AUTOMATIC QUANTUM
COMPUTER PROGRAMMING
A Genetic Programming Approach
GENETIC PROGRAMMING SERIES

Series Editor

John Koza
Stanford University

Also in the series:


## Contents

Preface vii  

1. THE POWER OF QUANTUM COMPUTING  1  
   1. What is Quantum Computing?  1  
   2. Possibilities Count  4  
   3. The Role of Automatic Programming  10  

2. QUANTUM COMPUTER SIMULATION  13  
   1. Bits, Qubits, and Gates  13  
   2. Gate-Level Simulation  18  

3. QUANTUM COMPUTER PROGRAMMING  23  
   1. QGAME: Quantum Gate and Measurement Emulator  23  
   2. Visualization  28  
   3. Example: Grover’s Database Search Algorithm  31  

4. GENETIC AND EVOLUTIONARY COMPUTATION  37  
   1. What is Genetic and Evolutionary Computation?  37  
   2. Genetic Algorithms  39  
   3. Scalability via Parallelism  41  
   4. Applicability of Genetic and Evolutionary Computation  41  

5. GENETIC PROGRAMMING  43  
   1. Programming by Genetic Algorithm  43  
   2. Traditional Program Representations  45  
   3. Traditional Genetic Operators  48  
   4. Example: Symbolic Regression  49  
   5. Obtaining Genetic Programming Results  52
AUTOMATIC QUANTUM COMPUTER PROGRAMMING

6. EVOLUTION OF COMPLEX PROGRAMS 55
   1 Types, Modules, and Development 56
   2 The Push Programming Language 59
   3 Push Examples 63
   4 PushGP: Genetic Programming with Push 68
   5 Autoconstructive Evolution 71

7. EVOLUTION OF QUANTUM PROGRAMS 75
   1 Program Representations 75
   2 Fitness 80
   3 Operators and Refinements 82

8. EVOLVED QUANTUM PROGRAMS 87
   1 The 1-bit Deutsch-Jozsa (XOR) Problem 88
   2 Grover’s Database Search Problem 95
   3 Scaling Majority-ON 104
   4 The OR and AND/OR Problems 108
   5 Gate Communication Problems 114
   6 Significance of These Results 118

9. CONCLUSIONS AND PROSPECTS 123

Appendix A
   QGAME source code 127
   References 141
   About the Author 149

Index 151
This is a book about the frontiers of computer science that have recently been opened by work in quantum mechanics, but it is also a book about the use of recently developed automatic programming technologies to explore those frontiers. The automatic programming technologies themselves issue from another interdisciplinary frontier of computer science — one born of the intersection of computer science with evolutionary biology. So this is a book about two frontiers of computer science, one being used primarily for the sake of exploring the other.

The selection of topics in this book was made with the intention of showing how genetic programming can be usefully applied to certain problems in quantum computing. To this end, it provides a basic introduction to quantum computing for non-physicists and it also provides a basic introduction to genetic programming for non-computer-scientists. These treatments should be comprehensible to scientifically literate readers who have, at minimum, a passing familiarity with undergraduate-level computer science (e.g. programming concepts) and mathematics (e.g. simple linear algebra). No background in physics is assumed.

Neither the introduction to quantum computing nor the introduction to genetic programming is intended to be comprehensive or even “balanced.” Coverage of each field is limited to relatively narrow slices that support the demonstrations found later in the book — those demonstrations show how certain genetic programming techniques can be applied to certain problems in quantum computing. Citations are provided where appropriate to sources that provide more comprehensive and detailed coverage.

The first chapter contains an introduction to quantum computing for non-physicists. The intention is to provide readers with a sense of how quantum computers could possibly deliver the surprising benefits that many researchers envision.
The second chapter details a mathematical (matrix-based) model of quantum computation and describes how this model can be used to simulate quantum computations on classical computers. Such simulation is necessarily inefficient — if we could simulate quantum computers efficiently on classical computers then there’d be little reason to study quantum computing in the first place! But for small computations simulation is indeed possible; this model allows us to use simulation in the “fitness assessment” step of a genetic programming algorithm, described later in the book.

The third chapter describes one particular quantum computer simulation system, the author’s QGAME (“Quantum Gate and Measurement Emulator”) system, and presents a few of the ways in which quantum programs and quantum computer states can be visually displayed. It concludes with a detailed example of the simulation of a quantum program for Grover’s database search problem.

The fourth chapter introduces genetic and evolutionary computation, with a focus on the traditional genetic algorithm. It also discusses, in general terms, the use of parallelism to scale genetic and evolutionary computation technologies up for complex applications, and the applicability of these technologies for various types of problems including those related to quantum computing.

