edited by
Anil Menon

FRONTIERS OF EVOLUTIONARY COMPUTATION

GENAGENAGENA
GENAGENAGENA
Genetic Algorithms and Evolutionary Computation

Kluwer Academic Publishers

FRONTIERS OF EVOLUTIONARY COMPUTATION

Genetic Algorithms and Evolutionary Computation

***Consulting Editor,* David E. Goldberg**
University of Illinois at Urbana-Champaign
deg@uiuc.edu

Additional titles in the series:

Efficient and Accurate Parallel Genetic Algorithms, Erick Cantú-Paz ISBN: 0-7923-7221-2

Estimation of Distribution Algorithms: *A New Tool for Evolutionary Computation,* edited by Pedro Larrañaga, Jose A. Lozano ISBN: 0-7923-7466-5

Evolutionary Optimization in Dynamic Environments, Jürgen Branke ISBN: 0-7923-7631-5

Anticipatory Learning Classifier Systems, Martin V. Butz ISBN: 0-7923-7630-7

Evolutionary Algorithms for Solving Multi-Objective Problems, Carlos A. Coello Coello, David A. Van Veldhuizen, and Gary B. Lamont ISBN: 0-306-46762-3

OmeGA: A Competent Genetic Algorithm for Solving Permutation and Scheduling Problems, Dimitri Knjazew ISBN: 0-7923-7460-6

The Design of Innovation: *Lessons from and for Competent Genetic Algorithms,* David E. Goldberg ISBN: 1-4020-7098-5

Noisy Optimization with Evolution Strategies, Dirk V. Arnold ISBN: 1 -4020-7105-1

Classical and Evolutionary Algorithms in the Optimization of Optical Systems, Darko Vasiljević: ISBN: 1-4020- 7140-X

Evolutionary Algorithms for Embedded System Design, edited by Rolf Drechsler, Nicole Drechsler: ISBN: 1-4020- 7276-7

Genetic Algorithms and Evolutionary Computation publishes research monographs, edited collections, and graduate-level texts in this rapidly growing field. Primary areas of coverage include the theory, implementation, and application of genetic algorithms (GAs), evolution strategies (ESs), evolutionary programming (EP), learning classifier systems (LCSs) and other variants of genetic and evolutionary computation (GEC). Proposals in related fields such as artificial life, adaptive behavior, artificial immune systems, agent-based systems, neural computing, fuzzy systems, and quantum computing will be considered for publication in this series as long as GEC techniques are part of or inspiration for the system being described. Manuscripts describing GEC applications in all areas of engineering, commerce, the sciences, and the humanities are encouraged. http://www.wkap.nl/prod/s/GENA

GENA**GENA**GENA
GENAGENA**GENA**
Genetic Algorithms and Evolutionary Computation

FRONTIERS OF EVOLUTIONARY COMPUTATION

edited by

Anil Menon
ProductSoft, Inc.
Pittsburgh, Pennsylvania, USA

KLUWER ACADEMIC PUBLISHERS
NEW YORK, BOSTON, DORDRECHT, LONDON, MOSCOW

eBook ISBN: 1-4020-7782-3
Print ISBN: 1-4020-7524-3

Created in the United States of America

Visit Kluwer Online at: http://kluweronline.com
and Kluwer's eBookstore at: http://ebooks.kluweronline.com

Contents

This page intentionally left blank

List of Figures

List of Tables

This page intentionally left blank

Preface

This book is a collection of essays, authored by eminent scholars in evolutionary computation (EC), artificial intelligence (AI), operations research, complexity theory and mathematics. Each essay revolves around important, interesting and *unresolved* questions in the field of evolutionary computation.

The book is designed to be a resource to at least three categories of readers. First of all, graduate students will find this book a rich source of open research issues. Imagine participating in an EC research seminar conducted by some of the best scholars in and around the field! The book also gives experts a chance to compare and contrast their understanding of the fundamental issues in EC with the perspectives of their peers. Finally, to the interested scholar it offers a sample of the kind of problems that are considered worth solving in EC.

Much has been written about how great solutions often have a certain aesthetic appeal (symmetry, simplicity, originality, unity and so on). In sharp contrast, characteristics of great problems remain something of a mystery. It is useful to think of a problem as existing in at least one of four states: undiscovered, unsolved, solved and hibernating. However, truly interesting problems — great problems — manage a simultaneous, contrary existence in all four quadrants. A great problem, to echo Walt Whitman, is often large and contains multitudes. Every mature field has its great problems. Even fields with a progressive tradition, like Physics and Mathematics, have problems that refuse to stay solved. The problem of explaining the directionality of the thermodynamic arrow of time, and the debate over whether mathematical objects are invented or discovered are but two examples that comes to mind. Great problems act as co-ordinate systems for the geography of our imaginations and explain why we do what we do.

So it is gratifying (rather than alarming) that EC is also evolving its own collection of really hard problems. For example, is an evolutionary process an algorithmic process (in the sense of Church-Turing)? Are building blocks theoretical rather than empirical constructs? Which results in EC are dependent on problem representation and which ones independent of it? What precise role does crossover play? Is there a way to unify the different formalisms used

to model evolutionary processes? What are the characteristics of problems solvable by EC? Some of these problems are discussed at length in this volume.

This book grew out of a proposed session for the 2001 International Conference on Artificial Intelligence in Las Vegas, Nevada. I had thought that a collection of authoritative essays, each devoted to the description of a substantially unsolved problem in EC, could help bring coherence to the field, clarify its important issues, and provoke imaginations. The session was jokingly dubbed the 'Hilbert session' in memory of David Hilbert's outstanding example almost a century ago. Unfortunately, time constraints prevented the session from from going forward. But the highly positive response from the invitees, as well as from others who had heard about the idea, suggested that a book could be an alternate and appropriate forum for implementing the idea. The stray mentions of Hilbert in some of the essays thus hark back to the origins of this book. Needless to say, the essays were not written with the aim of being either as definitive or as predictive as Hilbert's talk turned out to be.

The authors in this collection are wonderfully varied in their backgrounds, writing styles and interests. But their essays are related by several common goals: extensions to EC theories, discussion of various formalisms, summaries of the state of the art, and careful speculation on what *could* be done to resolve various issues. The essays also leave no doubt that the ferment caused by active trading is producing a watershed event in the marketplace of ideas. Witness for example, the import of ideas from evolutionary theory into Algorithmics (such as: population thinking, inheritance and recombination), and the export of ideas from mathematics and computer science into evolutionary theory (such as: stochastic models, complexity theory, computability). Ideally, I would have liked to triple the size of the book, include at least a dozen more authors, and reprint essays from relevant collections. On the other hand, progress is a side-effect of achieving the possible. While the sample of ideas and authors herein is certainly not comprehensive, it is very much representative of what is possible in our field.

EC is a young discipline, and consequently, it is still a field that has the rare chance to be defined in terms of its unsolved problems, rather than its solved ones. No doubt, the many encounters offered in this book, the journeys it will inspire, and the inevitable predilection of problems to get solved, will change this situation in the next few decades. But till then, this book is meant to serve as a beckoning toward the roads still not taken.

Contributing Authors

Bahrain Alidaee is an associate professor of Operations Management at the Business School, the University of Mississippi. His research interests include combinatorial optimization, heuristic programming, and game theory. He has published more than 40 articles in journals such as *Management Science*, *Transportation science, IEEE Transactions, European Journal of OR, Journal of Operational Research, Computers and Operations Research, Production and Operations Management* and other journals. He is a member of INFORMS, DSI, APICS, and POMS.

Lee Altenberg is Associate Professor of Information and Computer Sciences at the University of Hawaii at Manoa. His interest is in systems phenomena, and he has focused on the dynamics of evolutionary processes. Of particular interest is the emergence of the representation problem in evolution, the evolution of the genotype-phenotype map, and evolutionary dynamics of modularity. His publications bridge the fields of mathematical population genetics and evolutionary computation. Recent civic projects include restoration of dryland Hawaiian biodiversity, reduction of light pollution, and control of alien ungulates on Maui.

Vivek Balaraman works as a research scientist in the Artificial Intelligence Group of the Tata Research Development and Design Centre, Pune, India where he has been since 1989. Prior to that he worked at the Knowledge Based Computer Systems Laboratory, Department of Computer Science, Indian Institute of Technology, Madras. Since 1995 he has led the Case-Based Reasoning research and development team at TRDDC. The research has led to the domain independent CBR kernel engine which has been applied successfully on a variety of problems, among them diagnosis, experiential knowledge management and cognitive structured search in tasks like directory assistance and job search at portals. Patent applications have been filed for several aspects of this work. Vivek's research interests include machine learning, evolutionary theory and cognitive memory models.

Wolfgang Banzhaf is Associate Professor of Applied Computer Science at the University of Dortmund, Germany. He is lead author of the textbook *Genetic Programming — An Introduction* and editor-in-chief of the Kluwer journal *Genetic Programming and Evolvable Machines.* He has published more than 80 refereed conference and journal articles.

Carlos Cotta received the M.Sc. and Ph.D. degrees in 1994 and 1998, respectively, in Computer Science from the University of Málaga (UMA), Spain. He is currently affiliated to the Department of "Lenguajes y Ciencias de la Computación" of the UMA, where he holds an Associate Professorship in Programming Languages and Computer Systems. He has been previously appointed as Lecturer (1995–1999), and Assistant Professor (1999–2001) in this institution.

His research interests are primarily in evolutionary algorithms, both from the algorithmic (design techniques, theoretical foundations, parallelism, and hybridization) and the applied (combinatorial optimization, data mining, and bioinformatics) standpoint. He is the author or co-author of over 40 articles on these topics.

He is a member of the European Network of Excellence on Evolutionary Computing (EvoNet), the European Chapter on Metaheuristics (EU/ME), and the ACM Special Interest Group on Applied Computing (ACM SIGAPP) among other research societies and organizations. He has also served in the Programme Committee of the major conferences in the field (GECCO, CEC, and PPSN among others), and has refereed articles for scientific journals such as the *Journal of Heuristics,* and *IEEE Transactions on Evolutionary Computation* among others.

Kenneth A. De Jong is Professor of Computer Science at George Mason University, and a member of the research faculty at the Krasnow Institute. He received his PhD at the University of Michigan under the direction of John Holland. Dr. De Jong's research interests include evolutionary computation, adaptive systems, and machine learning. He is an active member of the evolutionary computation research community with a large number of papers, Ph.D. students, and presentations in this area. He is also involved in the organization of many of the workshops and conferences on evolutionary computation, and the founding Editor-in-chief of the journal Evolutionary Computation, published by MIT Press. He is currently serving on the executive council of the International Society for Genetic and Evolutionary Computation. Dr. De Jong is head of the Evolutionary Computation Laboratory at GMU, consisting of a group faculty members and graduate students working on a variety of research projects involving the application of evolutionary algorithms to difficult computational problems such as visual scene analysis and programming complex

robot behaviors. This group is also involved in extending current evolutionary computation models to include more complex mechanisms such as speciation, co-evolution,and spatial extent. These ideas are being developed to improve both the applicability and scalability of current evolutionary algorithms to more complex problem domains. Funding for the lab comes from a variety of sources including DARPA, ONR, NRL, NSF, and local area companies. Further details can are available at www.cs.gmu.edu/ eclab.

Fred Glover is the MediaOne Chaired Professor in Systems Science at the University of Colorado, Boulder, and Distinguished Researcher and co-founder of the Hearin Center for Enterprise Science at the University of Mississippi. He has authored or co-authored more than three hundred published articles and six books in the fields of mathematical optimization, computer science and artificial intelligence, with particular emphasis on practical applications in industry and government. In addition to holding editorial and advisory posts for journals in the U.S. and abroad, Dr. Glover has been featured as a National Visiting Lecturer by the Institute of Management Science and the Operations Research Society of America and has served as a host and lecturer in the U.S. National Academy of Sciences Program of Scientific Exchange. Professor Glover is the recipient of the distinguished von Neumann Theory Prize, a member of the National Academy of Engineering, and has received numerous other awards and honorary fellowships, including those from the American Association for the Advancement of Science (AAAS), the NATO Division of Scientific Affairs, the Institute of Operations Resarch and Management Science (INFORMS), the Decision Sciences Institute (DSI), the U.S. Defense Communications Agency (DCA), the Energy Research Institute (ERI), the American Assembly of Collegiate Schools of Business (AACSB), Alpha Iota Delta, and the Miller Institute for Basic Research in Science. He serves on the advisory boards of several organizations and is co-founder of OptTek Systems, Inc.

Martin A. Keane received a Ph.D. in Mathematics from Northwestern University in 1969. He worked for Applied Devices Corporation until 1972, in the Mathematics Department at General Motors Laboratory until 1976, and was Vice-President for Engineering of Bally Manufacturing Corporation until 1986. He is currently chief scientist of Econometrics Inc. of Chicago and a consultant to various computer-related and gaming-related companies.

Gary A. Kochenberger is the Professor of Operations Management, University of Colorado at Denver (since 1989). In recent years, his focus has been on problems of a combinatorial nature as commonly found in logistical management, operations management, and related areas. He has published three

books and more than 40 refereed articles in top journals in his field including Management Science, Mathematical Programming, Journal of Optimization Theory and Applications, Operation Research, Computers and Operations research, Naval Research Logistics Quarterly, Journal of the Operational Research Society, Interfaces, Operations Research Letters, and the Journal of the Production and Operations Management Society and Transportation Science. Moreover, he is actively engaged in several major journals including positions as area editors for both *INTERFACES* and the *Journal of the Production and Operations Management Society* (POMS).

John R. Koza received his Ph.D. in Computer Science from the University of Michigan in 1972 under the supervision of John Holland. He was co-founder, Chairman, and CEO of Scientific Games Inc. from 1973 through 1987 where he co-invented the rub-off instant lottery ticket used by state lotteries. He has taught a course on genetic algorithms and genetic programming at Stanford University since 1988. He is currently a consulting professor in the Biomedical Informatics Program in the Department of Medicine at Stanford University and a consulting professor in the Department of Electrical Engineering at Stanford University.

William Macready is a senior scientist with the Research Institute for Advanced Computer Science at NASA Ames Research Center. William has published on topics including optimization, landscapes, molecular evolution, machine learning, economics, and methods of quantifying complexity. He is interested in both the theoretical and practical aspects of optimization. Before joining RIACS/NASA, William worked in industry solving optimization problems in logistics, supply chains, scheduling, and designed efficient optimization approaches for clearing high-dimensional marketplaces.

Anil Menon received his Ph.D in Computer Science from Syracuse University in 1998. His thesis *Replicators, Majorization and Probabilistic Databases: New Approaches For The Analysis Of Evolutionary Algorithms* was awarded the Syracuse All-University Best Doctoral Thesis Award in 1999. His research interests lie in the areas of evolutionary computation and nonlinear optimization, he has been published in the peer-reviewed *International J. of Neural Networks, IEEE Transactions On Evolutionary Computation,* and the *Foundations of Genetic Algorithms.* He has more than eight years of software development experience in a wide variety of industries including, advanced data access applications, supply chain solutions, and computer aided design and manufacturing, His technical expertise lie in the areas of distributed databases, process optimization, and automated code generation. Till recently, he was a

Distinguished Engineer at Whizbang Research Labs, based in Provo, Utah. He currently consults for ProductSoft, a Pittsburgh based software startup.

Julian Miller is a lecturer in the School of Computer Science, University of Birmingham. He is author of around 75 research publications. He has an international reputation in the research fields of evolvable hardware and genetic programming. He is a regular programme chair and session chair for international conferences on evolvable hardware, and Genetic Programming, and evolutionary computation. He is an associate editor of the Journal of Genetic Programming and Evolvable Machines and IEEE Transactions on Evolutionary Computation. He teaches advanced MSc modules on Quantum and Molecular Computation and a Nature Inspired Design and a first-year undergraduate course on Artificial Intelligence Programming.

Heinz Mühlenbein is research manager of the team 'Adaptive Sytems' at the Institut of Autonomous intelligent Systems of the Fraunhofer Gesellschaft in Germany. He has worked in the areas of time-sharing operating systems, computer networks, parallel computing, and since 1987, soft computing. He is on the editorial board of journals in physics, parallel computing, heuristics, and evolutionary computation. In addition he has published more than 30 refereed journal articles.

Pablo Moscato is currently affiliated with The University of Newcastle, Australia. A native from Argentina, he received a degree in Physics from Universidad Nacional de La Plata in 1987 and a PhD degree from UNICAMP, Brazil. In 1988-89 was a Visiting Graduate Student at the California Institute of Technology and a member of the core research team of the Caltech Concurrent Computation Program where he first introduced the denomination of "memetic algorithms" in the computing literature. He has been Visiting Professor at Universidad del Centro de la Provincia de Buenos Aires, Tandil, Argentina (1995-1996), and UNICAMP, Campinas, Brazil (1996) where he lectured on metaheuristics for combinatorial optimization. He has published in *Journal of Computer and System Sciences, Physics Letters A, Lecture Notes in Computer Science, Annals of Operations Research, Applied Mathematics Letters, Neural Networks World, Chaos, Solitons and Fractals, Production Planning & Control, INFORMS Journal on Computing, Pesquisa Operacional, and European Journal of Operations Research as well as several international conferences. He is a member of the Editorial Board of* The Journal of Mathematical Modelling and Algorithms and a member of the Program Committee of several international conferences. He is author or co-author of twelve chapters of books. He also acted as co-ordinating editor of the section dedicated

to Memetic Algorithms of "New Ideas in Optimization", McGraw-Hill, UK, 1999.

Riccardo Poli is a Professor of Computer Scince at the University of Essex. He has co-authored the book *Foundations of Genetic Programming* and around 130 refereed publications (including 10 conference proceedings) on genetic programming, genetic algorithms, image processing and neural networks and other areas. He is an associate editor of *Evolutionary Computation* and of the *Journal of Genetic Programming and Evolvable Machines.* He has been programme committee member of over 40 international events and has presented invited tutorials at 8 international conferences.

César Rego received his Ph.D. in Computer Science from the University of Versailles and INRIA - France, after earning a MSc in Operations Research and Systems Engineering from the Technical School (IST) of the University of Lisbon. His undergraduate degree in Computer Science and Applied Mathematics is from the Portucalense University in Portugal. Part of his academic career was done in the Portucalense University and he also taught at IST and Faculty of Sciences of the University of Lisbon. Dr. Rego received an award from the Portuguese Operational Research Society (APDIO) for his MSc thesis. He also received the IFORS-Lisbon award for the best international paper published by members of APDIO, an investigation over a four-year period. Finally, he received an award as Researcher/Scholar of the year in School of Business, MIS/POM area, University of Mississippi. Professor Rego's publications have appeared in books on metaheuristics and in leading journals on optimization such as *European Journal of Operational Research* (EJOR), *Journal of Operational Research Society* (JORS), *Parallel Computing,* and *Management Science.* He has developed some of the most efficient algorithms that currently exist for the Traveling Salesman and Vehicle Routing Problems. In the practical realm, he has designed and implemented computer software for solving real-world problems for several major companies. His main research interest is the creation and empirical validation of optimization algorithms for solving complex and practical problems. He is a member the APDIO, the INFORMS, a senior researcher of the Hearin Center for Enterprise Science (HCES) and Associate Professor of Management Information Systems and Operations Management in the School of Business of The University of Mississippi.

Lothar M. Schmitt teaches Mathematics and Computer Science at The University of Aizu (Japan). He holds the Dr.rer.nat. title from the Universitaet des Saarlandes (Saarbruecken) and the Dr.rer.nat.habil. title from Universitaet Osnabrueck and is currently associate professor in Aizu. His work includes

contributions in functional analysis, operator algebra theory, non-commutative integration, quantum physics, biomechanical modeling, genetic algorithms and optimization, language analysis and UNIX-based interactive teaching systems. Otherwise, he enjoys family life, swimming, playing the piano, the arts and fine dining. In 2003, he is listed in "Who's Who in the World."

Chris Stephens is Professor at the Institute for Nuclear Sciences of the UNAM (Universidad Nacional Autonoma de Mexico) - the oldest university in the Americas. He has had visiting positions at various leading academic institutions, including the Weizmann Institute, the Joint Institute for Nuclear Research, Dubna, the University of Birmingham and the University of Essex. He is also a founding partner of Adaptive Technologies Inc. and Adaptive Technologies SA de CV - research companies dedicated to the production of agent-based technologies for dynamical optimization in finance and industry. Chris's research interests are very broad, having published over 70 research articles in a wide array of international journals - ranging from Classical and Quantum Gravity to the Journal of Molecular Evolution.

Matthew J. Streeter received a Masters degree in Computer Science from Worcester Polytechnic Institute in 2001. His Masters thesis applied genetic programming to the automated discovery of numerical approximation formulae for functions and surfaces. His primary research interest is applying genetic programming to problems of real-world scientific or practical importance. He is currently working at Genetic Programming Inc. as a systems programmer and researcher.

This page intentionally left blank

Chapter 1

TOWARDS A THEORY OF ORGANISMS AND EVOLVING AUTOMATA

Open Problems and Ways to Explore

Heinz Mühlenbein
FhG-AiS D-53754 Sankt Augustin
muehlenbein@gmd.de

Abstract We present 14 challenging problems of evolutionary computation, most of them derived from unfinished research work of outstanding scientists such as Charles Darwin, John von Neumann, Anatol Rapaport, Claude Shannon, and Alan Turing. The problems have one common theme: Can we develop a unifying theory or computational model of organisms (natural and artificial) which combines the properties structure, function, development, and evolution? There exist theories for each property separately as well as for some combinations of two. But the combination of all four properties seems necessary for understanding living organisms or evolving automata. We discuss promising approaches which aim in this research direction. We propose stochastic methods as a foundation for a unifying theory.

1. INTRODUCTION

The aim of this book is very ambitious. Its title is not: important problems of evolutionary computation, but *Hilbert problems* in evolutionary computation. What makes Hilbert's problems so famous and unique? Hilbert designed his problems with the goal that "they could serve as examples for the kinds of problems the solutions of which would lead to advancements of disciplines in mathematics." If we have a closer look at Hilbert's twenty-three problems today, then we observe that some of the problems indeed lead to important research, but a few of them did not. One of the reasons seems to be how the problems have been formulated. Most of them are well defined, but some are more vaguely posed, making a solution difficult.

In fact, the paper became famous because of question number two: *Can it be proven that the axioms of arithmetic are consistent?* Hilbert's question is a

sub-problem of the general research program Hilbert had in mind: *Can mathematics be axiomatized?* The general problem was taken on by Russel and Whitehead and lead to three volumes of the *Principia Mathematica.* Gödel dealt with the more specific problem two and proved that the answer is negative. This put an end to the effort of Russel and Whitehead. The implication of Gödel's result with regard to mathematics and the theory of computation in general is still a subject of hot discussions.

In contrast, problem number six just reads: *Can physics be axiomatized?* In the explanation of the question Hilbert writes: "to axiomatize those physical disciplines, in which mathematics already plays a dominant role; these are first and foremost probability and mechanics." To our surprise we see the calculus of probability as a part of physics! A closer inspection reveals that Hilbert's moderate goal was a mathematically sound application of probability to kinetic gas theory. This research has been carried out by physicists, but without ever referring to a Hilbert problem. It lead to statistical physics as it appears today.

My goal is modest. I will propose problems, mainly in evolutionary computation, and name each after a famous scientist who has formulated or investigated the problem. This does not imply that the problem so named is the most important the scientist has worked on. Nor do I claim that the scientist has considered the problem to be the most important one he has worked on. I only want to demonstrate that most of the challenging problems have been identified very early and are with us for quite a time. And my second message is: we have to look much more often into older papers. Older scientific papers should not be considered as "fossils". It is a fundamental misconception that science is continuously accumulating all the important available knowledge and condensing the knowledge in surveys or textbooks. Many important scientific ideas and papers enter main stream science after 20 or more years.

I will consider in the paper both – natural and artificial organisms. The emphasis will be on artificial automata. In order not just to summarize the problems, I will describe in the more technical sections 11 till 13 a theory I consider as a promising candidate for solving some of the problems presented. It is the theory of probability, used and extended in scientific disciplines as different as *probabilistic logic, statistical physics, stochastic dynamical systems and function optimization using search distributions.* These sections will be fairly selfish, because in selecting from the huge available literature the work of my research group will be over-represented.

2. EVOLUTIONARY COMPUTATION AND THEORIES OF EVOLUTION

The goal of evolutionary computation is to make the development of powerful problem solving programs easier. There have been tried at least three approaches to achieve this goal.

1 **Use a theory** - develop a theory of problem solving and implement it on a computer

2 **Copy the brain** - analyze the human brain and make a copy of it on a computer

3 **Copy natural evolution** - analyze natural evolution and implement the most important evolutionary forces on a computer

In the history of artificial intelligence research one of the three approaches was dominant at any one time. Evolutionary computation belongs to the third approach. Today this approach is gaining momentum. It relies on theories of evolution and of computation. The theory of computation is well advanced, so the problems of evolutionary computation lie in theories of evolution. If there existed a convincing constructive theory of evolution, then evolutionary computation would be just a matter of implementation - which of the major evolutionary forces to implement in what detail.

But do we possess a constructive theory of evolution? Here the opinions differ extremely. The main stream theory of evolution is called *New* or *Modern Synthesis.* Its followers claim that it reconciles Darwin's idea of continuous small variations with the concept of gene flows derived from population genetics. The second major force of the Modern Synthesis is still Darwin's concept of *natural selection.* But are these two forces sufficient to explain the wonders of evolution at least in some broad terms?

There is no doubt that the modern synthesis is able to explain the change of gene frequencies on a small time scale. *If there is enough diversification, then the theory correctly predicts further changes for a short time.* But can it explain evolution for a long time? Here the crucial question is: How could it come to such a diversification, starting from a tiny cell? I like to formulate the problem with Darwin's famous ending sentence of *The Origin of Species by Means of Natural Selection* ((Darwin, 1859)).

"There is grandeur in this view of life, with its several powers, having been originally breathed into a few forms or into one; and that, whilst this planet has gone cycling on according to the fixed laws of gravity, from so simple a beginning endless forms most beautiful and most wonderful have been, and are being, evolved."

