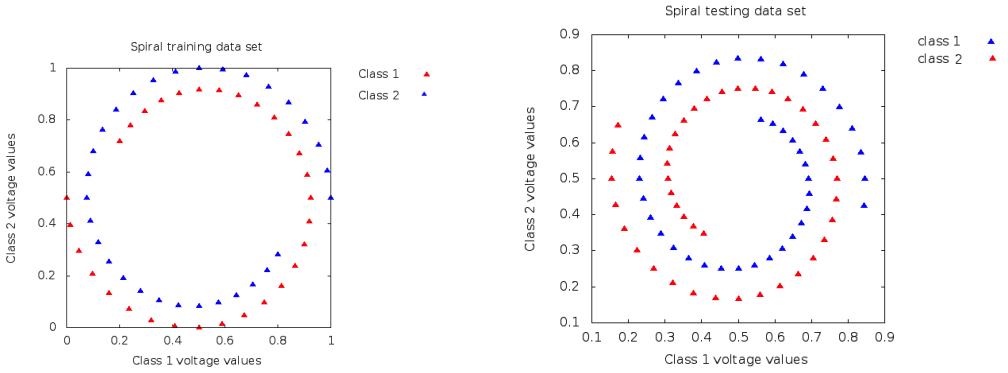
The SWCNTs/polymer based materials are trained as data classifiers by solving the optimisation problem describe in above section. After the successful optimisation process which is when the desired fitness value is achieved, testing is performed. During the test phase, the material is charged with the optimal configuration voltages, using the optimal pin configurations (9.5.5), the attributes from the test data set are applied as inputs. The output is collected from the material while its being charged with the configuration and input voltages and the selected classification rules classify this instance. In the case of KNN ball tree algorithm, the algorithm calculates the nearest neighbour of given instance using training data. It should be noted that test data set is not used to calculate nearest neighbours during the testing phase. The results of this methodology with various classification rules and datasets are discussed in the following sections.

9.6.1 Test problems

Binary class test problems

Spiral dataset:

The Spiral data set is a binary dataset with 193 train and 193 test instances. These instances belong to one of the spirals on a $2D$ surface. The data set is one of the benchmark in machine learning problems. It was developed by Lang and Witbrock [187]. The x-y plots of the train and test dataset are shown in Figure 9.3a, 9.3b. For current experiments, only 120 instances are used from the full dataset, where 60 instances are used for training phase and 60 instances are used for the testing phase.

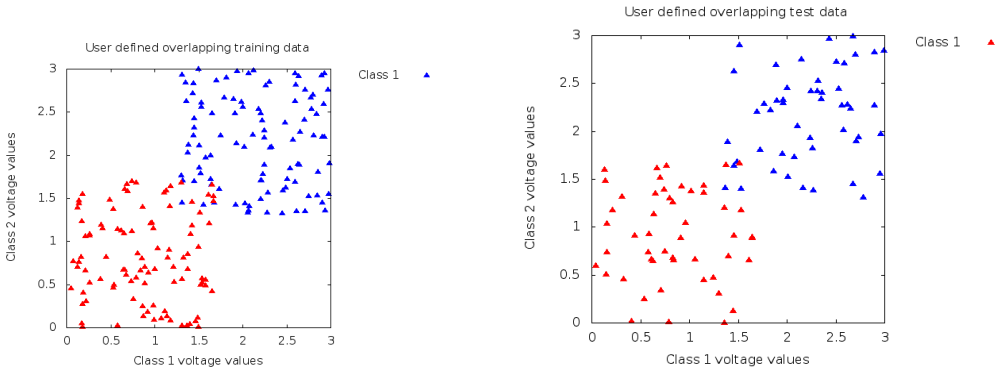


(a) Spiral training data set (b) Spiral test dataset

Figure 9.3: Spiral training and test data set.

Box dataset:

It is a user-defined binary data set with 300 instances. The data set is divided in to training set and testing set with equal distribution of both classes in the both data sets. For current experiments, 200 instances are used for training, where as, 100 instances are used for testing purposes. The x-y plots of training and test



(a) Box training data set (b) Box test dataset

Figure 9.4: Box training and test data set.

data set are shown in Figure 9.4a, 9.4b. It can be seen that the two classes are shaped in the form of squares with an overlapping area of almost 1.5%.

Multiple class test problems

Different test problems were considered for multiple class data classification. All the problems are from Machine Learning repository, UCI Irvine [182]. The problems considered in this study are briefly describe as follows.

The Iris dataset has 150 instances from three Iris flower species which define the three classes and each class has equal distribution of instances in the data set. For current experiment, the data set was randomly divided into 80 training and 70 test instances with approximately equal distribution of each class in each dataset. The 4 attributes of Iris dataset were used as input voltage signals, hence only 4 configuration signals can be applied on to the material. Similarly, 4 output signals were collected.

The Balance data set was generated to model psychological experimental results. Each example is classified as having the balance scale tip to the right, tip to the left, or be balanced. The data set has 625 patterns and 4 real valued

Properties of data sets and experimental parameters					
Datasets	Data	Train	Test	Attributes	Class
Iris	150	90	60	3	3
Balance	625	375	250	4	3
Thyroid	215	129	86	5	3
Appendicitis	106	64	42	7	2

Table 9.1: Description of multiple class data sets and their attributes

attributes and 3 classes. The real value attributes were converted to amplitude values of analogue voltages and were applied on the material. The data set was divided in to training and testing datasets in ratio of 60 : 40, i.e. 375 instances were used as training data and 250 instances were used as test data.

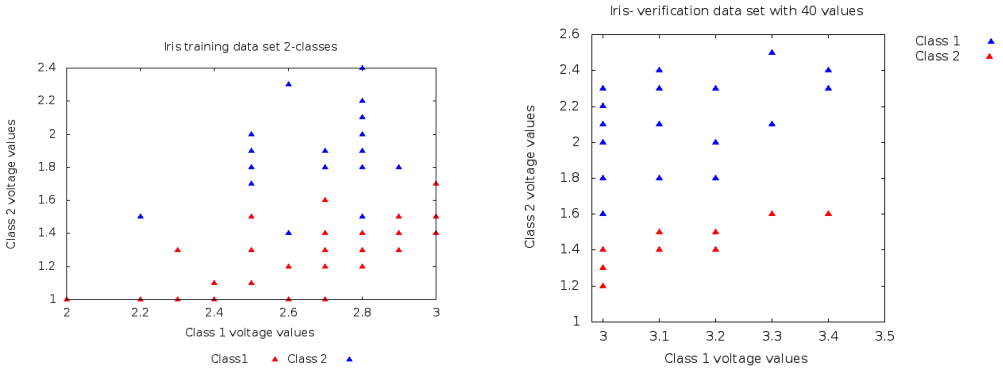
The thyroid data set is about the thyroid disease. The task is to detect if a given patient is normal or suffers from hyperthyroidism or hypothyroidism. The data set has 5 attributes, 3 classes and is based on new-thyroid data which contains 215 patterns. In order to train the material as a data classifier, 5 inputs and 3 configuration signals were used and 5 output locations were used to gather the material's response.

The appendicitis data set has the data of 106 patients where 7 attributes describe the measures taken to classify if the patient has appendicitis or not. It should be noted that there are 7 input signals and 1 configuration signal, which is used to effect the material's conductive behaviour for producing a response that can classify data. The description of these data sets is summarised in Table 9.1.

9.7 Results and discussion

9.7.1 Comparison of four different classification rules for Binary data classification

The binary classification classifies the data elements from a given set into two groups using a classification rule. In the first sets of experiments the material sample (0.1%) SWCNTs/(14.8)% PMMA is trained as a binary data classifier using four different classification rules. The Iris data set with two different classes is used. The original Iris data set has 150 instances which belong to three classes. However, for current experiment, 100 instances which belong to two classes are used from the full dataset. The data set is divided into training and testing data set, where 60 instances are used for training and 40 instances are used for testing purposes. The x-y plots of these training and test dataset are shown in Figure 9.5a, 9.5b:



(a) Iris training data set (b) Iris test dataset

Figure 9.5: Iris training and test data set.

PSO is used to find the optimal configuration voltage values. The swarm size was set as 7, and the termination criteria was the fitness value <= 3 and 100 maximum iterations. These values were kept constant for all the experiments.

The PSO algorithm starts randomly in each run. The value of k is set to 1 in kNN and kNN ball tree algorithm.

In the first experiment, the two inputs ($V_{in} = V_1, V_2$) from the given data set and 6 configuration voltages $V_q, q = 1, \cdots, 6$ are applied on the material and an output response V_{out} is measured at one location. This output response is compared with a user-defined threshold value $\theta = 0.04$ and V_{in} is classified using the rule in equation 9.2.1.

In the second experiment, the classification rule 9.3 is used, where ($V_{in} = V_1, V_2$) from the given data set and 6 configuration voltages are applied on to the material and the response $V_{out} = V_{out(1)}, V_{out(2)}$ is measured at two different locations. These two measurements form the basis of comparison scheme for deciding the class of V_{in} using the equation 9.3.2.

In the third and fourth experiment, classification rules 9.4 and 9.4 are used. ($V_{in} = V_1, V_2$) from the data set along with 6 configuration voltages are applied on to the material and the response $V_{out} = V_{out(1)}, V_{out(2)}$ is collected at two output locations while the inputs and configurations are still being applied. These measurements are passed to kNN algorithm and k nearest neighbours for the given instance are calculated. The class for the given instance is predicted using the class that is represented by maximum numbers of neighbours. The two variations of kNN algorithm i.e. kNN and kNN ball tree in Python's Scipy ® package [185] are used.

