GENETIC PROGRAMMING THEORY AND PRACTICE VII

GENETIC PROGRAMMING THEORY AND PRACTICE VII

Edited by

RICK RIOLO
Center for the Study of Complex Systems
University of Michigan

UNA-MAY O'REILLY
Computer Science and Artificial Intelligence Laboratory
Massachusetts Institute of Technology

TRENT MCCONAGHY
Solido Design Automation, Inc.

Kluwer Academic Publishers
Boston/Dordrecht/London

Contents

Contributing Authors

Peter C. Andrews is a software engineer in the Computational Genetics Laboratory at Dartmouth Medical School (Peter.C.Andrews@Dartmouth.edu).

Wolfgang Banzhaf is a professor and chair of the Department of Computer Science at Memorial University of Newfoundland, St. John's, NL, Canada (simonh@cs.mun.ca).

Ying Becker is Vice President, Head of Global Quantitative Active Equity Research, Advanced Research Center at State Street Global Advisors, Boston, MA USA (Ying_Becker@ssga.com).

Josh Bongard is an assistant professor of Computer Science in the College of Engineering and Mathematical Sciences at the University of Vermont, USA (josh.bongard@uvm.edu).

Caterina Cinel is a psychologist and principal research officer within the School of Computer Science and Electronic Engineering of the University of Essex, UK (ccinel@essex.ac.uk).

Luca Citi is a research officer within the School of Computer Science and Electronic Engineering of the University of Essex, UK (lciti@essex.ac.uk).

John Doucette is an undergraduate at the Faculty of Computer Science, Dalhousie University, NS, Canada (jdoucett@cs.dal.ca).

Peng Fei is a senior quantitative research analyst for the Advanced Research Center at State Street Global Advisors (SSgA), the investment management arm of State Street Corporation, Boston, MA (peng_fei@ssga.com).

Casey S. Greene is a graduate student in the Computational Genetics Laboratory at Dartmouth Medical School (Casey.S.Greene@Dartmouth.edu).

Malcolm Heywood is a Professor of Computer Science at Dalhousie University, Halifax, NS, Canada (mheywood@cs.dal.ca).

Douglas P. Hill is a computer programmer in the Computational Genetics Laboratory at Dartmouth Medical School (Douglas.P.Hill@Dartmouth.EDU).

Gregory Hornby is a Senior Scientist with the University of California Santa Cruz working in the Intelligent Systems Division at NASA Ames Research Center (gregory.s.hornby@nasa.gov).

Janine Imada recently completed graduate studies in Computer Science at Brock University, St. Catharines, ON, Canada (jimada@bell.net).

Minkyu Kim is a doctoral candidate in the Department of Electrical Engineering and Computer Science at Massachusetts Institute of Technology (minkyu@mit.edu).

Michael F. Korns is Chief Technology Officer at Investment Science Corporation (mkorns@korns.com).

Mark E. Kotanchek is Chief Technology Officer of Evolved Analytics, a data modeling consulting and systems company (mark@evolved-analytics.com).

Peter Lichodzijewski is a graduate student in the Faculty of Computer Science at Dalhousie University, Halifax, Nova Scotia, Canada (piotr@cs.dal.ca).

Hod Lipson is an Associate Professor in the school of Mechanical and Aerospace Engineering and the school of Computing and Information Science at Cornell University, Ithaca, NY (hod.lipson@cornell.edu).

Trent McConaghy is Chief Scientific Officer at Solido Design Automation Inc. and recently completed his PhD at ESAT-MICAS, Katholieke Universiteit Leuven, Belgium (trent_mcconaghy@yahoo.com).

Jason H. Moore is the Frank Lane Research Scholar in Computational Genetics and Associate Professor of Genetics at Dartmouth Medical School (Jason.H.Moore@Dartmouth.edu).

Tomoharu Nagao is a Professor of the Faculty of Environment and Information Sciences at Yokohama National University, Japan (nagao@ynu.ac.jp).

Una-May O'Reilly is a Principal Research Scientist in the Computer Science and Artificial Intelligence Laboratory at Massachusetts Institute of Technology (unamay@csail.mit.edu).

Riccardo Poli is a Professor of Computer Science in the School of Computer Science and Electronic Engineering at the University of Essex, UK (rpoli@essex.ac.uk).

Rick Riolo is Director of the Computer Lab and Associate Research Scientist in the Center for the Study of Complex Systems at the University of Michigan (rlriolo@umich.edu).

Brian J. Ross is a Professor of Computer Science at Brock University, St. Catharines, ON, Canada (bross@brocku.ca).

Michael Schmidt is a graduate student in computational biology at Cornell University, Ithaca, NY (mds47@cornell.edu).

Shinichi Shirakawa is a Reseacher of the Graduate School of Environment and Information Sciences at Yokohama National University and a Research Fellow of the Japan Society for the Promotion of Science (shirakawa@nlab.sogo1.ynu.ac.jp).

Guido F. Smits is a Research and Development Leader in the New Products Group within the Core R&D Organization of the Dow Chemical Company (gfsmits@dow.com).

Ekaterina Vladislavleva is a postdoctoral researcher in the Computer Arithmetics and Numerical Techniques Group in the Department of Mathematics and Computer Science, Antwerp University, Belgium (katya@vanillamodeling.com).

Garnett Wilson is a postdoctoral fellow in the Department of Computer Science at Memorial University of Newfoundland, St. John's, NL, Canada (gwilson@cs.mun.ca).

Preface

The work described in this book was first presented at the Seventh Workshop on Genetic Programming, Theory and Practice, organized by the Center for the Study of Complex Systems at the University of Michigan, Ann Arbor, 14-16 May 2009. The goal of this workshop series is to promote the exchange of research results and ideas between those who focus on Genetic Programming (GP) theory and those who focus on the application of GP to various real-world problems. In order to facilitate these interactions, the number of talks and participants was small and the time for discussion was large. Further, participants were asked to review each other's chapters *before* the workshop. Those reviewer comments, as well as discussion at the workshop, are reflected in the chapters presented in this book. Additional information about the workshop, addendums to chapters, and a site for continuing discussions by participants and by others can be found at http://cscs.umich.edu/gptp-workshops/gptp2009 .

We thank all the workshop participants for making the workshop an exciting and productive three days. In particular we thank the authors, without whose hard work and creative talents, neither the workshop nor the book would be possible. We also thank our keynote speaker Margaret J. Eppstein, Associate Professor in Computer Science and Director of the Complex Systems Center at the University of Vermont. Maggie's talk inspired a great deal of discussion among the participants throughout the workshop.