Let me be more specific and cite some major problems which a theory of evolution would have to explain. Maynard (Smith and Szathmary, 1995), have called them the *the major transitions in evolution* (see table 1.1).

before	→	after
replicator molecules	→	population of molecules in compartments
independent replicator	→	chromosomes
RNA as gene and enzyme	→	DNA and protein
procaryote	→	eucaryote
asexual clones	→	sexual population
protist	→	plants, animals, fungi
solitary individuals	→	colonies
societies of primates	→	human societies

Table 1.1. Major transitions in evolution; Maynard (Smith and Szathmary, 1995)

The authors "solve" some of the problems with a very narrow version of the modern synthesis. "We are supporters of the gene centered approach proposed by Williams and refined by (Dawkins, 1989)." In the gene centered approach, also called the *selfish gene concept,* the genes are the major actors. They possess an internal force to proliferate as much as possible.

This caricature of a theory of evolution is used by the authors to explain the transition from solitary individuals to colonies, for example. The argument is as follows: If a female produces two offspring, but n females can produce 3n offspring, then cooperation between the females pays off. Even if there is a fight between females and one becomes a queen, cooperation is still preferred ($1/n$ of $3n$ is larger than 2). Thus in the gene centered analysis a colony with a single queen has a selective advantage.

There are many flaws in the selfish gene concept. It is not constructive, it does not investigate if the selection advantage of a particular gene can be realized in a phenotype. Rabbits with wings would obviously have a selective advantage. Why did it not happen? Two genes can also oppose each other - gene 1 might increase by action a_1, and gene 2 by the opposite action a_2. Which gene wins? Consider a female and its offspring as an example. The offspring are threatened. Should the mother protect the offspring, even on the risk of her life? The notorious formula of Hamilton gives the result that the mother should sacrifice her life if more than two offspring are threatened (Maynard (Smith and Szathmary, 1995)). Hamilton argues as follows: in each offspring there are only one half of the genes of the mother. Thus the genes of the mother multiply if she protects at least three offspring. Ironically Darwin itself has devoted a whole chapter of his "The Origin of Species" to the problem insect colonies pose to natural selection. His explanation is constructive. He shows how many small changes in behavior can lead to very peculiar behavior, even to slave making ants! This example shows dramatically the extreme simplifi-

cation done by the selfish gene concept. It is my strong opinion that the selfish gene concept does not enrich Darwin's theory, but reduces it to a caricature.

The selfish gene concept has been opposed by a small group in biology, most notably by the late Stephen J. Gould. Recently even philosophers of science formulate a basic critic. I just cite (Griffiths, 2002). "The synthetic theory bypassed what were at the time intractable questions of the actual relationship between stretches of chromosomes and phenotypic traits. Although it was accepted that genes must, in reality, generate phenotypic differences through interaction with other genes and other factors in development, genes were treated as *black boxes* that could be relied on to produce phenotypic variation with which they were known to correlate."

I will discuss this problem later with my proposal of a system theory of evolution. The major conclusion of this section is: *there exists no general theory of evolution today.* The "theory" its proponents call "Modern Synthesis" is an extremely simplified version of Darwin's theory. It separates organisms and environment. Natural selection is modeled by a *fitness function,* whereas Darwin used the term only in a metaphoric sense. In fact, Darwin noticed the misinterpretation of his theory even during his life. He wrote in the last (1872) edition of "The Origin of Species": "As my conclusions have lately been much misrepresented, and it has been stated that I attribute the modification of species exclusively to natural selection, I may be permitted to remark that in the first edition of this work, and subsequently, I placed in a most conspicuous position — namely at the close of the Introduction — the following words: "I am convinced that natural selection has been the main but not the exclusive means of modification." This has been of no avail. Great is the power of steady misinterpretation."

Therefore evolutionary computation has to be largely experimental. This was already pointed out by John (von Neumann, 1954). "Natural organism are, as a rule, much more complicated and subtle, and therefore much less well understood in detail, than are artificial automata. Nevertheless, some regularities, which we observe in the organization of the former may be quite instructive in our thinking and planning of the latter; and conversely, a good deal of our experiences and difficulties with our artificial automata can be to some extend projected on our interpretations of natural organisms."

3. DARWIN'S CONTINENTAL CYCLE CONJECTURE

I will describe my first problem in Darwin's terms. In the chapter "Circumstances favorable to Natural Selection" Darwin writes: "A large number of individuals by giving a better chance for the appearance within any given period of profitable variations, will compensate for a lesser amount of vari-

ability in each individual, and is, I believe, an extremely important element of success."

On the other hand Darwin observes that a large number of individuals in a large continental area will hinder the appearance of new adaptations. This happens more likely in small isolated areas. He writes: "Isolation, also, is an important element in the process of natural selection. In a confined or isolated area, if not large, the organic and inorganic conditions of life will be in a great degree uniform; so that natural selection will tend to modify all individuals of a varying species throughout the area in the same manner in relation to the same conditions. But isolation probably acts more efficiently in checking the immigration of better adapted organisms. Lastly, isolation, by checking immigration and consequently competition, will give time for any new variety to be slowly improved."

Darwin then continues: "Hence an oceanic island at first sight seems to have been highly favorable for the production of new species." But Darwin notes a conflict: "to ascertain whether a small isolated area or a large open area like a continent, has been most favorable for the production of new organic forms, we ought to make the *comparison within equal times*; and this we are incapable of doing. "

Despite of the above observation Darwin concludes: "I conclude, that for terrestrial productions a large continental area, which will probably undergo many oscillations of level, and which consequently will exist for long periods in a broken condition, will be the most favorable for the production of many new forms of life, likely to endure long and spread widely." Darwin reasons as follows: "For the area will first have existed as a continent, and the inhabitants, at this period numerous in individuals and kinds, will have been subjected to very severe competition. When converted by subsidence into large separate islands, there will still exist many individuals of the same species on each island;... and time will be allowed for the varieties in each to become well modified and perfected. When by renewed elevation, the islands shall be re-converted into a continental area, there will be again severe competition: the most favored or improved varieties will be enabled to spread: there will be much extinction of the less improved forms . . ."

I am very impressed about Darwin's continental cycle conjecture, which he made much earlier than Alfred Wegener in geology. Therefore I dedicate my first problem to Darwin.

Problem 1 [Darwin]: *Can we demonstrate or even prove the correctness of Darwin's Continent-Island cycle conjecture ?*

The reader should have observed how carefully Darwin discusses the arguments. I strongly recommend to read Darwin's "The Origin of Species".

The most profound critique of modern "Darwinism" can be found in Darwin's book![1]

It seems difficult to test Darwin's conjecture in nature. I propose therefore to use simulations as a first step. I have used the iterated prisoners dilemma game to investigate problem 1 ((Mühlenbein, 1991a)). The results indicate that Darwin's conjecture might be correct. But the simulation model needs a lot more refinement.

Darwin mentions at many places of the "Origin" that space is as important for evolution as time. This has been shown in the context of genetic algorithms by (Mühlenbein, 1991b). Space is also an important element of the *shifting balance theory* of evolution proposed by (Wright, 1937). Without referring to Darwin a subset of the problem, that is the difference of the evolution in a large continent and small isolated islands, has been recently investigated by (Parisi and Ugolini, 2002).

4. THE SYSTEM VIEW OF EVOLUTION

The next set of problems I will derive more abstract. The major weakness of "Darwinism" in the form of the modern synthesis is the separation of the individuals and the environment. In this model each individual O_i (mainly characterized by its genes) is assigned a fitness f predicting the performance of this individual within the environment E and given the other individuals. This can be written as:

$$\begin{aligned} O_i(t+1) &= f(\mathbf{O}, E(t)) \\ E(t+1) &= g(E(t) \end{aligned}$$

It seems impossible to obtain numerical values for the fitness. Therefore theoretical biology has made many simplifications. The environment is kept fixed, i.e $g(E(t)) = const$, the influence of other individuals is described by some averages of the population, etc.. The shortcomings of the dichotomy individual-environment in the Modern Synthesis have already been discussed. The problem is still more difficult because each individual is in addition *developing in a close interaction with its environment.* The development problem has been addressed recently by (Oyama, 2000), in her *developmental system theory.* Unfortunately the theory is very informal, it has been formulated from a philosopher's point of view. Therefore I will describe the next problem as it has been stated in the final address of Anatol Rapaport, the then retiring president of General System Science Society
((Rapaport, 1970)).

[1]In addition I recommend the essays of Stephen J. Gould.

Problem 2 [Rapaport+1]: *Can we formulate a theory of organisms, which incorporates being, acting, evolving, and developing?*

I have named the problem Rapaport+1 because Rapaport identified only three properties. He combined evolving and developing into a single property becoming. The problem needs an explanation. It goes back to (Whitehead, 1948). In his book "Science and the Modern World" Whitehead warned that the store of fundamental ideas on which the then contemporary science was based was becoming depleted. Whitehead suggested that the concept of *organism,* hitherto neglected in physical science, might be a source of new ideas. Whitehead tried to define what an organism characterizes.

We will describe the definition of Rapaport. "According to a soft definition, a system is a portion of the world that is perceived as a unit and that is able to maintain its identity in spite of changes going on in it. An example of a system par excellence is a living organism. But a city, a nation, a business firm, a university are organisms of a sort. These systems are too complex to be described in terms of succession of states or by mathematical methods. Nevertheless they can be subjected to methodological investigations."

Rapaport then defines: "Three fundamental properties of an organism appear in all organism-like systems. Each has a *structure.* That is, it consists of inter-related parts. It maintains a short-term steady state. That is to say, it reacts to changes in the environment in whatever way is required to maintain its integrity. It *functions.* It undergoes slow, long term changes. It grows, develops, or evolves. Or it degenerates, disintegrates, dies.

Organisms, ecological systems, nations, institutions, all have these three attributes: *structure, function, and history,* or, if you will, *being, acting, and becoming.*"

Rapaport's becoming captures both – the development of an organism from the fertilized egg to the grown-up organism, and the evolution of the species in a succession of many generations. There is no doubt that the relationship between the two properties is a very close one. Ernst Haeckel even postulated in 1890 a *biogenetic law: Individual development is a shortened recapitulation of the history of the phylum.* Subsequent research has shown that there is some truth in the law, but as a general statement it is incorrect. In my opinion it is very important to distinguish between the development of an individual and the evolution of a species.

To my knowledge, Rapaport's talk did not lead to a scientific effort to build such a theory of organisms. The reader will guess the reason: it is the sheer complexity of the task! Instead research in biology remained concentrated on a single property or to a combination of two properties. Thus population genetics combines being and evolving, population dynamics combines being and

acting. The developmental system theory mentioned earlier combines being and developing.

The investigation of the above problem leads to another problem: In what language should we frame a theory of organisms? Three approaches can be tried:

- The descriptive approach, using natural language
- The micro-simulation approach
- The mathematical approach

Today the descriptive approach has gained momentum, characterized by the developmental system theory mentioned above (Oyama, 2000). Artificial Life uses micro-simulation. But in micro-simulations it is very difficult to distinguish between the microscopic event and the more general pattern happening in many simulations. Rapaport and, earlier, von Neumann advocated the mathematical approach. I go a step further and propose stochastic system theory as the research foundation. Stochastic analysis has been successfully used in population genetics for at least 75 years. But population dynamics is still mainly investigated with the help of deterministic differential equations. Thus I partition Rapaport's problem into three problems.

Problem 3a: *Can we develop a stochastic system theory, combining the properties being and acting of organisms or automata in a 2-d space?*

Problem 3b: *Can we develop a stochastic system theory, combining the properties being and developing of organisms or automata in a 2-d space?*

Problem 3c: *Can we develop a stochastic system theory, combining the aspects being, acting and evolving of organisms or automata in a 2-d space?*

The answer to the first question is a definite yes. It is already an active area of research. We will discuss the state of the art in stochastic analysis in the technical sections 11 till 13. Problem 3b was first investigated by von Neumann.

5. VON NEUMANN'S SELF-REPRODUCING AUTOMATA

Von Neumann started his research with the concept of *"complification"*. He used the term very informally. We will proceed in the same way. It is outside the scope of this paper to discuss all the measures proposed for complexity. Also the term automaton will be used in a broad manner. Von Neumann observed: "If automaton A can produce B, then A in some way must have contained a complete description of B. In this sense some decrease in complexity

must be expected as one automaton makes another automaton." But organisms reproduce themselves with no decrease in complexity. Moreover, organisms are indirectly derived from others which had lower complexity.

Problem 4 [von Neumann]: *Can we construct automata which are able to produce automata more complex than themselves?*

Von Neumann tried several approaches to enable a scientific investigation of the above problem. The main theory was collected by Burns and expended into a theory of *self-reproducing automata* ((Burns, 1970)). But it is more instructive to look at von Neumann's own description, summarized in the article "The General and Logical Theory of Automata" ((von Neumann, 1954)). Von Neumann started his research with a result of Turing. Turing wanted to give a precise definition of what is meant by a computing automaton. His solution was the *Universal Turing Machine* UTM. It consists of an automaton reading and writing symbols on an infinite tape. Von Neumann decided that his automaton should have the power to simulate the UTM in a discrete cellular 2-d space. Thus he investigated the problem how to construct an automaton which reproduces itself in 2-d space and has the power of UTM.

Von Neumann's construction proceeded as follows:

(a) *Construct an automaton A, which when furnished the description of any other automaton in terms of appropriate functions, will construct that entity.*

(b) *Construct an automaton B, which can make a copy of any instruction* α *that is furnished to it. This facility will be used when* α *furnishes a description of another automaton.*

(c) *Combine the automata A and B with a control mechanism* γ*, which does the following.* γ *will first cause A to construct the automaton which is described by* α*. Next* γ *will cause B to copy the instruction* α*. Finally* γ *will separate this construction from the system* $A + B + \gamma = D$

(d) *Form an instruction* α_D*, which describes this automaton D, and insert* α_D *into A within D. Call the aggregate which now results E.*

E is clearly self-reproducing. But E cannot do anything besides reproduction. It needs a program. Therefore von Neumann proposed an extension: Replace the instruction α_D by an instruction α_{D+F} which describes automaton D plus another automaton F. This automaton reproduces itself and then behaves like automaton F. Now if a "mutation" within the F part takes place, it changes E_F into $E_{F'}$. This "mutant" is still self-reproductive.

Von Neumann believed that with this construction he had made crude steps in the direction of a systematic theory of automata, especially towards forming a rigorous concept of what constitutes "complication." At a first glance, the construction seems to be the solution of the automatic programming problem. But why did von Neumann's self-reproducing automata not have any practical

relevance? The answer is simple: The construction does not solve the most important problem: How do the programs get into the machine? The *development of programs is the problem, not their self-reproduction.* Von Neumann's automata can in principle compute anything, but the programs have to be provided from the outside! Who provides these descriptions? A single built-in program F is surely not enough, because von Neumann did not introduce selection. Therefore the value of the mutant program *F'* for problem solving is not checked. Thus von Neumann solved only part of the problem. Therefore we extend problem 4.

Problem 5: *What conditions are required to enable von Neumann's automata to grow in complexity without external interventions?*
A worthwhile extension of von Neumann's approach would be to use a population of automata which interact with each other and which have to solve a set of problems to survive and produce offspring. Thus I believe that for a solution of problem 5 one needs both, Turing and Darwin. Turing provides the concept of a universal automaton and Darwin provides the concept of a changing environment metaphorically leading to natural selection.

The importance of von Neumann's construction for today's research has also been emphasized by (McMullin, 2001).

6. TURING'S INTELLIGENT MACHINE

Von Neumann's approach of using self-reproduction and the Universal Turing Machine was not the only method proposed to build intelligent machines. In fact, von Neumann discussed the use of artificial neural networks as another possibility. Before I describe this work, it is instructive to discuss how Turing himself approached the problem in his article "Computing machinery and intelligence" ((Turing, 1950)). At first Turing defined the concept of intelligence. A machine is intelligent if it passes a test Turing defined precisely: the *Turing test* is an "imitation" game, played by three objects A, B and C. C is the interrogator, A or B might be a machine. The machine passes the test if the interrogator is not able to find out that a machine answers to his questions. This gives our next problem.

Problem 6 [Turing]: *Is it possible to build machines which pass the Turing test?*

Turing believed that the answer to the above question is positive and proposed a method to construct such a machine. It is described in the section "Learning Machines" of the above cited paper. Turing's proposal seems to be almost unknown, although it is contained in this well-known article. I find

the proposal very fascinating. The arguments brought forward by Turing have been used a number of times in artificial intelligence research, but obviously without knowing that Turing already formulated them.

"As I have explained, the problem is mainly one of programming. Estimates of the storage capacity of the brain vary from 10^{10} and 10^{15}. I would be surprised if more than 10^{9} was required to satisfactory playing of the imitation game ... At my present rate I produce about a thousand digits of program a day, so that about sixty workers, working steadily through the fifty years might accomplish the job, if nothing went into the wastepaper basket. Some more expeditious method seems desirable."

Turing did not try to formalize a possible solution to problem 6. Any program passing the test will do. It is the *efficiency problem* which leads Turing to consider natural organisms, in this case the human mind. "In the process of trying to imitate an adult mind we are bound to think a good deal about the process which has brought it to the state that it is in. We may notice three components:

(a) The initial state of the mind, say at birth,
(b) The education to which it has been subjected,
(c) Other experience

Instead of trying to produce a program to simulate the adult mind, why not rather try to produce one which simulates the child's?.... We have thus divided our problem into two parts, the child programme and the education process. These two remain very closely connected. We cannot expect to find a good child machine at the first attempt... There is an obvious connection between this process and evolution, by the identifications

Structure of the child machine	=	hereditary material
Changes of the child machine	=	mutations
Natural selection	=	judgment of the experimenter

One may hope, however that this process will be more expeditious than evolution. The survival of the fittest is a slow method for measuring advantages... Opinions may vary as to the complexity which is suitable in the child machine. One might try to make it as simple as possible consistently with the general principles. Alternatively one might have a complete system of logical inference programmed in."

Turing reported: "*I have done some experiments with one such child machine, but the teaching method was too unorthodox for the experiment to be considered really successful.*" The imitation game is the final test, one needs some intermediate goals. "We may hope that machines will eventually compete with men in all purely intellectual fields. But which are the best ones to

start with?... Many people think that a very abstract activity, like the playing of chess, would be the best. It can also be maintained that it is best to provide the machine with the best sense organs that money can buy, and then teach it to understand and speak English." Today chess playing has been solved by brute force programming. This solution is feasible due to the strict rules of chess that enable fast and efficient game tree search. The progress in games like *GO* is much slower. But we are still left with the language understanding problem.

Problem 7 [Turing]: *Is it possible to create a machine which can be taught to understand English?*

Turing's proposal belongs to the "copy the evolution" approach. In contrast to nature the artificial evolution does not start with a cell, but with a well-designed child. Turing's approach is very informal, he believed that he could program an intelligent system using about 10^9 bits. I call this attitude the *programmer's approach.* The system is programmed without a theory. One just assumes that anything can be programmed. This attitude seems to be dominant today. For Turing evolution is just a technique to shorten the programming time.

7. WHAT CAN BE COMPUTED BY AN ARTIFICIAL NEURAL NETWORK?

We now turn back to von Neumann and his approach to machine intelligence. In contrast to Turing, von Neumann works more like a natural scientist. He tries to formalize solution strategies. Thus his solutions are not programs, but theories. In 1948 formal neural networks were already very popular in the research community because of the work of McCulloch and Pitts. John (von Neumann, 1954), investigated the power of neural networks in his famous talk "The general and logical theory of automata". In the section "Formal Neural Networks" von Neumann notes: "The McCulloch-Pitts result[2] proves that anything that can be exhaustively and unambiguously described, anything that can be completely and unambiguously put into words, is ipso facto realizable by a suitable finite neural network... Thus the remaining problems are these two. First, if ascertain modes of behavior can be effected by a finite neural network, the question still remains whether *the network can be realized within a practical size...* Second, the question arises whether every existing modes of behavior can be put completely and unambiguously into words...

There is no doubt that any special phase of any conceivable form of behavior can be described completely and unambiguously in words. This description

[2]McCulloch-Pitts had proven that their formal neural networks are equivalent to a Turing machine.

may be lengthy, but it is always possible... It is, however, an important limitation, that this applies only to every element separately, and it is far from clear how it will apply to the entire syndrome of behavior."

Von Neumann then discusses more specifically the concept of identification of analogous geometrical entities. He takes as example the concept of a triangle.

"There is no difficulty in describing how an organism might be able to identify any two rectilinear triangles, which appear on the retina, as belonging to the category "triangle". There is also no difficulty in adding to this, that numerous other objects, will also be classified and identified as triangles — triangles whose sides are curved, triangles whose sides are not full drawn ... This, in turn, however constitutes only a small fragment of the more general concept of *analogy.* Nobody would attempt to describe and define within any practical amount of space the general concept of analogy which dominates our interpretation of vision. There is no basis for saying whether such an enterprise would require thousands or millions or altogether impractical numbers of volumes. Now it is perfectly possible that the simplest and only practical way actually to say what constitutes a visual analogy consist in *giving a description of the connections of the visual brain.*"

This discussion leads to the next problem.

Problem 8 [von Neumann]: *Can an artificial neural network of practical size be designed which gives similar results on visual problems as the human brain?*

Turing also used an "analysis" of the human brain in order to show that an intelligent machine can be programmed in 10^9 bits. He wrongly assumed that the performance of the brain can be characterized by its number of neurons, about 10^9. He did not consider the interconnection structure as relevant. The only problem left to him is to obtain this program of 10^9 digits. Von Neumann is much more careful. It is not the number of neurons which matters, but their interconnection structure. Today we know that even the interconnection structure is not sufficient to define uniquely how the neurons process the visual input. We need to know the *dynamic interaction of all the neurons involved.*

8. LIMITS OF COMPUTING AND COMMON SENSE

I consider von Neumann's discussion about computability extremely important. To summarize his research on problem 8: First, von Neumann has serious doubts that the concept of visual analogy can be formulated in a finite number of words. Second, even if it can be formulated in a finite number of words, it

might be that this number is greater than a practical size. The practical size is defined by the real world. Finiteness is not enough, we need practical time and practical space in our real world. The finiteness of our world puts an upper limit to the largest program which can be computed in our world. The actual determination of the practical size turned out to be very difficult. Therefore *complexity theory* is formulated without an actual limit. The set of problems which can be computed is defined differently. It lead to the distinction of *Ptime* and *NPtime* problems: given a problem whose solution can be verified in polynomial time, is there an algorithm which actually finds such a solution (this means in polynomial time according to the size of the input.)? If both conditions can be proven, we have a problem from class Ptime or short **P**, if only the first condition is fulfilled we have an NPtime or short **NP** problem. Polynomial time means $T \approx O(n^k)$, exponential time $T \approx a^{f(n)}$. But if n is very large, even $O(n^k)$ can be a very large number, meaning that the problem cannot be computed in reasonable time. Nevertheless all problems in class **P** are considered to be easy computable.

The question if **P** is equal or not equal to **NP** is one of the most important open questions in complexity theory. The basic classification has been refined in a number of ways. I just mention the inclusion

$$\text{LOGTIME} \subseteq \text{LOGSPACE} \subseteq \text{PTIME} \subseteq \text{NPTIME} \subseteq \text{PSPACE} \subseteq \text{EXPTIME}$$

There have been several attempts to compute the practical size of space or the limit of time, using the finiteness of the universe and the laws of physics. Bremermann was one of the first to compute explicitly an upper limit.

Bremermann's bound: *No data processing system, whether artificial or living, can process more than* $2 * 10^{47}$ *bits per second per gram of its mass.*

Bremerman used this bound to calculate the total number of bits processed by a hypothetical computer the size of the earth within a time period equal to the estimated age of the earth. He computed 10^{93} bits. Then he calculated the mass of the universe and obtained his bound. The above limit is small by mathematical means. I call any number between 10^{47} and 10^{93} *Bremermann's limit* ((Mtihlenbein, 1996)). Programs which are finite, but require more than 10^{93} steps for solving, do not finish in our universe. This implies that the mathematical class of finite programs has to be divided into those below Bremermann's limit and above the limit.

Von Neumann had serious doubts that complex behaviors like the concept of visual analogy can be described by a reasonable number of words, meaning that the description can be read and processed in a lifetime. Despite the warning issued by von Neumann there have been many attempts to put so much knowledge into a machine that it could behave intelligently. The earliest pro-

posal was made by (McCarthy, 1959), in his article "Programs with common sense"[3]. The most recent effort is due to (Lenat, 1995). With a team of up to 10 people he tried to code "common sense" knowledge into a rule-based database. After almost 10 years of effort, he was still far away from the goal, formulated as the next problem.

Problem 9 [McCarthy,Lenat]: *Is it possible to put so much knowledge into a computer, that it is able to read a newspaper and improve itselffrom thereon ?*

Looking back to von Neumann's discussion, I believe that the answer to this question is negative. I do not recommend to work on this problem, because proving that something is impossible is very difficult. Instead I recommend a sub-problem, formulated in the paper "Computers and Automata" by (Shannon, 1953).

Problem 10 [Shannon]: *Can we organize machines into a hierarchy of levels, as the brain appears to be organized, with the learning of the machine gradually progressing up through the hierarchy?*

Hierarchy is used by Shannon very informally. He means levels of abstractions. Each level might use a different calculus. The machine should be able to do inference on a lower level after a limited number of examples. This feature should then be used for learning at the next level. Up to now there are no convincing theories how to solve this problem.

9. A LOGICAL THEORY OF ADAPTIVE SYSTEMS

In the paper "Outline for a Logical Theory of Adaptive Systems" (Holland, 1970b), tried to continue the scientific endeavor initiated by von Neumann. Holland wrote: "The theory should enable to *formulate key hypotheses and problems particularly from molecular control and neurophysiology. The work in theoretical genetics should find a natural place in the theory. At the same time, rigorous methods of automata theory, particularly those parts concerned with growing automata should be used.*"

Thus Holland's proposal is a very early attempt to work on the general problem 2, a constructive theory of the evolution of automata. It tries to combine *being, acting, developing, and evolving.* This proposal so important that I will describe it in detail. Holland's emphasis (like von Neumann's) is foremost

[3]The discussion of the talk started with a remark of Bar-Hillel: "Dr. McCarthy's paper belongs in the Journal of Half-Baked Ideas, the creation of which was recently proposed by Dr. I.J. Good."

on theories and systems, he does not claim to solve grand challenge applications with the proposed methods. This can be tried after the theories have been developed.