The average PCE(s) for training and testing from 5 different runs for each experiment using the four classification rules are listed in Table 9.2. It can be seen that the PCE using kNN and its variant kNN ball tree is much lower as compare to the other two classification rules. Where as, kNN ball tree efficiently classifies the given instances of the Iris data set using the given material. The error is $20 - 23\%$ in case of using first two classification rules, where as it is $8\% - 0\%$ using kNN and

Classification rule	PCE (Training)	PCE (Testing)
Comparison of output with a threshold value	19%	20%
Comparison of two outputs	21%	24%
kNN algorithm	6%	6%
kNN ball tree algorithm	0%	0%

Table 9.2: Classification of Binary data using material: SWCNT(0.1%)/PMMA and three different classification rules

its variant kNN ball tree algorithm. This can be attributed to materials complex electrical properties and their relation to the current classification problem. It can be seen in Figure 9.5a, that there are certain data points of two classes that are not distinguishable by design i.e. the two classes sort of merge in to one another. Although the design difference of two classes is distinguishable in testing data set 9.5b. However, the threshold rule and comparison scheme using two outputs have not effectively work with the material to train it as a binary data classifier.

This can be attributed to the fact that the material is fixed in nature and the optimisation process tries to find effective conductive pathways within the material structure, so that a response can be produced which can be translated into a binary data classifier. Also, the two classes of the data set have some sort of confusion areas between them, therefore, a response that is interpreted into a classifier using a threshold rule does not work. Hence, an effective classifier is required to handle this complexity. The kNN algorithm, on the other hand, tries to find the nearest data point using distance metrics, based on the output response from the material when input and configuration voltages are applied at specific points on the material. Hence, kNN and its variant kNN ball tree algorithm work intelligently with the optimisation process to train the material as a binary data classifier.

Data set	PCE (Training)	PCE (Testing)	Number of iterations	Number of function evaluations
Spiral	0%	0%	61	427
Box	2.8%	4.3%	8	56

Table 9.3: PCE during training and testing phase for various binary data sets

The results of these experiments demonstrated that effective use of SWCNT/PMMA based material for data classification problem using EIM. The materials evolved towards a state where measurements of electrical current can be interpreted for data classification following different classification rules. However, these experiments identified a suitable classification method in the form of kNN in order to train these materials as Binary data classifiers using EIM. The next section demonstrates the successful use of the same methodology and kNN algorithm for two other binary data classification problems.

9.7.2 Binary data classification using kNN ball tree algorithm

In the next set of experiments three different data sets have been used to train the material as a binary data classifier using kNN ball tree algorithm. The description of these data sets is given in section 9.6.1.

The results of training the SWCNT 0.1%/PMMA material using these data sets and kNN ball tree algorithm are given in Table 9.3. In case of spiral data set, the training completed after 68 iterations (428 function evaluations) and 100% accuracy was observed in the test phase. Where as, for the box data set the training completed after 8 iterations and PCE was 2.8% during training phase and 4.3% during testing phase. This error can be attributed to the confusion

present in the data set where two classes mere into one another. It can be seen in figure 9.4a that the two classes shared an area of approximately 1.5%, which makes it difficult for optimisation and classifier to train the material as a classifier for the current problem.

In order to explore this observation further, following set of experiments are performed to study the relationship between varying percentage of confusion present in the data set and its effect on training the material as a binary data classifier.

Comparison of PCE and overlap area of two classes while training the material as a binary data classifier

In the first set, the data sets were generated with three different shared areas between the two classes. In second set of experiments, four different data sets were generated with varied gap between the two classes. The first data set from this set as shown in Figure 9.6d, has neither a gap nor a shared area between the two classes. The x-y plots of these all these datasets are shown in Figure 9.6.

Each data sets consist of 1000 instances and divided in to training and test datasets in ratio of $60 : 40$ respectively, with an equal distribution of both classes in both datasets. The material used is 0.1% SWCNTs/PMMA. and PSO algorithm with swarm size 7 is used where as, maximum 100 iterations and a fitness value $>= 3$ are set as the termination criteria.

In the first set of experiments, the material was trained as a binary data classifier using three different data sets with a shared area of 6.6%, 13% and 16.6%. These overlapped areas can be seen in Figures 9.6a, 9.6b and 9.6c respectively. In other words, the overlapped area was gradually increased and the PCE for training and testing was calculated. For all three considered cases, the training was terminated after maximum number of iterations were reached and PCE for

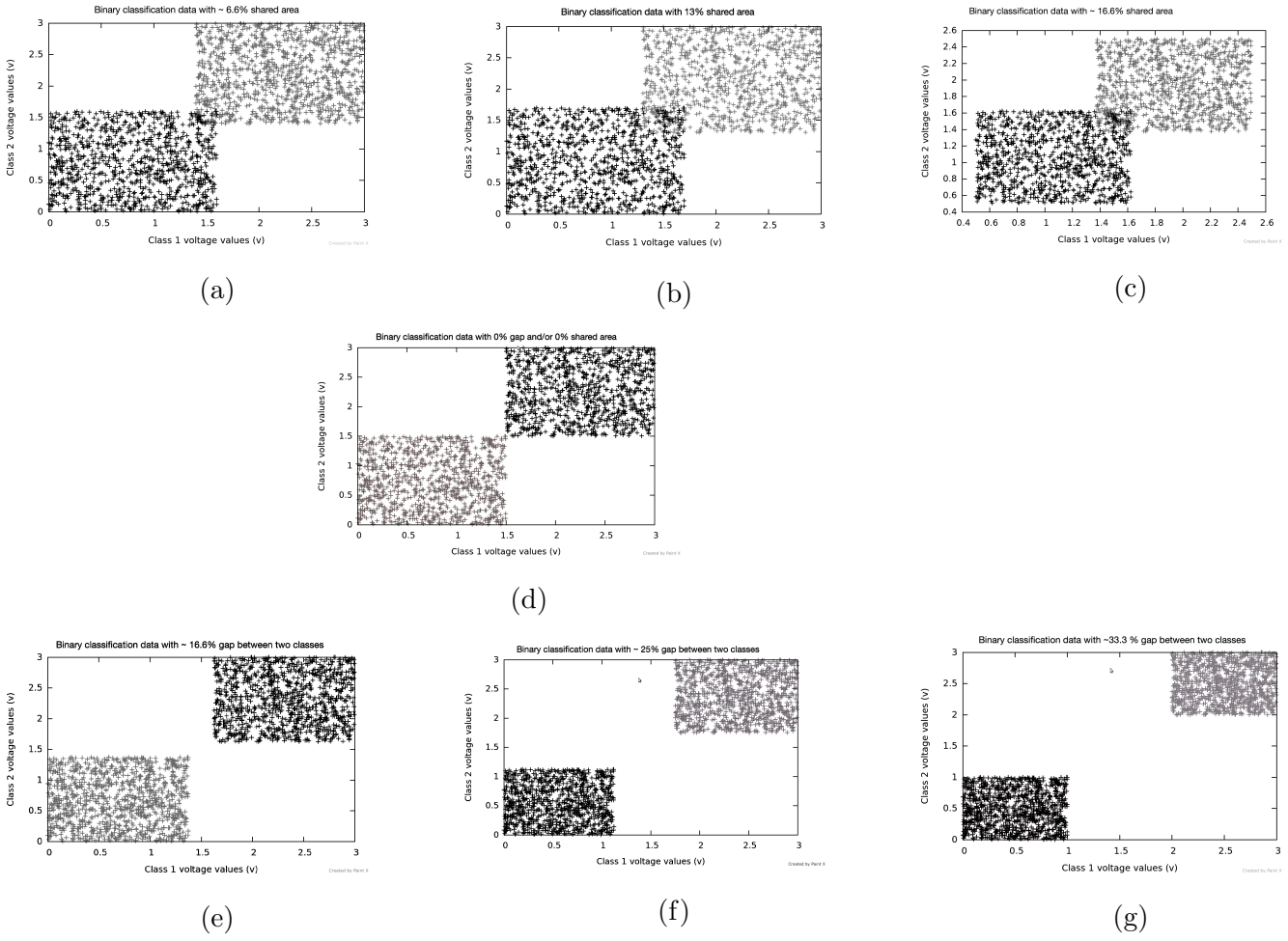


Figure 9.6: Data sets with varied shared areas and varied gaps

training is reported. The results of these experiments are given in Table 9.4. It can be seen that as the confusion was increased in the dataset the training error was also increased and the same is observed during the testing phase.

In the second set of experiments, the first data set was generated where the distance between the two classes is 0%, i.e. there is no shared area but there is no gap as well, as seen in Figure 9.6d. Later, in three different data sets this gap was increased to 16%, 25% and 33.3% respectively as seen in figures 9.6e, 9.6f and 9.6g. The material is trained for each data set and the PCE for training

Percentage overlap of two classes	PCE (Training)	PCE (Testing)	No. of iterations
0.6%	4 %	6%	100
1.2 %	5%	7%	100
1.8 %	7 %	9%	100

Table 9.4: Comparison of percentage overlap area of two classes in the given data set and PPCE

and testing was recorded. These results of four different cases are listed in Table 9.5. It can be seen that the PCE for training and testing remained 0% in all the four cases. The training completed just after first iteration where the gap was 33% and 25%. However, the number of iterations were increased when the gap between the two classes was decreased. This a clear indication that it took longer for the optimisation algorithm to find suitable configurations in order to train the material as a binary data classifier.

These simple experiments shed a light on the SWCNTs/PMMA capability to be trained as a data classifier. However, these experiments also showed that as the confusion in the data set is increased it becomes difficult to manipulate the material's conductive properties to generate an electrical response where data classification can be performed.

Later, in order to explore SWCNT based materials capabilities to solve some other classification problems, different multi-class data sets were also considered. The details of these experiments are discuss later in this chapter.