The workshop received support from these sources:

- The Center for the Study of Complex Systems (CSCS);
- Third Millennium Venture Capital Limited;
- Michael Korns, Investment Science Corporation;
- State Street Global Advisors, Boston, MA;
- Evolved Analytics;
- Computational Genetics Laboratory at Dartmouth College;
- Biocomputing and Developmental Systems Group, Computer Science and Information Systems, University of Limerick; and
- William and Barbara Tozier of Vague Innovation LLC.

We thank all of our sponsors for their kind and generous support for the workshop and GP research in general.

A number of people made key contributions to running the workshop and assisting the attendees while they were in Ann Arbor. Foremost among them was Howard Oishi, who makes GPTP workshops run smoothly with his diligent efforts before, during and after the workshop itself. During the workshop Bill Worzel moderated some of the sessions. After the workshop, many people provided invaluable assistance in producing this book. Special thanks go to Sarah Cherng, Patrick Hooper and Laura Tomassi, who did a wonderful job working with the authors, editors and publishers to get the book completed very quickly. Thanks to William Tozier for assisting in copy-editing many of the chapters. Jennifer Maurer and Melissa Fearon's editorial efforts were invaluable from the initial plans for the book through its final publication. Thanks also to Deborah Doherty of Springer for helping with various technical publishing issues. Finally, we thank Carl Simon, Director of CSCS, for his support for this endeavor from its very inception.

Rick Riolo, Una-May O'Reilly and Trent McConaghy

Foreword

Genetic programming (GP) has emerged as an important computational methodology for solving complex problems in a diversity of disciplines. The recent success of GP can be attributed to the highly innovative computer scientists that have developed and extended the approach over the last 20 years and the numerous investigators and analysts that have been on the front line of applying these algorithms to difficult problems in their specific domains. This is all supported by a vibrant and highly collaborative research community that includes numerous GP conferences and workshops, numerous GP journals and book series and a wealth of open-source software and internet-based resources. The continued growth of this community speaks to the impact of GP and its important place in the future of computation and analysis.

An important challenge for any discipline is to foster close relationships and collaborations between those that develop computational theory and those that practice the art of computation. The annual Genetic Programming Theory and Practice (GPTP) Workshop organized by the Center for the Study of Complex Systems at the University of Michigan in Ann Arbor was first organized and held in 2003 to specifically bring theorists and practitioners together to communicate and collaborate in an effort to transform GP from an innovative algorithm to a general computational strategy for solving complex problems. The positive impact of this workshop on those that attend has been substantial. There are several reasons for this. First and foremost, there is an openness and general sense of collegiality that is lacking at many other scientific venues. The attendees are genuinely interested in sharing their cutting edge ideas for group discussion. This selflessness combined with enthusiastic discussion and participation creates an environment that significantly fosters innovation. Second, there is an incredible sense of memory from past GPTP workshops. Attendees take what they have learned from previous years and integrate the new ideas into their work. This cascading of ideas from year to year provides a rare opportunity to synthesize innovation. Third, each of the attendees is truly interested in solving complex problems in their respective domains. This is important because the investigators are open to any idea that helps solve the problem at hand. The lack of dogmatism and openness to change is refreshing and has made this workshop a huge success.

This was my fourth year at GPTP and each year has been a tremendous learning experience. Perhaps the single most eye-opening GPTP event was learning that Michael Korns of Korns Associates was successfully using GP to make real life financial investment decisions. This to me is the ultimate endorsement for the use of GP for solving problems such as security analysis and stock ranking. There was a sense at GPTP this year that GP has turned the corner from an innovative algorithm used only by computer scientists to a truly useful discovery tool used by many. In fact, William Tozier made the prediction that 2010 marks the beginning of a GP bubble characterized by an exponential shift from art to craft. That is, within the next few years we will increasingly see GP being used by domain-specific experts such as bioinformaticists, economists, engineers and meteorologists to solve hard problems. If this is true, the next 10 years of GPTP will be more important than ever.

I encourage you to read and digest each of the chapters in this volume and those from all the previous volumes. I promise you will come away with a notebook full of new ideas for using GP to solve your domain-specific problem.

Jason H. Moore, Ph.D.
Frank Lane Research Scholar in Computational Genetics
Professor of Genetics and Community and Family Medicine
Dartmouth Medical School, Lebanon, NH, USA
July, 2009

Chapter 1

GPTP 2009: AN EXAMPLE OF EVOLVABILITY

Una-May O'Reilly[1], Trent McConaghy[2] and Rick Riolo[3]

[1]*Computer Science and Artificial Intelligence Laboratory, Massachusetts Institute of Technology;* [2]*Solido Design Automation Inc., Canada;* [3]*Center for Study of Complex Systems, University of Michigan.*

Abstract

This introductory chapter gives a brief description of genetic programming (GP); summarizes current GP algorithm aims, issues, and progress; and finally reviews the contributions of this volume, which were presented at the GP Theory and Practice (GPTP) 2009 workshop.

This year marks a transition wherein the aims of GP algorithms – reasonable *resource* usage, high quality *results*, and *reliable* convergence – are being consistently realized on an impressive variety of "real-world" *applications* by skilled practitioners in the field. These aims have been realized due to GP researchers' growing collective understanding of the nature of GP problems which require search across spaces which are massive, multi-modal, and with poor locality, and how that relates to long-discussed GP issues such as bloat and premature convergence. New ways to use and extend GP for improved computational resource usage, quality of results, and reliability are appearing and gaining momentum. These include: reduced resource usage via rationally designed search spaces and fitness functions for specific applications such as induction of implicit functions or modeling stochastic processes arising from bio-networks; improved quality of results by explicitly targeting the interpretability or trustworthiness of the final results; and heightened reliability via consistently introducing new genetic material in a structured manner or via coevolution and teaming. These new developments highlight that GP's challenges have changed from simply "making it work" on smaller problems, to consistently and rapidly getting high-quality results on large real-world problems. GPTP 2009 was a forum to advance GP's state of the art and its contributions demonstrate how these aims can be met on a variety of difficult problems.

1. The Workshop

In the beautiful, springtime charm of Ann Arbor, the seventh annual Genetic Programming Theory and Practice (GPTP) workshop was held at the University of Michigan campus from May 14-16, 2009.

We are grateful to all sponsors and acknowledge the importance of their contributions to such an intellectually productive and regular event. The workshop is generously founded and sponsored by the University of Michigan Center for the Study of Complex Systems (CSCS) and receives further funding from the following people and organizations: Michael Korns of Freeman Investment Management, Ying Becker of State Street Global Advisors, John Koza of Third Millenium, Bill and Barbara Tozier of Vague Innovation, Mark Kotanchek of Evolved Analytics, Jason Moore of the Computational Genetics Laboratory of Dartmouth College and Conor Ryan of the Biocomputing and Developmental Systems Group of the University of Limerick.