The fifth chapter specializes the treatment of genetic algorithms to genetic programming, which is the use of genetic algorithms for automatic programming. It includes a detailed example and a discussion of the steps one must generally take to obtain and understand useful results from a genetic programming system.

The sixth chapter moves beyond traditional genetic programming, and describes the ways in which one can evolve programs that include, for example, multiple data types, modules, and developmental components. Some of these capabilities are particularly useful for the evolution of quantum programs. Emphasis is placed on the author’s Push programming language for genetic and evolutionary computation, which provides some of the desired advanced capabilities in unusually simple ways. This chapter concludes with a description of the PushGP genetic programming system, which evolves Push programs, and a brief description of some more radically self-adaptive “autoconstructive evolution” techniques that are enabled by Push.

The seventh and eighth chapters bring the materials from all of the preceding chapters together, first with a discussion of specific strate-
gies for quantum program evolution,¹ and then with concrete examples in which interesting quantum programs were evolved using QGAME, PushGP and related technologies. These examples document a few specific ways in which genetic programming has already helped to explore the power of quantum computing.

The ninth chapter provides a brief summary of the main points of the book and discusses prospects for new discoveries made with the aid of automatic quantum computer programming technologies.

Source code, in Common Lisp, for a minimal version of QGAME is included in the Appendix. Additional related source code is available online from addresses that are cited within the text. Most of these files are also linked to the author's public "code" page.²

This book would not have been possible without the close working relationships enjoyed by the author with colleagues and students at Hampshire College in Amherst, Massachusetts. Several of the results that are used as examples in the book emerged from joint work of the author with Herbert J. Bernstein, Howard Barnum, and Nikhil Swamy. Although specific joint results are acknowledged where they occur in the text, these citations do not by themselves fully convey the extent of the influence of these colleagues. Similarly, the novel technologies that are described in the text owe much to the contributions of Chris Perry, Jon Klein, Mark Feinstein, Raymond Coppinger, Alan Robinson, Raphael Crawford-Marks, and Manuel Nickschas. Many of these colleagues also commented on the manuscript of this book, leading to substantial improvements. Additional substantial comments were provided by John Koza, Sameer H. Al-Sakran, and Rennie Nelson. Rebecca S. Neimark provided essential assistance in many phases of the project, including the creation of several of the figures and the design of the cover, which uses an image created by Chris Perry. James Hendler provided critical encouragement and advice, and Leni Bowen and Paula Harmon provided invaluable administrative support.

Some of the materials used in this book derive from those prepared by the author for a series of tutorials on quantum computing presented over several years at the Genetic and Evolutionary Computation Confer-

¹Note that the term "evolution" is used here and throughout this book in a sense derived from its biological usage: it refers to a process in which a population undergoes variation and natural selection. Some physicists use "evolution" in a more general sense, to describe any change in a system over time. The phrase "quantum program evolution" in this book refers to the generation of quantum programs using techniques derived from biological evolutionary processes.
²http://hampshire.edu/lspector/code.html
ences (GECCO), for an invited presentation on “Quantum Computation and Artificial Intelligence” at the 1999 National Conference on Artificial Intelligence (AAAI), for a seminar in the Chevron TechNet Advanced Information-Based Modeling seminar series, for a seminar presented at BBN Technologies, and for a course called “Quantum Computing with No Prerequisites of Any Kind” taught at Hampshire College.

This work was supported by a National Science Foundation Director's Award for Distinguished Teaching Scholars, by National Science Foundation grant EIA-0216344, and by the Defense Advanced Research Projects Agency and Air Force Research Laboratory, Air Force Materiel Command, under agreement number F30502-00-2-0611.
Graspings:  
wholes and not wholes,  
convergent divergent,  
consonant dissonant,  
from all things one and  
from one thing all.  
—Heraclitus
Chapter 1

THE POWER OF QUANTUM COMPUTING

This chapter provides a brief, non-technical introduction to quantum computing and outlines both the potential power and the enigmatic nature of quantum computers. It also makes the case for the application of automatic programming technologies to problems in quantum computing, arguing that such technologies can play a unique and important role in the future of this emerging field. The discussion here is general; mathematical and computational details are deferred to later chapters.

1. What is Quantum Computing?

What physical principles govern the processes of computation? Physicists studying this question have recently made a remarkable series of discoveries. These discoveries imply that it may be possible to build quantum computers — that is, computers that take advantage of certain quantum mechanical phenomena — that are more powerful, in a fundamental sense, than any other computers previously designed. More than that, they may be more powerful than any other computers previously imagined, in the sense that they obey new and more permissive laws of computational complexity.