Percentage gap between two classes	PCE (Training)	PCE (Testing)	No. of iterations
0%	0%	4%	10
0.5 %	0 %	6%	9
0.75 %	0 %	1%	1
1 %	0 %	1%	1

Table 9.5: Comparison of percentage gap between two classes in the given data set and PCE

Comparison of different concentrations of SWCNTs with fixed polymer to train as Binary data classifiers

The results presented in Chapter 6, showed that the concentration of SWCNTs in the polymer affects the materials capabilities to solve computational problems. The results presented in this section describe the relationship between the concentration of SWCNTs in polymer matrix and solution of classification problem. Five different concentrations of SWCNTs in fixed polymer PBMA, were used to solve the problem of binary data classification. The dataset considered was Iris data set with two classes and 100 instances as shown in Figure 9.5a. Particle swarm algorithm was used for finding optimal configurations with swarm size 7 and 100 iterations were set as the termination criteria. The problem was solved using above describe problem formulation and kNN ball tree algorithm. The results of these experiments are listed in Table 9.6 for these material samples.

It was observed that the materials with concentration of SWCNTs lower than 0.99% showed a very high PCE during the training and testing phase. Where as, the PCE was 0% for all other concentrations. This is similar to what has been observed for the solution of logic gates and circuits in Chapter 6, specifically

Binary data classification: Training and test accuracies of different concentrations of SWCNTs in fixed polymer(PBMA)		
Concentration of SWCNTs (wt%)/PBMA	Train	Test
0.51	40%	40%
0.74	40%	40%
0.99	0%	2%
1.49	0%	2%
2.37	0%	2%

Table 9.6: Binary data classification: (PCE) Training and (PCE) test of different concentrations of SWCNTs in fixed polymer(PBMA)

discussed in Section 6.3, Table 6.3. The results of this study reinforced that there is a certain threshold of SWCNTs concentration which is more suitable for the solution of classification problem as well.

9.7.3 Multiple class data classification

In the next step, the experiments have been performed to train the different material samples as multiple class data classifiers. These experiments aimed at evaluating SWCNTs based materials capabilities as data classifiers using different data sets of varying sizes and difficulties. The Mecobo board allows fixed 8 inputs and 8 outputs, therefore the datasets with maximum 7 attributes can be used in these experiments, hence the choice of data set was limited. The materials are trained using the same methodology as used for training the materials as binary data classification. The number of attributes determined the number of inputs

(n) and the remaining locations were used for configuration signals. Where as, the number of outputs (m) is same as number of inputs, i.e. ($n = m$). The PSO algorithm with swarm size 7 was used to provided the values of configuration signals as well as the input, output and configuration pin locations. The kNN ball tree algorithm with $k = 1$ was used as a classification rule. Initially, the material used was 0.1% SWCNT/PMMA. The following sections discuss these experiments in detail.

The description of all the data sets used for multiple class data classification is given in section 9.6.1. For each problem, the attributes of the dataset were applied as input voltages and remaining contact locations were used for the application of configuration signals. The number of output locations were same as number of inputs. The configuration signals effect the material's conductive properties in order to achieve a response that leads to correct kNN ball tree algorithm to predict the respective class for the given instance.

During training and testing phase the PCE was reported, which is the percentage of incorrectly classified patterns of train or test data sets. Each predicted outcome by kNN ball tree algorithm was compared with the actual class and if they were not exactly same, the pattern was said to be incorrectly classified.

The average PCE from five different runs for training and testing for all the considered data sets are listed in Table 9.7. The material sample 0.1% SWCNT/PMMA successfully classify all the considered data sets. For instance, the PCE reported for Iris data during training phase and testing phase is 4.3% and 4.8% respectively, which is almost equivalent to the results classification of Iris data in literature [188]. Similarly, the PCE reported during training for Balance [189], Thyroid [190] and Appendicitis data [191] is 14.4%, 5.5% and 16.7%, respectively. All of these results compare favourably with the classification results of these data sets using kNN in literature [192], [193], [194], [195].

Multiple class data classification: Average PCE for different datasets		
Dataset	Train	Test
Iris	4.4%	4.75%
Balance	14.45%	14.24%
Thyroid	5.45%	6.74%
Appendicitis	15.97%	17.92%

Table 9.7: Average PCE for some benchmark classification problems

In order to observe the role of material as a classifier, and verify the effectiveness of the EIM methodology several different test were performed. All the verification test were conducted with PSO algorithm with swarm size 7, and nearest neighbour $k = 1$ in kNN method.

In the first test, the Iris data set was used to test the efficiency of kNN ball tree method for data classification, but without the involvement of hardware platform and SWCNTs material. The kNN ball tree method effectively classified the Iris data set with PCE 96.8% accuracy in training and 94% accuracy during testing phase. The results of this test are listed in Table 9.8.

In the second test, the above described methodology for training the material sample was used, but this time there was no material sample on the hardware board (Mecobo). The attributes from the data sets were sent as input signals, the configurations are provided by the PSO algorithm and outputs were gathered. These outputs were then passed on to kNN ball tree algorithm to predict the class of given the instance. The results of this test are summarised in Table 9.8. This test resulted in PCE of 40% in both training and testing phase respectively. The results of this test are compared with the result of classification of Iris data using (0.1%)SWCNT/PMMA material sample in Table 9.7, it is clear that the above methodology was unsuccessful without any material. It was the combination of

Verification tests results using Iris data set and kNN Ball tree algorithm	Training error	Testing error
Test 1 (kNN ball tree alone)	96.8%	94%
Test 2 (No material on board)	40%	40%
Test 3 (using 6 volts as constant configurations)	40%	40%
Test 4 (using 0 volts as constant configurations)	40%	40%

Table 9.8: Different tests to verify the effectiveness of methodology used to train the SWCNTs materials as a data classifier

the material itself with its complex electrical and physical properties and optimal inputs signals that resulted in successful classification.

Similarly, in the two other tests, the constant voltages of 0 volts and 6 volts were applied to the material instead of configuration voltages. The attributes from training Iris data sets were applied as input signals and the response was gathered and subsequently sent to kNN algorithm to predict the respective class. In both cases, the experiments were run for several iterations (approximately 20) and a constant error of 40% was observed during training phase. This clearly showed that in the absence of configuration signals the SWCNT based material's complex properties were not explored in a way that can perform classification. Hence, it can be concluded that the specific material with its unique conductive properties as well as the configuration signals are core to train the material as a data classifier. In the last set of experiments different concentrations of SWCNTs in a fixed polymer (PBMA) were trained for above describe classification problems. The results of these experiments are summarised in Table 9.9. It can be seen that the materials with SWCNTs concentration lower than 0.9% failed to perform

as multi class data classifiers. Where as other concentration showed a varying success rate. The results of these experiments are consistent with binary data classification where material samples with SWCNTs concentration lower than 0.99% failed to trained as binary data classifiers. It can be concluded from these experiments that there is a percolation threshold of SWCNTs, below which there are not enough connections among carbon nanotubes that can produce a meaningful voltage response to solve the classification problem.

wt% SWCNTs/PBMA	PCE	Iris	Balance	Appendicitis	Thyroid
0.51%	Train	40%	50%	60%	40%
	Test	40%	50%	60%	40%
0.74%	Train	40%	40%	40%	40%
	Test	40%	40%	40%	40%
0.99%	Train	4.2%	15.2%	18.4%	6.4%
	Test	4.5%	16.7%	19.8%	5.5%
1.49%	Train	4.3%	18.9%	21.3%	7.5%
	Test	5.2%	20.3%	24.4%	8.1%
2.37%	Train	5.5%	15.5%	17.1%	6.4%
	Test	5.8%	15%	18.3%	7.2%

Table 9.9: Multiple data classification: Training and test accuracies of different concentrations of SWCNTs /PBMA

9.8 Conclusions

This chapter presented the results of various investigations of training the SWCNTs/polymer based materials as data classifiers following EIM. Different voltage levels were applied at various location on the material's body which produced a favourable

response to solve various data classification problems.

Different classification rules were considered and their results were compared to be used effectively for training SWCNTs based materials as data classifier. The results showed that kNN and kNN ball tree algorithm performed better as compared to other methods and it was kNN ball tree that produced favourable results that can be compared with data classification results presented in literature for similar problems.

The material with 0.1 wt% SWCNT/PMMA was successfully trained as a binary data classifier as well multi class data classifier. Later, different concentration of SWCNTs in a fixed polymer ratio were used to compare the effect of SWCNTs concentration on solution of classification problem. The material samples with SWCNT concentration below 0.99% performed unfavourably for classification problem, where as, the materials with 0.99%, 1.49% and 2.37% showed a successful response while being trained as a data classifier for various considered data classification problems.

Chapter 10

Conclusions and future work

This main aim of this thesis was to explore SWCNTs based materials for their computational capabilities using the idea of 'evolution in materio'. A set of four research questions was generated in Chapter 1, showing the aim of achieving the objective. Each question is answered in one or more chapters of this thesis.

This chapter provides the overall conclusion of the work done for this thesis and some direction for future exploration areas.

10.1 Conclusions

This thesis demonstrated the successful use of SWCNTs based materials to implement the idea of EIM. It explored this paradigm of unconventional computing in mainly three directions, i.e finding suitable combinations of SWCNTs, finding suitable optimisation methods to train these materials and explore suitable computational problems that can be solved using these materials and following EIM. The work done in this thesis showed that the SWCNTs can be successfully trained to solve simple and complex computational problems. This has been achieved by answering the research questions presented in section 1.1 as follows.

10.1.1 Material systems

Research question What are the suitable nano-material systems for extracting computation?

The SWCNTs are seen as an attractive material in modern electronics due to their electrical, optical and mechanical properties. The thin films of SWCNTs based materials are used in the experiments described in this thesis. Various SWCNTs and polymer based combinations have been studied for three different computational problems. The early experiments with SWCNTs describe in chapter 4, demonstrated that these materials can be trained as simple logic gates/circuits. However, later experiments shed light on the fact that with the more suitable methods of optimisation these materials can be trained to solve some complex logic circuits as well.