To make the results of the workshop useful to even a relative novice in the field of GP, we start the chapter with a brief overview of genetic programming (GP). Sections 3 and 4 describe current GP challenges and progress in GP. Sections 5 and 6 then organize and summarize the contributions of chapters in this volume from two perspectives: according to how contributed empirical research is informing GP practice, then according to the domains of application in which success through best practices has been reported. We conclude with a discussion of observations that emerged from the workshop and potential avenues of future work.

2. A Brief Introduction to Genetic Programming

Genetic programming (GP) is a search and optimization technique for executable expressions that is modeled on natural evolution. Natural evolution is a powerful process that can be described by a few central, general mechanisms; for an introduction, see (Futuyma, 2009). A population is composed of organisms which can be distinguished in terms of how fit they are with respect to their environment. Over time, members of the population breed in frequency proportional to their fitness. The new offspring inherit the combined genetic material of their parents with some random variation, and may replace existing members of the population. The entire process is iterative, adaptive and open ended. GP and other evolutionary algorithms typically realize this central description of evolution, albeit in somewhat abstract forms. GP is a set of algorithms that mimic of survival of the fittest, genetic inheritance and variation, and that iterate over a “parent”population, selectively “breeding” them and replacing them with offspring.

Though in general evolution does not have a problem solving goal, GP is nonetheless used to solve problems arising in diverse domains ranging from engineering to art. This is accomplished by casting the organism in the population as a candidate program-like solution to the chosen problem. The organism is represented as a computationally executable expression (aka structure), which is considered its genome. When the expression is executed on some supplied set of inputs, it generates an output (and possibly some intermediate results). This execution behavior is akin to the natural phenotype. By comparing the expression's output to target outputs, a measure of the solution's quality is obtained. This is used as the "fitness" of an expression. The fact that the candidate solutions are computationally executable structures (expressions), not binary or continuous coded values which are elements of a solution, is what distinguishes GP from other evolutionary algorithms (O'Reilly and Angeline, 1997). GP expressions include LISP functions (Koza, 1992; Wu and Banzhaf, 1998), stack or register-based programs (Kantschik and Banzhaf, 2002; Spector and Robinson, 2002), graphs (Miller and Harding, 2008; Mattiussi and Floreano, 2007; Poli, 1997), programs derived from grammars (Ryan et al., 2002; Whigham, 1995; Gruau, 1993), and generative representations which evolve the grammar itself (Hemberg, 2001; Hornby and Pollack, 2002; O'Reilly and Hemberg, 2007). Key steps in applying GP to a specific problem collectively define its search space: the problem's candidate solutions are designed by choosing a representation; variation operators (mutation and crossover) are selected (or specialized); and a fitness function (objectives and constraints) which expresses the relative merits of partial and complete solutions is formulated.

3. Genetic Programming Challenges

Current challenges for GP include economizing on GP *resource* usage, ensuring better quality *results*, extracting more *reliable* convergence, or applying GP to a challenging *problem domain*.

Economic Resource Usage includes shorter runtime, reduced usage of processor(s), and reduced memory and disk usage. Achieving it has traditionally been a major issue for GP. A key reason is that GP search spaces are astronomically large, multi-modal, and have poor locality. Poor locality means that a small change in the individual's genotype often leads to large changes in the fitness, introducing additional difficulty into the search effort. For example, the GP "crossover" operation of swapping the subtrees of two parents might change the comparison of two elements from a "less than" relationship to an "equal to" relationship. This usually gives dramatically different behavior and fitness. To handle such challenging search spaces, significant exploration is needed (e.g. large population sizes). This entails intensive processing and memory needs. Exacerbating the problem, fitness evaluations (objectives and constraints) of real-world problems tend to be expensive. Finally, because GP expressions have variable length, there is a tendency for them to "bloat"— to grow rapidly without a corresponding increase in performance. Bloat can be a significant drain on available memory and CPU resources.

Ensuring Quality Results. The key question is: "can a GP result *be used* in the target application?" This *usability* criteria may be more difficult to attain than evident at first glance because the result may need to be human-interpretable, trustworthy, or predictive on dramatically different inputs—and attaining such qualities can be challenging. Ensuring quality results has always been perceived as an issue, but the goal is becoming more prominent as GP is being applied to more real world problems. Practitioners, not GP, are responsible for deploying a GP result in their application domain. This means that practitioners (and potentially their clients) must trust the results sufficiently to be comfortable using them. Human-interpretability (readability) of the result is a key factor in trust. This can be an issue when deployment of the result is expensive or risky, such as analog circuit design (McConaghy and Gielen, 2005); when customers' understanding of the solution is crucial such as portfolio strategies (Becker et al., 2007); when the result must be inspected or approved; or to gain acceptance of GP methodology, e.g. for use of symbolic regression for modeling industrial processes (Kordon et al., 2005).

Reliable convergence means that the GP run can be trusted to return reasonable results, without the practitioner having to worry about premature convergence or whether algorithm parameters like population size were set correctly. GP can fail to capably identify sub-solutions or partially correct solutions and thus be unable to successfully promote, combine and reuse them to generate good solutions with effective structure. The default approach has been to use the largest population size possible, subject to time and resource constraints. However, this invariably implies high resource usage, and still gives no guarantee of hitting useful results even if such results exist.

Problem domains present both opportunities and challenges for GP. Due to its evolution of executable expressions, GP has a far broader set of problem domain opportunities than other EAs and optimization approaches. But expression spaces are non-trivial to search across and selecting the expression primitives is non-trivial. GP representation and variation operator designs must generate syntactically valid expressions. But that's the easy part! The design must be done thoughtfully. Poor choices will lead to high resource usage and poor quality results. Thoughtfully designed representations and operators can lead to orders of magnitude difference in speed or quality; e.g. as shown in (Poli and Page, 2000; McConaghy et al., 2007).