We use the phrase “quantum computing” to describe computational processes that rely for their efficacy on specifically quantum mechanical properties of information-processing hardware. Of course all computing relies on quantum mechanics in some sense, since quantum mechanics is currently our best theory for describing all physical processes. As Rolf Landauer has made clear (Landauer, 1999), “information is inevitably physical,” and this means, among other things, that the laws of physics (and in particular the laws of quantum mechanics) underlie all information processing. But as of this writing most information processing
can be understood using only classical physics and classical information theory, and the specifically quantum mechanical properties of the hardware can be ignored. Quantum computing is computing in which the specifically quantum mechanical properties matter a great deal, usually because they are being leveraged to allow us to do things that are not permitted by the classical theories. We call computers that can be understood in terms of the classical theories classical computers, and computers that can be understood only in terms of quantum mechanics quantum computers.

Why are we interested in quantum computing? One reason is that the size of computing elements continues to shrink at an exponential rate (following "Moore's Law"), with the result that we will be storing bits on devices roughly the size of atoms within the next decade. At these sizes, specifically quantum mechanical effects predominate and we will be doing quantum computing whether we want to or not! But most of the excitement surrounding quantum computing comes not from its inevitability but rather from the discovery that quantum computers can do things beyond the reach of classical computers.

What can quantum computers do that classical computers cannot? This question is still largely open and under active investigation. Indeed, the primary motivation for this book is to provide new tools for the exploration of this question. But we do already know that quantum computers can outperform classical computers in a few specific ways.

At the time of this writing the most spectacular known advantage of quantum over classical computers is the complexity advantage demonstrated by Peter Shor's algorithm for factoring large numbers (Shor, 1994), a problem with practical applications in cryptography and possibly in other areas. Although the classical computational complexity for factoring is not known with certainty, the best known classical factoring algorithms require an amount of time proportional to \(2^{\frac{n^3}{3}} \log(n)^{\frac{2}{3}}\), where \(n\) is the number of digits in the number to be factored. In contrast, Shor's quantum algorithm (Beckman et al., 1996; Shor, 1998) requires time proportional to only \(n^2 \log(n) \log \log(n)\). This is an exponential savings, assuming that the known classical factoring algorithms are near optimal (which is not known but is suspected by many to be true).

A rough calculation can give one a feel for just how spectacular this improvement is. Suppose we wish to factor a 5,000 digit number. If we crudely (but consistently) assume that the complexity functions in the previous paragraph are exact, that all logarithms are to be taken base 2, and that we can execute one instruction per nanosecond, then we would expect the best known classical algorithms to require about 80 billion years. This is many times the current age of the universe. By contrast,
under the same assumptions Shor’s quantum factoring algorithm would require less than two seconds! Of course we are ignoring many things that cannot really be ignored in these estimates, such as constants in the complexity functions and the possibility that quantum and classical hardware will support different clock rates. So the exact numbers in these estimates may be quite far off. But the dramatic nature of the exponential speedup is independent of these factors.

A more modest, but also more certain advantage of quantum over classical computers was discovered by Lov Grover. Grover’s quantum search algorithm (Grover, 1997) achieves a quadratic speedup over the best classical algorithms for finding a single “marked” item in a database. A classical algorithm must test, on average, half of the $n$ items in the database, while the quantum algorithm can find the item after making only about $\sqrt{n}$ queries. This is less spectacular than the apparent exponential savings of Shor’s algorithm, but the quantum complexity advantage is unquestionable, the algorithm has wide applicability, and the savings may be considerable in practice. In Section 3.3 we examine Grover’s algorithm in more detail, and in Section 8.2 we demonstrate the use of genetic programming to re-discover an instance of Grover’s algorithm.

Can quantum computers speed up other sorts of calculations? Several variants of Shor’s and Grover’s algorithms have been developed for related problems, and a few other, qualitatively different algorithms have also been discovered.\(^1\) For some types of problems we have also obtained, via mathematical analysis, specific bounds on the possible speedups. But overall our current knowledge is spotty; we know relatively little about what kinds of computations can be sped up, or how much, by the use of quantum hardware. These gaps in our knowledge provide one motivation for the development of technologies that can automatically discover new quantum algorithms.