Chapter 6 discussed the experiments with different concentrations of SWCNTs in a fixed polymer ratio. These results demonstrated that particular concentration of SWCNTs in a fixed polymer is essential for the solution of a particular computational problem. The detailed study of SWCNTs concentration and its effect on the solution of computational problem suggested that electrical and mechanical properties of SWCNTs are notably different below and above a percolation threshold. The same was observed for training these materials as logic gates/circuits. Three different SWCNTs concentrations were successfully trained as complex logic circuits. However, it was observed that the concentrations below 0.1% were unsuccessful. The three different concentrations were successful trained for the tone discrimination and data classification problems as well.

10.1.2 Suitable hardware

Research question What kind of hardware and software setup is appropriate to achieve the desired outcomes?

The initial experiments with different SWCNTs were performed to solve logic gates problem with a very basic setup. The results of these experiments have been discussed in Chapter 5. However, the experiments described in 5.3.5, demonstrated that the point of application of voltage signals on a material has a significant effect on the solution of logic gate problem. This highlighted the need of using a more flexible hardware, where the application of signals and various other factors affecting the properties of SWCNTs can be put under optimisation control. Later, the experiments performed with the Mecobo board was used which allowed the flexibility to put the application of signals under optimisation control. This hardware has helped to solve various complex computational problems such as complex logic circuits (chapter 8), various data classifications problems (chapter 9).These experiments demonstrated Mecobos functionality and suitability as an interface for evolvable materials.

10.1.3 Optimisation algorithms

Research question What are the suitable investigative methods for evolving the material? i.e. what kind of evolutionary/optimisation processes/algorithms can be used for different computational tasks?

The other line of this study was directed towards the exploration of different optimisation techniques. It aimed at developing and customising optimisation techniques for training the material to perform selected computational tasks. Initially, a very basic Nelder-Mead algorithm NM was used to solve basic logic gate problem. Later, it was observed that the NM algorithm was not very efficient when called to address more complex logic gate/circuits. In contrast, the other two population based algorithms i.e. Differential evolution and Particle Swarm Optimisation Algorithm proved to be successful in implementing complex computational problem with different SWCNTs based materials. PSO with SPV

rule was successfully used for the solution of complex logic circuits, tone discrimination problem and data classification problems.

10.1.4 Computational problems

Research question What are the suitable tasks/methods for extracting computation from the nano material systems?

Another line of investigation was to identify suitable computational problems and matching them with suitable SWCNTs based materials. The various computational problems of varying difficulty are studied with SWCNTs based materials. It was the first time that various complex logic circuits were implemented with SWCNTs based material using the EIM technique. While the work of this thesis was being carried out simple logic gates and an XOR gate using Genetic Algorithm and EIM were solved using mixed signals (square wave and analogue voltages) on Mecobo 3.0 [2]. Although tone discrimination problem was implemented with Liquid crystals using EIM, the problem was re-implemented with more difficulty. Instead of using two frequencies, 100 different frequency were randomly generated and material was trained to show a different response to frequencies below and above 100 KHz.

Similarly, data classification problem was studied with SWCNTs based materials. Various benchmark problems with varied difficulty are studied with these materials. It was the first time that k-nearest neighbour method and its different variations were studied as data classification rule. The results of these experiments showed that these techniques yielded in results which are more closer to the results of these benchmark problems found in literature.

Most of the material combinations were successful for simple computational problems, such as simple logic gates and tone discrimination problem. However, it was observed that only specific concentrations of SWCNTs and polymer combinations

were successful with more complex problems, such as complex logic circuits and multiple class data classification problems.

These results of these experiments have robustly supported the hypothesis presented in Chapter 1, as physical properties of SWCNTs based materials were exploited using computer controlled evolution to solve different computational problems. The work outlined in this thesis has been presented in some publications through conferences and workshop proceedings and journal papers. Of these, six papers have already been published.

10.2 Thesis conclusion

To summarise, this research work presented the successful use of SWCNTs based materials to solve various computer problems using the idea of EIM. It also presented the results of using Graphene based materials for the solution of Logic gates problem.

The study showed that a different problem formulation may be required to solve different computational problems. The threshold interpretation scheme was successful for the solution of Logic gates/ circuits problem. However, the study of data classification problem highlighted that due to complex nature of SWCNTs, the k-NN method of classification data is more suitable for these kinds of problems.

The work presented in this thesis also reported the implementation of evolutionary algorithms, specially PSO for EIM.

It has been found out that a certain percolation threshold of SWCNTs in the polymer is necessary to solve computational problems of varying difficulty. This concentration is directly related to the point where the conductivity of the material substrate starts increasing rapidly, which is the percolation threshold of the conductive network within the polymer. This indicated a clear link between

the material's physical properties and its ability to perform a computation.

The research also highlighted the importance of a flexible hardware platform to use for the further study of EIM.

10.3 Suggestions for future work

The work presented in this thesis has just touched the surface of this field of computing. The analysis of the results showed the potential of this study and provided a direction for future work. It is the first time that SWCNTs/polymer combinations and some graphene-based materials have been studied for various computational problems. The results of these experiments showed that these material systems can be studied for various other computational problems. Therefore, it is worth trying other SWCNTs/polymer combinations for future work. Similarly, there are materials which are conductive, exhibit non-linearity in response and have the properties of being configurable. Such material systems can lead to building useful systems.

The materials studied in this thesis have been mainly explored for their conductive properties and their use in computation. The thesis also reported the results of tone discrimination problem using the combination of square wave signals and static voltages. Similarly, different signal representations are evaluated in a different study for the solution of graph colouring problem [121]. A detail study of square wave signals for evolving CNTs based materials for computational problem $\frac{num}{den}m$ is presented in [120].

Future experiments may explore other physical properties of the materials to solve various computational problems. This can be done by exploring only one physical property such as temperature sensitivity or combination of various other physical properties. It would be interesting to see if the selection and combination

of these properties are put under optimisation control.

There are very few computational problems that have recently been studied with SWCNTs based materials using EIM, apart from computational problems presented in this thesis. These include machine learning classification problems [115], bin packing problems [112], function optimisation [113], evolving robot controllers [117], travelling sales man problem [118]. This list may have been updated while this thesis is being written and submitted. However, there are numerous other computational problems, of varying complexities and difficulties that can be studied with these novel materials. These studies will help to identify computational problems and their respective complexities, that are more suitable for studied with EIM.

The computational problems reported in this thesis can be studied further. For example, more complex logic circuits can be implemented, various other benchmark classification problems can be studied.

There were various limitations while studying different computational problems. The electrode slides consisted of either 12 or 16 electrodes. This limits the selection of computational problems with more inputs and output combinations. Similarly, Mecobo allowed only 8 input and 8 output locations, this also limited the selection of the computational problem. For example, there are many benchmark data classification problems with varying attributes, that require more input and output terminals, but with Mecobo and a limited number of electrodes on an electrode slide, this choice was limited to very few classification problems.

This thesis reported the results with one optimisation algorithm (Nelder-Mead algorithm) and two evolutionary algorithms (DE and PSO) for the current experiments. Previously, genetic algorithms were used for experiments with Liquid crystals [153], [40], [38], [39]. The experiments presented in [115], [112], [113], [117] used Cartesian genetic programming. Therefore, it is highly recommended to study

various other population-based methods and optimisation techniques with these materials and EIM. These investigations will help to identify suitable algorithms for extraction of computation form the materials.

The investigations with materials do not report how fast the material has performed a computation, i.e. the response time of the material. This is particularly useful for building future devices using these materials and techniques. Future investigations into this matter will give helpful insights. Similarly, the stability of results has been observed over few months time for all the computational problems. However, the stability should be tested for longer periods of time. The variations should be observed and the factors affecting the stability should also be investigated.

Bibliography

[1] NASCENCE project (ICT 317662), "Report on materials systems and electrical behaviour," 2013, deliverable D1.1.

[2] O. R. Lykkebø, S. Harding, G. Tufte, and J. Miller, "Mecobo: A hardware and software platform for in materio evolution," in *Unconventional Computation and Natural Computation*, ser. Lecture Notes in Computer Science, O. H. Ibarra, L. Kari, and S. Kopecki, Eds. Springer International Publishing, 2014, vol. 8553, pp. 267–279. [Online]. Available: http://dx.doi.org/10.1007/978-3-319-08123-6_22

[3] M. Massey, A. Kotsialos, F. Qaiser, D. Zeze, C. Pearson, D. Volpati, L. Bowen, and M. Petty, "Computing with carbon nanotubes: Optimization of threshold logic gates using disordered nanotube/polymer composites," *Journal of Applied Physics*, vol. 117, no. 13, p. 134903, 2015.

[4] F. Cancare, S. Bhandari, D. B. Bartolini, M. Carminati, and M. D. Santambrogio, "A bird's eye view of fpga-based evolvable hardware," in *Adaptive Hardware and Systems (AHS), 2011 NASA/ESA Conference on*. IEEE, 2011, pp. 169–175.

[5] J. F. Miller, S. L. Harding, and G. Tufte, "Evolution-in-materio: evolving computation in materials," *Evolutionary Intelligence*, vol. 7, no. 1, pp. 49–67, 2014.

[6] S. Carrara, "Nano-bio-technology and sensing chips: new systems for detection in personalized therapies and cell biology," *Sensors*, vol. 10, no. 1, pp. 526–543, 2010.

[7] A. K. Geim and K. S. Novoselov, "The rise of graphene," *Nature materials*, vol. 6, no. 3, pp. 183–191, 2007.

[8] rgo synthesis illustration. [Online]. Available: https://www.utu.fi/en/units/sci/units/chemistry/research/mcca/PublishingImages/GO%20rGO.jpg

[9] B. J. Copeland, "What is computation?" *Synthese*, vol. 108, no. 3, pp. 335–359, Sep 1996. [Online]. Available: https://doi.org/10.1007/BF00413693

[10] A. M. Turing, "On computable numbers, with an application to the entscheidungsproblem," *J. of Math*, vol. 58, no. 345-363.