"Unrestricted adaptability (assuming nothing is known of the environment) requires that the adaptive system be able initially to *generate any of the programs of some universal computer*... With each generation procedure we associate the population of programs it generates;... In the same vein we can treat the environment as a population of problems." It is especially the last sentence which relates Holland's ideas to Darwin's. Now let us have a closer look at Holland's model. First, there is a finite set of generators (programs) $(g_1, \ldots, g_k)$. The generation procedure is defined in terms of this set and a *graph* called a *generation tree.* Each permissible combination of generators is represented by a vertex in the generation tree. Holland now distinguishes between auxiliary vertices and main vertices. Each auxiliary vertex will be labeled with two numbers, called the *connections* and *disconnection probabilities.* This technique enables to create new connections or to delete existing connections. Each main vertex is labeled with a variable referred to as *density.* The interested reader is urged to read the original paper ((Holland, 1970b)).

Holland claims that from the generation tree and the transition equations of any particular generation procedure, one can *calculate the expected values of the densities of the main vertices as a function of time.* Holland writes: "From the general form of the transition equations one can determine such things as conditions under which the resulting generation procedures are *stationary processes.*" Thus Holland already tried to formulate a stochastic theory of program generation! This is an idea still waiting to be explored.

Holland's next extension of the system is similar in spirit to von Neumann's self-reproducing automata. Holland introduces *supervisory programs* which can construct *templates* which alter the probabilities of connections. Templates play the role of catalysts or enzymes. Thus program construction is also influenced by some kind of "chemical reactions." The above process is not yet adaptive. Adaptation needs an environment posing problems. Therefore Holland proposes that the *environment is treated as a population of problems.* These problems are presented by means of a finite set of initial statements and an algorithm for checking whether a purported solution of the problem is in fact a solution. "When we consider the interaction of an adaptive system with its environment we come very soon to questions of partial solutions, subgoals etc. The simplest cases occur when there is an a priori estimate of the nature of the partial solution and a measure of the closeness of its approach to the final solution."

Holland then observes that a *rich environment* is crucial for the adaptation. "Mathematical characterization of classes of rich environments relative to a given class of adaptive systems constitutes one of the major questions in the

study of adaptive systems. … An adaptive system could enhance its rate of adaptation by somehow enriching the environment. Such enrichment occurs if the adaptive system can generate subproblems or subgoals whose solution will contribute to the solution of the given problems of the environment."

It is very interesting to note that Holland distinguished three kinds of programs – supervisory programs, templates, and programs for the problem solution. The supervisory programs use a probabilistic generation tree to generate programs, the templates are used as catalyst to "skew" the generation process. Holland perceived a hierarchy of programs ((Holland, 1970a)):

1 productive systems – the generator system is able to produce other generators

2 autocatalytic systems – the generator system produces generators which are used in the construction

3 self-duplicating systems – the generator system produces duplicates of itself

4 general adaptive systems – has still to be defined

"The beginning of such a definition (of adaptive systems) lies in the following consideration: with the help of concepts such as autocatalytic and self-duplicating generator systems it is possible to define such concepts as steady-state equilibria and homeostasis for embedded automata… If the generator system for such an automaton has a hierarchical structure, then a small change in structure produces a small change in proportion to the "position" of the change in the hierarchy… By making changes first at the highest level and then at progressively lower levels of the hierarchy, it should be possible to narrow down rather quickly to any automaton in this category having some initially prescribed behavior."

I believe that Holland's very first proposal is a very good starting point for future research. It puts forward many ideas not yet contained in current research. Holland's proposal to use stochastic systems, their steady-state equilibria and homeostasis is in my opinion still a very promising approach for a constructive evolution theory of automata. Holland itself never implemented his general model. It remained a theoretical design. Therefore the next problem is still open.

Problem 11 [Holland]: *Try to implement Holland's model and prove its usability by a convincing application.*

After working about eight years on this theory Holland turned to a simpler evolution model, in fact the Modern Synthesis mentioned before. The environment is hidden in a *fitness function.* Evolution reduces then to an optimization

problem. This research lead to *genetic algorithms.* Holland believed that his genetic algorithms have an almost optimal adaptation rate taking into account the information which is available ((Holland, 1973; Holland, John H., 1975)). But we will prove in Section 13 that it is our Boltzmann distribution algorithm which fulfills his criterion for optimality!

Nobel laureate Gell-Man criticized at the Santa Fe institute that genetic algorithms are unsuited to investigate self-organized evolution, because they use a simple fitness function for a genotype. Therefore (Holland, John H., 1975), later developed *Echo.* Unfortunately *Echo* lacks the theoretical foundation of Holland's first proposal. Therefore I will not discuss it in this paper.

10. THE λ-CALCULUS FOR CREATING ARTIFICIAL INTELLIGENCE

In another chain of reasoning we might ask ourselves: Maybe there is a way of creating human like intelligence without copying nature too much. Instead of starting with the Universal Turing Machine, we can start with the calculus developed by Church and later called the **λ-calculus.** It was implemented as part of the LISP language by John McCarthy. The **λ-calculus** has the same computational power as the Turing machine, but it is based on substitution. LISP is an interpretative language, thus the LISP environment can be seen as a very complex self-reproducing automaton. For the next problem I recommend to read Minsky's survey "Steps toward artificial intelligence" ((Minsky, 1961)). I only cite: "It is my conviction that no scheme for learning, or for pattern recognition, can have very general utility unless there are provisions for recursive, or at least hierarchical, use of previous results. We cannot expect a learning system to come to handle very hard problems without preparing it with a reasonable graded sequence of problems of growing difficulty. The first problem must be one which can be solved in reasonable time with the initial resources. The next must be capable of solution in reasonable time by using reasonably simple and accessible combinations of methods developed in the first, and so on."

In my opinion we have even to go a step further. There seems to be no big gain if the set of problems is hand crafted by a human. The program itself should create some of the sub-problems. We now have to formulate a task for this model. I rephrase a question from (Shannon, 1953):

Problem 12 [Shannon]: *Can we program a digital computer so that eventually 99 percent of the orders it follows are written by the computer itself and which solves difficult problems (e.g performs comparable to the human eye or understands the English language?)*

I added the two applications in brackets, because Shannon forgot in his question to specify the applications to be solved. But without an application the above problem can easily be solved by a program which randomly generates instructions.

LISP was the first language used by Koza for *Genetic Programming.* But within the framework of our discussion, Koza's model is too restricted. It works only for one problem at a time. For each problem we need examples describing the input-output relations of the problem to be solved. The population of solutions is changed according to the mechanisms used by genetic algorithms.

11. PROBABILISTIC LOGIC

All problems up to now have been formulated in the very early days of electronic computers. For the early researcher a possible solution of these problems was either a theory or a successful application in pattern recognition or language understanding. Furthermore, in order to develop and understand the model, either *classical mathematics* or *abstract automata* defined by a flexible language have been used. Several times *stochastic systems* have been proposed for the mathematical analysis. Von Neumann explicitly expressed the feeling, having in mind artificial automata as model organisms, that a new theory is urgently needed ((von Neumann, 1954)): "This new system of formal logic will move closer to another discipline which has been little linked in the past with logic. This is *thermodynamics,* primarily in the form it was received from Boltzmann, and is that part of theoretical physics which comes nearest in some of its aspects to manipulating and measuring information. Its techniques are much more analytical than combinatorial."

Von Neumann's prediction has become true. Probability has been extended to *probabilistic logic.* But first we will describe two early attempts in this area.

11.1 VON NEUMANN'S PROBABILISTIC LOGICS

To my knowledge von Neumann was the first to use the term probabilistic logic in his paper "Probabilistic Logics and the Synthesis of Reliable Organisms from Unreliable Components" (von Neumann, 1956). I shortly describe his model.

"With every basic organ is associated a number ϵ such that in any operation the organ will fail to function correctly ... Suppose the organ receives a stimulation at time t and no later ones. Let the probability that the organ is still excited after s cycles be denoted by p_s. Then the recursion formula

$$p_{s+1} = (1 - \epsilon)p_s + \epsilon(1 - p_s)$$

is valid." It is easy to show that the equation has the solution

$$p_{s+1} = (1 - 2\epsilon)^{s+1}(p_0 - \frac{1}{2}) + \frac{1}{2}$$

Therefore von Neumann concludes that $p_s \rightarrow 0.5$ for $s \rightarrow \infty$, meaning in von Neumann's opinion that the component functions randomly. But let us now investigate the problem in a precisely defined automaton setting. The automaton has two states {0,1}. At each step the automaton changes with probability ϵ from the given state to the opposite state. If we observe the automaton, we would see that the automaton changes states only after ϵ^{-1} steps on the average. Such a behavior is very different from that of a random automaton, which changes states at each step with probability 0.5. But both automata have a *limit distribution* with $p_s \approx 0.5$. The difference between the distributions becomes apparent if joint distributions like $p(x_s; x_{s-1})$ are considered, where x_s denotes the state of the automaton at step s.

Von Neumann's analysis did not capture the reliability problem. Therefore his "solution" to the problem of unreliable components did not have any practical value. Von Neumann approached probabilistic logic from the most difficult point of view, namely the stochastic view. This means to define logic with time dependent dynamics! It is much easier to define probabilistic logic from the logic point of view, without time and dynamics. This is discussed in the next sections.

11.2 THE CONDITIONAL PROBABILITY COMPUTER

The importance of conditional probabilities for the *classification of objects* given a vector of features was first recognized by (Uttley, 1959). In its simplest form Uttley's *conditional probability computer* consists of n binary input units $\mathbf{x} = (x_1, \ldots, x_n)$ and m output units $\mathbf{y} = (y_1, \ldots, y_n)$.

Definition 1 *Let* $0 \leq p(\mathbf{x}) \leq 1$ *denote the probability of* $\mathbf{x}$. *Then* $p_i(x_k, t) = \sum_{\mathbf{x}, X_i = x_k} p(\mathbf{x}, t)$ *defines the univariate marginal distributions of variable* X_i. *Let* $\mathbf{x}_\xi$ *be a sub-vector of* $\mathbf{x}$. *Then the marginal distribution is defined as* $p(\mathbf{x}_\xi, t) = \sum_{\mathbf{x}, X_\xi = x_\xi} p(\mathbf{x}, t)$ *Let* $\mathbf{y}$, $\mathbf{z}$ *be disjoint sub-vectors of* $\mathbf{x}$. *Then conditional probabilities are defined as* $p(\mathbf{y}|\mathbf{z}) = p(\mathbf{y}, \mathbf{z})/p(\mathbf{z})$ *for* $p(\mathbf{z}) > 0$.

Given an input vector of features $\mathbf{x}$, the conditional computer looks for

$$\max_{j=1,m} p(y_j|\mathbf{x})$$

For the computation of the maximum, Uttley proposed to compute *all possible* conditional probabilities $p(y_j|\mathbf{x})$ for a learning set. As Uttley observed,

a conditional probability computer would allow to compute all logical inferences, if we identify "from $\mathbf{y}$ follows $\mathbf{z}$" by the condition $p(\mathbf{z}|\mathbf{y}) > 0.5$. The drawback of this proposal is that it needs $2^n - 1$ units.Thus the computation is exponential in time and space.

There have been several attempts to use less units and also to deal with *incomplete input.* Most notably are the early efforts of Minsky and Selfridge, and independently by Papert (both papers have been published in (Cherry, 1961)). In both papers the assumption is made that all x_i's are independent. This is very unrealistic. It needed a long time to solve the computation problem and the incomplete input problem.

11.3 MODERN PROBABILISTIC LOGIC

Modern probabilistic logics can be seen as a candidate for von Neumann's new system of formal logic. It connects probability theory with logic by assigning probabilities to clauses.

Definition 2 *A probabilistic statement that* $\mathbf{z}$ *is true given* $\mathbf{y}$ *is a conditional probability with "truth" value* $0 \leq r \leq 1$

$$p(\mathbf{z}|\mathbf{y}) = r$$

Thus probabilistic logic is just probability theory with a different interpretation. Let n be the number of binary concepts. In addition let a number of clauses be specified. The specifications are called the *constraints.*

For any specification we have a set of probability models (P-models) which can either be empty (i.e the constraints violate the laws of probability), contain a single P-model, or contain a number of P-models (the specification is *incomplete.*) If the P-model is unique, we can compute the probability of an arbitrary prepositional sentence by summing up probabilities. The probability of a conditional statement $p(\mathbf{z}|\mathbf{y})$ can be obtained by dividing the probability $p(\mathbf{z}, \mathbf{y})$ by the probability $p(\mathbf{y})$.

But unique P-models are unrealistic. The specification has to set all of the $2^n - 1$ variables defining the distribution. Consequently, for incomplete specifications the missing information must be added by some automatic completion procedure. This is achieved by the *maximum entropy principle.* The entropy of a distribution is defined by

$$H(p) = -\sum_x p(\mathbf{x}) \ln(p(\mathbf{x})) \tag{1.1}$$

The maximum entropy principle formulates the *principle of indifference.* If no constraints are specified, the uniform random distribution is assumed. The principle has been first proposed by (Jaynes, 1957).

Maximum entropy principle: *Find the maximal entropy distribution for* $p(\mathbf{x})$ *which satisfies the given marginals.*

This principle has a long history in physics and probabilistic logic. The interested reader is referred to (Jaynes, 1957). The following theorem has been proven by ((Cover and Thomas, 1989)).

Theorem 3 *If the given constraints are consistent, then there exists a unique distribution* $q(\mathbf{x})$ *of maximum entropy.*

Consistent means that the marginal distributions derived from the constraints fulfill all the constraints which can be derived from the laws of probability theory. The most popular algorithm to compute the maximum entropy distribution is called *iterative proportional fitting.* To give the reader a flavor of the theory we present a simple example.

Example: Given the three expressions 'having a full-time job' ft, 'working in a technical domain' t and 'male' m, the following information is specified

$$P(t|ft) = 0.55$$
$$P(t|m) = 0.55$$
$$P(t|ft \cup m) = 0.45$$

Then the maximum entropy solution gives, for instance $P(t|ft \cap m) \approx 0.84$ (see [4]).

The maximum entropy principle solves the incomplete data problem. But iterative proportional fitting scales exponentially in the number of variables. Thus a simpler technique has to be found. Such a method has recently been discovered. It uses the principle of *conditional independence.* Its graphical representation is called a *graphical model.* For our discussion the following definition is sufficient.

Definition 4 *A graphical model is a graph G, where two variables are connected by an edge if they appear together in one constraint.*

The new method tries to find a factorization of the distribution. There is lots of literature available how this can be done, we just mention (Lauritzen, 1996). The algorithm computes *cliques* and generates a *junction tree J.* A junction tree is an undirected tree the nodes of which are clusters of variables. The clusters satisfy the *junction property*: For any two clusters a and b and any cluster h on the unique path between a and b in the junction tree the relation

$$a \cap b \subseteq h \tag{1.2}$$

[4] Whether this very precise value is justified by logical arguments is still a subject of hot discussions.

is true. The edges between the clusters are labeled with the intersection of the adjacent clusters; we call these labels *separating sets* or *separators* s. Then the probability can be factored into

$$p(\mathbf{x}) = \prod_{a,s} \frac{p(\mathbf{x}_a)}{p(\mathbf{x}_s)} \tag{1.3}$$

The modified iterative proportional fitting algorithm uses only the computed clusters of the factorization as marginals. This algorithm produces exactly the same result as the standard iterative proportional fitting. If all factors of the factorization have a number of variables which is independent of the global number n, then the algorithm is polynomial.

The crucial question remains: Which graphical models lead to bounded factorizations? We give here just one negative result ((Mühlenbein and Mahnig, 2003)):

Theorem 5 *Graphical model models which are 2-D grids lead to factorizations which have at least one factor with $\sqrt{n}$ variables. Thus for these problems the computational amount to compute the maximum entropy distribution is still exponential.*

12. STOCHASTIC ANALYSIS OF CELLULAR AUTOMATA

Another new application of stochastic systems and probabilistic logic are cellular automata. The stochastic analysis of cellular automata was already advocated by (Wolfram, 1994), in his paper "Twenty Problems in the Theory of Cellular Automata". The next problem combines Wolfram's problems ten and eleven.

Problem 13 [Wolfram]: *What is the correspondence between cellular automata and stochastic systems, and how are cellular automata affected by noise and other imperfections ?*

We have worked on this problem. In order to provide the reader with more detailed information, I will discuss a simple example. It is taken from ((Mühlenbein and Höns, 2002)).

12.1 THE NONLINEAR VOTER MODEL

We consider a model of two species (or two opinions). For the spatial distribution we assume a one-dimensional stochastic cellular automaton (SCA) defined by a circle of n cells. Each cell is occupied by one individual, thus each cell is characterized by a discrete value $\sigma_i \in \{0, 1\}$. We set $x_{n+1} := x_1$

and $x_0 := x_n$. The state of cell x_i at time $t+1$ is defined by the states of cells x_{i-1}, x_i, x_{i+1} at time t. The state transitions of the voter model depend only on $k(t) = \sigma_{i-1}(t) + \sigma_i(t) + \sigma_{i+1}(t)$. This class of automata is called *totalistic.* For the stochastic voter model the transitions are defined as follows.

$k(t)$	$p(\sigma_i(t+1) = 1 \mid k(t))$
3	$1-\epsilon$
2	$1-\alpha$
1	α
0	ϵ

$p(\sigma_i = 1|k(t)))$ denotes the transition probability given k. ϵ is a small stochastic disturbance parameter. The model is defined by α. If $\alpha < 0.5$ one speaks of positive frequency dependent invasion. This model is also called the *majority vote model,* because the individuals join the opinion of the majority in the neighborhood. For $\alpha > 0.5$ the model is called a negative frequency dependent invasion process. In this case the minority opinion has more weight. The deterministic cellular automata are given by $\epsilon = 0$ and $\alpha = 0, 1$. The voter model has been intensively investigated by micro simulations.

We will first analyze the voter model by the theory of Markov chains. Let $\mathbf{x} = (x_1, \ldots, x_n)$ denote a vector, $x_i \in \Lambda_i = \{0, 1, 2, \ldots, m_i\}$. We use the following conventions. Capital letters X_i denote the names of variables, lower case letters x_i assignments. The *distinction between the name of a variable and an assignment* is essential for the definition of marginal distributions. When there cannot be a confusion between name or assignment, we will use lower case letters and abbreviations. For notational simplicity we will assume *binary variables* $x_i \in \{0, 1\}$.
The time evolution of the distribution is given for one step by the equation

$$p(\mathbf{x}, t+1) = \sum_{x'} p(\mathbf{x}, t+1|\mathbf{x}', t) p(\mathbf{x}', t) \tag{1.4}$$

$M(t) = (p(\mathbf{x}, t+1|\mathbf{x}', t))$ defines a $2^n \times 2^n$ matrix.

Definition 6 *The stochastic process is a Markov process if* $M(t)$ *is independent of* t.

The stochastic voter model is a Markov process. For a Markov process we have

$$p(\mathbf{x}, t) = M^t p(\mathbf{x}, 0) \tag{1.5}$$

For $0 < \epsilon, \alpha < 1$ we have $p(\mathbf{x}|\mathbf{x}') > 0$. Therefore the *theorem of Frobenius-Perron* can be applied. The largest eigenvalue of the matrix is 1. Its unique eigenvector defines the stationary distribution. Thus we have the following theorem.

Theorem 7 *The stochastic voter model with* $0 < \alpha, \epsilon < 1$ *has a unique limit distribution. It is given by the left eigenvector belonging to the eigenvalue* $\lambda_1 = 1$.

It is numerically impossible to analyze a large cellular automaton by standard Markov techniques. It takes an exponential amount of computation to compute the exact stationary distribution.

We propose a different approach. We approximate the distribution $p(\mathbf{x}, t)$ by distributions using a small number of parameters. For this approximation we use the theory of graphical models mentioned before.

12.2 STOCHASTIC ANALYSIS OF ONE DIMENSIONAL SCA

For notational convenience we set $\theta_i := x_i(t+1)$, and $\sigma_i := x_i(t)$. We will now derive difference equations involving marginal distributions with a few number of parameters. We obtain from the definition of the voter model for the von Neumann neighborhood in 1-D

$$p(\theta_i) = \sum_{\sigma_{i-1},\sigma_i,\sigma_{i+1}} p(\theta_i|\sigma_{i-1},\sigma_i,\sigma_{i+1}) p(\sigma_{i-1},\sigma_i,\sigma_{i+1}) \tag{1.6}$$

$p(\theta_i)$ gives the probability of cell i containing a 1. The conditional distribution $p(\theta_i|\sigma_{i-1},\sigma_i,\sigma_{i+1})$ is uniquely defined by the transitions of the cellular automaton, in our case by the voter model with parameters ϵ and α. But on the right side tri-variate marginals appear. For these we obtain

$$\begin{aligned} p(\theta_{i-1},\theta_i,\theta_{i+1}) \quad = \quad & \sum_{\sigma_{i-2},\ldots,\sigma_{i+2}} p(\theta_{i-1},\theta_i,\theta_{i+1}|\sigma_{i-2},\ldots,\sigma_{i+2}) \\ & p(\sigma_{i-2},\sigma_{i-1},\sigma_i,\sigma_{i+1},\sigma_{i+2}) \end{aligned} \tag{1.7}$$

Thus now marginal distribution of size 5 enter. In order to stop this expansion we approximate the marginal distributions of order 5 by marginal distributions of order 3. From the definition of the SCA we obtain

$$\begin{aligned} p(\theta_{i-1},\theta_i,\theta_{i+1}|\sigma_{i-2},\sigma_{i-1},\sigma_i,\sigma_{i+1},\sigma_{i+2}) = p(\theta_{i-1}|\sigma_{i-2},\sigma_{i-1},\sigma_i) \\ p(\theta_i|\sigma_{i-1},\sigma_i,\sigma_{i+1}) p(\theta_{i+1}|\sigma_i,\sigma_{i+1},\sigma_{i+2}) \end{aligned}$$

From the theory of graphical models we obtain the approximation

$$p(\sigma_{i-2},\ldots,\sigma_{i+2}) \approx p(\sigma_{i-1},\sigma_i,\sigma_{i+1}) p(\sigma_{i-2}|\sigma_{i-1},\sigma_i) p(\sigma_{i+2}|\sigma_i,\sigma_{i+1})$$

Inserting the last two equations into equation (1.6) gives the difference equations for the *tri-variate marginal distributions.* The approximations have to

fulfill constraints derived from probability theory.

$$\sum_{\sigma_{i-1},\sigma_i,\sigma_{i+1}} p(\sigma_{i-1},\sigma_i,\sigma_{i+1}) = 1$$

$$\sum_{\sigma_{i-1}} p(\sigma_{i-1},\sigma_i,\sigma_{i+1}) = \sum_{\sigma_{i+2}} p(\sigma_i,\sigma_{i+1},\sigma_{i+2})$$

In the same manner approximations of different precision can be obtained. We just discuss the simplest approximation, using *uni-variate marginal distributions.* Here equation (1.6) is approximated by

$$p(\theta_i) \approx \sum_{\sigma_{i-1},\sigma_i,\sigma_{i+1}} p(\theta_i|\sigma_{i-1},\sigma_i,\sigma_{i+1})p(\sigma_{i-1})p(\sigma_i)p(\sigma_{i+1}) \qquad (1.8)$$

The approximation by univariate marginal distributions leads to n difference equations only, but these difference equations are nonlinear. It seems very unlikely that analytical solutions of these equations can be obtained. For *spatially homogeneous* problems we have $p(\theta_i) = p(\theta_{i+1})$. In this case the probabilities do not depend on the locus of the cell. This is the *mean-field limit* known from statistical physics ((Opper and Saad, D., editors, 2001)). With $x = 1/n \sum_i p(x_i = 1, t)$ we obtain the *mean-field equation*

$$x(t+1) = (1-\epsilon)x^3 + \epsilon(1-x)^3 + 3(1-\alpha)x^2(1-x) + 3\alpha x(1-x)^2 \quad (1.9)$$

For $\epsilon \approx 0$ and $\alpha < 1/3$ the equation has stable fix-points at $x \approx 0$ and $x \approx 1$. For $\alpha > 1/3$ the equation has a stable attractor at $x = 0.5$. Thus the mean-field limit approximation indicates a *bifurcation* for $\alpha = 1/3$. This interpretation is tempting, but not quite correct. The relation between the attractors of the SCA and the fix-points of equation (1.9) is much more complicated ((Mühlenbein and Höns, 2002)).

The approximation of 2-D spatial distributions is much more difficult than the approximation of 1-D automata. Here the junction tree algorithm is needed. The interested reader is referred to ((Mühlenbein and Höns, 2002)).

13. STOCHASTIC ANALYSIS OF EVOLUTIONARY ALGORITHMS

The broad applicability of the new developments in probability theory can be demonstrated by another example, namely *evolutionary algorithms* (Mühlenbein and Mahnig, 2000). This application is easier than the analysis of cellular automata. The distribution remains focused because of selection.

Let a function $f : X \rightarrow \mathbb{R}_{\geq 0}$ be given. We consider the optimization problem

$$\mathbf{x}_{opt} = \text{argmax}\, f(\mathbf{x}) \qquad (1.10)$$

For the solution Holland proposed in 1973 an algorithm called *genetic algorithm* ((Holland, John H., 1975)). The following discussion is taken from ((Mühlenbein and Mahnig, 2003)).

Genetic algorithms are defined on a microscopic level. Given two strings, a new point is generated by recombination/crossover. A stochastic analysis of a genetic algorithm requires the computation of a recurrence equation

$$p(\mathbf{x}, t+1) = \sum_{\mathbf{x}'} p(\mathbf{x}|\mathbf{x}', t)p(\mathbf{x}', t) \tag{1.11}$$

Here $\mathbf{x}$ and $\mathbf{x}'$ denote genotypes (binary vectors), $p(\mathbf{x}|\mathbf{x}', t)$ denotes the probability for a transition from $\mathbf{x}'$ to $\mathbf{x}$ at generation t. Because of selection the transition probabilities are time dependent.