4. Progress in Genetic Programming

The field of GP is making progress in addressing the challenges described in the last section. Resource usage has been decreased by improved algorithm design, improved design of representation and operators in specific domains. Its impact has been lessened by Moore's Law and increasing availability of parallel computational approaches, meaning that computational resources become exponentially cheaper over time. Results quality has improved for the same reasons, and due to a new emphasis by GP practioners on getting interpretable or trustworthy results. Reliability has been improved via algorithm techniques that support continuous evolutionary improvement in a systematic or structured fashion. For example, by using hierarchical fair competition (HFC) and Age-Layered Population Structure (ALPS) (Hu et al., 2003; Hornby, 2006), the practitioner no longer has to "hope" that the algorithm isn't stuck. Finally, practice in thoughtful design of expression representation and genetic operators, for general and specific problem domains, has led to GP systems achieving human-competitive performance. In the 2008 ACM SIGEVO annual Genetic and Evolutionary Computation Conference (GECCO) Humies competition GP was used to generate a novel synthetic RTL benchmark circuit (Pecenka et al., 2008) and to evolve terms with special properties for a large finite algebra (Spector et al., 2008). GP has been adopted for industrial scale modeling, data analysis , design and discovery (Kotanchek et al., 2007; Terry et al., 2006). In GPTP, we have seen applications ranging from finance to biology to antennae: (Kim et al., 2008; Korns, 2007; Driscoll et al., 2003; Lohn et al., 2005).

Despite these achievements, GP's computer-based evolution does not demonstrate the potential associated with natural evolution, nor does it always satisfactorily solve important problems we might hope to use it on. Even when using best-practice approaches to manage challenges in resources, results, and reliability, the computational load may still be too excessive and the final results may be inadequate. To achieve success in a difficult problem domain takes a great deal of human effort toward thoughtful design of representations and operators.

In the two sections that follow we provide two perspectives on the GPTP workshop's intellectual contributions and on the trends we observed with respect to resource economization, results quality and reliable convergence. First, we review how the empirical research contributions have informed GP practice . Second, we review how GP has achieved successful application by the employment of "best practice" approaches.

5. Empirical Research Informing Practice

The intent of GPTP has been to bring together practitioners and theorists in order to unify the challenges practitioners face with the questions theorists study.

As well, GPTP provides a focused group setting where practitioners describe to theorists their problems, their GP system, and the issues they have encountered. This helps the theorists to better appreciate the nature of a problem, examine the practical outcome of an approach and, with immediacy, suggest how and why something is happening and what could be done about it. With the theorists present, there is an opportunity for practitioners to ask whether their theoretical findings are illustrated in some aspect of their implementation, and whether a theoretical result can shed light on a problem they face.

One of the trends our readers might notice this year is fewer "conventional" theory submissions. Conventional GP theory is difficult by nature of GP's variable length genome representation, executable phenotype character, and stochasticity. It does not proceed as quickly in terms of novelty and major impact as practice.

This year marks contributions that inform practice, yet are not strictly pen-and-paper theorems and calculations. With Chapter 6 as an example, test problems are chosen to appropriately challenge a proposed technique, and the analysis provides an understanding of how it works. GPTP workshop participants have embraced this sort of study because it focuses on one issue while elegantly eliminating unrelated complexity and confounding factors. The theory is in the form of techniques that are measurably better, more transparently analyzed and better explained and deduced. This kind of result promotes a general (applicable across GP problem domains) best-practice approach and has occurred in approaches to designing representations, operators or fitness functions, or approaches to enhanced reliability, quality of results, and resource usage. Development of a best-practice approach is arguably "empirical research" theory.

The contributions of this volume can be organized accordingly:

- One best-practice approach to enhanced reliability and results quality is to reduce and modulate selection pressure on a specific cohort of the population. Modulation could be applied to new genetic material, to genetic material that is not the norm, or to expressions that trade off strict functionality with solution complexity. One specific technique which is gaining common use is Age-Layered Population Structure (ALPS) (Hornby, 2006). It provides a structured way for new genetic material to continually enter the population, allowing new individuals time to improve before they have to compete against older, more fit individuals. Because this approach is capable and also makes a run's success less sensitive to population size, the number of research groups adopting ALPS or similar mechanisms is growing (Hu et al., 2003; McConaghy et al., 2007; Patel and Clack, 2007; Sun et al., 2007; Willis et al., 2008; Korns and Nunez, 2008; Kotanchek et al., 2008; Slany, 2009). In Chapter 6, Hornby presents the steady-state variant of the ALPS.

- With respect to best practices in design of GP fitness functions, there are four papers which describe how fitness function design was the key to make each respective problem tractable for GP. In Chapter 4, Kotanchek *et al.* describe the "Data Balancing " technique, which, among other benefits, can reduce the cost of symbolic regression fitness functions by reducing the training data to a smaller yet representative set. In Chapter 10, Ross and Imada describe multi-objective techniques that can exploit feature tests which provide different dynamical-system descriptions of stochastic, noisy time series. In Chapter 5, Schmidt and Lipson describe fitness functions that provide GP with sufficient selectivity to evolve implicit functions. In Chapter 9, Citi *et al.* describe a mapping from genotype to fitness function for Electroencephalography (EEG) signal classification.
- With respect to best practices in representation and operator design on specific problems, there are four papers. In Chapter 3, Doucette *et al.* describe how to decompose a high-dimensional classification problem into subproblems that can be solved by a *team* of GP individuals. In Chapter 7, McConaghy *et al.* describe a technique to transform a high-dimensional symbolic regression problem into a 1-dimensional problem. This dramatically simplifies the problem that GP has to solve. In Chapter 11, Shirakawa and Nagao describe a simple, easy-to-apply representation for evolving register-based software programs, a general-purpose problem-solving method. In Chapter 13, Korns describes an operator to create a conditional expression of two subtrees in a behavior preserving fashion which enhances locality, and he also describes operators to locally explore symbolic regression functional spaces.
- With respect to best practices in *general* design of representations and operators , there are three papers. In Chapter 2, Greene *et al.* apply a GP system with a hierarchical organization of search operator control: evolving a single scalar for mutation probability at the top level, and at successively lower levels, evolving more fine-grained control down to the level where individuals themselves are manipulated. In Chapter 8, Wilson and Banzhaf apply the "PAM DGP" approach which adapts the mapping from genotype to phenotype *during* evolution. In Chapter 12, Bongard describes a "functional crossover" operator which aims to enhance the locality of search by restricting allowable subtree swaps to subtrees with similar output ranges.

6. GPTP 2009: Application Successes Via Best Practices

As discussed earlier, progress in the field of GP can be characterized by GP successes in attacking challenging, industrial-strength, human-competitive problem domains. In attacking such problems and sharing their experiences at forums like GPTP, the best practices emerging from the successes are propagated and improved, leading to further successes in a variety of domains. This section organizes the papers in the volume according to problem domain. The problem domain groupings are: GP as a “discovery engine”, time-domain modeling, high-dimensional symbolic regression and classification, financial applications, and design of graph-based structures. In this book, each domain is represented by multiple papers.