[11] J. Von Neumann, "First draft of a report on the edvac," *IEEE Annals of the History of Computing*, no. 4, pp. 27–75, 1993.

[12] S. Stepney, S. L. Braunstein, J. A. Clark, A. Tyrrell, A. Adamatzky, R. E. Smith, T. Addis, C. Johnson, J. Timmis, P. Welch *et al.*, "Journeys in non-classical computation i: A grand challenge for computing research," *International Journal of Parallel, Emergent and Distributed Systems*, vol. 20, no. 1, pp. 5–19, 2005.

[13] S. Stepney, "Introduction to unconventional computing," in *Guide to Unconventional Computing for Music*. Springer, 2017, pp. 1–21.

[14] S. B. Cooper, B. Löwe, and A. Sorbi, *New computational paradigms: Changing conceptions of what is computable*. Springer Science & Business Media, 2007.

[15] G. Rozenberg, T. Bck, and J. N. Kok, *Handbook of natural computing*. Springer Publishing Company, Incorporated, 2011.

[16] L. N. de Castro and F. J. Von Zuben, "From biologically inspired computing to natural computing," *Recent developments in biologically inspired computing*, pp. 1–8, 2004.

[17] P. Marrow, "Nature-inspired computing technology and applications," *BT Technology Journal*, vol. 18, no. 4, pp. 13–23, 2000.

[18] N. Kurzawa, C. Summerfield, and R. Bogacz, "Neural circuits trained with standard reinforcement learning can accumulate probabilistic information during decision making," *Neural computation*, 2017.

[19] H. T. Bui, "Localized dna computation," Ph.D. dissertation, 2017.

[20] N. M. Linke, D. Maslov, M. Roetteler, S. Debnath, C. Figgatt, K. A. Landsman, K. Wright, and C. Monroe, "Experimental comparison of two quantum computing architectures," *Proceedings of the National Academy of Sciences*, p. 201618020, 2017.

[21] R. Mansour, "Evolutionary computing enriched computer aided diagnosis system for diabetic retinopathy: A survey," *IEEE Reviews in Biomedical Engineering*, 2017.

[22] M. Dorigo, M. Birattari, X. Li, M. López-Ibáñez, K. Ohkura, C. Pinciroli, and T. Stützle, *Swarm Intelligence: 10th International Conference, ANTS 2016, Brussels, Belgium, September 7-9, 2016, Proceedings*. Springer, 2016, vol. 9882.

[23] P. Arena and L. Patane, "Cellular computation in the insect brain," in *CNNA 2016; 15th International Workshop on Cellular Nanoscale Networks and their Applications; Proceedings of.* VDE, 2016, pp. 1–2.

[24] D. Alistarh, J. Aspnes, D. Eisenstat, R. Gelashvili, and R. L. Rivest, "Time-space trade-offs in molecular computation," Technical Report, Tech. Rep., 2016.

[25] M. Canayaz and A. Karci, "Cricket behaviour-based evolutionary computation technique in solving engineering optimization problems," *Applied Intelligence*, vol. 44, no. 2, pp. 362–376, 2016.

[26] H. Abelson, D. Allen, D. Coore, C. Hanson, G. Homsy, T. F. Knight, Jr., R. Nagpal, E. Rauch, G. J. Sussman, and R. Weiss, "Amorphous computing," *Commun. ACM*, vol. 43, no. 5, pp. 74–82, May 2000. [Online]. Available: http://doi.acm.org/10.1145/332833.332842

[27] J. W. Rozenblit and P. L. Jankowski, "An integrated framework for knowledge-based modeling and simulation of natural systems," *Simulation*, vol. 57, no. 3, pp. 152–165, 1991.

[28] S. Stepney, "Local and global models of physics and computation," *International Journal of General Systems*, vol. 43, no. 7, pp. 673–681, 2014.

[29] J. A. Martín H., J. Lope, and D. Maravall, ""adaptation, anticipation and rationality in natural and artificial systems: computational paradigms mimicking nature"," *"Natural Computing"*, vol. 8, no. 4, pp. 757–775, 2008.

[30] B. J. MacLennan, "Artificial morphogenesis as an example of embodied computation." *IJUC*, vol. 7, no. 1-2, pp. 3–23, 2011.

[31] M. Burgin, "Super-recursive algorithms as a tool for high performance computing," in *Proceedings of High Performance Computing Symposium (San Diego, 1999)*, 1999, pp. 224–228.

[32] A. Adamatzky, P. Arena, A. Basile, R. Carmona-Galán, B. D. L. Costello, L. Fortuna, M. Frasca, and A. Rodríguez-Vázquez, "Reaction-diffusion navigation robot control: from chemical to vlsi analogic processors," *Circuits and Systems I: Regular Papers, IEEE Transactions on*, vol. 51, no. 5, pp. 926–938, 2004.

[33] A. Adamatzky, "Hot ice computer," *Physics Letters A*, vol. 374, no. 2, pp. 264–271, 2009.

[34] J. W. Mills, "The nature of the extended analog computer," *Physica D: Nonlinear Phenomena*, vol. 237, no. 9, pp. 1235–1256, 2008.

[35] L. M. Adleman, "Molecular computation of solutions to combinatorial problems," *Science*, vol. 266, no. 5187, pp. 1021–1024, 1994.

[36] N. C. Seeman, "Dna in a material world," *Nature*, vol. 421, no. 6921, pp. 427–431, 2003.

[37] J. F. Miller and K. Downing, "Evolution in materio: Looking beyond the silicon box," in *Evolvable Hardware, 2002. Proceedings. NASA/DoD Conference on.* IEEE, 2002, pp. 167–176.

[38] S. Harding and J. Miller, "Evolution in materio: Initial experiments with liquid crystal," in *Evolvable Hardware, 2004. Proceedings. 2004 NASA/DoD Conference on.* IEEE, 2004, pp. 298–305.

[39] S. Harding and J. F. Miller, "Evolution in materio: A tone discriminator in liquid crystal," in *Evolutionary Computation, 2004. CEC2004. Congress on*, vol. 2. IEEE, 2004, pp. 1800–1807.

[40] ——, "Evolution in materio: Evolving logic gates in liquid crystal," in *Proc. Eur. Conf. Artif. Life (ECAL 2005), Workshop on Unconventional Computing: From cellular automata to wetware*. Beckington, UK, 2005, pp. 133–149.

[41] ——, "Evolution in materio: A real-time robot controller in liquid crystal," in *Evolvable Hardware, 2005. Proceedings. 2005 NASA/DoD Conference on*. IEEE, 2005, pp. 229–238.

[42] H. Broersma, F. Gomez, J. Miller, M. Petty, and G. Tufte, "Nascence project: Nanoscale engineering for novel computation using evolution," *International journal of unconventional computing*, vol. 8, no. 4, pp. 313–317, 2012.

[43] J. Liu, T. Dietz, S. R. Carpenter, M. Alberti, C. Folke, E. Moran, A. N. Pell, P. Deadman, T. Kratz, J. Lubchenco *et al.*, "Complexity of coupled human and natural systems," *science*, vol. 317, no. 5844, pp. 1513–1516, 2007.

[44] R. Pfeifer, M. Lungarella, and F. Iida, "Self-organization, embodiment, and biologically inspired robotics," *Science*, vol. 318, no. 5853, pp. 1088–1093, 2007.

[45] R. Rosen, *Life Itself - A Comprehensive Enquiry into the Nature, Origin and Fabrication of Life.* New York, NY, USA: Columbia University Press, 1991.

[46] J. W. Mills, M. Parker, B. Himebaugh, C. Shue, B. Kopecky, and C. Weilemann, "Empty space computes: The evolution of an unconventional supercomputer," in *ACM International Conference on Computing Frontiers*, 2006.

[47] S. Lloyd, "Ultimate physical limits to computation," *Nature*, vol. 406, no. 6799, pp. 1047–1054, 08 2000.

[48] S. Stepney, "The neglected pillar of material computation," *Physica D: Nonlinear Phenomena*, vol. 237, no. 9, pp. 1157–1164, 2008.

[49] P. Kahlem and E. Birney, "Dry work in a wet world: computation in systems biology," *Molecular systems biology*, vol. 2, no. 1, p. 40, 2006.

[50] S. Tsuda, K.-P. Zauner, and Y.-P. Gunji, "Robot control with biological cells," in *Sixth International Workshop on Information Processing in Cells and Tissues*, 2005, pp. 202–216, event Dates: Aug. 30–Sept. 1, 2005.

[51] M. Amos, D. A. Hodgson, and A. Gibbons, "Bacterial self-organisation and computation," 2005.

[52] S. Harding, "Evolution in materio," University of York, Tech. Rep., 2005.

[53] S. Prasad, M. Yang, X. Zhang, C. Ozkan, and M. Ozkan, "Electric field assisted patterning of neuronal networks for the study of brain functions," *Biomedical Microdevices*, vol. 5, no. 2, pp. 125–137, 2003.

[54] S. Harding, J. Neil, K. peter Zauner, J. F. Miller, and K. Clegg, "A framework for the automatic identification and extraction of computation from materials."

[55] H.-K. Janssen, "On the nonequilibrium phase transition in reaction-diffusion systems with an absorbing stationary state," *Zeitschrift für Physik B Condensed Matter*, vol. 42, no. 2, pp. 151–154, 1981.

[56] A. M. Turing, "The chemical basis of morphogenesis," *Philosophical Transactions of the Royal Society of London B: Biological Sciences*, vol. 237, no. 641, pp. 37–72, 1952.

[57] A. Adamatzky, B. D. L. Costello, and T. Asai, *Reaction-diffusion computers*. Elsevier, 2005.