(Vose, 1999), has derived such an equation for the Simple Genetic Algorithm with proportionate selection, crossover, and mutation. The computation of the crossover probabilities are especially difficult. Since crossover operates on two arbitrary strings $\mathbf{x}$ and $\mathbf{y}$ of the selected population, one has to use the joint distribution $p(\mathbf{x}; \mathbf{y})$ in equation (1.11). But even for the binary case, the transfer matrix $p(\mathbf{x}|\mathbf{x}')$ is of size $2^n \times 2^n$. It is extremely difficult to analyze the distribution using this general equation.

But let us proceed further. Equation (1.11) should not be the end result of a stochastic analysis, but just the beginning. We will concentrate on distributions which are defined by a small number of parameters or can be approximated by distributions with a small set of parameters. Since we treat the marginal distributions as *deterministic* variables, the analysis is valid for *infinite* populations only. Fluctuations arising by virtue of finite populations can be investigated in principle, but it is extremely difficult. Due to of the sampling theory in statistics our analysis can be seen as the limit case of large finite populations where the size goes to infinity.

A good candidate for optimization using a search distribution is the Boltzmann distribution.

Definition 8 *For $\beta \geq 0$ define the* Boltzmann distribution *of a function $f(x)$ as*

$$p_\beta(x) := \frac{e^{\beta f(x)}}{\sum_y e^{\beta f(y)}} =: \frac{e^{\beta f(x)}}{Z_f(\beta)} \tag{1.12}$$

where $Z_f(\beta)$ is the partition function. To simplify the notation β and/or f can be omitted.

The Boltzmann distribution is usually defined as $e^{-\frac{g(x)}{T}}/Z$. The term $g(x)$ is called the energy and $T = 1/\beta$ the temperature. The Boltzmann distribution is suited for optimization because it concentrates with increasing β around the global optima of the function. In theory, if it were possible to sample efficiently from this distribution for arbitrary β, optimization would be an easy task.

13.1 BOLTZMANN SELECTION

Our proposed algorithm incrementally computes the Boltzmann distribution by using Boltzmann selection.

Definition 9 *Given a distribution* p *and a selection parameter* $\Delta\beta$, Boltzmann selection *calculates the distribution of the selected points according to*

$$p^s(x) = \frac{p(x)e^{\Delta\beta f(x)}}{\sum_y p(y)e^{\Delta\beta f(y)}} \tag{1.13}$$

Algorithm 1: *BEDA* – Boltzmann Estimated Distribution Algorithm

1 $t \Leftarrow 1$. Generate N points according to the uniform distribution $p(x, 0)$ with $\beta(0) = 0$.
2 **do** {
3 With a given $\Delta\beta(t) > 0$, let
$$p^s(x,t) = \frac{p(x,t)e^{\Delta\beta(t)f(x)}}{\sum_y p(y,t)e^{\Delta\beta(t)f(y)}}.$$
4 Generate N new points according to the distribution $p(x, t+1) = p^s(x, t)$.
5 $t \Leftarrow t + 1$.
6 } **until** (stopping criterion reached)

We can now define the *BEDA* (Boltzmann Estimated Distribution Algorithm). It can easily be proven that *BEDA* converges to the set of all global optima if $\sum_t \Delta(\beta(t)) \rightarrow \infty$ ((Mühlenbein and Mahnig, 2002b)). *BEDA* is a conceptional algorithm, because the calculation of the distribution requires a sum over exponentially many terms. We next transform *BEDA* into a practical algorithm. This means to reduce the number of parameters of the distribution and to compute an adaptive schedule for β.

13.2 FACTORIZATION OF THE DISTRIBUTION

In this section the factorization method introduced for graphical models is applied.

Definition 10 *Let* $s_1, \ldots, s_m$ *be index sets,* $s_i \subseteq \{1, \ldots, n\}$. *Let* f_i *be functions depending only on the variables* x_j *with* $j \in s_i$. *Then*

$$f(\mathbf{x}) = \sum_{i=1}^{m} f_i(\mathbf{x}_{s_i}) \tag{1.14}$$

is an additive decomposition *of the fitness function* f.

From the additive decomposition we construct a *graphical model* by connecting those variables which are contained in the same sub-function. This definition is identical to the graphical model earlier introduced in probabilistic logic. In addition we need the following definitions:

Definition 11 *Given* $s_1, \ldots, s_m$, *we define for* $i = 1, \ldots, m$ *the sets* d_i, b_i *and* c_i:

$$d_i := \bigcup_{j=1}^{i} s_j, \quad b_i := s_i \setminus d_{i-1}, \quad c_i := s_i \cap d_{i-1} \tag{1.15}$$

We set $d_0 = \emptyset$.

In the theory of decomposable graphs, d_i are called *histories,* b_i *residuals* and c_i *separators* ((Lauritzen, 1996)). (Mühlenbein et al., 1999) have proven the following theorem.

Theorem 12 (Factorization Theorem) *Let* $p_\beta(\mathbf{x})$ *be a Boltzmann distribution with*

$$p_\beta(\mathbf{x}) = \frac{e^{\beta f(\mathbf{x})}}{Z_f(\beta)} \tag{1.16}$$

and $f(\mathbf{x}) = \sum_{i=1}^{m} f_{s_i}(\mathbf{x})$ *be an additive decomposition. If*

$$b_i \neq \emptyset \quad \forall i = 1, \ldots, m; \quad d_m = \{x_1, \ldots, x_n\}, \tag{1.17}$$

$$\forall i \geq 2 \ \exists j < i \ \textit{such that} \ c_i \subseteq s_j \tag{1.18}$$

then,

$$p_\beta(\mathbf{x}) = \prod_{i=1}^{m} p_\beta(\mathbf{x}_{b_i} | \mathbf{x}_{c_i}) = \frac{\prod_{i=1}^{m} p_\beta(\mathbf{x}_{b_i}, \mathbf{x}_{c_i})}{\prod_{i=2}^{m} p_\beta(\mathbf{x}_{c_i})} \tag{1.19}$$

The constraint defined as equation (1.18) is called the *running intersection property.* This severe assumption is identical to the junction property defined in equation (1.2).

With the help of the factorization theorem, we can turn the conceptional algorithm *BEDA* into *FDA,* the Factorized Distribution Algorithm. If the conditions of the factorization theorem are fulfilled, the convergence proof of *BEDA* is valid for *FDA* also. *FDA* can in principle be used with any selection scheme, but then the convergence proof is no longer valid. Therefore we believe that Boltzmann selection is an essential part in using the *FDA*.

Since *FDA* uses finite samples of points to estimate the conditional probabilities, convergence to the optimum will depend on the size of the samples (the population size). *FDA* has experimentally proven to be very successful on

Algorithm 2: *FDA* – Factorized Distribution Algorithm

1 Calculate b_i and c_i from the decomposition of the function.
2 $t \Leftarrow 1$. Generate an initial population with N individuals from the uniform distribution.
3 **do** {
4 Select $M \leq N$ individuals using Boltzmann selection.
5 Estimate the conditional probabilities $p(x_{b_i}|x_{c_i}, t)$ from the selected points.
6 Generate new points according to $p(x, t+1) = \prod_{i=1}^{m} p(x_{b_i}|x_{c_i}, t)$.
7 $t \Leftarrow t + 1$.
8 } **until** (stopping criterion reached)

a number of functions where standard genetic algorithms fail to find the global optimum.

For the interested reader we give a short overview of additional work. The scaling behavior for various test functions as well as the computation of the graphical model by sampling data instead of using the structure of the fitness function in investigated in (Mühlenbein and Mahnig, 1999). An early survey can be found in (Mühlenbein and Mahnig, 2000). A large application is solved in (Mühlenbein and Mahnig, 2002a). For a recent survey the reader is referred to (Mühlenbein and Mahnig, 2003).

13.3 HOLLAND'S SCHEMA ANALYSIS AND THE BOLTZMANN DISTRIBUTION

We now turn to the very first analysis of genetic algorithms made by (Holland, John H., 1975). We will introduce here Holland's terminology. But first we will show that this terminology was unnecessary.

Remark: *Marginal distributions define schemata* For the researchers working on the theory of genetic algorithm it is important to mention that marginal distributions are equivalent to *schema* probabilities introduced in (Holland, John H., 1975). We just give an example for $n = 5$. Let $\xi = (1, 0, *, *, *)$ define a schema. Then the probability of the instances of schema ξ in the population $P(t)$ is by definition equal to the marginal distribution $p(X_1 = 1, X_2 = 0, t)$. Thus Holland's schema analysis is nothing else than a stochastic analysis in the space of marginal distributions. We prefer to use the notation common in probability theory. In fact, one of the main reasons that schema theory did not come very far is the imprecise terminology. In our stochastic analysis con-

ditional probabilities play an essential role. But the concept of conditional schema probabilities has not yet entered the traditional schema theory.

Thus in Holland's terminology ξ defines a schema and $P(\xi, t)$ its probability. This probability is in our notation the marginal distribution $p(\mathbf{x}_\xi, t)$.) Holland derived the following conjecture about a good population based search algorithm.

((Holland, John H., 1975),p.88): *Each (schema) ξ represented in (the current population) $B(t)$ should increase (or decrease) in a rate proportional to its "observed" "usefulness" $\hat{\mu}_\xi(t) - \hat{\mu}(t)$ (average fitness of schema ξ minus average fitness of the population)*

$$\frac{dP(\xi, t)}{dt} = (\hat{\mu}_\xi(t) - \hat{\mu}(t))P(\xi, t) \tag{1.20}$$

Holland claimed that the simple genetic algorithm behaves according to the above equation. This is not true. Instead we have the surprising result:

Theorem 13 *The Boltzmann distribution $p(\mathbf{x}, t) = e^{tf(\mathbf{x})}/Z_f(t)$ with $P(\xi, t) = \sum_{X|X_\xi = x_\xi} p(\mathbf{x}, t)$ fulfills Holland's equation (1.20).*

Proof: Taking the derivative we easily obtain

$$\frac{p(\mathbf{x}, t)}{dt} = p(\mathbf{x}, t)(f(\mathbf{x}) - \bar{f}(t)) \tag{1.21}$$

Let ξ define a schema, $\mathbf{x}_\xi$ the corresponding marginal distribution. Then

$$\begin{aligned} \frac{dP(\xi, t)}{dt} = \frac{dp(\mathbf{x}_\xi, t)}{dt} &= p(\mathbf{x}_\xi, t)\left(\frac{1}{p(\mathbf{x}_\xi, t)} \sum_{X|X_\xi = x_\xi} p(\mathbf{x}, t)(f(\mathbf{x}) - \bar{f}(t))\right) \\ &= P(\xi, t)(\hat{\mu}_\xi(t) - \hat{\mu}(t)) \end{aligned}$$

Thus the Boltzmann distribution with the fixed *annealing schedule* $\beta(t) = t$ fulfills Holland's equation. *According to Holland's analysis FDA with us schedule should be an almost optimal algorithm!*

In addition, our factorization theorem can be seen as a mathematically complete *schema theorem.* It tells which schemata are necessary to generate the *whole* distribution. The usual schema theorems describe only the evolution of single schemata, but not how the distribution can be generated.

I hope this short discussion demonstrates that we now have a solid theory of genetic algorithms. But we are still far away from Holland's "logical theory of adaptive systems."

14. STOCHASTIC ANALYSIS AND SYMBOLIC REPRESENTATIONS

We will use the stochastic analysis on more and more complex models. Finally we hope to analyze Holland's general model (problem 11) with the stochastic techniques presented above. Cellular automata can be seen as special cases of Holland's model. All automata perform in the same way, that is we have just one generator. Instead of a tree we have a one or two dimensional space. Selection can be modeled between neighboring automata. The reader has noticed that the stochastic analysis of cellular automata is already fairly difficult. This indicates that the analysis of Holland's model will be really difficult.

But in order to make progress in creating more intelligent machines, still another big step has to be done. In the paper we mainly advocated the probability calculus. With probabilistic logic a first connection is made between symbolic propositions and quantitative variables. This connection has to be extended. It is apparent that ultimately we have to *combine stochastic analysis with more general symbolic representations.* It might be that cellular automata can be used as a first test. This was already proposed by (Wolfram, 1994).

Problem 14 [Wolfram]: *What higher-level descriptions of information processing in cellular automata can be given?*

"One approach is statistical in nature. It consists in devising and describing attractors for the global evolution of cellular automata. All initial configurations in a particular basin of attraction may be thought of as instances of some pattern, so that their evolution towards the same attractor may be considered as a recognition of the pattern.

Another approach is to use symbolic representations for various attributes or components of cellular automaton configurations…perhaps data could be represented by an object like a graph, on which transformations can be performed in parallel…it seems likely that a radically new approach is needed." ((Wolfram, 1994)). To my knowledge Wolfram did not publish any proposal how to solve his problem.

15. CONCLUSION

In my opinion, the big problems in the theory of organisms and artificial automata have been recognized from the very beginning. In biology it is connected to Darwin, Waddington and Mayr, in electronic computation to von Neumann, Turing, Shannon. Some of the proposals for solving the challenging problems in computation have been far too optimistic, other proposals have not been implemented because the implementation was too difficult. There-

fore subsequent developments have lead to a fragmentation and specialization of research. This is true for biology as well as for computer science. Today evolutionary computation is divided into genetic algorithms, evolutionary algorithms, genetic programming, artificial life, and evolvable hardware – not to mention more specialized models like ant colony optimization, memetic algorithms, or classifier systems. But each model itself is too simple to solve the problems presented.

The challenging problems faded away, less difficult problems and simpler models have come into the center of attention. An exception is the problem of all problems: "Can we produce artificial intelligence comparable to or even surpassing human intelligence?" Researchers have often been too optimistic about the time scale to solve this problem. Whereas in the 60's many researcher's predicted a solution in about 10 years, the time scale has now been increased to about 50 years! I am very skeptical that the above goal can be reached. But in my opinion there will be no progress at all, unless some of the problems presented here will have been solved during this 50 years.

REFERENCES

Burns, A. W. (1970). *Essays on Cellular Automata.* University of Illinois Press, Urbana.

Cherry, C. (1961). *Information Theory: Fourth London Symposium.* Butterworth, London.

Cover, T. M. and Thomas, J. A. (1989). *Elements of Information Theory.* Wiley, New York.

Darwin, C. (1859). *The Origins of Species by Means of Natural Selection.* Penguin Classics, London.

Dawkins, R. (1989). *The Selfish Gene.* Oxford University Press, Oxford. (Second Edition).

Griffiths, P. E. (2002). The philosophy of molecular and developmental biology. In *Blackwell Guide to Philosophy of Science.* Blackwell Publishers.

Holland, J. H. (1970a). Iterative circuit computers. In Burns, A. W., editor, *Essays on Cellular Automata,* pages 277–296. University of Illinois Press, Urbana.

Holland, J. H. (1970b). Outline for a logical theory of adaptive systems. In Burns, A. W., editor, *Essays on Cellular Automata,* pages 296–319. University of Illinois Press, Urbana.

Holland, J. H. (1973). Genetic algorithms and the optimal allocation of trials. *Siam Journal on Computing,* 2(2): 88–105.

Holland, J. H. (1975). *Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control, and Artificial In-*

telligence. University of Michigan Press, Ann Arbor, MI. Second edition. Cambridge, MA. The MIT Press 1992.

Jaynes, E. T. (1957). Information theory and statistical mechanics. *Phys. Review,* 6:620–643.

Lauritzen, S. L. (1996). *Graphical Models.* Clarendon Press, Oxford.

Lenat, D. B. (1995). Cyc: A large-scale investment in knowledge infrastructure. *Comm. of the ACM,* 38:924–948.

McCarthy, J. (1959). Programs with common sense. In *Mechanisation of Thought Processes,* pages 75–84. Her Majesty's Stationery Office, London.

McMullin, B. (2001). John von Neumann and the evolutionary growth of complexity: Looking backward, looking forward... *Artificial Life,* 6:347–361.

Minsky, M. (1961). Steps toward artificial intelligence. *Proc. of IRE,* 49:8–30.

Mühlenbein, H. (1991a). Darwin's continent cycle theory and its simulation by the Prisoner's Dilemma. *Complex Systems,* 5:459–478.

Mühlenbein, H. (1991b). Evolution in time and space - the parallel genetic algorithm. In Rawlins, G., editor, *Foundations of Genetic Algorithms,* pages 316–337. Morgan Kaufmann, San Mateo.

Mühlenbein, H. (1996). Algorithms, data and hypothese: Learning in open worlds. In Mahler, G., May, V., and Schreiber, M., editors, *Molecular Electronics: Properties, Dynamics, and Applications,* pages 5–22. Marcel Dekker, New York.

Mühlenbein, H. and Höns, R. (2002). Stochastic analysis of cellular automata with application to the voter model. *Advances in Complex Systems,* 5((2-3)):301–337.

Mühlenbein, H. and Mahnig, T. (1999). FDA – a scalable evolutionary algorithm for the optimization of additively decomposed functions. *Evolutionary Computation,* 7(4):353–376.

Mühlenbein, H. and Mahnig, T. (2000). Evolutionary algorithms: From recombination to search distributions. In Kallel, L., Naudts, B., and Rogers, A., editors, *Theoretical Aspects of Evolutionary Computing, Natural Computing,* pages 137–176. Springer Verlag, Berlin.

Mühlenbein, H. and Mahnig, T. (2002a). Evolutionary optimization and the estimation of search distributions with applications to graph bipartitioning. *Journal of Approximate Reasoning,* 31(3):157–192.

Mühlenbein, H. and Mahnig, T. (2002b). Mathematical analysis of evolutionary algorithms. In Ribeiro, C. C. and Hansen, P., editors, *Essays and Surveys in Metaheuristics,* Operations Research/Computer Science Interface, pages 525–556. Kluwer Academic Publisher, Norwell.

Mühlenbein, H. and Mahnig, T. (2003). Evolutionary algorithms and the Boltzmann distribution. In De Jong, K., Poli, R., and Rowe, J., editors, *Foundations of Genetic Algorithms 7.* Morgan Kaufmann Publishers, San Francisco.

Mühlenbein, H., Mahnig, T., and Ochoa, A. R. (1999). Schemata, distributions and graphical models in evolutionary optimization. *Journal of Heuristics,* 5(2):215–247.

Opper, M. and Saad, D. (editors) (2001). *Advanced Mean Field Methods.* MIT Press, Cambridge.

Oyama, S. (2000). *Evolutions's Eye.* Duke University Press, Durham.

Parisi, D. and Ugolini, M. (2002). Living in enclaves. *Complexity,* 7:21–27.

Rapaport, A. (1970). Modern systems theory – an outlook for coping with change. *General Systems,* XV:15–25.

Shannon, C. E. (1953). Computers and automata. *Proc. of IRE,* 41:1234–1241.

Smith, J. M. and Szathmary, E. (1995). *The Major Transitions in Evolution.* W.H. Freeman, Oxford.

Turing, A. M. (1950). Computing machinery and intelligence. *Mind,* 59:433–460.

Uttley, A. M. (1959). Conditional probability computing in a nervous system. In *Mechanisation of Thought Processes,* pages 19–152. Her Majesty's Stationery Office, London.

von Neumann, J. (1954). The general and logical theory of automata. In *The world of mathematics,* pages 2070–2101. Simon and Schuster, New York.

von Neumann, J. (1956). Probabilistic logics and the synthesis of reliable organs from unreliable components. *Annals of Mathematics Studies,* 34:43–99.

Vose, M. D. (1999). *The Simple Genetic Algorithm: Foundations and Theory.* MIT Press, Cambridge.

Whitehead, A. N. (1948). *Science and the Modern World.* Pelican Books, New York.

Wolfram, S. (1994). *Cellular Automata and Complexity.* Addison-Wesley, Reading.

Wright, S. (1937). The distribution of gene frequencies in populations. *Proc. Nat. Acad. Sci.,* 24:253–259.

Chapter 2

TWO GRAND CHALLENGES FOR EC

Unification and Expansion

Kenneth De Jong
Computer Science Department
George Mason University
Fairfax, VA 22030
kdejong@gmu.edu

Abstract The field of evolutionary computation has developed and matured significantly over the past 40 years. As with other disciplines attempting to understand complex adaptive systems, this progress has raised as many new and interesting questions as it has answered. In this chapter I describe some of the key open questions by organizing them in the form of two grand challenges: unification and expansion.

Keywords: Evolutionary computation, evolutionary algorithms, open research issues, grand challenges.

1. INTRODUCTION

Although more than 40 years old, the field of evolutionary computation (EC) continues to grow at a rapid pace. This growth, in turn, places a certain amount of healthy "stress" on the field as current understanding and traditional approaches are stretched to the limit by challenging new problems and new areas of application.

So, an occasion like this is an opportunity to reflect on where the field is and the challenges that lie ahead. In the following sections I attempt to do so by first noting the important historical events that have strongly influenced the field as we see it today, and then describing some of the key open questions by organizing them in the form of two grand challenges: unification and expansion.

2. HISTORICAL DIVERSITY

Although one can certainly find earlier activities, I believe that there is general agreement that the 1960s was a key historical period that has significantly shaped the field of evolutionary computation. During that period several groups around the world including Rechenberg and Schwefel at the Technical University of Berlin (Rechenberg, 1964), Fogel, et al. at General Dynamics in San Diego (Fogel et al., 1966), and Holland at the University of Michigan in Ann Arbor (Holland, 1962) were captivated by the potential of taking early simulation models of evolution a step further and harnessing these evolutionary processes in computational forms that could be used for complex computer-based problem solving.

Rechenberg and Schwefel were motivated by the need to solve difficult engineering optimization problems and came up with an approach they called *evolutionsstrategie*, or evolution strategies (ESs). Fogel and his colleagues sought to use evolutionary techniques to bypass the bottleneck of building intelligent agents by hand, which they named evolutionary programming (EP). In Holland's case, the motivation was the design and implementation of robust adaptive systems, capable of dealing with an uncertain and changing environment, and lead to the development of genetic algorithms (GAs).

GAs owe their name to an early emphasis on representing and manipulating individuals in terms of their genetic makeup rather than using a phenotypic representation. Much of the early work used a universal internal representation involving fixed-length binary strings with "genetic" operators such as mutation and crossover defined to operate in a domain-independent fashion at this level without any knowledge of the phenotypic interpretation of the strings (Holland, 1975; De Jong, 1975).

By contrast, evolution strategies (ES) and evolutionary programming (EP) were developed initially using more problem-specific "phenotype" representations. In the case of ES the focus was on building systems capable of solving difficult real-valued parameter optimization problems (Schwefel, 1981). The "natural" representation was a vector of real-valued "genes" that was manipulated primarily by mutation operators designed to perturb the real-valued parameters in useful ways. The early work in EP centered on systems for evolving finite state machines capable of responding to environmental stimuli, and developing operators (primarily mutation) for effecting structural and behavioral change over time (Fogel et al., 1966).

These early beginnings have had an enormous influence on the field. In fact an indication of their inspirational power is that these historical labels are no longer all that useful in describing the enormous variety of current activities on the field. GA practitioners are seldom constrained to universal fixed-length binary implementations. ES practitioners have incorporated recombination op-

erators into their systems. EP is used for much more than just the evolution of finite state machines. Entire new subareas such as genetic programming (Koza, 1992) have developed, and the literature is filled with provocative new terms and ideas such as "messy GAs" (Goldberg, 1991).

As a consequence, the field today is highly diversified with many new and exciting application areas, but at the same time generating many new challenges as well. I see these challenges as falling into two primary categories that constitute "grand" challenges for the field, and discuss each of them in the remainder of this chapter.

3. THE CHALLENGE OF UNIFICATION

The diversity of the EC field today can be viewed as both a blessing and a curse in the sense that it reflects both the vitality of the field and the difficulty in articulating a cohesive view. However, in my opinion, developing a unifying EC framework is a key requirement for continued growth and development of the field.

One strategy for achieving this is to focus on the core set of features and issues common to any EC activity. This allows one to understand the relationships between different approaches when contrasted in a common framework, it facilitates the transfer of ideas from one approach to another, and it serves as a solid platform from which to develop new approaches. Given the wide diversity of the field, this may seem like a hopeless task. However, developments over the past few years suggest that considerable progress can be made in this direction by adopting an object-oriented, class hierarchy point of view. In this section I briefly summarize this approach.

At the highest level of abstraction, the EC community shares the common goal of solving difficult computational problems using an evolutionary algorithm (EA) as a key element of the approach. In order to compare and contrast different approaches, we need a common framework for describing EAs. In my opinion, all EAs share the following basic features:

- A population of individuals
- A notion of fitness
- A notion of population dynamics (births, deaths) biased by fitness
- A notion of inheritance of properties from parent to child

Focusing on EAs at this level of abstraction not only helps one to compare and contrast specific instances, but also helps to identify and clarify a number of critical issues that are common to the entire field. I summarize a few of these in the following sections.

3.1 MODELING THE DYNAMICS OF POPULATION EVOLUTION

At a high level of abstraction we think of evolutionary processes in terms of the ability of more-fit individuals to have a stronger influence on the future makeup of the population by surviving longer and by producing more offspring that continue to assert influence after the parents have disappeared. How these notions are turned into computational models varies quite dramatically within the EC community. This variance hinges on several important design decisions discussed briefly in the following subsections.

3.1.1 Choosing Population Sizes. Most current EAs assume a constant population size N which is specified as a user-controlled input parameter. So called "steady state" EAs rigidly enforce this limit in the sense that each time an offspring is produced resulting in $N + 1$ individuals, a selection process is invoked to reduce the population size back to N. By contrast, "generational" EAs permit more elasticity in the population size by allowing $K \gg 1$ offspring to be produced before a selection process is invoked to delete K individuals.

Although we understand that the size of an EA's population can affect its ability to solve problems, we have only the beginnings of a theory strong enough to provide *a priori* guidance in choosing an appropriate fixed size (e.g., (Goldberg et al., 1992), not much theory regarding appropriate levels of elasticity (K), and even less understanding as to the merits of dynamically adjusting the population size.