GP as a Discovery Engine

The fact that GP can return an *interpretable* expression has been recognized as important for a long time (Koza, 1992), due to its implications for scientific discovery and engineering analysis (Keijzer, 2002). This volume marks two important steps towards broad use of GP: (1) capturing a new, broad class of functional forms which underpin many types of scientific theories, and (2) an easy-to-use GP system with novel data analysis capabilities, built directly into a world-standard mathematical package.

In Chapter 5, Schmidt and Lipson describe how many types of scientific problems have an *implicit* functional form: the functions are not merely a mapping from input variables to output variables, but instead a system of equations describes relationships among variables. For example, $x^2 + y^2 = z^2$ describes the equation for a circle; there is no single output variable. The challenge in discovering such functional forms is that a traditional least-squares comparison between target values and actual values is not meaningful, because the true problem involves capturing a surface (manifold) embedded within a multi-dimensional space. Simplistic fitness functions do not provide enough differentiation among candidate functions, making it hard for GP to find good initial designs and even harder to refine designs. To solve the problem, the authors propose the use of local finite-element analysis to measure gradients in the manifold, and then apply a least-squares error measure to differences in gradients. The authors demonstrate how the approach can successfully capture the dynamics in classical pendulum physics models, as well as capturing dynamics of more complex pendulum models for which closed-form equations describing dynamics are unknown.

In Chapter 4, Kotanchek *et al.* describe the use of a highly visual, easy-to-use GP symbolic regression system that is embedded in Mathematica. The visual, exploratory nature of the system leads to a truly iterative, interactive means to use GP to explore data in real time. The paper describes techniques to detect outliers in either a data-based or model-based fashion, measure relative importance among variables, detect regions in an input-output mapping space which are over- or under-represented by the training data at hand and rank the importance of each datapoint. They also describe a "Data Balancing" technique which is a key tool for many of these techniques.

Time-Domain Modeling

This year GPTP had three papers addressing three very different problems related to time-series signals: EEG time series classification, modeling stochastic reaction processes, and time series with many state variables.

In Chapter 9, Citi *et al.* classify time domain Electroencephalography (EEG) signals with the aim of improving brain-computer interfaces (BCIs). The approach focuses on Event-Related Potentials (ERPs) which are well-defined events within EEG signals. EEG signals during an ERP have characteristic waveforms that provide the possibility of accurate classification. While ERPs have been explored extensively, an issue is the large number of human-in-the-loop training trials. In past work Citi *et al.* have partly alleviated this using a simple binning technique but this moved the issue to selection of the bin properties themselves. In this volume they use GP to evolve probabilistic membership functions for the bins which yields promising improvement in performance.

GP is well suited to learning models that synthesize reaction processes because a language from the domain and domain dependent operations on the data can be transfered quite directly to the GP function and terminal set. This is the case with pi-calculus and process algebra structures that model reactions of bio-networks. However when the reaction process is stochastic, rather than deterministic, a challenge arises in specifying fitness objectives. Just using the error between model prediction and real data fails to account for the statistical features in the time series that arise from stochastic timing and variance. In Chapter 10, Ross and Imada discuss and evaluate how different statistical feature tests can be used simultaneously via multi-objective GP.

In Chapter 12, building on past work, Bongard applies GP to reverse engineering a broad set of dynamical systems. Because the systems are deterministic, and known in advance, Bongard's measure of success is whether GP can successfully recapture the original differential equations. While the focus of the paper is a novel crossover operator, the paper reconfirms that GP is consistently effective at capturing the system dynamics for a variety of problems.

High-Dimensional Symbolic Regression and Classification

GP modeling approaches have typically attacked problems in the range of 1 to roughly 20 dimensions. But it is well known that the nature of a problem dramatically shifts past 20 dimensions, because every training data point is effectively "very far away" from every other datapoint (Hastie et al., 2001). Problems with 100, 1000, or 10,000 input dimensions have very different properties. In this book, we have three diverse problems with high-dimensional inputs: high-dimensional classifier design, high-dimensional regressor design and identification of key input variable interactions (i.e., epistasis).

In Chapter 2, Greene *et al.* tackle what they deem a “needle in a haystack” problem: 10,000+ input variables but only a few have an effect, and the variable interactions have more effect than single-variable effects. The combination of high dimensionality and epistasis means one tough problem. The application is for DNA analysis, to identify which sequence variations predict disease risk in human populations. The authors approach the problem via a GP system with hierarchical operator control, and demonstrate that GP is indeed able to extract expressions of great use to geneticists.

In supervised problems, where a model has to be learned from a class of exemplars with a domain of attributes, GP has been successfully used to find a single binary classifier that automatically identifies the relevant subset of attributes. However, for domains of large numbers of attributes, it is more natural to consider grouping the exemplars and learning a set of cooperative classifiers that function in a non-overlapping way over the subgroups. Different (and overlapping) sets of attributes are appropriate to each classifier. In Chapter 3, Doucette *et al.* show how to extend GP so it can accomplish this kind of classification without requiring any preliminary *ad hoc* intervention to group the exemplars or attributes. Furthermore, the resulting classifier set is a product of a single GP run. This is more efficient than using multiple runs to incrementally learn binary classifiers for multiple classes.

In Chapter 7, McConaghy *et al.* describe a class of regression problems where the input variables cannot be heavily pruned to a few key variables, because most variables have some effect. This class of problems includes modeling the effect of manufacturing variation in analog electronic circuits. The paper shows that traditional GP approaches fail badly on such a problem, along with many other well-known regression and data-mining techniques. It then proposes a “latent variable” solution, in which the input vector is transformed to a scalar via a linear transformation, then the scalar is passed through a nonlinear GP expression to get the output. The process is repeated on the residuals. The challenge is in determining the linear transformation vectors, and the final expression; the result is demonstrated to have effective prediction on unseen inputs.

Financial Applications

GPTP has regularly reported contributions from the domain of finance (Zhou, 2003; Yu et al., 2004; Caplan and Becker, 2004; Becker et al., 2006; Korns, 2006; Becker et al., 2007; Korns, 2007; Chen et al., 2008; Korns and Nunez, 2008; Kim et al., 2008). This year marks two new papers advancing the state of the art of GP application in the area of finance.

Over a number of years, a large-scale, industrial-strength, symbolic regression-classification GP system used for trading models developed by Investment Science Corporation has been revised, extended and improved. It combines standard genetic programming with abstract expression grammars, particle swarm optimization, differential evolution, context aware crossover and age-layered populations. Chief designer, Michael Korns, now of Freeman Investment Management, has stated that its design and analysis has been guided by insights gained from theoretical findings presented at GPTP. He also credits observations and analyses arising during cross-connecting discussions by participants. Korn's contribution this year, in Chapter 13, targets techniques for improving symbolic regression in cases where the target expression contains conditionals. The system is enhanced with pessimal vertical slicing, splicing of uncorrelated champions via abstract conditional expressions, and abstract mutation and crossover.

