

Preface

This is the first book on Cartesian genetic programming (CGP). CGP is a form of automatic evolution of computer programs and other computational structures using ideas inspired by Darwin's theory of evolution by natural selection.

I still remember vividly, in 1993, when I first heard the term 'genetic algorithms' from a friend while I was working at Napier University in Edinburgh. I had been working at the time in the area of digital logic synthesis and optimization. I and some of my colleagues were designing and implementing new algorithms that would allow digital circuits to be built efficiently from a specification. It often took many months, even years, to design and implement these algorithms. My friend got to hear about this and explained the idea that computational problems could be solved using an algorithm that was inspired by Darwinian evolution. It was a Friday, and after he explained the basic idea he presented me with a copy of Lawrence David Davis's 'Handbook of Genetic Algorithms'. I asked him how long he thought it would take me to write a computer program using genetic algorithms to try to solve the optimization problem I had been working on. He said, 'Oh, a couple of hours, I would think'. That weekend, I devoured the book and wrote my program. By Monday, I had a working program and it was producing excellent results, better even than the purpose-built algorithm I and my colleagues had designed and published the previous year. When I met up with my friend again, the following week, I was excited, but also a little shamefaced, as he had said it should take a few hours, but I had spent the best part of the weekend working on the program! I explained, however, that the method was working well and giving better results than my previous algorithm. He then revealed that he had been joking about it requiring a couple of hours' work, and he too became very excited. From that moment to this, I have been fascinated by evolutionary algorithms and their power to solve problems efficiently, often in unexpected and extraordinary ways.

The contributors to this book include all the leading exponents of the subject. Although the volume is primarily aimed at postgraduates, researchers and academics, it is hoped that it may be useful to undergraduates who wish to learn about one of the leading techniques in genetic programming. For those completely new to evolu-

tionary computation and genetic programming (GP), the introduction should give a sufficient, albeit brief, background. Five well-known representations used in GP are discussed. However, in reality there are many more that have been published in the research literature. Following up the many books and research references in these chapters would be an excellent place to start gaining a broader and more detailed understanding of the ever fascinating topic of evolutionary computing.

Chapter 2 explains the classic form of CGP. Three examples are given of how CGP genotypes could be used to represent computational structures in different domains. The three domains are digital circuits, mathematical equations and artworks. A detailed explanation of how CGP genotypes are decoded is given. The chapter also gives precise pseudocode algorithms for determining which nodes are active, for efficiently decoding a genotype and for calculating fitness, and a suitable evolutionary algorithm. Discussion is given of a suitable method of mutating genotypes, and there is also a brief discussion of recombination. A further section is included that gives advice about choosing parameter settings when running a CGP evolutionary algorithm. It is hoped that this will enable students and researchers to write their own programs so that they can evolve solutions to problems they are interested in. There is also a discussion of the important issue of genetic redundancy and how this leads to fitness neutrality. It is these properties that allow the evolutionary strategy to be a highly effective search method. CGP has largely been used for evolving programs without iteration or recursion. However, this is not a fundamental limitation of CGP. The chapter ends with a brief discussion of this topic. Given that it is hitherto relatively unexplored, it would form an interesting topic for further research.

One of the most powerful methods of problem solving is divide-and-conquer. This is where we solve a harder problem by breaking it down into a number of smaller, easier subproblems. This problem-solving methodology has long been utilized in forms of genetic programming and it is referred to by several names. Examples are automatically defined functions, automatically defined macros, module acquisition and program teams. Such methods are important in making GP systems *scale* up to larger or harder problem instances. Chapter 3 discusses problem decomposition in CGP, and how subfunctions can be automatically evolved and acquired. In CGP, these subfunctions are called *modules*. Two methods are described: one is called embedded CGP, and the other is modular CGP. The chapter includes details of experiments on many benchmark problems. It shows that CGP is a highly efficient and competitive technique when compared with other GP approaches. In addition, a number of methods of genetic recombination are discussed in this chapter including the use of *multiple* chromosomes in CGP.