3.1.2 Deletion Strategies. The processes used to delete individuals varies significantly from one EA to another and includes strategies such as uniform random deletion, deletion of the K worst, and inverse fitness-proportional deletion. It is clear that "elitist" deletion strategies that are too strongly biased towards removing the worst can lead to premature loss of diversity and stagnation at suboptimal solutions. It is equally clear that too little fitness bias results in unfocused and meandering search. Finding a proper balance is important but difficult to determine *a priori* with current theory.

3.1.3 Parental Selection. Similar issues arise with respect to choosing which parents will produce offspring. Biasing the selection too strongly towards the best individuals results in too narrow a search focus, while too little bias produces a lack of needed focus. Current methods include uniform random selection, rank-proportional selection, and fitness-proportional selection.

We understand these selection strategies in isolation quite well (Back, 1995; Blickle and Thiele, 1995). However, it is clear that parental selection and indi-

vidual deletion strategies must complement each other in terms of the overall effect they have on the exploration/exploitation balance. We have some theory here for particular cases such as Holland's "optimal allocation of trials" characterization of traditional GAs (Holland, 1975), and the "1/5" rule for ESs (Schwefel, 1981), but much stronger results are needed.

3.1.4 Reproduction and Inheritance. In addition to these selection processes, the mechanisms used for reproduction also affect the balance between exploration and exploitation. At one extreme one can imagine a system in which offspring are exact replicas of parents (asexual reproduction with no mutation) resulting in rapid growth in the proportions of the best individuals in the population, but with no exploration beyond the initial population members. At the other extreme, one can imagine a system in which the offspring have little resemblance to their parents, maximizing exploration at the expense of inheriting useful parental characteristics.

The EC community has focused primarily on two reproductive mechanisms which fall in between these two extremes: 1-parent reproduction with mutation and 2-parent reproduction with recombination and mutation. Historically, the EP and ES communities have emphasized the former while the GA community has emphasized the latter.

However, these traditional views are breaking down rapidly. The ES community has found recombination to be useful, particularly in evolving adaptive mutation rates (Bäck and Schwefel, 1993). Various members of the GA community have reported improved results by not using recombination (de Garis, 1990), by not using mutation (Koza, 1992), or by adding new and more powerful mutation operators (Eshelman and Schaffer, 1991). More recently the virtues of N-parent recombination ($N > 2$) have been explored (Eiben, 1996).

As before, we have the tantalizing beginnings of a theory to help understand and guide the use and further development of reproductive mechanisms. Beginning with Holland's initial work, the GA community has analyzed in considerable detail the role of crossover and mutation (see, for example, (De Jong, 1975; Goldberg, 1989; Vose and Liepins, 1991; Booker, 1992; Spears, 1998). The ES community has developed theoretical models for optimal mutation rates with respect to convergence and convergence rates in the context of function optimization (Schwefel, 1995).

However, the rapid growth of the field is pressing these theories hard with "anomalous results" (Forrest and Mitchell, 1992) and new directions not covered by current theory. One of the important issues not well understood is the benefit of adaptive reproductive operators. There are now a variety of empirical studies that show the effectiveness of adaptive mutation rates (e.g., (Fogarty, 1989), (Bäck and Schwefel, 1993), or (Fogel, 1995b)) as well as adaptive

recombination mechanisms (e.g., (Schaffer and Morishima, 1987) or (Davis, 1989)).

3.2 CHOICE OF REPRESENTATION

One of the most critical decisions made in applying evolutionary techniques to a particular class of problems is the specification of the space to be explored by an EA. This is accomplished by defining a mapping between points in the problem space and points in an internal representation space.

The EC community differs widely on opinions and strategies for selecting appropriate representations, ranging from universal binary encodings to problem-specific encodings for TSP problems and real-valued parameter optimization problems. The tradeoffs are fairly obvious in that universal encodings have a much broader range of applicability, but are frequently outperformed by problem-specific representations which require extra effort to implement and exploit additional knowledge about a particular problem class (see, for example, (Michalewicz, 1994)). An intriguing idea being explored is to allow the representation to adapt to the particular characteristics of a problem (e.g., "messy GAs" (Goldberg et al., 1991)).

Although there are strong historical associations between GAs and binary string representations, between ESs and vectors of real numbers, and between EP and finite state machines, it is now quite common to use representations other than the traditional ones in order to effectively evolve more complex objects such as symbolic rules, Lisp code, or neural networks. Claiming one EA approach is better than another on a particular class of problems is not meaningful any more without motivating and specifying (among other things) the representations chosen.

What is needed, but has been difficult to obtain, are theoretical results on representation theory. Holland's schema analysis (Holland, 1975) and Radcliffe's generalization to formae (Radcliffe, 1991) are examples of how theory can help guide representation choices. Similarly "fitness correlation" (Manderick et al., 1991) and operator-oriented views of internal fitness landscapes (Jones, 1995) emphasize the tightly coupled interaction between choosing a representation for the fitness landscape and the operators used to explore it. Clearly, much more work is required if effective representations are to be easily selectable.

3.3 CHARACTERISTICS OF FITNESS LANDSCAPES

The majority of the EC applications to date has been with problem domains in which the fitness landscape is time-invariant and the fitness of individuals can be computed independently from other members of the current population.

This is a direct result of the pervasiveness of optimization problems and the usefulness of evolutionary algorithms (EAs) in solving them. This has led to considerable insight into the behavior of EAs on such surfaces including such notions as "GA-easy", "GA-hard", and "deception".

Much of this work has involved optimization problems that are unconstrained or lightly constrained (e.g., upper and lower bounds on the variables). The situation becomes more difficult as the complexity of the constraints increases. The ability to exploit constraint knowledge is frequently the key to successful applications, and that in turn can imply creative, non-standard representations and operators (Michalewicz and Schoenauer, 1996). How to do this effectively is still an interesting and open research issue.

Things become even more interesting and open ended if we attack problem classes in which the fitness landscape varies over time. There are at least three important problem classes of this type for which research results are badly needed: autonomously changing landscapes, the evolution of cooperative behavior, and ecological problems.

Problems involving autonomously changing landscapes frequently arise when fitness is defined in terms of one or more autonomous entities in the environment whose behavior can change independently of any of the search activity of an EA. Typical examples are mechanical devices that age, breakdown, etc, or changes in weather patterns which dramatically change the "fitness" of a particular ship on the open sea. If we apply typical optimization-oriented EAs to such problems, the strong pressures to converge generally result in a loss of the population diversity needed to respond to such changes. We currently have very little insight regarding how to design EAs for such problems.

Rule-learning systems (Holland, 1986; Grefenstette et al., 1990), iterated prisoner's dilemma problems (Axelrod, 1987; Fogel, 1995a), and immune system models (Forrest et al., 1993) are examples of problems in which fitness is a function of how well an individual complements other individuals in the population. Rather than searching for a single optimal individual, the goal is to evolve groups of individuals (generalists, specialists, etc.) that collectively solve a particular problem.

If we apply typical optimization-oriented EAs to such problems, the strong pressures towards homogeneity in the population make it difficult to maintain different but cooperative individuals. Additional mechanisms for rewarding groups of individuals seem to be required (e.g., bucket brigades, profit sharing), but we have little in the way of theory to guide us.

Ecology-oriented problems present a third and perhaps most difficult class of landscapes in which the shape of the fitness landscape is directly affected by the evolutionary process itself. Perhaps a better way to think of this is in co-evolutionary terms in which multiple interacting evolutionary processes are at work modeling the availability of resources (Holland, 1992), prey-predator

relationships, host-parasite interactions (Hillis, 1990), and so on. Very few of our insights from the optimization world appear to carry over here.

4. THE CHALLENGE OF EXPANSION

In the previous section, we argued that a unified EC framework is needed to provide a deeper understanding of the many forms of evolutionary computation that exist and their use as effective problem solvers. However, the interest in using EAs to solve problems that violate traditional assumptions, continues to grow. We already have examples of EAs which are powerful function optimizers, but which are completely ineffective for evolving cooperative behavior or tracking a changing landscape. Modified EAs are now being developed for these new problem classes, but are also much less useful as traditional optimizers.

This presents us with our second grand challenge: how to extend and expand our repertoire of EAs in an effective and principled manner. My answer is that we use the unified framework discussed in the previous section as the platform for doing so. In this section I illustrate this approach by describing several key expansion areas.

4.1 REPRESENTATION AND MORPHOGENESIS

In the earlier section on representation issues we discussed the tradeoffs between problem-independent and problem-specific representations. Closely related to this is the biological distinction between the more universal genotypic descriptions of individuals in the form of plans for generating them and the phenotypic descriptions of the actual generated structures.

Historically, much of the EA work has involved the evolution of fairly simple structures that could be represented in phenotypic form or be easily mapped onto simple genotypic representations. However, as we attempt to evolve increasingly more complex structures such as Lisp code (Koza, 1992) or neural networks (de Garis, 1990), it becomes increasingly difficult to define forms of mutation and recombination which are capable of producing structurally sound and interesting new individuals. If we look to nature for inspiration, we don't see many evolutionary operators at the phenotype level (e.g., swapping arms and legs!). Rather, changes occur at the genotype level and the effects of those changes instantiated via growth and maturation. If we hope to evolve such complexity, we may need to adopt more universal encodings coupled with a process of morphogenesis (e.g., (Harp et al., 1989), or (Stanley and Miikkulainen, 2002)).

4.2 NON-RANDOM MATING AND SPECIATION

Currently, most EAs incorporate a random mating scheme in which the species or sex of an individual is not relevant. One problem with this, as with real biological systems, is that the offspring of parents from two species are often not viable. As we move to more complex systems which attempt to evolve cooperating behavior and which may have more than one evolutionary process active simultaneously, the roles of non-random mating and speciation will become an important issue.

Some solutions to these problems have been suggested, such as crowding (De Jong, 1975), sharing (Goldberg and Richardson, 1987), and tagging (Booker, 1982). Unfortunately, these solutions tend to make fairly strong assumptions, such as the number of species and/or the distribution of niches in the environment. For some problems these assumptions are reasonable. However, in many cases such properties are not known *a priori* and must evolve as well (Spears, 1994).

4.3 DECENTRALIZED, HIGHLY PARALLEL MODELS

Because of the natural parallelism within an EA, much recent work has concentrated on the implementation of EAs on both fine and coarse grained parallel machines. Clearly, such implementations hold promise of significant decreases in the execution time of EAs.

More interestingly, though, for the topic of this paper, are the evolutionary effects that can be naturally implemented with parallel machines, namely, speciation, nicheing, and punctuated equilibria. For example, non-random mating may be easily implemented by enforcing parents to be neighbors with respect to the topology of the parallel architecture. Species emerge as local neighborhoods within that topology. Subpopulations in equilibrium are "punctuated" by easily implemented migration patterns from neighboring subpopulations.

However, each such change to an EA significantly changes its semantics and the resulting behavior. Our admittedly weak theory about traditional EAs needs to be strengthened and extended to help us in better understanding and designing these parallel implementations. In the case of finely grained, neighborhood models some significant progress is being made along these lines (see, for example, (Sarma, 1998)).

4.4 SELF-ADAPTING SYSTEMS

Another theme that has been arising with increasing frequency is the inclusion of self-adapting mechanisms with EAs to control parameters involving

the internal representation, mutation, recombination, and population size. This trend is due in part to the absence of strong predictive theories that specify such things *a priori.* It is also a reflection of the fact that EAs are being applied to more complex and time-varying fitness landscapes.

Some important issues that need to be solved involve the self-adaptation mechanism itself. For example, do we use an EA or some other mechanism? If we use an EA, how do we use fitness as a performance feedback for self-adaptation?

On a positive note, the EC community has already empirically illustrated the viability of self-adaptation of mutation and recombination as noted earlier, as well as adaptive representations like Argot (Shaefer, 1987), messy GAs (Goldberg et al., 1991), dynamic parameter encoding schemes (Schraudolph and Belew, 1992), and Delta coding (Whitley et al., 1991). Thesis work of Turner (Turner, 1998) suggests that simple performance-based mechanisms can be effectively used to dynamically tune parent selection and operator usage.

4.5 COEVOLUTIONARY SYSTEMS

Hillis' work (Hillis, 1990) on the improvements achievable by co-evolving parasites along with the actual individuals of interest gives an exciting glimpse of the behavioral complexity and power of such techniques. Holland's Echo system (Holland, 1992) reflects an even more complex ecological setting with renewable resources and predators. More recently, Rosin (Rosin and Belew, 1995) and Potter (Potter, 1997) have shown the benefits of both "competitive" and "cooperative" co-evolutionary models.

Each of these systems suggests an important future role for co-evolution in EAs, but they raise more questions than they answer concerning a principled method for designing such systems as well as the kinds of problems for which this additional level of complexity is both necessary and effective. One promising approach currently being explored is the use of evolutionary game theory (Ficici and Pollack, 2000; Wiegand et al., 2002).

4.6 INCLUSION OF LAMARCKIAN PROPERTIES

Although EAs may be inspired by biological systems, many interesting properties arise when we include features not available to those systems. One common example is the inclusion of Lamarckian operators, which allow the inheritance of characteristics acquired during the lifetime of an individual.

In the EC world this is beginning to show up in the form of hybrid systems in which individuals themselves go through a learning and/or adaptation phase as part of their fitness evaluation, and the results of that adaptation are passed on to their offspring (e.g., see (Turney et al., 1996)). Although initial empirical

results are encouraging, we presently have no good way of analyzing such systems at a more abstract level.

4.7 MODELING EVOLUTIONARY SYSTEMS

With few exceptions this entire discussion so far has been presented from a computer science and engineering perspective, namely, the use of EAs as computational tools to solve difficult computer science and engineering problems. This is, to a great extent, a reflection of the individuals in the field and their interests. It should be clear, however, that an equally plausible direction is to use EAs as models of biological and other evolving systems.

However, there is a problem here in that it is difficult to achieve both computational utility and biological plausibility in a single model. As a consequence, most of the computationally oriented EAs that have been developed over the past 40 years are quite inadequate as modeling tools. Rather, significant development effort is usually required to rework an existing EA for systems modeling efforts (see, for example, (Burke et al., 1998)). What I have seen in the past few years is a significant growth of interest in these kinds of EA applications, and a growing sense of a need to fill the gap between currently developed EAs and the kinds of EAs needed for effective evolutionary systems modeling tools.

5. SUMMARY AND CONCLUSIONS

This is an exciting time for the EC field. The increased level of activity has resulted in an infusion of new ideas and applications that are challenging old tenets and requiring fundamental changes in the ways in which we model and use evolutionary algorithms.

I have attempted to summarize this in the form of two grand challenges: unification and expansion. I believe that progress in these areas is critically important for the continued growth of the field.

REFERENCES

Axelrod, R. (1987). The evolution of strategies in the iterated prisoner's dilemma. In Davis, L., editor, *Genetic Algorithms and Simulated Annealing,* pages 32–41. Morgan Kaufmann.

Bäck, T. (1995). Generalized convergence models for tournament and (*mu, lambda*) selection. In Eshelman, L., editor, *Proceedings of the Sixth International Conference on Genetic Algorithms,* pages 2–9. Morgan Kaufmann.

Bäck, T. and Schwefel, H.-P. (1993). An overview of evolutionary algorithms for parameter optimization. *Evolutionary Computation,* 1(1): 1–23.

Blickle, T. and Thiele, L. (1995). A mathematical analysis of tournament selection. In Eshelman, L., editor, *Proceedings of the Sixth International Conference on Genetic Algorithms,* pages 9–16. Morgan Kaufmann.

Booker, L. (1982). *Intelligent Behavior as an adaptation to the task environment.* PhD thesis, University of Michigan, Ann Arbor.

Booker, L. (1992). Recombination distributions for genetic algorithms. In Whitley, D., editor, *Foundations of Genetic Algorithms 2*, pages 29–44. Morgan Kaufmann.

Burke, D., De Jong, K., Grefenstette, J. J., Ramsey, C. L., and Wu, A. (1998). Putting more genetics into genetic algorithms. *Evolutionary Computation,* 6(4):387–410.

Davis, L. (1989). Adapting operator probabilities in genetic algorithms. In Schaffer, J. D., editor, *Proceedings of the Third International Conference on Genetic Algorithms*, pages 60–69. Morgan Kaufmann.

de Garis, H. (1990). Genetic programming: modular evolution for Darwin machines. In *Proceedings of the International Joint Conference on Neural Networks,* pages 194–197. Lawrence Erlbaum.

De Jong, K. (1975). *Analysis of the behavior of a class of genetic adaptive systems.* PhD thesis, University of Michigan, Ann Arbor.

Eiben, G. (1996). Multi-parent's niche: N-ary crossovers on NK-landscapes. In *Proceedings of the Fourth International Conference on Parallel Problem Solving from Nature,* pages 319–335. Springer Verlag.

Eshelman, L. and Schaffer, D. (1991). Preventing premature convergence in genetic algorithms by preventing incest. In Belew, R. K. and Booker, L. B., editors, *Proceedings of the Fourth International Conference on Genetic Algorithms,* pages 115–122. Morgan Kaufmann.

Ficici, S. and Pollack, J. (2000). A game-theoretic approach to the simple coevolutionary algorithm. In Schoenauer, M., Deb, K., Rudolph, G., Yao, X., Lutton, E., Merelo, J., and Schwefel, H.-P., editors, *Proc. of the Sixth Conference on Parallel Problem Solving from Nature (PPSN VI)*, pages 467–476. Springer-Verlag.

Fogarty, T. (1989). Varying the probability of mutation in the genetic algorithm. In *Proceedings of the Third International Conference on Genetic Algorithms,* pages 104–109. Morgan Kaufmann.

Fogel, D. B. (1995a). On the relationship between the duration of an encounter and the evolution of cooperation in the iterated prisoner's dilemma. *Evolutionary Computation,* 3(3):349–363.

Fogel, D. B. (1995b). *Evolutionary Computation.* IEEE Press.

Fogel, L., Owens, A., and Walsh, M. (1966). *Artificial intelligence through simulated evolution.* John Wiley.

Forrest, S., Javornik, B., Smith, R., and Perelson, A. (1993). Using genetic algorithms to explore pattern recognition in the immune system. *Evolutionary Computation,* 1(3): 191–212.

Forrest, S. and Mitchell, M. (1992). Relative building block fitness and the building block hypothesis. In Whitley, D., editor, *Foundations of Genetic Algorithms 2,* pages 109–126. Morgan Kaufmann.

Goldberg, David E., Deb, Kalyanmoy, and Clark, J. (1992). Accounting for noise in sizing of populations. In Whitley, D., editor, *Foundations of Genetic Algorithms 2,* pages 127–140. Morgan Kaufmann.

Goldberg, David E., Deb, Kalyanmoy, and Korb, B. (1991). Don't worry, be messy. In Belew, R. K. and Booker, L. B., editors, *Proceedings of the Fourth International Conference on Genetic Algorithms,* pages 24–30. Morgan Kaufmann.

Goldberg, David E. and Richardson, J. (1987). Genetic algorithms with sharing for multimodal function optimization. In Grefenstette, J. J., editor, *Proceedings of the Second International Conference on Genetic Algorithms,* pages 41–49. Lawrence Erlbaum.

Goldberg, David E. (1989). *Genetic Algorithms in Search, Optimization, and Machine Learning.* Addison-Wesley.

Grefenstette, J. J., Ramsey, C. L., and Schultz, A. (1990). Learning sequential decision rules using simulation models and competition. *Machine Learning,* 5(4):355–381.

Harp, S., Samad, T., and Guha, A. (1989). Towards the genetic synthesis of neural networks. In *Proc. of the Third International Conference on Genetic Algorithms*, pages 360–369. Morgan Kaufmann.

Hillis, D. (1990). Co-evolving parasites improve simulated evolution as an optimization procedure. *Physica D*, 42:228–234.

Holland, J. H. (1962). Outline for a logical theory of adaptive systems. *JACM,* 9:297–314.

Holland, J. H. (1975). *Adaptation in natural and artificial systems.* University of Michigan Press.

Holland, J. H. (1986). Escaping brittleness: The possibilities of general-purpose learning algorithms applied to parallel rule-based systems. In Michalski, R., Carbonell, J., and Mitchell, T, editors, *Machine Learning,* volume 2, pages 593–624. Morgan Kaufmann.

Holland, J. H. (1992). *Adaptation in natural and artificial systems, second edition.* MIT Press.

Jones, T. (1995). *Evolutionary algorithms, fitness landscapes, and search.* PhD thesis, University of New Mexico.

Koza, John R. (1992). *Genetic Programming: On the programming of computers by means of natural selection.* Bradford Books, Cambridge.

Manderick, B., de Weger, M., and Spiessens, P. (1991). The genetic algorithm and the structure of the fitness landscape. In Belew, R. K. and Booker, L. B., editors, *Proceedings of the Fourth International Conference on Genetic Algorithms,* pages 143–150. Morgan Kaufmann.

Michalewicz, Z. (1994). *Genetic Algorithms + Data Structures = Evolution Programs.* Springer-Verlag.

Michalewicz, Z. and Schoenauer, M. (1996). Evolutionary algorithms for constrained optimization problems. *Evolutionary Computation,* 4(1): 1–32.

Potter, M. (1997). *The Design and Analysis of a Computational Model of Cooperative Coevolution.* PhD thesis, George Mason University.

Radcliffe, N. J. (1991). Forma analysis and random respectful recombination. In Belew, R. K. and Booker, L. B., editors, *Proceedings of the Fourth International Conference on Genetic Algorithms,* pages 222–229. Morgan Kaufmann.

Rechenberg, I. (1964). Cybernetic solution path of an experimental problem. Library Translation 1122, August 1965. Farnborough Hants: Royal Aircraft Establishment. English translation of lecture given at the Annual Conference of the WGLR at Berlin in September, 1964.

Rosin, C. and Belew, R. K. (1995). Methods for competitive co-evolution: Finding opponents worth beating. In Eshelman, L., editor, *Proceedings of the Sixth International Conference on Genetic Algorithms,* pages 373–380. Morgan Kaufmann.

Sarma, J. (1998). *An Analysis of Decentralized and Spatially Distributed Genetic Algorithms.* PhD thesis, George Mason University.

Schaffer, D. and Morishima, A. (1987). An adaptive crossover mechanism for genetic algorithms. In Grefenstette, J. J., editor, *Proceedings of the Second International Conference on Genetic Algorithms,* pages 36–40. Morgan Kaufmann.

Schraudolph, N. and Belew, R. K. (1992). Dynamic parameter encoding for genetic algorithms. *Machine Learning,* 9(l):9–22.

Schwefel, H.-P. (1981). *Numerical optimization of computer models.* John Wiley and Sons.

Schwefel, H.-P. (1995). *Evolution and optimum seeking.* Wiley.

Shaefer, C. (1987). The argot strategy: adaptive representation genetic optimizer technique. In Grefenstette, J. J., editor, *Proceedings of the Second International Conference on Genetic Algorithms,* pages 50–58. Lawrence Erlbaum.

Spears, W. M. (1994). Simple subpopulation schemes. In *Proc. of the Evolutionary Programming Conference,* pages 296–307. World Scientific.

Spears, W. (1998). *The Role of Mutation and Recombination in Evolutionary Algorithms.* PhD thesis, George Mason University.

Stanley, K. and Miikkulainen, R. (2002). Evolving neural networks through augmenting topologies. *Evolutionary Computation,* 10(2):99–127.

Turner, M. (1998). *Performance-based Self-adaptive Evolutionary Behavior.* PhD thesis, George Washington University.

Turney, P., Whitley, D., and Anderson, R. (1996). Evolution, learning, and instinct: 100 years of the baldwin effect. *Evolutionary Computation,* 4(3).

Vose, M. D. and Liepins, G. E. (1991). Schema disruption. In Belew, R. K. and Booker, L., editors, *Proceedings of the Fourth International Conference on Genetic Algorithms,* pages 237–242. Morgan Kaufmann.

Whitley, D., Mathias, K., and Fitzhorn, P. (1991). Delta coding: an iterative search strategy for genetic algorithms. In Belew, R. K. and Booker, L. B., editors, *Proceeding of the Fourth International Conference on Genetic Algorithms,* pages 77–84. Morgan Kaufmann.

Wiegand, P., Liles, W., and De Jong, K. (2002). Analyzing cooperative coevolution with evolutionary game theory. In Fogel, D., editor, *The IEEE World Congress on Computational Intelligence,* pages 1600–1606. IEEE Press.

This page intentionally left blank

Chapter 3

EVOLUTIONARY COMPUTATION: CHALLENGES AND DUTIES

Carlos Cotta
Dept. Lenguajes y Ciencias de la Computación, Universidad de Málaga
ETSI Informática (3.2.49), Campus de Teatinos, 29071-Málaga, Spain
ccottap@lcc.uma.es

Pablo Moscato
School of Electrical Engineering and Computer Science,
University of Newcastle,
Callaghan, NSW, 2308 Australia
moscato@densis.fee.unicamp.br

Abstract Evolutionary Computation (EC) is now a few decades old. The impressive development of the field since its initial conception has made it one of the most vigorous research areas, specifically from an applied viewpoint. This should not hide the existence of some major gaps in our understanding on these techniques. In this essay we propose a number of challenging tasks that –according to our opinion– should be attacked in order to fill some of these gaps. They mainly refer to the theoretical basis of the paradigm; we believe that an effective cross-fertilization among different areas of Theoretical Computer Science and Artificial Intelligence (such as Parameterized Complexity and Modal Logic) is mandatory for developing a new corpus of knowledge about EC.

1. INTRODUCTION

On July 2, 2002, we made several queries using one of the most popular databases for retrieval of research publications, the *Web of Science*[1]. Undoubtedly, one of the best jewels of the crown of Theoretical Computer Science is the theory of *NP*-completeness (Garey and Johnson, 1979), so we thought that it would be relevant to identify how many papers have been published and catalogued in this database that include either the terms *'NP-hard'* or *'NP-*

[1] http://www.isinet.com

complete' in either the title or the abstract or even as a keyword. It has been reported elsewhere that "thousands" of problems have been already catalogued as *NP*-hard, so we thought that this search would at least help to indirectly quantify the presence of *NP*-completeness theory in the scientific and technological literature. Surprisingly for us, only 4,111 documents contained at least one of these terms. We expected this number to be larger provided the significance and widespread usefulness of this classification.