GPTP also welcomes a new team working on financial modeling. In Chapter 8, co-authors Wilson and Banzhaf consider day trading where a hold, buy or sell decision is made for each security on a daily basis. Predictions of returns are based on the recent past. The system addressing the problem is a developmental co-evolutionary genetic programming approach called PAM DGP. It was demonstrably better than with standard linear genetic programming in terms of profitable buys, but not necessarily protective sells, in particular stock price trend scenarios.

Design of Graph-Based Structures

In Chapter 11, Shirakawa and Nagao propose a method called Graph Structured Program Evolution (GRAPE). GRAPE expressions are graphs capable of expressing conditional branches and loops, which can be executed in a register-based computational machine. Graphs are complemented with a data set for each of the multiple data types GRAPE supports. The genotype is a linear string of integers. GRAPE is evaluated on problems emblematic of iterative and conditional requirements: factorial, exponentiation, and list sorting. While it can solve these instances, challenges remain with the number of evaluations required and the complexity of the solutions.

In Chapter 6, Hornby describes the application of ALPS to two problems: evolving a NASA X-Band antenna, and evolving the structure of a table. The generative representation used for tables and antennae (GENRE) is general enough to handle graph-based structures. While the focus of the paper was ALPS itself, the improved quality of the results themselves is notable.

7. Themes, Summary and Looking Forward

The consensus among the participants this year was that genetic programming has reached a watershed in terms of practicality for a well defined range of applications. With appropriate determination of algorithm techniques, representation, operators, and fitness function, GP has applicability to such challenging problems as scientific discovery and data modeling, time-domain modeling, high-dimensional symbolic regression and classification, financial applications, and design of graph-based structures. In this book, each domain is represented by two to three papers.

The participants expressed confidence, based on experience, that there are successful technical approaches that alleviate commonly occurring problems such as premature evolutionary convergence, bloat, and scalability. Employing these approaches has become "standard practice" among the participants, though they admittedly are experts. This convergence on approaches has arisen over the course of multiple annual GPTP meetings. Participants first proposed diverse solutions, some of which were stimulated by GP theory. Then, when brought into the GPTP forum, the solutions were collectively analyzed for key similarities, differences and capabilities. This enabled those present to arrive at an understanding of central principles and to unify their ideas into recognizable broader technical approaches with theoretical and empirical foundations. It is this process that has bolstered the participants' confidence in new techniques and from which best-practice approaches have emerged.

There will always be tradeoffs among results, resources, reliability and human up-front setup effort in designing representation, operators, and fitness functions. The workshop seems to herald a transition away from these largely-explored issues toward those that arise from using GP for other purposes. The new directions for GP that are exciting and present their unique challenges are, for example:

- What fundamental contributions will allow GP to be adopted into broader use beyond that of expert practitioners? For example, how can GP be scoped so that it becomes another standard, off-the-shelf method in the "toolboxes" of scientists and engineers around the world? Can GP follow in the same vein of linear programming? Can it follow the example of support vector machines and convex optimization methods? One challenge is in formulating GP so that it provides more ease in laying out a problem. Another is determining how, by default—without parameter tuning—GP can efficiently exploit specified resources to return results reliably.

- Success with GP often requires extensive human effort in capturing and embedding the domain knowledge. How can this up-front human effort be reduced while still achieving excellent results? Are there additional automatic ways to capture domain knowledge for input to GP systems?

How can a system of evolutionary modules interact to exploit domain knowledge?

- Scalability is always relative. GP has attacked fairly large problems, but how can GP be improved to solve problems that are 10x, 100x, 10,000x or 1,000,000x harder?
- How can the inherent distributed nature of GP be better and more easily exploited, especially in the current era of multicore CPUs, GPUs, and cloud computing? What are the implications of distribution in terms of algorithm dynamics and capabilities?
- How can GP be extended with more sophisticated evolutionary mechanisms such as co-evolution or speciation to improve its ability to generate solutions that exhibit complex properties such as module formation, module reuse and self-organization into hierarchies and high level systems?
- What other "uncrackable" problems await a creative GP approach?

These questions and their answers will provide the fodder for future GPTP workshops. We wish you many hours of stimulating reading of this volume's contributions.

References

Becker, Ying, Fei, Peng, and Lester, Anna M. (2006). Stock selection : An innovative application of genetic programming methodology. In Riolo, Rick L., Soule, Terence, and Worzel, Bill, editors, *Genetic Programming Theory and Practice IV*, volume 5 of *Genetic and Evolutionary Computation*, chapter 12, pages 315–334. Springer, Ann Arbor.

Becker, Ying L., Fox, Harold, and Fei, Peng (2007). An empirical study of multi-objective algorithms for stock ranking. In Riolo, Rick L., Soule, Terence, and Worzel, Bill, editors, *Genetic Programming Theory and Practice V*, Genetic and Evolutionary Computation, chapter 14, pages 241–262. Springer, Ann Arbor.

Caplan, Michael and Becker, Ying (2004). Lessons learned using genetic programming in a stock picking context. In O'Reilly, Una-May, Yu, Tina, Riolo, Rick L., and Worzel, Bill, editors, *Genetic Programming Theory and Practice II*, chapter 6, pages 87–102. Springer, Ann Arbor.

Chen, Shu-Heng, Zeng, Ren-Jie, and Yu, Tina (2008). Co-evolving trading strategies to analyze bounded rationality in double auction markets. In Riolo, Rick L., Soule, Terence, and Worzel, Bill, editors, *Genetic Programming Theory and Practice VI*, Genetic and Evolutionary Computation, chapter 13, pages 195–215. Springer, Ann Arbor.

Driscoll, Joseph A., Worzel, Bill, and MacLean, Duncan (2003). Classification of gene expression data with genetic programming. In Riolo, Rick L. and Worzel, Bill, editors, *Genetic Programming Theory and Practice*, chapter 3, pages 25–42. Kluwer.

Futuyma, Douglas (2009). *Evolution, Second Edition*. Sinauer Associates Inc.

Gruau, Frederic (1993). Cellular encoding as a graph grammar. *IEE Colloquium on Grammatical Inference: Theory, Applications and Alternatives*, (Digest No.092):17/1–10.