[58] A. Adamatzky and B. D. L. Costello, "Experimental logical gates in a reaction-diffusion medium: The xor gate and beyond," *Physical Review E*, vol. 66, no. 4, p. 046112, 2002.

[59] D. Tolmachiev and A. Adamatzky, "Chemical processor for computation of voronoi diagram," *Advanced Functional Materials*, vol. 6, no. 4, pp. 191–196, 1996.

[60] L. A. Rubel, "The extended analog computer," *Advances in Applied Mathematics*, vol. 14, no. 1, pp. 39–50, 1993.

[61] C. G. Langton, "Computation at the edge of chaos: phase transitions and emergent computation," *Physica D: Nonlinear Phenomena*, vol. 42, no. 1, pp. 12–37, 1990.

[62] J. Miller and K. Downing, "Evolution in materio: looking beyond the silicon box," in *Evolvable Hardware, 2002. Proceedings. NASA/DoD Conference on*, 2002, pp. 167–176.

[63] S. H. Mahdavi and P. J. Bentley, "Evolving motion of robots with muscles," in *Applications of Evolutionary Computing*. Springer, 2003, pp. 651–660.

[64] M. Oltean, "Switchable glass: a possible medium for evolvable hardware," in *Adaptive Hardware and Systems, 2006. AHS 2006. First NASA/ESA Conference on*. IEEE, 2006, pp. 81–87.

[65] L. Bottou, "Large-scale machine learning with stochastic gradient descent," in *Proceedings of COMPSTAT'2010*. Springer, 2010, pp. 177–186.

[66] J. R. Shewchuk *et al.*, "An introduction to the conjugate gradient method without the agonizing pain," 1994.

[67] T. G. Kolda, R. M. Lewis, and V. Torczon, "Optimization by direct search: New perspectives on some classical and modern methods," *SIAM review*, vol. 45, no. 3, pp. 385–482, 2003.

[68] M. R. Hestenes, "Multiplier and gradient methods," *Journal of optimization theory and applications*, vol. 4, no. 5, pp. 303–320, 1969.

[69] D. E. Goldberg *et al.*, *Genetic algorithms in search, optimization, and machine learning*. Addison-wesley Reading Menlo Park, 1989, vol. 412.

[70] E. Aarts and J. Korst, "Simulated annealing and boltzmann machines: a stochastic approach to combinatorial optimization and neural computing," 1988.

[71] S. S. Rao and V. Singh, "Optimization," *Systems, Man and Cybernetics, IEEE Transactions on*, vol. 9, no. 8, pp. 447–447, 1979.

[72] D. M. Olsson and L. S. Nelson, "The nelder-mead simplex procedure for function minimization," *Technometrics*, vol. 17, no. 1, pp. 45–51, 1975.

[73] J. A. Nelder and R. Mead, "A simplex method for function minimization," *Computer journal*, vol. 7, no. 4, pp. 308–313, 1965.

[74] M. Box, "A new method of constrained optimization and a comparison with other methods," *The Computer Journal*, vol. 8, no. 1, pp. 42–52, 1965.

[75] R. O'Neill, "Algorithm as 47: function minimization using a simplex procedure," *Applied Statistics*, pp. 338–345, 1971.

[76] A. E. Eiben and J. E. Smith, *Introduction to evolutionary computing*. springer, 2003.

[77] J. R. Koza, F. H. Bennett III, D. Andre, and M. A. Keane, "Genetic programming: Turings third way to achieve machine intelligence," *Evolutionary Algorithms in Engineering and Computer Science*, pp. 185–197, 1999.

[78] D. Fogel, *Artificial intelligence through simulated evolution*. Wiley-IEEE Press, 2009.

[79] J. H. Holland, *Adaptation in natural and artificial systems: An introductory analysis with applications to biology, control, and artificial intelligence*. University of Michigan Press, 1975.

[80] H.-P. P. Schwefel, *Evolution and optimum seeking: the sixth generation*. John Wiley & Sons, Inc., 1993.

[81] Z. Michalewicz and M. Michalewicz, "Evolutionary computation techniques and their applications," in *Intelligent Processing Systems, 1997. ICIPS '97. 1997 IEEE International Conference on*, vol. 1, Oct 1997, pp. 14–25 vol.1.

[82] Z. W. Geem, J. H. Kim, and G. Loganathan, "A new heuristic optimization algorithm: harmony search," *Simulation*, vol. 76, no. 2, pp. 60–68, 2001.

[83] M. G. Omran and M. Mahdavi, "Global-best harmony search," *Applied Mathematics and Computation*, vol. 198, no. 2, pp. 643 – 656,

2008. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0096300307009320

[84] C. A. C. Coello, G. B. Lamont, D. A. Van Veldhuizen *et al.*, *Evolutionary algorithms for solving multi-objective problems*. Springer, 2007, vol. 5.

[85] A. Eiben and M. Schoenauer, "Evolutionary computing," *Information Processing Letters*, vol. 82, no. 1, pp. 1 – 6, 2002, evolutionary Computation. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0020019002002041

[86] A. E. Eiben and J. E. Smith, *Introduction to Evolutionary Computing*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2015, ch. Representation, Mutation, and Recombination, pp. 49–78. [Online]. Available: http://dx.doi.org/10.1007/978-3-662-44874-8_4

[87] H. de Garis, "Evolvable hardware genetic programming of a darwin machine," in *Artificial neural nets and genetic algorithms*. Springer, 1993, pp. 441–449.

[88] M. Takechi and T. Tokunaga, "Evolving hardware with genetic learning: A first step towards building a darwin machine," in *From Animals to Animats 2: Proceedings of the Second International Conference on Simulation of Adaptive Behavior*. MIT Press, 1993, pp. 417–424.

[89] H. Garis, "Evolvable hardware genetic programming of a darwin machine," in *Artificial Neural Nets and Genetic Algorithms*, R. Albrecht, C. Reeves, and N. Steele, Eds. Springer Vienna, 1993, pp. 441–449.

[90] A. Thompson, "Silicon evolution," in *Stanford University*. MIT Press, 1996, pp. 444–452.

[91] R. Computing, "The theory and practice of fpga-based computation/ed. by scott hauck and andre dehon," 2008.

[92] X. Yao and T. Higuchi, "Promises and challenges of evolvable hardware," *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, vol. 29, no. 1, pp. 87–97, 1999.

[93] P. C. Haddow and G. Tufte, "An evolvable hardware fpga for adaptive hardware," in *Evolutionary Computation, 2000. Proceedings of the 2000 Congress on*, vol. 1. IEEE, 2000, pp. 553–560.

[94] T. Higuchi, M. Iwata, I. Kajitani, H. Iba, Y. Hirao, T. Furuya, and B. Manderick, "Evolvable hardware and its application to pattern recognition and fault-tolerant systems," in *Towards evolvable hardware*. Springer, 1996, pp. 118–135.

[95] T. Higuchi, M. Murakawa, M. Iwata, I. Kajitani, W. Liu, and M. Salami, "Evolvable hardware at function level," in *Evolutionary Computation, 1997., IEEE International Conference on*. IEEE, 1997, pp. 187–192.

[96] A. Thompson, "An evolved circuit, intrinsic in silicon, entwined with physics," in *Evolvable Systems: From Biology to Hardware*. Springer, 1997, pp. 390–405.

[97] W. Wolf, *FPGA-based system design*. Pearson education, 2004.

[98] D. E. Van den Bout, J. N. Morris, D. Thomae, S. Labrozzi, S. Wingo, and P. Hallman, "Anyboard: An fpga-based, reconfigurable system," *IEEE Design & Test*, vol. 9, no. 3, pp. 21–30, 1992.

[99] A. Thompson, "Evolving electronic robot controllers that exploit hardware resources," in *In*. Springer-Verlag, 1995, pp. 640–656.

[100] D. Keymeulen, K. Konaka, M. Iwata, Y. Kuniyoshi, and T. Higuchi, "Robot learning using gate-level evolvable hardware," in *Learning Robots*. Springer, 1997, pp. 173–188.

[101] P. Haddow and G. Tufte, "Evolving a robot controller in hardware," in *In Proc. of the Norwegian Computer Science Conference (NIK-99)*, 1999, pp. 141–150.

[102] K. Tan, C. Chew, K. Tan, L. Wang, and Y. Chen, "Autonomous robot navigation via intrinsic evolution," in *Evolutionary Computation, 2002. CEC'02. Proceedings of the 2002 Congress on*, vol. 2. IEEE, 2002, pp. 1272–1277.

[103] L. Huelsbergen, E. Rietman, and R. Slous, "Evolution of astable multivibrators in silico," in *Evolvable Systems: From Biology to Hardware*. Springer, 1998, pp. 66–77.

[104] A. Thompson, "On the automatic design of robust electronics through artificial evolution," in *Evolvable Systems: From Biology to Hardware*. Springer, 1998, pp. 13–24.

[105] A. Thompson and P. Layzell, "Evolution of robustness in an electronics design," in *Evolvable Systems: From Biology to Hardware*. Springer, 2000, pp. 218–228.

[106] P. Layzell, "A new research tool for intrinsic hardware evolution," in *Evolvable Systems: From Biology to Hardware*. Springer, 1998, pp. 47–56.

[107] A. Thompson, "On the automatic design of robust electronics through artificial evolution," in *Proceedings of the Second International Conference on Evolvable Systems: From Biology to Hardware*, ser. ICES '98. London,

UK, UK: Springer-Verlag, 1998, pp. 13–24. [Online]. Available: http://dl.acm.org/citation.cfm?id=645508.656773

[108] J.Crooks, "Evolvable analogue hardware, meng project report," The University Of York, Tech. Rep., 2002.