This result is curious, we hoped to get a larger figure, since, for comparison purposes, the same database retrieved 1,361 documents containing either *'salesman problem'* or *'salesperson problem',* in general referring to a particular problem member of the *NP* Optimization class (the traveling salesman problem, MIN TSP).

Regarding the current use of "single-agent" metaheuristic optimization methods, two of them take the lead with *"Simulated Annealing"* (Kirkpatrick et al., 1983) (4,676) and *"Tabu Search"* (Glover and Laguna, 1997) (856). These metaheuristics have been introduced at least one decade after the theory of *NP*-completeness and they are widely used in practice. Noting that any metaheuristic method turns into a heuristic when applied to a particular problem, it is also relevant to query for *"heuristic"* or *"heuristics"* giving an impressive number of 15,933 documents in the database, most of them on algorithmic approaches to solve a problem modeled in formal mathematical terms.

There are many possible interpretations for the results of these database queries, and each of these interpretations is the amalgamation of a number of factors and conjectures. The reader may agree with us in that the great success of metaheuristics in solving in practice many hard optimization problems is certainly one of the circumstances to take into account. In our opinion, a subtle shift in research focus is also a major factor in this result. More precisely, it may be that the relative weight of *applied* research (recall that most of the works dealing with metaheuristics are of applied nature) has increased with respect to *fundamental* research. The wide availability of computing resources is crucial in this sense: testing and comparing different approaches for solving a problem can be much more amenable than complex mathematical analysis. This philosophy could be summarized in *"try to get probably good solutions to your problem, for provably good solutions are overwhelmingly hard to obtain "*. For most problem domains, we should take extreme care in order to define what can be a challenging instance, since it may be extremely easy to find optimal solutions (Krivelevich, 2002), biasing the chosen scenario favoring exact methods (see the discussion in (Berretta and Moscato, 1999)).

While the lack of a proper mathematical analysis is not something to be inherently criticized from a scientific point of view, it is true that the lack of solid theoretical basis for most metaheuristics will jeopardize their successful utilization in the 21^{st} Century. Quoting Lewis and Papadimitriou (Lewis and Papadimitriou, 1998):

> *"Explaining and predicting the impressive empirical success of some of these algorithms is one of the most challenging frontiers of the theory of computation today."*

Indeed, developing formal theories for grasping the optimization dynamics of these algorithms, and to devise appropriate metaheuristics for solving specific problems appear as the major challenges researchers have to face. This is specifically true in the field of evolutionary algorithms (Bäck et al., 1997) (EAs), one of the metaheuristics families with stronger impetus, yet whose foundation-knowledge corpus remains very incomplete.

In this essay, we will try to identify some of the principal challenges whose solution we believe may constitute important milestones for EA development. Each of these challenges will be described in a different section. It must be noted that their numbering is not intended to represent any relevance order. On the contrary, the reader is invited to rank them according to his/her particular vision of the field.

2. CHALLENGE #1: HARD PROBLEMS FOR THE PARADIGM – EPISTASIS AND PARAMETERIZED COMPLEXITY

It is absolutely necessary to identify and understand the relative "hardness" of finding appropriate algorithms for specific problems, in particular with respect to the computational complexity classes to which the problems belong. We believe that it would be a better attitude, particularly toward building a bridge with Theory of Computation, to try firstly to identify hard problems for EAs in relationship with known computational classes. At present, the approach of creating "toy problems" that are "hard" for the paradigm, while partly useful for identifying some particular issues that need consideration, does not lead to an articulated, systematic approach to understand for which problems, or problem instances, the EA approach is competitive or even superior, to exact approaches or other metaheuristics. We may ask:

> *Is there any way to find efficient algorithms based on evolutionary search principles which always give good approximations to optimization problems to which it is hard to find the optimal solution ?*

From some perspective, the answer of this question is most probably "No", since under the commonly believed assumption that $P \neq NP$, we know that there are some problems that can not be approximated with efficient algorithms at all. On the other hand, this question can be relativized by answering: *"It depends on the problem"* since we know that for some problems that are equally hard to be solved to optimality, some can be very well approximated with efficient algorithms. This leaves some room for the possibility that some problems can be efficiently approximated using algorithms based on evolutionary techniques while others do not.

To study the central question presented above, we identify three complementary research directions:

- Identify NP-Optimization problems for which the evolutionary search paradigm has proved not to be competitive against the best heuristic or approximation algorithm known for those problems.
- Identify which optimization problems can be approached using an evolutionary search paradigm and identify the reasons.
- For the problems of the two groups mentioned above, it will be important to find links with the theory of computational complexity and the complexity classes (regarding approximability and, in particular, parameterized complexity (Fellows, 2002)) that those problems belong to.

Ideally, the outcome of the above research will also provide interesting clues in terms of relating computational complexity classes with the typical measures of "EA-hardness" such as epistasis (Davidor, 1991; Forrest and Mitchell, 1991). This phenomenon –the non-additive fitness dependence among several *genes*– has a direct influence in the difficulty an EA faces for solving a certain problem (defined as the combination of a particular representation and fitness function). It is customary to quantify epistasis by means of a integer fixed parameter, say $p > 0$. The existence of such a parameter in the context of the discussion about complexity classes mentioned above immediately suggest a possible connection with the paradigm of *parameterized complexity* (PC) (for an interesting introduction to the general topic see (Downey et al., 1999)). This paradigm extends the classical paradigm by analyzing the complexity of problems with respect to a certain parameter (or set of parameters). Recall that classical classifications such as the conspicuous $P - NP$ dichotomy are based in a worst-case scenario. For instance, the paradigmatic SAT problem is known to be easily solvable in general (for a particular type or randomly-generated instances) except for instances located at the phase transition between satisfiability and non-satisfiability (Gent and Walsh, 1994). The existence of structural parameters upon which to base the complexity analysis can be very useful to isolate such scenarios. The PC paradigm establishes a hierarchy of parameterized complexity classes $FPT \subseteq W[1] \subseteq W[2] \subseteq \cdots \subseteq W[t] \subseteq W[SAT] \subseteq W[P]$ that allows discriminating problems of different complexity according to the chosen parameter. For example, problems in the *FPT* (*fixed-parameter tractable*) class have algorithms whose worst-case complexity is $O(f(k)n^c)$, where k is the parameter, $f(k)$ and arbitrary function of k only, and c is a constant. In contrast, the complexity of solving problems in $W[1]$ is $O(f(k)n^{g(k)})$, substantially harder in general.

A prototypical example of an NP-complete problem whose parameterized version is fixed-parameter tractable is VERTEX COVER. This problem can be

defined as follows:

k-VERTEX COVER

- **Instance:** An undirected graph $G(V, E)$, with $|V| = n$, an integer $k > 0$.

- **Question:** Does there exist a set $V' \subseteq V$ of k vertices, such that for every $(u, w) \in E$, it holds that $u \in V'$ or $w \in V'$?

If the size of the set V' is taken as a parameter, this problem can be shown to be in *FPT* (Downey and Fellows, 1995), existing algorithms for solving it in $O(1.278^k + n)$, i.e., linear in n for fixed k, and polynomial in n for $k \in O(\log n)$. This surprising result can be achieved by combining the results of (Chen et al., 1999) and the speed-up method of (Niedermeier and Rossmanith, 2000). Notice that while VERTEX COVER would be dismissed as "probably intractable" according to its *NP*-hardness, it turns out to be perfectly solvable for a wide range of values for its structural parameter.

A lesson can be extracted, since we may apply these algorithms for recombination operators. They appear to be greatly advantageous when the population has begun to converge to similar individuals. We will return to this issue in the next challenge.

As mentioned above, the *W*-hierarchy allows encapsulating problems of increasing difficulty. The membership of a certain problem to a precise PC class is established by means of Boolean circuits. These are traditional networks of logical gates that take a potential solution as an input, and output a Boolean value indicating whether that is a solution for the problem considered. The structure of the circuit obviously depends on the problem, and its complexity determines the precise PC class to which membership is established. More precisely, an important parameter is the *weft* of the circuit. This is the maximum number of logical gates whose fan-in is unrestricted (depends on the problem data) in an input-output path. The higher the weft for a constant depth of the circuit, the higher the class the circuit belongs to.

At this point, the resemblance between the weft of a Boolean circuit and the structural interdependence of genes in epistatic representations suggests that some deep connection may exist between PC and the yet informal notion of "EA-hardness". Disentangling this connection (if it effectively exists) constitutes a very attractive challenge both for Computer Science theoreticians and EA researchers.

3. CHALLENGE #2: SYSTEMATIC DESIGN OF PROVABLY GOOD RECOMBINATION OPERATORS

Recombination is undoubtedly a major component of evolutionary algorithms, at least in the case of genetic algorithms (GAs). While its intuitive role has been always clear (to combine the information present in a set of solutions to create new solutions), the guidelines for designing effective recombination operators have experienced a remarkable evolution. It is increasingly accepted that instead of directly manipulating the syntactic units used to encode solutions, the operator must extract *"relevant"* information from these solutions and recombine it (with independence of whether solutions are encoded on the basis of these particular information pieces or not). Unfortunately, the concept of "relevance" is hardly defined in formal terms. For instance, solutions of the *NP* optimization problem MIN TRAVELING SALESMAN (MIN TSP) can be encoded as permutations or even as binary strings. However, operators working directly on these encodings such as *cycle crossover* (Oliver et al., 1987) or *uniform crossover* (Syswerda, 1989) will in general provide worse performance than operators extracting the relevant information. For the symmetric instances of the MIN TSP it has been shown that the preservation of some features, in particular *common edges* is indeed a good strategy (Mathias and Whitley, 1992), and this lead to the proposal of several *"edge recombination"* methods (Moscato, 1999).

We will refer to these relevant "pieces of information" as *features.* We note, however, that in most of the cases where the problem to be solved is intractable, these features generally correspond to *predicates computable in polynomial-time.* Back to the symmetric instance of the MIN TSP example, when a recombination operator requires to *"find all* common *edges of the* $k_{par} \geq 2$ *parents"* this can be understood as checking $k_{par} \times n(n-1)/2$ predicates (where n denotes the number of cities). Each one of them corresponds to one edge between two cities, and we return the edges for which the associate predicate returns a 'Yes' for all parents.

After having identified the relevant features (let us suppose we managed to find all features of a set of parent solutions in polynomial-time), the next and obviously important step is deciding how we can use this information. While *blind* recombination operators that randomly shuffle the set of features were more typical in the past, the addition of problem-domain knowledge to guide the process is becoming increasingly popular. The terms *hybrid* GAs and *memetic algorithms* (MAs) (Moscato, 1989) (Moscato, 1999) have been coined to denote these methods that use *smarter* reproductive operators and periods of single-agent optimization.

There exist a plethora of mechanisms to create these *smart* recombination operators, e.g., (Cotta and Troya, 1998; Radcliffe and Surry, 1994), but, up to the best of our knowledge, no complexity results for some of the decision problems involved have been reported. For instance, suppose we have a number of $m \geq 3$ tours from a relatively large population of size $Pop \gg m$. Let us also suppose that $m - 1$ of them have lengths values which are below the current population's average length value, but one has a value well above average. To strengthen the argument we can even suppose that it is actually the longest tour in the entire population. While the preservation of edges/features present in all m parents can still make some sense, we notice that the preservation of edges/features present in the best $m - 1$ parents *and not* present in the worst tour, seems also a valuable heuristic to create new solutions. Analogously, the *avoidance* of a feature present in the worst tour *and not* present in the other $m - 1$ tours is certainly another appealing heuristic.

It is clear that, while there might be other heuristics of interest for special cases, the previous example clearly depicts the existence of a more general problem: *given a set of parents, find the optimal subset of features to avoid and to preserve.* This problem already appears when we have parents that can be categorized in two different classes. A natural measure of optimality is the cardinality of the set, since we expect that m is already a small number in comparison with the size of the instance, then we only expect to make a valid inference if the number of chosen features is also small.

We think that the EC community may critique itself in having not yet defined a systematic effort to understand how to extract useful features from populations of solutions. Although some *ad-hoc* approaches work for particular problems, most recombination approaches are naive attempts to solve a more fundamental issue, that of extracting particular characteristics/features that the optimal solutions might have and, possibly more important, which features *might not* be present in them.

Related with this latter point, it must be noted that we still lack a formal framework for recombination, similar for instance to that we have for Local Search (Johnson et al., 1988; Yannakakis, 1997). In this sense, an interesting new direction for theoretical research arose after the introduction of two computational complexity classes, the PMA class (for *Polynomial Merger Algorithms problems)* and its unconstrained analogue, the uPMA class. We will define the classes PMA and uPMA by referring to three analogous algorithms to the ones that define the class of *Polynomial Local Search* problems (PLS). These definitions (specially for PMA) are particularly dependent on an algorithm called ***k-merger,*** that will help to formalize the notion of *recombination of a set of* k *given solutions,* as generally used by most MAs (as well as other population approaches). The input of a k-merger algorithm is a set S_{par} of $k \geq 2$ *feasible* solutions. They can be informally called "parent" solutions

and, if successful, the **k-merger** delivers as output *at least one* feasible solution (with some constraints). For the uPMA class the construction of the new solution is less restricted than for PMA. In general, recombination processes can be very complex with many side restrictions involving the detection, the preservation or avoidance, and the feasible combination of *features* already present in the parent solutions.

Definition (uPMA). Let x be an instance of an optimization problem P. With $\mathcal{M}_P(S_{par}, x)$ $\mathcal{M}_P(S_{par}, x)$ we denote the set of all possible outputs (i.e., feasible solutions) that the ***k-merger*** algorithm can give if it receives as input the pair (S_{par}, x) for problem P.

A recombination problem $P/\mathcal{M}$ belongs to uPMA if there exist three polytime algorithms ***p-starter***, ***p′-evaluator***, and ***k-merger*** (where p, p' and k are integer numbers such that $p' \geq p \geq k \geq 2$) that satisfy the following properties:

- Given an input x (formally a string $\in \{0,1\}^*$), the ***p-starter*** determines whether x is an instance of problem P and in this case produces a set of p different feasible solutions $\{y_1, y_2, \ldots, y_p\}$.

- Given an instance x of P and an input (formally a string $\in \{0,1\}^*$), the ***p′-evaluator*** determines whether this input represents a set of feasible solutions, i.e. $\{y_1, y_2, \ldots, y_{p'}\}$ and in that case it computes the value of the objective function associated to each one of them, i.e. $m_P(y_j, x), \forall j = 1, \ldots, p'$.

- Given an instance x of P and a set of k feasible solutions S_{par}, the ***k-merger*** determines whether the set S_{par} is a ***k-merger*** *optimum,* and, if it is not, it outputs at least one feasible solution $y' \in \mathcal{M}_P(S_{par}, x)$ with strictly better value of m_P (i.e. $m_P(y', x) < max\{m_P(y_1, x), m_P(y_2, x), \ldots, m_P(y_k, x)\}$ for a minimization problem, and $m_P(y', x) > min\{m_P(y_1, x), m_P(y_2, x), \ldots, m_P(y_k, x)\}$ for a maximization problem).

Analogously, the PMA class is more restricted since it embodies a particular type of recombination. For uPMA the type of recombination is implicit in the way the group neighborhood $\mathcal{M}$ is defined. However, the definition for PMA is still general enough to encompasses most of the recombination procedures used in practical population-based approaches.

Definition (PMA). A recombination problem $P/\mathcal{M}$ belongs to PMA if there exist three polynomial-time algorithms ***p-starter***, ***p′-evaluator***, and ***k-merger*** (where p, p' and k are integer numbers such that $p' \geq p \geq k \geq 2$),

such that the ***p-starter*** and ***p′-evaluator*** satisfy the same properties required by the uPMA class but the ***k-merger*** is constrained to be of a particular type, i.e.:

- Given an instance x of P and a set of k feasible solutions S_{par}, the ***k-merger*** determines whether the set S_{par} is a ***k-merger*** *optimum,* and, if it is not, it does the following:
  - For each $y \in S_{par}$, it solves n_1 polynomial-time decision problems $\{\Pi_1(y), \Pi_2(y), \ldots, \Pi_{n_1}(y)\}$. Let D be a matrix of $k \times n_1$ Boolean coefficients formed by the output of all these decision problems, i.e. $D_{i,j} = \Pi_j(y_i)$.
  - It creates a set of n_2 *constraints C,* such that C can be partitioned in two subsets, i.e. $C = C_{in} \cup C_{out}$. Each constraint $c \in C$ is represented by a predicate π_c such that its associated decision problem $\Pi_c(y)$ can be solved in polynomial-time for every feasible solution y. Any predicate π_c is a polynomial-time computable function that has as input the Boolean matrix D and the instance x. It is required that *at least one* predicate π_c^* to be a non-constant function of *at least two different elements of* S_{par}.
  - It outputs at least one *offspring,* i.e. another feasible solution $y' \in \mathcal{M}_P(S_{par}, x)$ with strictly better value of m_P, (i.e. $m_P(y', x) <$ *max* $\{m_P(y_1, x), m_P(y_2, x), \ldots, m_P(y_k, x)\}$ for a minimization problem, and $m_P(y', x) > min\ \{m_P(y_1, x), m_P(y_2, x), \ldots, m_P(y_k, x)\}$ for a maximization problem) subject to

$$\max_{y'}[(\sum_{(c \in C_{in}) \wedge \Pi_c(y')} w_c) - (\sum_{(c \in C_{out}) \wedge \Pi_c(y')} w_c)] \qquad (3.1)$$

    where w_c is an integer weight associated to constraint c.

Conducting research to identify problems, and their associated recombination procedures, such that membership, in either PMA or uPMA, can be proved is a definitely important task. It is also hoped that after some initial attempts on challenging problems completeness and reductions for the classes can be properly defined.

We should also note that the definition are such that they would naturally give several new interesting parameterized complexity problems. So, while proving NP-hardness is a good start, we hope that the research focus should be directed towards proving many problems to be fixed-parameter tractable. In essence, that would lead toward developing "optimal" recombination operators, that while exponential on the parameters, can be polynomial on the instance size.

4. CHALLENGE #3: USING MODAL LOGIC AND LOGIC PROGRAMMING METHODS TO GUIDE THE SEARCH

Looking ahead one of the possible directions that EC can take, after checking the current trends, it is then reasonable to affirm that increasingly more complex schemes evolving solutions, agents, as well as representations, will soon be implemented. The way they would handle information (actually it is a "distributed" information for it is carried by a population of solutions, which can be transmitted, recombined, and analyzed) have some points in common with *Blackboard Systems* (Englemore and Morgan, 1988). This has been recognized in the past yet it is conspicuously hardly being mentioned in the current metaheuristics literature. We are proposing to call these new methods as *Belief Search* and to show they can work in an EC setting, we will resort to two illustrative examples. We will assume that the formula $\mathbf{B}_a^i\phi$ has the following meaning *"agent i believes with strength (at least) a that ϕ is true"*, such that the strength values a are restricted to be rational numbers in [0,1]. Let us also suppose we accept as an axiom that from ψ being true we can deduce $\mathbf{B}_1^i\psi$ for all i. Now let us suppose that our agents are trying to solve a MIN TSP and that the particular instance being considered is Euclidean and two-dimensional. Let ϕ_k represent the proposition *"edge e_k is present in the optimum tour"* and let $\chi_{k,l}$ be true if edges e_k and e_l cross each other, and false otherwise. It can be proved (a "folk theorem") that for such particular type of TSP instances (a form of problem-domain, or better, instance-domain knowledge) *"if edges e_k and e_l cross each other, then e_k and e_l can not both be present in the optimal tour"*. Then we can assume that this is known by all agents, and by the previous axiom we can deduce that agent 2 now believes $\mathbf{B}_1^2(\chi_{k,l} \rightarrow \neg(\phi_k \wedge \phi_l))$. Now let us suppose that agent 1 believes, with strength 0.4, that *"either edge e_k or e_l, but not both, is present in the optimal tour"*. We will represent this as $\mathbf{B}_{0.4}^1\phi_{k,l}$. We will not enter into the discussion of how that agent reached that belief and we take it as a fact. Now let us suppose that another agent believes, at a level 0.7 that $\chi_{k,l} \rightarrow \phi_{k,l}$, then we write $\mathbf{B}_{0.7}^3(\chi_{k,l} \rightarrow \phi_{k,l})$. This is curious, since this kind of assumption confuses our common sense. In general we do not see any relationship between the fact that two edges cross and that we can deduce that as a consequence one of them should be present in the optimum tour. We can take this as a fact, as if a "co-evolving" algorithm, is generating these predicates to guide the search. However, note that agent 3 believes in this relationship (at a 0.7 level) for *a particular* pair of edges e_k and e_l. Now, what can we say about the *distributed belief* of this group of three agents ? How can we recombine this information ? At this point we need to introduce a logic to recombine belief information. Discussions on which particular type of logic to guide heuristic search process is a much more elegant

and useful method than keeping on discussing values of parameters based on trial-and-error experimental tests. It may also lead to improved convergence in *Estimation-of-Distributions* (EDA) metaheuristics (Larrañaga and Lozano, 2001).

According to one possible selection for such a logic, just picked to exemplify the discussion, we can use $\mathbf{PL}_n^{\otimes}$, a multi-agent epistemic logic recently introduced by Boldrin and Saffiotti, the opinions shared by a set of n different agents can be *recombined* in a distributed belief. Using $\mathbf{PL}_n^{\otimes}$ we can deduce $\mathbf{D}_{0.82}\phi_{k,l}$. The distributed belief about proposition $\phi_{k,l}$ is then stronger than any individual belief about it, and is even stronger than what you would get if any agent would believe the three facts. We offer now two examples on its application.

4.1 EXAMPLE 1

In $\mathbf{PL}_n^{\otimes}$ we have the following axioms and inference rules, where ϕ and ψ range over formulas of $\mathcal{L}$; a, b and c over rational numbers in [0,1]; and $i = 1, \ldots, n$. The five axioms are:

(A0)	Axiom 0	All propositional tautologies	
(A1)	Axiom 1	$\mathbf{B}_0^i \perp$	
(A2)	Axiom 2	$\mathbf{B}_a^i(\phi \rightarrow \psi) \rightarrow (\mathbf{B}_b^i\phi \rightarrow \mathbf{B}_c^i\psi)$	$c \leq \min\{a, b\}$
(A3)	Axiom 3	$\mathbf{D}_a(\phi \rightarrow \psi) \rightarrow (\mathbf{D}_b\phi \rightarrow \mathbf{D}_c\psi)$	$c \leq \min\{a, b\}$
(A4)	Axiom 4	$(\wedge_{i=1}^n \mathbf{B}_{a_i}^i\phi) \rightarrow \mathbf{D}_c\phi$	$c = \oplus_{i=1}^n a_i$

The three inference rules are :

(MP)	Modus Ponens	from ϕ and $\phi \rightarrow \psi$ deduce ψ
(NEC)	Necessitation	from ϕ deduce $\mathbf{B}_1^i\phi$
(US)	Uniform substitutions	

A formula ϕ is said to be a theorem of $\mathbf{PL}_n^{\otimes}$ written $\vdash \phi$, if ϕ is obtained from **A0-A4** by a finite number of applications of **MP**, **NEC** and uniform substitutions. Then, if $\Gamma \subseteq \mathcal{L}$ we will write $\Gamma \vdash \phi$ to mean $\vdash (\wedge_{\phi \in \Gamma}) \rightarrow \phi$.

Proposition: Given $\Gamma = \{(a), (b), (c)\}$ (see below) then $\Gamma \vdash (\mathbf{D}_{0.82}\phi_{k,l} \wedge \mathbf{D}_{0.82}(\chi_{k,l} \rightarrow \phi_{k,l}))$.

(a)$\mathbf{B}_{0.4}^1\phi_{k,l}$,
(b)$\chi_{k,l}$,
(c)$\mathbf{B}_{0.7}^3(\chi_{k,l} \rightarrow \phi_{k,l})$.

Proof:

(1)$\phi_{k,l} \rightarrow (\chi_{k,l} \rightarrow \phi_{k,l})$	**A0**
(2)$\mathbf{B}_1^1(\phi_{k,l} \rightarrow (\chi_{k,l} \rightarrow \phi_{k,l}))$	**NEC**, 1
(3)$\mathbf{B}_1^1(\phi_{k,l} \rightarrow (\chi_{k,l} \rightarrow \phi_{k,l}))$ $\rightarrow (\mathbf{B}_{0.4}^1\phi_{k,l} \rightarrow \mathbf{B}_{0.4}^1(\chi_{k,l} \rightarrow \phi_{k,l}))$	**A2, US**
(4)$\mathbf{B}_{0.4}^1\phi_{k,l} \rightarrow \mathbf{B}_{0.4}^1(\chi_{k,l} \rightarrow \phi_{k,l})$	**MP**, 2, 3
(5)$\mathbf{B}_{0.4}^1(\chi_{k,l} \rightarrow \phi_{k,l})$	**MP**, (a), 4
(6)$\perp \rightarrow (\chi_{k,l} \rightarrow \phi_{k,l})$	**A0**
(7)$\mathbf{B}_1^2(\perp \rightarrow (\chi_{k,l} \rightarrow \phi_{k,l}))$	**NEC**, 6
(8)$\mathbf{B}_0^2 \perp$	**A1**
(9)$\mathbf{B}_1^2(\perp \rightarrow (\chi_{k,l} \rightarrow \phi_{k,l}))$ $\rightarrow (\mathbf{B}_0^2 \perp \rightarrow \mathbf{B}_0^2(\chi_{k,l} \rightarrow \phi_{k,l}))$	**A2, US**
(10)$\mathbf{B}_0^2 \perp \rightarrow \mathbf{B}_0^2(\chi_{k,l} \rightarrow \phi_{k,l})$	**MP**, 7, 9
(11)$\mathbf{B}_0^2(\chi_{k,l} \rightarrow \phi_{k,l})$	**MP**, 8, 10
(12)$\mathbf{B}_0^2(\chi_{k,l} \rightarrow \phi_{k,l}) \rightarrow (\mathbf{B}_1^2\chi_{k,l} \rightarrow \mathbf{B}_0^2\phi_{k,l})$	**A2, US**
(13)$\mathbf{B}_1^2\chi_{k,l} \rightarrow \mathbf{B}_0^2\phi_{k,l}$,	**MP**, 11, 12
(14)$\mathbf{B}_1^2\chi_{k,l}$	**NEC**, (b)
(15)$\mathbf{B}_0^2\phi_{k,l}$	**MP**, 14, 13
(16)$\mathbf{B}_1^3\chi_{k,l}$	**NEC**, (b)
(17)$\mathbf{B}_{0.7}^3(\chi_{k,l} \rightarrow \phi_{k,l}) \rightarrow (\mathbf{B}_1^3\chi_{k,l} \rightarrow \mathbf{B}_{0.7}^3\phi_{k,l})$	**A2, US**
(18)$\mathbf{B}_1^3\chi_{k,l} \rightarrow \mathbf{B}_{0.7}^3\phi_{k,l}$	**MP**, (c), 17
(19)$\mathbf{B}_{0.7}^3\phi_{k,l}$	**MP**, (c), 18
(20)$\mathbf{D}_{0.82}(\chi_{k,l} \rightarrow \phi_{k,l})$	**A4**, (5), (11), (c)
(21)$\mathbf{D}_{0.82}\phi_{k,l}$	**A4**, (a), 15, 19

We can leave to the reader the task of checking this example following section 3.3. of (Boldrin and Saffiotti, 1999).