Hastie, Trevor, Tibshirani, Robert, Friedman, Jerome, and Franklin, James (2001). *The Elements of Statistical Learning*. Springer, New York, 2nd edition.

Hemberg, Martin (2001). GENR8 - A design tool for surface generation. Master's thesis, Department of Physical Resource Theory, Chalmers University, Sweden.

Hornby, Gregory S. (2006). ALPS: the age-layered population structure for reducing the problem of premature convergence. In Keijzer, Maarten, Cattolico, Mike, Arnold, Dirk, Babovic, Vladan, Blum, Christian, Bosman, Peter, Butz, Martin V., Coello Coello, Carlos, Dasgupta, Dipankar, Ficici, Sevan G., Foster, James, Hernandez-Aguirre, Arturo, Hornby, Greg, Lipson, Hod, McMinn, Phil, Moore, Jason, Raidl, Guenther, Rothlauf, Franz, Ryan, Conor, and Thierens, Dirk, editors, *GECCO 2006: Proceedings of the 8th annual conference on Genetic and evolutionary computation*, volume 1, pages 815–822, Seattle, Washington, USA. ACM Press.

Hornby, Gregory S. and Pollack, Jordan B. (2002). Creating high-level components with a generative representation for body-brain evolution. *Artificial Life*, 8(3):223–246.

Hu, Jianjun, Goodman, Erik D., and Seo, Kisung (2003). Continuous hierarchical fair competition model for sustainable innovation in genetic programming. In Riolo, Rick L. and Worzel, Bill, editors, *Genetic Programming Theory and Practice*, chapter 6, pages 81–98. Kluwer.

Kantschik, Wolfgang and Banzhaf, Wolfgang (2002). Linear-graph GP—A new GP structure. In Foster, James A., Lutton, Evelyne, Miller, Julian, Ryan, Conor, and Tettamanzi, Andrea G. B., editors, *Genetic Programming, Proceedings of the 5th European Conference, EuroGP 2002*, volume 2278 of *LNCS*, pages 83–92, Kinsale, Ireland. Springer-Verlag.

Keijzer, Maarten (2002). *Scientific Discovery using Genetic Programming*. PhD thesis, Danish Technical University, Lyngby, Denmark.

Kim, Minkyu, Becker, Ying L., Fei, Peng, and O'Reilly, Una-May (2008). Constrained genetic programming to minimize overfitting in stock selection. In Riolo, Rick L., Soule, Terence, and Worzel, Bill, editors, *Genetic Programming Theory and Practice VI*, Genetic and Evolutionary Computation, chapter 12, pages 179–195. Springer, Ann Arbor.

Kordon, Arthur, Castillo, Flor, Smits, Guido, and Kotanchek, Mark (2005). Application issues of genetic programming in industry. In Yu, Tina, Riolo, Rick L., and Worzel, Bill, editors, *Genetic Programming Theory and Practice III*, volume 9 of *Genetic Programming*, chapter 16, pages 241–258. Springer, Ann Arbor.

Korns, Michael F. (2006). Large-scale, time-constrained symbolic regression. In Riolo, Rick L., Soule, Terence, and Worzel, Bill, editors, *Genetic Programming Theory and Practice IV*, volume 5 of *Genetic and Evolutionary Computation*, chapter 16, pages –. Springer, Ann Arbor.

Korns, Michael F. (2007). Large-scale, time-constrained symbolic regression-classification. In Riolo, Rick L., Soule, Terence, and Worzel, Bill, editors, *Genetic Programming Theory and Practice V*, Genetic and Evolutionary Computation, chapter 4, pages 53–68. Springer, Ann Arbor.

Korns, Michael F. and Nunez, Loryfel (2008). Profiling symbolic regression-classification. In Riolo, Rick L., Soule, Terence, and Worzel, Bill, editors, *Genetic Programming Theory and Practice VI*, Genetic and Evolutionary Computation, chapter 14, pages 215–229. Springer, Ann Arbor.

Kotanchek, Mark, Smits, Guido, and Vladislavleva, Ekaterina (2007). Trustable symoblic regression models. In Riolo, Rick L., Soule, Terence, and Worzel, Bill, editors, *Genetic Programming Theory and Practice V*, Genetic and Evolutionary Computation, chapter 12, pages 203–222. Springer, Ann Arbor.

Kotanchek, Mark, Smits, Guido, and Vladislavleva, Ekaterina (2008). Exploiting trustable models via pareto GP for targeted data collection. In Riolo, Rick L., Soule, Terence, and Worzel, Bill, editors, *Genetic Programming Theory and Practice VI*, Genetic and Evolutionary Computation, chapter 10, pages 145–163. Springer, Ann Arbor.

Koza, John R. (1992). *Genetic Programming: On the Programming of Computers by Means of Natural Selection*. MIT Press, Cambridge, MA, USA.

Lohn, Jason D., Hornby, Gregory S., and Linden, Derek S. (2005). Rapid re-evolution of an X-band antenna for NASA's space technology 5 mission. In Yu, Tina, Riolo, Rick L., and Worzel, Bill, editors, *Genetic Programming Theory and Practice III*, volume 9 of *Genetic Programming*, chapter 5, pages 65–78. Springer, Ann Arbor.

Mattiussi, Claudio and Floreano, Dario (2007). Analog genetic encoding for the evolution of circuits and networks. *IEEE Transactions on Evolutionary Computation*, 11(5):596–607.

McConaghy, Trent and Gielen, Georges (2005). Genetic programming in industrial analog CAD: Applications and challenges. In Yu, Tina, Riolo, Rick L., and Worzel, Bill, editors, *Genetic Programming Theory and Practice III*, volume 9 of *Genetic Programming*, chapter 19, pages 291–306. Springer, Ann Arbor.

McConaghy, Trent, Palmers, Pieter, Gielen, Georges, and Steyaert, Michiel (2007). Genetic programming with reuse of known designs. In Riolo, Rick L., Soule, Terence, and Worzel, Bill, editors, *Genetic Programming Theory and Practice V*, Genetic and Evolutionary Computation, chapter 10, pages 161–186. Springer, Ann Arbor.

Miller, Julian Francis and Harding, Simon L. (2008). Cartesian genetic programming. In Ebner, Marc, Cattolico, Mike, van Hemert, Jano, Gustafson, Steven, Merkle, Laurence D., Moore, Frank W., Congdon, Clare Bates, Clack, Christopher D., Moore, Frank W., Rand, William, Ficici, Sevan G., Riolo, Rick, Bacardit, Jaume, Bernado-Mansilla, Ester, Butz, Martin V., Smith, Stephen L., Cagnoni, Stefano, Hauschild, Mark, Pelikan, Martin, and Sastry, Kumara, editors, *GECCO-2008 tutorials*, pages 2701–2726, Atlanta, GA, USA. ACM.