In most forms of genetic programming, the genotype is composed of a *single* chromosome. Genetic recombination, or crossover, is carried out at a *gene* level. However, in biology, recombination also happens at the higher level of whole chromosomes. Humans have 23 pairs of chromosomes, while the kingfisher bird has 66 pairs, and the goldfish 51. In sexual recombination in humans, a child inherits each of the 23 chromosomes (which get duplicated) from *either* one of the parents and many inherited genetic traits are attributable to one parent or the other. The clearest example of this is in the inheritance of sex. If a sperm cell containing an X chromo-

some fuses with an ova (which always contains an X chromosome), then a female offspring is created, while if a sperm containing a Y chromosome fuses, it produces a male child. How does this relate to CGP? Well, many computational problems do not have a single output. A good example of this is provided by electronic circuits; for instance Chap. 3 considers an arithmetic logic circuit with 17 outputs. One has a choice to try to evolve a genotype with a single chromosome with 17 outputs, or one could evolve a genotype consisting of 17 chromosomes (each with a single output). In the latter case, it turns out that one can define an optimal form of sexual recombination for this case, where the offspring inherits the best chromosome from each of its parents.

Human programmers rarely, if ever, write programs that while being executed, change themselves. For one thing, such programs would be fiendishly difficult to debug. However, this does not mean that the automatic evolution of such programs cannot cope with such a problem. Chapter 4 takes this idea forward and discusses a relatively new form of CGP, called self-modifying CGP (SMCGP). In SMCGP, an evolved program (a phenotype) can actually modify itself to produce a new phenotype. In this way, a single evolved genotype can be *iterated* to produce a possibly infinite series of phenotypes, each of which is a program. This means that instead of evolving a program that solves one instance of a problem, one can attempt to evolve solutions to a possibly infinite sequence of problems. It turns out that in certain cases it can be proved mathematically that an evolved solution solves all instances, that is to say it is a general solution.

CGP developed from a method for evolving digital circuits, so it is not surprising that ever since researchers have been enhancing and modifying the technique and applying it to more challenging problems in electronic circuit design. Chapter 5 brings the reader up to date on this research. Although it has been known for some time that evolutionary algorithms are capable of producing innovative new designs that traditional synthesis methods were not capable of, there were always intractable limitations on the size of such evolved circuits. This issue is discussed in the chapter. The chapter also discusses the design of polymorphic circuits (where logic gates can assume more than one function depending on an additional physical variable). This is a good application area for CGP, since conventional circuit synthesis tools cannot handle such devices. Interestingly, the chapter describes how a specially designed chip, a hardware implementation of CGP, was used to speed up the evaluation of the design quality (fitness).

Another good area for application of CGP is in the design of specialized complex arithmetic circuits such as multiple-constant-multiplier circuits. These are circuits that multiply their single input by a number of constants. They are very useful in implementing low power finite-impulse response filter circuits. Finding optimal circuits of this type is known to be NP-complete. The chapter shows that CGP can be used to find designs that improve on those found by the state-of-the-art heuristic algorithm

For some time now, we have all been enjoying the benefits of Moore's famous law on the exponentially increasing transistor densities in integrated circuits. However, these densities have increased to the point that devices are approaching atomic

sizes. This is causing increasing numbers of failures and lower production yields. Using models of transistors which incorporate random intrinsic variability together with a specially designed representation of CGP combined with a multi-objective algorithm is shown to be capable of designing novel CMOS circuit topologies.

Finally, Chap. 5 discusses two hardware implementations of CGP. The first uses embedded CGP (see Chap. 3) to design classifiers for electromyographic signals. Such methods are essential for the control of prosthetic hands. Results show that this approach can produce hardware prosthesis control classifiers that achieve accuracies close to that of state-of-the-art classification algorithms. In addition, CGP-evolved hardware classifiers offer compactness, fast computation and self-adaptability. The second hardware application uses CGP to implement application-specific caches. Not only does this approach significantly improve cache performance, but also the evolved mapping functions generalize very well. The technique also achieves faster execution times and consumes less energy than conventional cache architectures.

Chapter 6 presents three applications in which CGP can automatically generate novel image-processing algorithms whose performance is comparable to, or even exceeds, that of the best known conventional solutions. The first application deals with the automatic design of low-level image filters that are comparable in quality to conventional filters but with a lower implementation cost. In the second, CGP is used to design more advanced image operators such as dilation/erosion filters, using relatively complex elementary functions (sine, square root, etc.) Such designs may be useful in advanced image-processing software tools. The third application uses evolved CGP graphs to define transformations on medical images. It is shown that CGP image transformations can significantly improve classification accuracy for a number of predefined image features.

Chapter 7 describes a complete CGP design system implemented in hardware (on a field-programmable gate array). This has advantages over a CGP system running on a general processor in that it can achieve a significant speed-up for many applications and could be used in small low-power adaptive embedded systems.