Another interesting exercise is the following: let ϕ_k (ϕ_l) be the predicate *"edge e_k (respectively, e_l) is present in the optimal tour"*. The task is then to deduce according to $\mathbf{PL}_n^{\otimes}$ what can be distributively believed about individual edges e_k and e_l if in addition to the three previous agents there are also two other agents, such that $\mathbf{B}_{0.54}^4\phi_k, \mathbf{B}_{0.27}^4\phi_l, \mathbf{B}_{0.63}^5\phi_k, \mathbf{B}_{0.15}^5\phi_l$.

4.2 EXAMPLE 2

The following are theorems of $\mathbf{PL}_n^{\otimes}$:

(TH1) $(\mathbf{B}_a^i\phi \wedge \mathbf{B}_b^i\psi) \rightarrow \mathbf{B}_c^i(\phi \vee \psi) \quad c = \max\{a, b\}$
(TH2) $(\mathbf{B}_a^i\phi \wedge \mathbf{B}_b^i\psi) \rightarrow \mathbf{B}_c^i(\phi \wedge \psi) \quad c = \min\{a, b\}$

If the agents are trying to solve an Euclidean, 2-dimensional instance of the TSP, then we also have the *instance-dependent axioms* or **IDA**s.

$$\textbf{IDA1} \quad \chi_{k,l} \rightarrow \neg(\phi_k \wedge \phi_l)$$

In addition, we also know that:

$$\textbf{IDA2} \quad \phi_{k,l} \equiv ((\phi_k \rightarrow \neg\phi_l) \wedge (\neg\phi_k \rightarrow \phi_l))$$

Proposition: *Given* $\Gamma' = \{(a), \ldots, (g)\}$ *(see below) then* $\Gamma' \vdash \boldsymbol{D}_{0.96934}\phi_{k,l}$.

(a)$\mathbf{B}^1_{0.4}\phi_{k,l}$
(b)$\chi_{k,l}$
(c)$\mathbf{B}^3_{0.7}(\chi_{k,l} \rightarrow \phi_{k,l})$
(d)$\mathbf{B}^4_{0.54}\phi_k$
(e)$\mathbf{B}^4_{0.27}\phi_l$
(f)$\mathbf{B}^5_{0.63}\phi_k$
(g)$\mathbf{B}^5_{0.15}\phi_l$

Proof:

(1)$\mathbf{B}^4_1(\chi_{k,l} \rightarrow \neg(\phi_k \wedge \phi_l))$	**IDA1, NEC**
(2)$\mathbf{B}^4_1(\chi_{k,l} \rightarrow (\phi_k \rightarrow \neg\phi_l))$	**A0**
(3)$\mathbf{B}^4_1\chi_{k,l}$	(b), **NEC**
(4)$\mathbf{B}^4_1(\chi_{k,l} \rightarrow (\phi_k \rightarrow \neg\phi_l)) \rightarrow (\mathbf{B}^4_1\chi_{k,l} \rightarrow \mathbf{B}^4_1(\phi_k \rightarrow \neg\phi_l))$	**A2, US**
(5)$\mathbf{B}^4_1\chi_{k,l} \rightarrow \mathbf{B}^4_1(\phi_k \rightarrow \neg\phi_l)$	**MP**, 2, 4
(6)$\mathbf{B}^4_1(\phi_k \rightarrow \neg\phi_l)$	**MP**, 3, 5
(7)$\mathbf{B}^4_1(\phi_k \rightarrow \neg\phi_l) \rightarrow (\mathbf{B}^4_{0.54}\phi_k \rightarrow \mathbf{B}^4_{0.54}(\neg\phi_k))$	**A2**
(8)$\mathbf{B}^4_{0.54}\phi_k \rightarrow \mathbf{B}^4_{0.54}(\neg\phi_l)$	**MP**, 6, 7
(9)$\mathbf{B}^4_{0.54}(\neg\phi_l)$	**MP**, (d), 8

analogously we can deduce:

(18)$\mathbf{B}^4_{0.27}(\neg\phi_k)$	**MP**, (e), 17
(27)$\mathbf{B}^5_{0.63}(\neg\phi_l)$	**MP**, (f), 26
(36)$\mathbf{B}^5_{0.15}(\neg\phi_k)$	**MP**, (g), 35

and now we will use one of the theorems:

(37)$(\mathbf{B}^4_{0.54}\phi_k \wedge \mathbf{B}^4_{0.27}\phi_l) \rightarrow \mathbf{B}^4_{0.54}(\phi_k \vee \phi_l)$	**TH1, US**
(38)$\mathbf{B}^4_{0.54}(\phi_k \vee \phi_l)$	**MP**, (d), (e), 37
(39)$(\mathbf{B}^5_{0.63}\phi_k \wedge \mathbf{B}^5_{0.15}\phi_l) \rightarrow \mathbf{B}^5_{0.63}(\phi_k \vee \phi_l)$	**TH1, US**
(40)$\mathbf{B}^5_{0.63}(\phi_k \vee \phi_l)$	**MP**, (f), (g), (39)
(41)$(\mathbf{B}^4_{0.54}(\neg\phi_l) \wedge \mathbf{B}^4_{0.27}(\neg\phi_k)) \rightarrow \mathbf{B}^4_{0.54}(\neg\phi_k \vee \neg\phi_l)$	**TH1, US**
(42)$\mathbf{B}^4_{0.54}(\neg\phi_k \vee \neg\phi_l)$	**MP**, 9, 18, 41
(43)$(\mathbf{B}^5_{0.63}(\neg\phi_l) \wedge \mathbf{B}^5_{0.15}(\neg\phi_k)) \rightarrow \mathbf{B}^5_{0.63}(\neg\phi_k \vee \neg\phi_l)$	**TH1, US**
(44)$\mathbf{B}^5_{0.63}(\neg\phi_k \vee \neg\phi_l)$	**MP**, 27, 36, 43

and now we combine the information using the other theorem:

(45)$(\mathbf{B}^4_{0.54}(\phi_k \vee \phi_l)) \wedge (\mathbf{B}^4_{0.54}(\neg\phi_k \vee \neg\phi_l))$	
$\rightarrow \mathbf{B}^4_{0.54}((\phi_k \vee \phi_l) \wedge (\neg\phi_k \vee \neg\phi_l))$	**TH2, US**
(46)$\mathbf{B}^4_{0.54}((\phi_k \vee \phi_l) \wedge (\neg\phi_k \vee \neg\phi_l))$	**MP**, 38, 42, 45
(47)$\mathbf{B}^4_{0.54}\phi_{k,l}$	**A0**
(48)$(\mathbf{B}^5_{0.63}(\phi_k \vee \phi_l)) \wedge \mathbf{B}^5_{0.63}(\neg\phi_k \vee \neg\phi_l)$	
$\rightarrow \mathbf{B}^5_{0.63}((\phi_k \vee \phi_l) \wedge (\neg\phi_k \vee \neg\phi_l))$	**TH2, US**
(49)$\mathbf{B}^5_{0.63}((\phi_k \vee \phi_l) \wedge (\neg\phi_k \vee \neg\phi_l))$	**MP**, 40, 44, 48
(50)$\mathbf{B}^5_{0.63}\phi_{k,l}$	**A0**
(51)$\mathbf{B}^3_{0.7}(\chi_{k,l} \rightarrow \phi_{k,l}) \rightarrow (\mathbf{B}^3_1\chi_{k,l} \rightarrow \mathbf{B}^3_{0.7}\phi_{k,l})$	**A2, US**
(52)$\mathbf{B}^3_1\chi_{k,l} \rightarrow \mathbf{B}^3_{0.7}\phi_{k,l}$	**MP**, (c), 51
(53)$\mathbf{B}^3_1\chi_{k,l}$	**NEC**
(54)$\mathbf{B}^3_{0.7}\phi_{k,l}$	**MP**, 53, 52
(55)$\mathbf{B}^2_1(\bot \rightarrow \phi_{k,l})$	**A0, NEC**
(56)$\mathbf{B}^2_1(\bot \rightarrow \phi_{k,l}) \rightarrow (\mathbf{B}^2_0 \bot \rightarrow \mathbf{B}^2_0\phi_{k,l})$	**A2, US**
(57)$\mathbf{B}^2_0 \bot \rightarrow \mathbf{B}^2_0\phi_{k,l}$	**MP**, 55, 56
(58)$\mathbf{B}^2_0 \bot$	**A1**
(59)$\mathbf{B}^2_0\phi_{k,l}$	**MP**, 58, 57
(60)$\mathbf{D}_{0.96934}\phi_{k,l}$	**A4**, (a), 47, 50, 54, 59

We note that, by computing the distributed belief of the set of solutions in an EA (or agents in an MA), *it is possible to use this information to bias constructive algorithms.* This said, a Belief-Search-based EA can also benefit from constructive heuristics already available in the literature. In addition, *exact search methods can prioritize some pending decisions based on the information that is distributively believed.* This may also allow parallel search by a set of agents, allowing the agents to have many alternatives instead of the *depth-first* or *best-first* guiding procedures generally used.

By no means we affirm that $\mathbf{PL}^{\otimes}_n$ *is* the definitive logic that should be used to guide EAs with Belief Search. We mention this, since $\mathbf{PL}^{\otimes}_n$ is related to multimodal logics of partial belief and it may be the case that some other forms for connectives are more appropriate than the T-norm proposed for merging information. However, $\mathbf{PL}^{\otimes}_n$ already embodies very interesting features that we would like to highlight and we have not noticed in other logics of belief. First, it allows *nested epistemic reasoning,* i.e., an expression like $\mathbf{B}^i_a\mathbf{D}_b\phi$ can be interpreted as *agent i believes at level a that ϕ is distributively believed at level B".* This is very interesting since some *ad-hoc* heuristics for generalizing recombination operators, like the *rebel, conciliator,* and *obsequent* behaviors (Berretta and Moscato, 1999), can be interpreted in terms of an underlying nested epistemic reasoning. Second, the negation is typically modal conveying the concept of absence of information. As remarked by Boldrin and Saffiotti, this contrast with the algebraic approach of other logics in which negation represents positive information on some "orthogonal" formula. Again, this is best illustrated with the MIN TSP as our favorite example. A strong belief on a subset of $O(n)$ edges to be in the optimal solution *does not necessary mean* that the $O(n^2)$ remaining edges might not be in the optimal solution. Modal

logic seems to have an interesting role in this respect. Finally, according to Boldrin and Saffiotti, $\mathbf{PL}_n^{\otimes}$ can be extended to also include a set of epistemic operators D^G, with G being a subset of the agents. The intended meaning of this is that they will combine the distributed belief of subsets of the agents.

5. CHALLENGE #4: LEARNING FROM OTHER METAHEURISTICS AND OTHER OPEN CHALLENGES

Evolutionary Computation metaheuristics are far from being the only method of choice to perform heuristic search. We have shown in the introduction how *Simulated Annealing* (SA) and *Tabu Search* (TS) are among the most popular "'single-agent" stochastic optimization methods. The key of the success is the simplicity of their implementation and the fact that for many optimization problems (and the problem instances under study) it is relatively easy to get very good solutions. One of the authors of this chapter, back in 1989, introduced the denomination of 'memetic algorithms' (MAs) as a paradigm aimed to liberate population-search methods from the current biologically motivated metaphors at that time. Several ideas were introduced, the use of single-agent metaheuristics for individual search optimization steps, the use of different neighborhoods for the different agents, the study of correlation of local optima, etc. After more than a decade from that work, we see that several ideas have been upraised up to the point of constituting new metaheuristics, like *variable neighborhood search* (VNS) (Hansen and Mladenović, 2001). We can quote from (Moscato, 1989):

> *Another advantage that can be exploited is that the most powerful computers in the network can be doing the most time-consuming heuristics, while others are using a different heuristics. The program to do local search in each individual can be different. This enriches the whole, since what is a* local minima *for one of the computers is not a local minima for another in the network. Different heuristics may be working fine due to different reasons. The collective use of them would improve the final output. In a distributed implementation we can think in a division of jobs, dividing the kind of moves performed in each computing individual. It leads to an interesting concept, where instead of dividing the physical problem (assignment of cities/cells to processors) we divide the set of possible moves. This set is selected among the most efficient moves for the problem.*

and also,

> *Is this the ultimate solution for the problems that the search involves ? Is it wise to use a set of many different moves, to continue adding different moves* ad infinitum *? Certainly not. Effective moves are those that, on the average, create a new configuration with similar values of the objective value, reflecting the efficient use of the correlation between the configurations given by the representation.*

Despite the clear coincidences present in these early discussions, and contrarily to what the reader might suspect, we are not interested in claiming that the VNS ideas were already contained in MAs. On the contrary, we view the systematic development of particular strategies as a healthy sign. If a simpler

metaheuristic (SA, TS, VNS, GRASP, etc.) performs the same as a more complex method (GAs, MAs, Ant Colonies, etc.) we should either resort to the simpler method, or to the one that has less free parameters, or to the one that is easier to implement. On the other hand, such a fact challenges us to adapt the more complex methodology to beat a simpler heuristic, if that is possible at all. What we do not consider as a healthy sign, however, are the attempts to encapsulate some metaheuristics on stretched confinements. For instance, a MA is not just a "hybrid" GA, or a "parallel GA", or a GA in which all solutions are local optima. Actually this latter strategy was not part of the proposed definitions, since already the MAs in 1988 and 1989 were using SA or stochastic methods and the solutions were far from being locally optimal at the time of recombination. Not every method that uses a population and a recombination operator is a GA, not every hybrid GA is a MA. An "ant colony" metaheuristic (Dorigo et al., 1996) is indeed a new idea, but when the "ants" use local search, the resulting algorithm exhibits a strong resemblance to an MA.

We think that there are several "learned lessons" from work in other metaheuristics. For instance, TS decides to accept another new configuration (whether a feasible solution or not) without restriction to the relative objective function value of the two solutions. This has lead to good performance in some configuration spaces where evolutionary methods and Simulated Annealing perform poorly. A classical example of this situation is the MIN NUMBER PARTITIONING problem (Berretta and Moscato, 1999). In addition, we have also identified some problems with evolutionary search methods in instances of the TSP in which the entries of the distance matrix have a large number of decimal digits. We believe that there is an inherent problem to be solved, for evolutionary methods to deal with fitness functions that have so many decimal digits. Traditional rank-based or fitness-based selection schemes to keep new solutions in the current population fail. It would be then reasonable to investigate whether some ideas from basic TS mechanisms could be adapted to allow less stringent selection approaches.

Problems like *STRIPS planning* (Bylander, 1994) or the less known *Sokoban* (Culberson, 1999) can provide good test-beds for the performance of EC methods in problems of other complexity classes. Unfortunately, although there are exceptions (Westerberg and Levine, 2001) they are seldom addressed. Other related challenges have been described in (Selman et al., 1997). Multi-objective optimization is another interesting field full of new challenges where several metaheuristics are being evaluated (Coello, 1999).

In (Selman et al., 1997) we can read in their second challenge:

> *Minsky (1967) was foundational in establishing the theory of computation, but after Hartmanis (1971) there has been a fixation with asymptotic complexity. In reality lots of problems we face in building real AI systems do not get out of hand in terms of the size of problems for individual modules–in particular with*

behavior-based systems most of the submodules need only deal with bounded size problems.

We have recently initiated work in an area which we have tentatively called *Evolutionary Analysis of Algorithms* (Cotta and Moscato, 2003). This approach deals with the problem of finding, for a fixed-size, the worst-case instance for a particular algorithm; there are problems that by their intrinsic nature have been defined with a natural upper-bound on the instance size. Then the real challenging problem is to find new methods allowing "co-evolution" between the tasks of designing a better algorithm and the worst-case instance. This hopefully will lead to more robust methodology for algorithms development.

6. CONCLUSIONS

By looking back at the development of Evolutionary Computation in the previous decades, we can say that it is a healthy field. The number of researchers and published articles is steadily growing at a superlinear rate (Alander, 1994). So is also the number of successful applications of these techniques. Hence, the field is now well grounded and mature enough to endeavor the challenging task of understanding how, when and why these techniques work or should be deployed on an specific problem.

We have proposed a number of challenges whose successful resolution will –in our opinion– provide major boosts for the vigorous development of the field. Obviously, these challenges are only a part of a bigger picture, as the reader will verify by reading other essays in this collection. They nevertheless reflect our view of the area, a view in which the lack of a solid theoretical corpus as well as insufficient connections with other areas of metaheuristic optimization (let alone with other areas of Theoretical Computer Science) constitute a Damocles' sword whose existence we have to face (and indeed solve).

It is up to us, EC researchers, to determine whether future EC practitioners will regard the field as a collection of elaborate recipes to be adapted to one's taste, or as a cooking book from which to learn how to cook the dish he/she likes. Admittedly, this is an ambitious objective. It is also true that some of the most optimistic perspectives about the capabilities of the paradigm a decade ago were dismissed by theoretical results such as Hart & Belew's hardness results (Hart and Belew, 1991) and Wolpert & Macready's No Free Lunch Theorem (Wolpert and Macready, 1997), so in principle, this could be the case for some of these challenges. However, we have to consider that these past experiences did not compromise the future of EC; on the contrary, they allowed redirecting efforts in more fruitful ways. Theoretical results cannot thus be negative, for they represent the underlying truth about the paradigm. It is to this underlying ground upon which we have to settle and adapt. Whatever the

outcome of the challenges we have depicted in this essay, this should be the philosophy with which we have to react.

REFERENCES

Alander, J. (1994). Indexed bibliography of genetic algorithms and neural networks. Technical Report 94-1-NN, University of Vaasa, Department of Information Technology and Production Economics.

Bäck, T., Fogel, D., and Michalewicz, Z. (1997). *Handbook of Evolutionary Computation.* Oxford University Press, New York NY.

Berretta, R. and Moscato, P. (1999). The number partitioning problem: An open challenge for evolutionary computation ? In Corne, D., Dorigo, M., and Glover, F., editors, *New Ideas in Optimization,* pages 261–278. McGraw-Hill, Maidenhead, Berkshire, England, UK.

Boldrin, L. and Saffiotti, A. (1999). A modal logic for merging partial belief of multiple reasoners. *Journal ofLogic and Computation,* 9(1):81–103. Online at `http://www.aass.oru.se/~asaffio/`.

Bylander, T. (1994). The computational complexity of propositional STRIPS planning. *Artificial Intelligence,* 69(1-2): 165–204.

Chen, J., Kanj, I., and Jia, W. (1999). Vertex cover: further observations and further improvements. In *Proceedings of the* 25^{th} *International Workshop on Graph-Theoretic Concepts in Computer Science,* number 1665 in Lecture Notes in Computer Science, pages 313–324. Springer-Verlag.

Coello, Coello, C. A. (1999). An updated survey of evolutionary multiobjective optimization techniques: State of the art and future trends. In Angeline, P. J., Michalewicz, Z., Schoenauer, M., Yao, X., and Zalzala, A., editors, *Proceedings of the Genetic and Evolutionary Computation Conference,* volume 1, pages 3–13, Piscataway, NJ. IEEE Press.

Cotta, C. and Moscato, P. (2003). A mixed evolutionary-statistical analysis of an algorithm's complexity. *Applied Mathematics Letters,* 16(l):41–47.

Cotta, C. and Troya, J. (1998). Genetic forma recombination in permutation flowshop problems. *Evolutionary Computation,* 6(1):25–44.

Culberson, J. (1999). Sokoban is PSPACE-complete. In Lodi, E., Pagli, L., and Santoro, N., editors, *Proceedings in Informatics 4, Fun With Algorithms,* pages 65–76, Waterloo. Carleton Scientific.

Davidor, Y. (1991). Epistasis variance: A viewpoint on GA-hardness. In Rawlins, G., editor, *Foundations of Genetic Algorithms,* pages 23–35. Morgan Kaufmann.

Dorigo, M., Maniezzo, V., and Colorni, A. (1996). The ant system: Optimization by a colony of cooperating agents. *IEEE Transactions on Systems, Man, and Cybernetics-Part B,* 26(1):29–41.

Downey, R. G. and Fellows, M. R. (1995). Fixed parameter tractability and completeness I: Basic theory. *SIAM Journal of Computing,* 24:873–921.

Downey, R. G., Fellows, M. R., and Stege, U. (1999). Computational Tractability: The View From Mars. *Bulletin of the European Association for Theoretical Computer Science,* 69:73–97.

Englemore, R. and Morgan, T. (eds.) (1988). *Blackboard Systems.* Addison-Wesley.

Fellows, M. R. (2002). Parameterized Complexity: The main ideas and connections to practical computing. *Electronic Notes in Theoretical Computer Science,* 61. available at http:/www.elsevier.nl/locate/entcs/volume61.html.

Forrest, S. and Mitchell, M. (1991). What makes a problem hard for a genetic algorithm? some anomalous results and their explanation. In Belew, R. and Booker, L., editors, *Proceedings of the 4th International Conference on Genetic Algorithms,* pages 182–189, San Mateo, CA. Morgan Kaufman.

Garey, M. and Johnson, D. (1979). *Computers and Intractability: A Guide to the Theory of N P-Completeness.* W.H. Freeman and Company, San Francisco.

Gent, I. and Walsh, T. (1994). The SAT phase transition. In Cohn, A., editor, *Proceedings of 11th European Conference on Artificial Intelligence,* pages 105–109. John Wiley & Sons.

Glover, F. and Laguna, M. (1997). *Tabu Search.* Kluwer Academic Publishers, Boston, MA.

Hansen, P. and Mladenović, N. (2001). Variable neighborhood search: Principles and applications. *European Journal of Operational Research,* 130(3) : 449–467.

Hart, W. and Belew, R. K. (1991). Optimizing an arbitrary function is hard for the genetic algorithm. In Belew, R. K. and Booker, L., editors, *Proceedings of the 4th International Conference on Genetic Algorithms,* pages 190–195, San Mateo CA. Morgan Kaufmann.

Johnson, D., Papadimitriou, C. H., and Yannakakis, M. (1988). How easy is local search ? *Journal of Computers and System Sciences,* 37:79–100.

Kirkpatrick, S., Gelatt Jr., C., and Vecchi, M. (1983). Optimization by simulated annealing. *Science,* 220(4598):671–680.

Krivelevich, M. (2002). Sparse graphs usually have exponentially many optimal colorings. *Electronic Journal of Combinatorics,* 9(1):#R27.

Larrañaga, P. and Lozano, J. A., editors (2001). *Estimation of Distribution Algorithms. A New Tool for Evolutionary Computation.* Kluwer Academic Publishers, Boston.

Lewis, H. and Papadimitriou, C. H. (1998). *Elements of the Theory of Computation.* Prentice-Hall, Inc., Upper Saddle River, New Jersey.

Mathias, K. and Whitley, D. (1992). Genetic operators, the fitness landscape and the traveling salesman problem. In Männer, R. and Manderick, B., edi-

tors, *Parallel Problem Solving From Nature II,* pages 259–268, Amsterdam. Elsevier Science Publishers B.V.

Moscato, P. (1989). On Evolution, Search, Optimization, Genetic Algorithms and Martial Arts: Towards Memetic Algorithms. Technical Report Caltech Concurrent Computation Program, Report. 826, California Institute of Technology, Pasadena, California, USA.

Moscato, P. (1999). Memetic algorithms: A short introduction. In Corne, D., Dorigo, M., and Glover, F., editors, *New Ideas in Optimization,* pages 219–234. McGraw-Hill, Maidenhead, Berkshire, England, UK.

Niedermeier, R. and Rossmanith, P. (2000). A general method to speed up fixed-parameter-tractable algorithms. *Information Processing Letters,* 73: 125–129.

Oliver, I., Smith, D., and Holland, J. R. C. (1987). A study of permutation crossover operators on the traveling salesman problem. In Grefenstette, J. J., editor, *Proceedings of the* 2^{nd} *International Conference on Genetic Algorithms,* pages 224–230, Hillsdale NJ. Lawrence Erlbaum Associates.

Radcliffe, N. J. and Surry, P. D. (1994). Fitness Variance of Formae and Performance Prediction. In Whitley, D. and Vose, M. D., editors, *Proceedings of the* 3^{rd} *Workshop on Foundations of Genetic Algorithms,* pages 51–72, San Francisco. Morgan Kaufmann.

Selman, B., Kautz, H. A., and McAllester, D. A. (1997). Ten challenges in propositional reasoning and search. In *Proceedings of the* 15^{th} *International Joint Conference on Artificial Intelligence (IJCAI'97),* pages 50–54.

Syswerda, G. (1989). Uniform crossover in genetic algorithms. In Schaffer, J., editor, *Proceedings of the* 3^{rd} *International Conference on Genetic Algorithms,* pages 2–9, San Mateo, CA. Morgan Kaufmann.