O'Reilly, Una-May and Angeline, Peter J. (1997). Trends in evolutionary methods for program induction. *Evolutionary Computation*, 5(2):v–ix.

O'Reilly, Una-May and Hemberg, Martin (2007). Integrating generative growth and evolutionary computation for form exploration. *Genetic Programming and Evolvable Machines*, 8(2):163–186. Special issue on developmental systems.

Patel, S. and Clack, C. D. (2007). ALPS evaluation in financial portfolio optimisation. In Srinivasan, Dipti and Wang, Lipo, editors, *2007 IEEE Congress on Evolutionary Computation*, pages 813–819, Singapore. IEEE Computational Intelligence Society, IEEE Press.

Pecenka, Tomas, Sekanina, Lukas, and Kotasek, Zdenek (2008). Evolution of synthetic rtl benchmark circuits with predeÞned testability. The 5th Annual (2008) ÒHUMIESÓ Awards.

Poli, Riccardo (1997). Evolution of graph-like programs with parallel distributed genetic programming. In Back, Thomas, editor, *Genetic Algorithms: Proceedings of the Seventh International Conference*, pages 346–353, Michigan State University, East Lansing, MI, USA. Morgan Kaufmann.

Poli, Riccardo and Page, Jonathan (2000). Solving high-order boolean parity problems with smooth uniform crossover, sub-machine code GP and demes. *Genetic Programming and Evolvable Machines*, 1(1/2):37–56.

Ryan, Conor, Nicolau, Miguel, and O'Neill, Michael (2002). Genetic algorithms using grammatical evolution. In Foster, James A., Lutton, Evelyne, Miller, Julian, Ryan, Conor, and Tettamanzi, Andrea G. B., editors, *Genetic Programming, Proceedings of the 5th European Conference, EuroGP 2002*, volume 2278 of *LNCS*, pages 278–287, Kinsale, Ireland. Springer-Verlag.

Slany, Karel (2009). Comparison of CGP and age-layered CGP performance in image operator evolution. In Vanneschi, Leonardo, Gustafson, Steven, Moraglio, Alberto, De Falco, Ivanoe, and Ebner, Marc, editors, *Proceedings* *of the 12th European Conference on Genetic Programming, EuroGP 2009*, volume 5481 of *LNCS*, pages 351–361, Tuebingen. Springer.

Spector, Lee, Clark, David M., Lindsay, Ian, Barr, Bradford, and Klein, Jon (2008). Genetic programming for finite algebras. In Keijzer, Maarten, Antoniol, Giuliano, Congdon, Clare Bates, Deb, Kalyanmoy, Doerr, Benjamin, Hansen, Nikolaus, Holmes, John H., Hornby, Gregory S., Howard, Daniel, Kennedy, James, Kumar, Sanjeev, Lobo, Fernando G., Miller, Julian Francis, Moore, Jason, Neumann, Frank, Pelikan, Martin, Pollack, Jordan, Sastry, Kumara, Stanley, Kenneth, Stoica, Adrian, Talbi, El-Ghazali, and Wegener, Ingo, editors, *GECCO '08: Proceedings of the 10th annual conference on Genetic and evolutionary computation*, pages 1291–1298, Atlanta, GA, USA. ACM.

Spector, Lee and Robinson, Alan (2002). Genetic programming and autoconstructive evolution with the push programming language. *Genetic Programming and Evolvable Machines*, 3(1):7–40.

Sun, Lei, Hines, Evor L., Green, Roger J., Leeson, Mark S., and Iliescu, D. Daciana (2007). Phase compensating dielectric lens design with genetic programming: Research articles. *International Journal of RF and Microwave Computer-Aided Engineering*, 17(5):493–504.

Terry, Michael A., Marcus, Jonathan, Farrell, Matthew, Aggarwal, Varun, and O'Reilly, Una-May (2006). GRACE: generative robust analog circuit exploration. In Rothlauf, Franz, Branke, Jurgen, Cagnoni, Stefano, Costa, Ernesto, Cotta, Carlos, Drechsler, Rolf, Lutton, Evelyne, Machado, Penousal, Moore, Jason H., Romero, Juan, Smith, George D., Squillero, Giovanni, and Takagi, Hideyuki, editors, *Applications of Evolutionary Computing, EvoWorkshops2006: EvoBIO, EvoCOMNET, EvoHOT, EvoIASP, EvoInteraction, EvoMUSART, EvoSTOC*, volume 3907 of *LNCS*, pages 332–343, Budapest. Springer Verlag.

Whigham, P. A. (1995). Grammatically-based genetic programming. In Rosca, Justinian P., editor, *Proceedings of the Workshop on Genetic Programming: From Theory to Real-World Applications*, pages 33–41, Tahoe City, California, USA.

Willis, Amy, Patel, Suneer, and Clack, Christopher D. (2008). GP age-layer and crossover effects in bid-offer spread prediction. In Keijzer, Maarten, Antoniol, Giuliano, Congdon, Clare Bates, Deb, Kalyanmoy, Doerr, Benjamin, Hansen, Nikolaus, Holmes, John H., Hornby, Gregory S., Howard, Daniel, Kennedy, James, Kumar, Sanjeev, Lobo, Fernando G., Miller, Julian Francis, Moore, Jason, Neumann, Frank, Pelikan, Martin, Pollack, Jordan, Sastry, Kumara, Stanley, Kenneth, Stoica, Adrian, Talbi, El-Ghazali, and Wegener, Ingo, editors, *GECCO '08: Proceedings of the 10th annual conference on Genetic and evolutionary computation*, pages 1665–1672, Atlanta, GA, USA. ACM.

Wu, Annie S. and Banzhaf, Wolfgang (1998). Introduction to the special issue: Variable-length representation and noncoding segments for evolutionary algorithms. *Evolutionary Computation*, 6(4):iii–vi.

Yu, Tina, Chen, Shu-Heng, and Kuo, Tzu-Wen (2004). Discovering financial technical trading rules using genetic programming with lambda abstraction. In O'Reilly, Una-May, Yu, Tina, Riolo, Rick L., and Worzel, Bill, editors, *Genetic Programming Theory and Practice II*, chapter 2, pages 11–30. Springer, Ann Arbor.

Zhou, Anjun (2003). Enhance emerging market stock selection. In Riolo, Rick L. and Worzel, Bill, editors, *Genetic Programming Theory and Practise*, chapter 18, pages 291–302. Kluwer.

Index