[109] S. Harding and J. Miller, "Evolution in materio: a real-time robot controller in liquid crystal," in *Evolvable Hardware, 2005. Proceedings. 2005 NASA/DoD Conference on*, 2005, pp. 229–238.

[110] ——, "Evolution in materio," in *Computational Complexity*, R. A. Meyers, Ed. Springer New York, 2012, pp. 1030–1042.

[111] ——, "Evolution in materio: a tone discriminator in liquid crystal," in *Evolutionary Computation, 2004. CEC2004. Congress on*, vol. 2, 2004, pp. 1800–1807 Vol.2.

[112] M. Mohid, J. F. Miller, S. L. Harding, G. Tufte, O. R. Lykkebo, M. K. Massey, and M. C. Petty, "Evolution-in-materio: Solving bin packing problems using materials," in *Evolvable Systems (ICES), 2014 IEEE International Conference on*. IEEE, 2014, pp. 38–45.

[113] J. F. Miller and M. Mohid, "Function optimization using cartesian genetic programming," in *Proceedings of the 15th annual conference companion on Genetic and evolutionary computation*. ACM, 2013, pp. 147–148.

[114] M. Mohid, J. F. Miller, S. L. Harding, G. Tufte, O. R. Lykkebo, M. K. Massey, and M. C. Petty, "Evolution-in-materio: A frequency classifier using materials," in *Evolvable Systems (ICES), 2014 IEEE International Conference on*. IEEE, 2014, pp. 38–45.

[115] M. Mohid, J. F. Miller, S. L. Harding, G. Tufte, O. R. Lykkebø, M. K. Massey, and M. C. Petty, "Evolution-in-materio: Solving machine learning classification problems using materials," in *Parallel Problem Solving from Nature–PPSN XIII*. Springer, 2014, pp. 721–730.

[116] M. Mohid and J. Miller, "Solving even parity problems using carbon nanotubes," in *Computational Intelligence (UKCI), 15th UK Workshop on. IEEE Press*, 2015.

[117] ——, "Evolving robot controllers using carbon nanotubes," in *Proceedings of the 13th European Conference on Artificial Life (ECAL2015)*, 2015, pp. 106–113.

[118] K. D. Clegg, J. F. Miller, K. Massey, and M. Petty, "Travelling salesman problem solved in materioby evolved carbon nanotube device," in *International Conference on Parallel Problem Solving from Nature*. Springer, 2014, pp. 692–701.

[119] K. D. Clegg, J. F. Miller, M. K. Massey, and M. C. Petty, "Practical issues for configuring carbon nanotube composite materials for computation," in *Evolvable Systems (ICES), 2014 IEEE International Conference on*. IEEE, 2014, pp. 61–68.

[120] O. R. Lykkebø, S. Nichele, and G. Tufte, "An investigation of square waves for evolution in carbon nanotubes material," in *Proceedings of the 13th European Conference on Artificial Life (ECAL2015)*, 2015, pp. 503–510.

[121] O. R. Lykkebø and G. Tufte, "Comparison and evaluation of signal representations for a carbon nanotube computational device," in *Evolvable Systems (ICES), 2014 IEEE International Conference on*. IEEE, 2014, pp. 54–60.

[122] E. Vissol-Gaudin, A. Kotsialos, M. K. Massey, C. Groves, C. Pearson, D. A. Zeze, and M. C. Petty, "Solving binary classification problems with carbon nanotube/liquid crystal composites and evolutionary algorithms," in *Evolutionary Computation (CEC), 2017 IEEE Congress on*. IEEE, 2017, pp. 1924–1931.

[123] E. Vissol-Gaudin, A. Kotsialos, M. K. Massey, D. A. Zeze, C. Pearson, C. Groves, and M. C. Petty, "Training a carbon-nanotube/liquid crystal data classifier using evolutionary algorithms," in *International Conference on Unconventional Computation and Natural Computation*. Springer, 2016, pp. 130–141.

[124] E. Vissol-Gaudin, A. Kotsialos, C. Groves, C. Pearson, D. Zeze, and M. Petty, "Computing based on material training: Application to binary classification problems," in *Rebooting Computing (ICRC), 2017 IEEE International Conference on*. IEEE, 2017, pp. 1–8.

[125] M. S. Dresselhaus, G. Dresselhaus, and P. C. Eklund, *Science of fullerenes and carbon nanotubes: their properties and applications*. Academic press, 1996.

[126] B. Yakobson, G. Samsonidze, and G. Samsonidze, "Atomistic theory of mechanical relaxation in fullerene nanotubes," *Carbon*, vol. 38, no. 11, pp. 1675–1680, 2000.

[127] J. W. Wilder, L. C. Venema, A. G. Rinzler, R. E. Smalley, and C. Dekker, "Electronic structure of atomically resolved carbon nanotubes," *Nature*, vol. 391, no. 6662, pp. 59–62, 1998.

[128] A. D. Franklin, M. Luisier, S.-J. Han, G. Tulevski, C. M. Breslin, L. Gignac, M. S. Lundstrom, and W. Haensch, "Sub-10 nm carbon nanotube transistor," *Nano letters*, vol. 12, no. 2, pp. 758–762, 2012.

[129] A. D. Franklin, "Electronics: The road to carbon nanotube transistors," *Nature*, vol. 498, no. 7455, pp. 443–444, 2013.

[130] L. Wei, D. J. Frank, L. Chang, and H.-S. Wong, "A non-iterative compact model for carbon nanotube fets incorporating source exhaustion effects," in *Electron Devices Meeting (IEDM), 2009 IEEE International*. IEEE, 2009, pp. 1–4.

[131] S. J. Kang, C. Kocabas, T. Ozel, M. Shim, N. Pimparkar, M. A. Alam, S. V. Rotkin, and J. A. Rogers, "High-performance electronics using dense, perfectly aligned arrays of single-walled carbon nanotubes," *Nature Nanotechnology*, vol. 2, no. 4, pp. 230–236, 2007.

[132] M. M. Shulaker, G. Hills, N. Patil, H. Wei, H.-Y. Chen, H.-S. P. Wong, and S. Mitra, "Carbon nanotube computer," *Nature*, vol. 501, no. 7468, pp. 526–530, 2013.

[133] V. N. Popov, "Carbon nanotubes: properties and application," *Materials Science and Engineering: R: Reports*, vol. 43, no. 3, pp. 61 – 102, 2004.

[134] N. Hamada, S.-i. Sawada, and A. Oshiyama, "New one-dimensional conductors: graphitic microtubules," *Physical Review Letters*, vol. 68, no. 10, p. 1579, 1992.

[135] R. Saito, G. Dresselhaus, M. S. Dresselhaus *et al.*, *Physical properties of carbon nanotubes*. World Scientific, 1998, vol. 4.

[136] T. W. Odom, J.-L. Huang, P. Kim, and C. M. Lieber, "Atomic structure and electronic properties of single-walled carbon nanotubes," *Nature*, vol. 391, no. 6662, pp. 62–64, 1998.

[137] J. W. Mintmire, B. I. Dunlap, and C. T. White, "Are fullerene tubules metallic?" *Phys. Rev. Lett.*, vol. 68, pp. 631–634, Feb 1992.

[138] P. L. McEuen, M. S. Fuhrer, and H. Park, "Single-walled carbon nanotube electronics," *IEEE transactions on nanotechnology*, vol. 1, no. 1, pp. 78–85, 2002.

[139] P. M. Ajayan, L. S. Schadler, C. Giannaris, A. Rubio *et al.*, "Single-walled carbon nanotube–polymer composites: strength and weakness," *Advanced materials*, vol. 12, no. 10, pp. 750–753, 2000.

[140] Z. Spitalsky, D. Tasis, K. Papagelis, and C. Galiotis, "Carbon nanotubepolymer composites: Chemistry, processing, mechanical and electrical properties," *Progress in Polymer Science*, vol. 35, no. 3, pp. 357 – 401, 2010. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0079670009000859

[141] R. Andrews and M. Weisenberger, "Carbon nanotube polymer composites," *Current Opinion in Solid State and Materials Science*, vol. 8, no. 1, pp. 31 – 37, 2004. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1359028603000925

[142] S. Stankovich, D. A. Dikin, R. D. Piner, K. A. Kohlhaas, A. Kleinhammes, Y. Jia, Y. Wu, S. T. Nguyen, and R. S. Ruoff, "Synthesis of graphene-based nanosheets via chemical reduction of exfoliated graphite oxide," *carbon*, vol. 45, no. 7, pp. 1558–1565, 2007.

[143] K. R. Paton, E. Varrla, C. Backes, R. J. Smith, U. Khan, A. ONeill, C. Boland, M. Lotya, O. M. Istrate, P. King *et al.*, "Scalable production of large quantities of defect-free few-layer graphene by shear exfoliation in liquids," *Nature materials*, vol. 13, no. 6, pp. 624–630, 2014.

[144] G. Eda and M. Chhowalla, "Chemically derived graphene oxide: towards large-area thin-film electronics and optoelectronics," *Advanced Materials*, vol. 22, no. 22, pp. 2392–2415, 2010.

[145] S. Harding and J. F. Miller, "Evolution in materio," in *Encyclopedia of Complexity and Systems Science*. Springer, 2009, pp. 3220–3233.

[146] V. Skakalova, U. Dettlaff-Weglikowska, and S. Roth, "Electrical and mechanical properties of nanocomposites of single wall carbon nanotubes with pmma," *Synthetic Metals*, vol. 152, no. 1-3, pp. 349–352, 2005.

[147] N. G. Sahoo, S. Rana, J. W. Cho, L. Li, and S. H. Chan, "Polymer nanocomposites based on functionalized carbon nanotubes," *Progress in polymer science*, vol. 35, no. 7, pp. 837–867, 2010.

[148] O. R. Lykkebø, S. Harding, G. Tufte, and J. F. Miller, "Mecobo: A hardware and software platform for in materio evolution," in *International Conference on Unconventional Computation and Natural Computation*. Springer, 2014, pp. 267–279.