It is well known that in evolutionary computation the evaluation of the quality of evolved solutions can be very time-consuming. In this regard, CGP is no exception. Chapter 8 describes how this problem can be significantly alleviated by exploiting the parallel processing capabilities of the graphics processing units (GPUs) that are found on modern graphics cards. GPUs are fast, highly parallel devices. GPUs are built for rapid processing of 3D graphics; however, it turns out that modern GPUs can also be programmed for more general-purpose computation. It is only recently that the genetic programming community has begun to start exploiting GPUs to accelerate the evaluation of encoded genetic programs. Indeed, CGP was the first GP technique implemented on GPUs that was suitable for a wide range of applications. The chapter describes how such an implementation has resulted in very significant performance increases.

Chapter 9 describes how CGP can be used to design an interesting new kind of artificial neural network (ANN), called the CGP Developmental Network (CGPDN). A neuron in the CGPDN is represented by seven CGP chromosomes encoding different desirable aspects of biological neurons (i.e. neuron body or soma, dendrites,

axons, synapses, and dendritic and axonal branches). The CGPDN neuron is developmental in nature and when the programs encoded in the evolved CGP chromosomes are executed, they build a dynamic network in which neurons can replicate, die and change. Also, the dendritic and axonal branches can change their morphology and efficiency while at the same time solving a computational problem. It is shown that CGPDN is capable of learning in two well-known problems in artificial intelligence. The long-term vision of this work is to evolve programs that encode a general learning *capability*, rather than encoding a specific learned response (as in conventional ANNs).

Chapter 10 describes how CGP can be used in the creative arts. The chapter gives a brief overview of a field known as evolutionary art. This is where evolutionary algorithms or genetic programming is used to produce works of visual art. It is argued that some aspects of CGP prove to be highly advantageous in obtaining *contextual focus*. This is where there is simultaneously a drive towards a precise goal and an exploration of wider possibilities. It is argued that contextual focus is essential for building systems that mimic human creativity. Appropriately, the techniques described are used to generate creative portraits of Charles Darwin.

Evolutionary algorithms have long been applied to various problems in medicine as they provide not only good performance but are also highly flexible and adaptable. The final chapter in this book gives an overview of the types of medical problems that may be addressed and illustrates this by considering in detail a number of published case examples. It also presents case studies of how CGP can be utilized in the diagnosis of three medical conditions: breast cancer, Parkinson's disease and Alzheimer's disease. It gives advice on the common pitfalls, benefits and rewards of medical applications and, particularly, the tricky problem of obtaining patient data.

The articles in this book give a good indication of the wide applicability of CGP. The underlying reason for this is primarily because CGP is a general technique for evolving solutions to all sorts of computational problems. However, there are a number of applications of CGP that have been studied that have not been represented in this book. Some have been carried out with or by research colleagues, others by students. Examples include applications to bioinformatics, helicopter control, artificial neural networks, function optimization, artificial life, robot control, object recognition, sensor failure recovery, fault-tolerant circuits and the solution of differential equations. I offer my apologies to the many authors of such work for the lack of inclusion of this work in this volume. Perhaps, in the future a new volume on CGP will be produced.

It is important to realize that CGP is not a fixed technique; that is to say, it is constantly under development. It began with the classic technique described in Chap. 2, which is still widely used, but it developed into embedded or modular CGP, described in Chap. 3, and still more recently into the self-modifying form described in Chap. 4. No doubt this process will continue. However it develops, one thing is clear; it will become more efficient and more widely applicable. The pioneer of GP, John Koza, pointed out that the power of GP is proportional to the availability of computational power. Thus we are likely to see even more impressive results coming from GP (and CGP) in the future.

My very great thanks to my friend Peter Thomson, who, while having a warm shower, came up with a genetic representation that would be suitable for evolving circuits. This inexorably led me to Cartesian genetic programming.

The idea for this book was formed sometime in May 2009. I would like to thank Jerry Liu and James Walker for their help with final corrections. I would like especially to thank Ronan Nugent, senior editor for Computer Science at Springer-Verlag, for his encouragement and his great patience, as my own pledged deadlines slipped. Thanks, Ronan, for keeping faith. I would like also to thank my colleagues and friends who are respected CGP users and researchers and have contributed to this book. May you continue to spread the faith in CGP and retain enough scepticism to keep improving and applying it!

March 2011

York, UK
Julian Miller



<http://www.springer.com/978-3-642-17309-7>

Cartesian Genetic Programming

Miller, J.F. (Ed.)

2011, XXII, 346 p., Hardcover

ISBN: 978-3-642-17309-7