Quantum computing technology may also provide other kinds of benefits, qualitatively different than those due to the computational complexity advantages that are exemplified by Shor’s algorithm and Grover’s algorithm. For example, quantum states are “tamper resistant” in a certain sense, and this property can be leveraged to provide secure communication channels upon which it is theoretically impossible to eavesdrop. Some of the schemes for such channels require relatively little in the way of quantum hardware engineering, and quantum information technology products for secure communications are already commercially available.

---

\(^1\)Additional quantum algorithms are described, for example, in (Hogg, 1998; Hogg, 2000; Hallgren, 2002; Hallgren et al., 2003; van Dam and Seroussi, 2002; van Dam et al., 2002).
Other possible technologies may result from the exploitation of phenomena such as quantum superdense coding, quantum teleportation or quantum error correction, allowing information to be moved and/or reconstructed in novel ways. Although these applications and potential applications appear to be quite different in nature from the “speedups” provided by Shor’s and Grover’s algorithms, all of these envisioned applications result from the discovery of quantum algorithms and protocols with novel properties. For this reason they may also benefit from technologies that can automatically discover new quantum algorithms.

Some theorists envision yet further benefits emerging from some future, deeper understanding of quantum mechanics, possibly stemming from new theories of quantum gravity that support new forms of computation. For example, Roger Penrose argues that human consciousness and creativity rely on quantum effects beyond those conceivably provided by current models of quantum computation (Penrose, 1989; Penrose, 1997). Regardless of the strength of these arguments — which appear to this author to rest on mistakes about the nature of human cognition — it seems reasonable to expect such exotic forms of quantum computation, if they ever exist, to present challenges to human algorithm designers that are at least as great as those posed by “ordinary” quantum computation. This would further increase the utility of technologies that automatically discover new quantum algorithms.

Several general introductions to quantum computing are available. These include, listed roughly from least to most technical, (Brown, 2000; Milburn, 1997; Brooks, 1999; Rieffel and Polak, 2000; Williams and Clearwater, 1998; Steane, 1998; Gruska, 1999; Nielsen and Chuang, 2000). John Preskill’s online lecture notes also provide a comprehensive introduction. Some foundational documents can be found in (Feynman, 1996) and (Hey, 1999). Current research in the quantum computing and quantum information theory is published in a wide range of journals (mostly physics journals) and conference proceedings (such as Shapiro and Hirota, 2003). Many contributions are distributed in pre-print form from the online “arXiv” archive.

2. Possibilities Count

How is it that quantum computers can outperform classical computers? That is, how can the specifically quantum mechanical properties of quantum computing hardware provide non-classical computing power? In Chapter 2 we look at a mathematical characterization of quantum
computing that provides, in some sense, the most complete answer to this question. But the mathematical characterization does little to pro-
vide intuitions about what's really going on. Indeed, such intuitions are hard to come by, since better-than-classical quantum algorithms exploit the "weird" aspects of quantum mechanics that have baffled nearly ev-
erybody for nearly a century. As Richard Feynman wrote in discussing quantum electrodynamics:

No, you're not going to be able to understand it. . . . You see, my physics students don't understand it either. That is because I don't understand it. Nobody does. . . . The theory of quantum electrodynamics describes Nature as absurd from the point of view of common sense. And it agrees fully with experiment. So I hope you can accept Nature as She is — absurd. (Feynman, 1985)

The same can be said in regard to some of the deepest questions in quantum computing: we can easily see how the mathematics produces the results, but that's a far cry from understanding how or why Nature conforms to the particular, counter-intuitive mathematics.

Nevertheless, in this section a brief attempt is made to ground at least some fundamental intuitions about how it is that quantum mechanical properties can provide computational advantages.

One perspective on the source of the power of quantum computing is that in quantum computing possibilities count, even if they never happen. Furthermore, in well-designed quantum algorithms, each of exponentially many possibilities can be used to perform a part of a computation at the same time.

At first blush this must appear to be a preposterous assertion. How can possibilities that never happen influence the outcome of a computa-
tion? But there is a sense in which this is literally true, and one can view many if not all of the novel effects of quantum computing as stemming from this fact.

Consider a beam splitter as shown in Figure 1.1, which might be made from a half-silvered mirror. Photons leave the light source on the left, hit the beam splitter in the center, and either reflect to detector A or pass through to detector B. When we turn the light up high, sending out a steady beam, half of the photons are detected at A and half are detected at B. When we turn the light down very low, so that there is only one photon in the system at a time, each photon is detected either at A or at B with 50% probability for each detector. That is, for any given photon there is a possibility that it will be reflected and a possibility that it will pass through. Quantum mechanics tells us that we cannot know which possibility will actually happen — that is, will eventually be detected