[149] A. Thompson, I. Harvey, and P. Husbands, "Unconstrained evolution and hard consequences," 1995.

[150] T. Higuchi, M. Iwata, D. Keymeulen, H. Sakanashi, M. Murakawa, I. Kajitani, E. Takahashi, K. Toda, N. Salami, N. Kajihara *et al.*, "Real-world applications of analog and digital evolvable hardware," *IEEE transactions on evolutionary computation*, vol. 3, no. 3, pp. 220–235, 1999.

[151] G. W. Greenwood and A. M. Tyrrell, *Introduction to evolvable hardware: a practical guide for designing self-adaptive systems.* John Wiley & Sons, 2006, vol. 5.

[152] J. Bird and P. Layzell, "The evolved radio and its implications for modelling the evolution of novel sensors," in *Evolutionary Computation, 2002. CEC '02. Proceedings of the 2002 Congress on*, vol. 2, 2002, pp. 1836–1841.

[153] S. L. Harding, J. F. Miller, and E. A. Rietman, "Evolution in materio: Exploiting the physics of materials for computation," *arXiv preprint cond-mat/0611462*, 2006.

[154] J. M. Parkinson and D. Hutchinson, "An Investigation into the Efficiency of Variants on the Simplex Method," in *Numerical Methods for Non-linear Optimization*, F. A. Lootsma, Ed. London and New York: Academic Press, 1972, pp. 115–135. [Online]. Available: http://www.ams.org/mathscinet-getitem?mr=48:10490

[155] M. H. Wright, "Direct search methods: Once scorned, now respectable," *Pitman Research Notes in Mathematics Series*, pp. 191–208, 1996.

[156] F. Heppner and U. Grenander, "A stochastic nonlinear model for coordinated bird flocks," *The ubiquity of chaos*, pp. 233–238, 1990.

[157] C. W. Reynolds, "Flocks, herds and schools: A distributed behavioral model," in *ACM SIGGRAPH computer graphics*, vol. 21, no. 4. ACM, 1987, pp. 25–34.

[158] J. Kennedy, J. F. Kennedy, R. C. Eberhart, and Y. Shi, *Swarm intelligence.* Morgan Kaufmann, 2001.

[159] J. Kenndy and R. Eberhart, "Particle swarm optimization," in *Proceedings of IEEE International Conference on Neural Networks*, vol. 4, 1995, pp. 1942–1948.

[160] Y. Shi *et al.*, "Particle swarm optimization: developments, applications and resources," in *evolutionary computation, 2001. Proceedings of the 2001 Congress on*, vol. 1. IEEE, 2001, pp. 81–86.

[161] R. S. Verma, V. Singh, and S. Kumar, "Dna sequence assembly using particle swarm optimization," *International Journal of Computer Applications*, vol. 28, no. 10, 2011.

[162] R. Storn and K. Price, "Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces," *Journal of global optimization*, vol. 11, no. 4, pp. 341–359, 1997.

[163] S. Das and P. N. Suganthan, "Differential evolution: a survey of the state-of-the-art," *IEEE transactions on evolutionary computation*, vol. 15, no. 1, pp. 4–31, 2011.

[164] S. Das, S. S. Mullick, and P. Suganthan, "Recent advances in differential evolution–an updated survey," *Swarm and Evolutionary Computation*, vol. 27, pp. 1–30, 2016.

[165] R. Storn, "On the usage of differential evolution for function optimization," in *Fuzzy Information Processing Society, 1996. NAFIPS., 1996 Biennial Conference of the North American*. IEEE, 1996, pp. 519–523.

[166] K. Price, R. M. Storn, and J. A. Lampinen, *Differential evolution: a practical approach to global optimization*. Springer Science & Business Media, 2006.

[167] D. Zaharie, "Critical values for the control parameters of differential evolution algorithms," in *Proceedings of MENDEL*, vol. 2002, 2002.

[168] J. Liu and J. Lampinen, "A fuzzy adaptive differential evolution algorithm," *Soft Computing*, vol. 9, no. 6, pp. 448–462, 2005.

[169] A. K. Qin and P. N. Suganthan, "Self-adaptive differential evolution algorithm for numerical optimization," in *2005 IEEE congress on evolutionary computation*, vol. 2. IEEE, 2005, pp. 1785–1791.

[170] M. E. H. Pedersen, "Good parameters for differential evolution," *Magnus Erik Hvass Pedersen*, 2010.

[171] S. Harding and J. F. Miller, "Evolution in materio: Evolving logic gates in liquid crystal," in *In Proceedings of the workshop on unconventional computing at ECAL 2005 VIIIth European*, 2005, p. 12.

[172] X. Hu, R. C. Eberhart, and Y. Shi, "Swarm intelligence for permutation optimization: a case study of n-queens problem," in *Swarm Intelligence Symposium, 2003. SIS'03. Proceedings of the 2003 IEEE.* IEEE, 2003, pp. 243–246.

[173] A. Kotsialos, M. K. Massey, F. Qaiser, D. Zeze, C. Pearson, and M. C. Petty, "Logic gate and circuit training on randomly dispersed carbon nanotubes." *International journal of unconventional computing.*, vol. 10, no. 5-6, pp. 473–497, 2014.

[174] M. C. Petty, "Material devices for nascence."

[175] H.-K. Jang, J. E. Jin, J. H. Choi, P.-S. Kang, D.-H. Kim, and G. T. Kim, "Electrical percolation thresholds of semiconducting single-walled carbon

nanotube networks in field-effect transistors," *Physical Chemistry Chemical Physics*, vol. 17, no. 10, pp. 6874–6880, 2015.

[176] M. Ieda, G. Sawa, and S. Kato, "A consideration of poole-frenkel effect on electric conduction in insulators," *Journal of Applied Physics*, vol. 42, no. 10, pp. 3737–3740, 1971.

[177] M. Mohid, J. F. Miller, S. L. Harding, G. Tufte, O. R. Lykkebø, M. K. Massey, and M. C. Petty, "Evolution-in-materio: A frequency classifier using materials," in *Evolvable Systems (ICES), 2014 IEEE International Conference on*. IEEE, 2014, pp. 46–53.

[178] A. Kotsialos, M. K. Massey, F. Qaiser, D. A. Zeze, C. Pearson, and M. C. Petty, "Logic gate and circuit training on randomly dispersed carbon nanotubes." *International Journal of Unconventional Computing.*, vol. 10, no. 5-6, pp. 473–497, September 2014.

[179] D. Volpati, M. K. Massey, D. Johnson, A. Kotsialos, F. Qaiser, C. Pearson, K. Coleman, G. Tiburzi, D. A. Zeze, and M. C. Petty, "Exploring the alignment of carbon nanotubes dispersed in a liquid crystal matrix using coplanar electrodes," *Journal of Applied Physics*, vol. 117, no. 12, p. 125303, 2015.

[180] R. C. Eberhart and J. Kennedy, "A new optimizer using particle swarm theory," in *Proceedings of the sixth international symposium on micro machine and human science*, vol. 1. New York, NY, 1995, pp. 39–43.

[181] A. Thompson, *Hardware Evolution: Automatic design of electronic circuits in reconfigurable hardware by artificial evolution*. Springer Science & Business Media, 2012.

[182] K. Bache and M. Lichman, "UCI machine learning repository," 2013. [Online]. Available: http://archive.ics.uci.edu/ml

[183] J. Han and M. Kamber, "Data mining: concepts and techniques (the morgan kaufmann series in data management systems)," 2000.

[184] J. K. Uhlmann, "Satisfying general proximity/similarity queries with metric trees," *Information processing letters*, vol. 40, no. 4, pp. 175–179, 1991.

[185] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[186] T. Liu, A. W. Moore, and A. Gray, "New algorithms for efficient high-dimensional nonparametric classification," *Journal of Machine Learning Research*, vol. 7, no. Jun, pp. 1135–1158, 2006.

[187] K. J. Lang, "Learning to tell two spirals apart," in *Proc. of 1988 Connectionist Models Summer School*, 1988.

[188] J. M. Keller, M. R. Gray, and J. A. Givens, "A fuzzy k-nearest neighbor algorithm," *IEEE transactions on systems, man, and cybernetics*, no. 4, pp. 580–585, 1985.

[189] H. Parvin, H. Alizadeh, and B. Minati, "A modification on k-nearest neighbor classifier," *Global Journal of Computer Science and Technology*, 2010.

[190] D.-Y. Liu, H.-L. Chen, B. Yang, X.-E. Lv, L.-N. Li, and J. Liu, "Design of an enhanced fuzzy k-nearest neighbor classifier based computer aided

diagnostic system for thyroid disease," *Journal of medical systems*, vol. 36, no. 5, pp. 3243–3254, 2012.

[191] M. Grochowski and N. Jankowski, "Comparison of instance selection algorithms ii. results and comments," in *International Conference on Artificial Intelligence and Soft Computing*. Springer, 2004, pp. 580–585.

[192] K. Q. Weinberger, J. Blitzer, and L. K. Saul, "Distance metric learning for large margin nearest neighbor classification," in *Advances in neural information processing systems*, 2006, pp. 1473–1480.

[193] J. Goldberger, G. E. Hinton, S. T. Roweis, and R. R. Salakhutdinov, "Neighbourhood components analysis," in *Advances in neural information processing systems*, 2005, pp. 513–520.

[194] M. L. Raymer, T. E. Doom, L. A. Kuhn, and W. F. Punch, "Knowledge discovery in medical and biological datasets using a hybrid bayes classifier/evolutionary algorithm," *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, vol. 33, no. 5, pp. 802–813, 2003.

[195] R. C. Holte, "Very simple classification rules perform well on most commonly used datasets," *Machine learning*, vol. 11, no. 1, pp. 63–90, 1993.