Westerberg, C. H. and Levine, J. (2001). Investigation of different seeding strategies in a genetic planner. In Boers, E., Cagnoni, S., Gottlieb, J., Hart, E., Lanzi, P., Raidl, G., Smith, R., and Tijink, H., editors, *Applications of Evolutionary Computing,* volume 2037 of *Lecture Notes in Computer Science,* pages 505–514. Springer-Verlag, Berlin Heidelberg.

Wolpert, D. H. and Macready, W. G. (1997). No free lunch theorems for optimization. *IEEE Transactions on Evolutionary Computation,* l(l):67–82.

Yannakakis, M. (1997). Computational complexity. In Aarts, E. and Lenstra, J., editors, *Local Search in Combinatorial Optimization,* pages 19–55. Wiley, Chichester.

Chapter 4

OPEN PROBLEMS IN THE SPECTRAL ANALYSIS OF EVOLUTIONARY DYNAMICS

Lee Altenberg
Information and Computer Sciences,
University of Hawai'i at Manoa,
Honolulu, HI U.S.A.
altenber@hawaii.edu

Abstract The dynamics of evolution can be completely characterized by the spectra of the operators that define the dynamics, under broad classes of selection and genetic operators, in both infinite and finite populations. These classes include frequency-independent selection, uniparental inheritance, and generalized mutation. Several open questions exist regarding these spectra:

1 For a given fitness function, what genetic operators and operator intensities are optimal for finding the fittest genotype? The concept of rapid first hitting time, and analog of Sinclair's "rapidly mixing" Markov chains, is examined.

2 What is the relationship between the spectra of deterministic infinite population models, and the spectra of the Markov processes derived from them in the case of finite populations?

3 Karlin proved a fundamental relationship between selection, rates of transformation under genetic operators, and the consequent asymptotic mean fitness o the population. Developed to analyze the stability of polymorphisms in subdivided populations, the theorem has been applied to unify the reduction principle for self-adaptation, and has other applications as well. Many other problems could be solved if it were generalized to account for the interaction of different genetic operators. Can Karlin's theorem on operator intensity be extended to account for mixed genetic operators?

INTRODUCTION

A general theory for the performance and design of evolutionary algorithms has proven difficult to achieve. This difficulty sets in even before we delve into search spaces with great complexity, or algorithm operators with great

complexity. We find it in the simplest "canonical" models of evolutionary algorithms owing to their nonlinear structure and stochastic dynamics.

Nonlinearity and stochasticity can be eliminated by making a variety of simplifying assumptions—in essence, exploring a subspace on the boundaries of the general problem. Linearity is produced by assuming constant selection and uniparental transmission (i.e. where the offspring type is determined by the type of its one parent). Determinism is produced by assuming an infinite population. A model with these assumptions produces a linear dynamical system whose trajectory and attractors can be described in closed form, and decomposed in terms of its spectrum of eigenvalues and eigenvectors.

Actual evolutionary algorithms depart from this boundary in two important ways: finite populations, and recombination between two (or more) parents.

Recombination is the main innovation of genetic algorithms, aimed at allowing combinations of partial solutions to be assembled. Recombination between two parents changes the dynamics of the infinite population model from linear to quadratic. In this case, we can no longer obtain a spectrum of eigenvalues and eigenvectors; the methods of nonlinear analysis must be employed, such as characterization of fixed points and their stability, domains of attraction, and Lyapunov functions. A great deal of work has been on the dynamics of recombination and selection for models at various points on the boundary of the general problem. A recent compendium can be found in (Christiansen, 2000). For more on quadratic dynamical systems see Rabinovich et al. (1992), Arora et al. (1994). . Progress has been made in the dynamics of recombination in the absence of selection, in both infinite and finite population models, by Rabani et al. (1995) , and for simple selection, by Rabinovich et al. (1999). Numerous analyses for other models on the boundary of the general problem can be found in the evolutionary computation and population genetics literature.

Evolutionary algorithms employ finite populations of a size considerably less than the cardinality of the search space, since a primary goal of the algorithms is to locate desired elements of the search space without exhaustive search.

Finite population algorithms typically use Bernoulli sampling to generate new samples of the search space. This changes the model of the algorithm from deterministic to stochastic, a Markov chain which has a linear state transition matrix, but whose dimensions are exponentially increased beyond the number of elements in the search space. The first model of finite population dynamics was developed based on Bernoulli sampling by Wright (1931) and Fisher (1930). In the Wright-Fisher model, the number of states in the Markov chain for the finite population model is $\mathcal{O}(N^{|\mathcal{S}|})$, compared to a dimension of $|\mathcal{S}|$ for the infinite population model, where $|\mathcal{S}|$ is the number of different genotypes, and N is the population size. Hence, the dimensionality of the state space

is vastly increased in the finite population model over the infinite population model.

This comparison can be made more concrete by describing the difference in terms of points in the $|\mathcal{S}| - 1$ dimensional simplex. In the infinite population model, the system state is represented as a single point in the simplex which moves deterministically one generation to the next. In the finite population model, the state is represented as a probability distribution over a cloud points in the simplex, restricted to the lattice of coordinates $\{\mathbf{x} : N x_i \in \{0, 1, \ldots, N\}, \sum_{i=1}^{n} x_i = 1\}$. The distribution of the cloud of points is what changes every generation.

Because the uniparental, infinite population model has a complete solution, in terms of the spectrum of the linear operators, it presents the logical starting point to try to understand a number of unanswered questions in the design and dynamics of evolutionary algorithms. So it is the uniparental, infinite population model that I begin with. There are three primary open questions I want to discuss:

1 What is the optimal transmission matrix for finding global optima of a search space?

2 What is the relationship between the spectrum of the infinite population model and the spectrum of the finite population model?

3 Can a key theorem of Karlin on the effects of operator intensity be generalized?

THE CANONICAL MODEL

The 'canonical' model I shall be referring to throughout is the model of an infinite population evolving with discrete, non-overlapping generations, under constant fitness coefficients and generalized uniparental transmission. Let x be the n-dimensional vector of frequencies of different types in the population, so $x_i \geq 0$, and $\sum_{i=1}^{n} x_i = 1$, which is to say that $\mathbf{x} \in \Delta_n$, the $n-1$-dimensional simplex. Then the recursion on x is:

$$\mathbf{x}' = \frac{1}{\overline{w}} T W \mathbf{x}, \tag{4.1}$$

where $\mathbf{x}'$ is the vector of frequencies in the next time step; W is the diagonal matrix of fitness coefficients, $w_i \geq 0$;

$$\overline{w} = \sum_{i=1}^{n} w_i x_i$$

is the mean fitness of the population, used as a normalizer to maintain the system state as frequencies; and

$$T = \left[T_{ij}\right]_{i,j=1}^{n}$$

is the n-by-n matrix of transmission probabilities, T_{ij}, the probability that type j produces an offspring of type i, so

$$\sum_{i=1}^{n} T_{ij} = 1 \; \forall j, \;\; T_{ij} \geq 0.$$

In vector form, these identities are:

$$\mathbf{1}^{\top}\mathbf{x} = 1, \;\; \mathbf{1}^{\top}T = \mathbf{1}, \text{ and } \overline{w} = \mathbf{1}^{\top}W\mathbf{x},$$

where

$$\mathbf{1} = \begin{bmatrix} 1 \\ \vdots \\ 1 \end{bmatrix}$$

The trajectory of the system is:

$$\mathbf{x}(t) = \frac{1}{\nu(t)}(TW)^{t}\,\mathbf{x}(0), \tag{4.2}$$

where $\nu(t) = \mathbf{1}^{\top}(TW)^{t}\,\mathbf{x}(0)$ is the normalizer.

1. OPTIMAL EVOLUTIONARY DYNAMICS FOR OPTIMIZATION

For an optimization problem, we assume that an objective function $f : \mathcal{S} \rightarrow \Re+$ is defined on each element of the search space; here, I assume that the goal is to find the element with maximum objective function value. Exhaustive search or random search of such a space will require on the average $n/2$ samples to have sampled an optimum if it is unique (which will be by assumption throughout unless specified otherwise). If an algorithm can find the optimum in an average of $\epsilon n/2$ samples, for some small constant $\epsilon \ll 1$, then it is clearly doing better than "blind search".

However, evolutionary algorithms can perform much better than $\mathcal{O}(n)$. The canonical example for an "evolutionary algorithm-easy" problem is the ONEMAX problem, where the fitness increases with the number of loci that have 1 as their allelic value (Ackley, 1987). The number of samples required by a simple mutation-selection algorithm to find the global optimum in the ONEMAX problem is $\mathcal{O}(L) = \mathcal{O}(\log(n))$, where L the number of loci, $n = |\mathcal{A}|^{L}$

is the size of the search space, $\mathcal{A}$ is the set of alleles for each locus, $|\mathcal{A}|$ the cardinality of $\mathcal{A}$ ($|\mathcal{A}| = 2$ for binary strings).

So, as a performance goal, we would like the time complexity our evolutionary search to be on the order of the ONEMAX problem, taking $\mathcal{O}(\log(n))$ samples in order to find the global optimum. To be a little more lenient with the performance requirements, we can relax the condition for "EA-easy" to polylogarithmic time, meaning that it takes $\mathcal{O}(P(\log(n)))$ samples to find the optimum, where $P(\log(n))$ is a polynomial in $\log(n)$.

So, we wish to know what conditions on an evolutionary algorithm will allow it to find the global optimum in $\mathcal{O}(P(\log(n)))$ samples. To be precise, we wish to know when the distribution of the first hitting times for producing the optimal individual has a median value (since expected hitting times are often be nonconvergent) of $\mathcal{O}(P(\log(n)))$ samples.

Evolutionary algorithms often have multiple domains of attraction (at least in the metastable sense (van Nimwegen et al., 1999)), which imposes a secondary search problem: finding the initial conditions that are in the domain of attraction containing the global optimum. The multiple-attractor problem is usually described as "multimodality" of the fitness function, but it must be understood that the fitness function by itself does not determine whether the EA has multiple domains of attraction—it is only the relationship of the fitness function to the variation-producing operators that produces multiple-attractors (Altenberg, 1995).

In order to preclude this secondary search problem, the algorithm must exhibit a single, global attractor that contains the global optimum.

So, I wish to find what spectral properties give rise to these characteristics of an evolutionary algorithm:

1 **Rapid First Hitting Time:** It finds the global optimum using a number of samples that are $\mathcal{O}(P(\log(n)))$ where n, is the cardinality of the search space. I will call this the *rapid first hitting time* property.

2 **Global Attraction:** It finds the global optimum regardless of the initial samples taken, i.e. the simplex must have one global attractor containing the optimum.

Search problem that present obstacles to 1. include *long path problems,* and the *needle-in-a-haystack.* Search problem that present obstacles to 2. include *deception, rugged adaptive landscapes,* and *multimodal objective functions.*

1.1 SPECTRAL CONDITIONS FOR GLOBAL ATTRACTION

For the canonical model Eq. (4.1), the *global attraction* condition, 2. above, can be stated precisely as:

$$\lim_{t\to\infty} \frac{1}{\nu(t)}(\boldsymbol{TW})^t \mathbf{x}(0) = \boldsymbol{\pi}, \text{ and } \pi_1 > 0, \forall \mathbf{x}(0) \in \Delta_n, \qquad (4.3)$$

where we index the global optimum type as 1.

Condition (4.3) is guaranteed if and only if $\boldsymbol{T}$ is primitive (irreducible and acyclic), i.e. there is some $k \geq 0$ such that $\boldsymbol{T}^k > 0$. From the Perron-Frobenius theorem (Gantmacher, 1959), primitiveness guarantees that there be a strictly positive eigenvector $\boldsymbol{\pi}$ corresponding to the leading eigenvalue of $\boldsymbol{TW}$. This eigenvector $\boldsymbol{\pi}$, normalized so $\langle \mathbf{1}, \boldsymbol{\pi} \rangle = \sum_i \pi_i = 1$, is the global attractor, since the composition of the population converges to it regardless of the initial composition x(0).

Primitiveness in the transmission matrix corresponds to the property of ergodicity.

It should be noted that when some types have a fitness of 0, then their frequency becomes irrelevant to the dynamics, so the transmission probabilities $\{T_{ij} : j \in \mathcal{N}\}$, where $\mathcal{N} = \{i : w_i = 0\}$, are also irrelevant. Hence, primitiveness is required only for the restriction

$$\boldsymbol{T}^+ = \left[T_{ij}\right]_{i,j\notin\mathcal{N}}.$$

For simplicity, I will henceforth assume all fitnesses are positive.

It should be noted that ergodicity in the infinite population model gives us little guarantee that the system in the finite population model will exhibit a global attractor, due to the phenomenon of metastability or *broken ergodicity* (Palmer, 1982). While ergodicity in the infinite population model is necessary for ergodicity in the finite population model, it is not sufficient. The Markov chain for the finite population model must in addition be *rapidly mixing* (Sinclair, 1992) to avoid broken ergodicity, as will be discussed later.

1.2 SPECTRAL CONDITIONS FOR RAPID FIRST HITTING TIMES

What properties of $\boldsymbol{T}$ and $\boldsymbol{W}$—which here completely define the canonical evolutionary algorithm—lead to rapid first hitting times? $\boldsymbol{W}$ incorporates the map between the objective function and the fitness values, w_i, and we could certainly focus on the properties of this map. I can pose the following (without belaboring its precise details):

Open Question: *For a given transmission matrix, $\boldsymbol{T}$, what is the optimum*

selection scheme to find the global optimum with a rapid first hitting time? ◇

Here, however, since the canonical model assumes that $\boldsymbol{W}$ is fixed, we wish to consider the problem for arbitrary $\boldsymbol{W}$. This leaves only $\boldsymbol{T}$, the transmission matrix, to be explored.

We can, without loss of generality, label the unique optimal point in the search space with $i = 1$, so

$$w_1 = \max_{i=1}^{n} w_i.$$

We can trivially guarantee a hitting time of 1 by simply constructing a transmission matrix that produces the optimum by mutation:

$$\boldsymbol{T} = \begin{bmatrix} 1 & 1 & \cdots & 1 \\ 0 & 0 & \cdots & 0 \\ \vdots & & & \vdots \\ 0 & 0 & \cdots & 0 \end{bmatrix}.$$

Transmission in this case is biased to find the optimum without any help from selection. Clearly, such *a priori* knowledge does not capture the nature of the implicit knowledge that an evolutionary algorithm must contain to have rapid first hitting times (Altenberg, 1995). The essence of evolutionary search is that *transmission in the absence of selection is unable to produce adaptation or optimization.* Only when selection and transmission are combined does adaptation occur. The translation of this principle into a condition on $\boldsymbol{T}$ would require that all types evolve to equal frequency in the absence of selection, i.e.

$$\lim_{t\to\infty} (\boldsymbol{T})^t \mathbf{x}(0) = \frac{1}{n}\,\mathbf{1},\ \forall \mathbf{x}(0) \in \Delta_n. \tag{4.4}$$

Condition (4.4) for "fair" transmission implies that

1 The transmission matrix is doubly stochastic, i.e. $\boldsymbol{T}\,\mathbf{1} = \mathbf{1}$;

2 The transmission matrix is primitive, i.e. irreducible and acyclic.

So, our question about the optimal characteristics of $\boldsymbol{T}$ can be posed thus:

Open Question: *Given a fitness Junction on a points in a search space, what "fair" transmission matrix is optimal for finding the global optimum with rapid first hitting time?* ◇

A rapid first hitting time refers to the number of samples that need to be taken before finding the global optimum. But in an infinite population, an

infinite number of samples are taken each generation. So clearly, to adapt the infinite population model to the problem of rapid first hitting time, we need a proper translation.

In a finite population, with discrete, non-overlapping generations, the number of samples, s^*, until the optimum is found is:

$$s^* = N \tau \mu,$$

where N is the population size, τ is the first hitting time (in generations), and μ is the fraction of the population each generation that comprise new samples. Hence, to achieve rapid first hitting times, the population size and the first hitting time itself must each be polylogarithmic in $n = |\mathcal{S}|$, the size of the search space, since $\mathcal{O}(P(\log(n))) * \mathcal{O}(P(\log(n))) = \mathcal{O}(P(\log(n)))$.

1.3 RAPID MIXING AND RAPID FIRST HITTING TIMES

Vitanyi (2000) has investigated the problem of rapid first hitting time in the finite population model, and proposes two criteria that will ensure rapid first hitting time:

1 the second-largest eigenvalue of the matrix representing the Markov process is bounded away far enough from 1 so that the Markov chain is rapidly mixing, as defined by Sinclair (1992).

2 the stationary distribution π gives probability greater than $1/P(\log(n))$ to the set of states that contain the global optima, where $P(\log(n))$ is a polynomial in the log of the size of the search space.

The identification of the second-largest eigenvalue as a measure of the speed of convergence of the Markov chain in evolutionary dynamics goes all the way back to Wright (1931) and Fisher (1930), who solved the second-largest eigenvalue for the Markov process representing the finite population model. This eigenvalue is $\lambda_3 = 1 - 1/N$ (since $\lambda_1 = \lambda_2 = 1$), where N is haplotype population size. It gives the rate of convergence to fixation on a single haplotype due to genetic drift, and is also the rate of decrease in the frequency of heterozygotes in the population. See Ewens (1997, pp. 17, 76, 79, 82, 85-90, 105-107, and Appendix B).

Other more recent work investigating the second-largest eigenvalue includes Suzuki (1995), (Rudolph, 1997), Schmitt et al. (2001a,2001b). .

The condition defined by Sinclair (1992) to produce what he calls *rapid mixing* in a Markov chain is as follows. Sinclair lays out his concept of rapid mixing by first defining the relative pointwise distance (r.p.d.) on a Markov

process with transition matrix $\boldsymbol{P}$ as:

$$d(t,n) = \max_{i,j\in\{1,\ldots,n\}} \frac{\left|\left[\boldsymbol{P}^t\right]_{ij} - \pi_i\right|}{\pi_i},$$

where n is the cardinality of the state space for the chain. Additionally, one defines

$$\tau(\epsilon) = \min\{t \in \mathcal{Z}^+ : \; d(t',n) \leq \epsilon, \;\; \forall t' \geq t\}.$$

The Markov chain is said to be rapidly mixing if there exists a polynomial $P(\log(n), \log(1/\epsilon))$ such that:

$$\max_{\epsilon\in(0,1]} \tau(\epsilon) \leq P(\log(n), \log(1/\epsilon))$$

(Sinclair, 1992, p. 56).

The second-largest eigenvalue determines the rate at which the components of the probability distribution that are orthogonal to the limiting distribution die away. Rapid mixing concerns the rate of convergence of a Markov chain to its limiting probability distribution. The definition of fast optimization which depends on rapid mixing I call *rapid first hitting time* by analogy.

I propose a slightly different set of criteria from Vitanyi (2000) to allow rapid first hitting time to be defined in the infinite population model. We can translate the above discussion into a condition for rapid first hitting time in the deterministic model thus:

Definition: Rapid First Hitting Time. *Consider a deterministic evolutionary algorithm with a unique global optimum, which we set to be type 1, so* $w_1 > w_i$ *for all* $i \in \{2, \ldots, n\}$. *Let*

$$\tau(\epsilon) = \max_{\mathbf{x}(0)\in\Delta_n} \min\{t \in \mathcal{Z}^+ : x_1(t) \geq \epsilon\}.$$

The evolutionary algorithm is said to possess a rapid first hitting time if there exist polynomials in $\log(n)$, $P_1(\log(n))$ *and* $P_2(\log(n))$, *such that*

$$\epsilon \geq \frac{1}{P_1(\log(n))} \quad \textit{and} \quad \tau(\epsilon) \leq P_2(\log(n))). \tag{4.5}$$

For the canonical evolutionary algorithm, $\mathbf{x}(t) = \frac{1}{\nu(t)}(\boldsymbol{TW})^t\,\mathbf{x}(0)$, *this requires that for all* $\mathrm{x}(0) \in \Delta_n$, *there exist polynomials* $P_1(\log(n))$ *and* $P_2(\log(n))$ *such that:*

$$x_1(P_2(\log(n))) = \frac{1}{\nu(t)}[1\;0\cdots\;0]\;(\boldsymbol{TW})^{P_2(\log(n))}\,\mathbf{x}(0) \geq \frac{1}{P_1(\log(n))}. \tag{4.6}$$

Of course, it must be emphasized that this 'translation' carries with it no presumption that the infinite population model adequately approximates the behavior the first hitting time in the finite population model. The first hitting time is a concept that properly belongs to stochastic processes; it is a random variable. The use of the infinite population model to approximate the first hitting time has been taken before in the "takeover time" models (Goldberg and Deb, 1991), where a deterministic, infinite population model is used to approximate the time to fixation of a genotype in a finite population. *It is clear* that this approximation will be inadequate and misleading under the very circumstances in which an evolutionary algorithm is of interest, namely, when it can find the fittest elements of the search space by sampling only a fraction of the search space. This circumstances will be discussed in Section 2.1. I claim only that this use of the infinite population model may lead us to results that may be worth investigating more rigorously in the finite population model.

1.4 SOME ANALYSIS

We can assume without significant loss of generality that $\boldsymbol{TW}$ permits a Jordan canonical representation as

$$\boldsymbol{TW} = \boldsymbol{Q}\boldsymbol{\Lambda}\boldsymbol{Q}^{\top}, \tag{4.7}$$

where the matrix $\boldsymbol{Q}$ consists of columns that are the eigenvectors of $\boldsymbol{TW}$, $\boldsymbol{Q}\boldsymbol{Q}^{\top} = \boldsymbol{Q}^{\top}\boldsymbol{Q} = \boldsymbol{I}$, and $\boldsymbol{\Lambda}$ is the diagonal matrix $\boldsymbol{\Lambda}_{ii} = \lambda_i$ of the eigenvalues of $\boldsymbol{TW}$. This assumption will simplify the analysis.

The condition applies if we assume that transition probabilities are symmetric, i.e. $T_{ij} = T_{ji}$, which is typical of the mutation operators used on data structures in evolutionary computation. This is verified by noting that since any symmetric matrix $\boldsymbol{S}$ has Jordan form $\boldsymbol{S} = \boldsymbol{P}\boldsymbol{\Lambda}\boldsymbol{P}^{\top}$, so we can take $\boldsymbol{S} = \boldsymbol{W}^{1/2}\boldsymbol{T}\boldsymbol{W}^{1/2} = \boldsymbol{P}\boldsymbol{\Lambda}\boldsymbol{P}^{\top}$, hence

$$\boldsymbol{TW} = \left(\boldsymbol{W}^{-1/2}\boldsymbol{P}\right)\boldsymbol{\Lambda}\left(\boldsymbol{P}^{\top}\boldsymbol{W}^{1/2}\right).$$

We must assume here that all fitnesses are non-zero, $w_i > 0$.

With this assumption we can then represent the trajectory of the population as:

$$\mathbf{x}(t) = \frac{1}{\nu(t)}\boldsymbol{Q}\boldsymbol{\Lambda}^{t}\boldsymbol{Q}^{\top}\mathbf{x}(0).$$

We can arbitrarily permute the indices so that $\lambda_1 > \lambda_2 \geq \cdots \geq \lambda_n > -\lambda_1$, and so that $w_1 > w_2 \geq \cdots \geq w_n$. Then for Q_{ij}, i follows the order of the fitnesses, while j follows the order of the eigenvalues. In particular,

$$q_1 = c\,\boldsymbol{\pi} > 0$$

is the strictly positive leading eigenvector of $\boldsymbol{TW}$, with $c = \langle \mathbf{1}, \boldsymbol{q}_1 \rangle$ (note that by definition $\langle \boldsymbol{q}_i, \boldsymbol{q}_i \rangle = 1$). Thus:

$$\boldsymbol{TW}\boldsymbol{\pi} = \lambda_1 \boldsymbol{\pi}.$$

The trajectory of the frequency of the optimal type is:

$$\begin{aligned} x_1(t) &= \frac{1}{\nu(t)} \sum_{i=1}^{n} q_{1i}\, \lambda_i^t \left[\boldsymbol{q}_i^\top \mathbf{x}(0) \right] \\ &= \frac{\lambda_1^t}{\nu(t)} \left(q_{11} \langle \boldsymbol{q}_1, \mathbf{x}(0) \rangle + \sum_{i=2}^{n} q_{1i} \left(\frac{\lambda_i}{\lambda_1} \right)^t \langle \boldsymbol{q}_i, \mathbf{x}(0) \rangle \right). \end{aligned} \tag{4.8}$$

Further evaluation of $\nu(t)$ yields:

$$\begin{aligned} \nu(t) &= \sum_{i=1}^{n} \mathbf{1}^\top \boldsymbol{q}_i \lambda_i^t \boldsymbol{q}_i^\top \mathbf{x}(0) \\ &= \lambda_1^t \left[\sum_{i=1}^{n} \langle \mathbf{1}, \boldsymbol{q}_i \rangle \left(\frac{\lambda_i}{\lambda_1} \right)^t \langle \boldsymbol{q}_i, \mathbf{x}(0) \rangle \right] \\ &= \lambda_1^t \left[\langle \mathbf{1}, \boldsymbol{q}_1 \rangle \langle \boldsymbol{q}_1, \mathbf{x}(0) \rangle + \sum_{i=2}^{n} \langle \mathbf{1}, \boldsymbol{q}_i \rangle \left(\frac{\lambda_i}{\lambda_1} \right)^t \langle \boldsymbol{q}_i, \mathbf{x}(0) \rangle \right] \\ &= \lambda_1^t \left[c \langle \boldsymbol{q}_1,\, x(0) \rangle + \sum_{i=2}^{n} \langle \mathbf{1}, \boldsymbol{q}_i \rangle \left(\frac{\lambda_i}{\lambda_1} \right)^t \langle \boldsymbol{q}_i, \mathbf{x}(0) \rangle \right], \end{aligned}$$

using $c = \langle \mathbf{1}, \boldsymbol{q}_1 \rangle$. So we obtain:

$$x_1(t) = \frac{q_{11} \langle \boldsymbol{q}_1, \mathbf{x}(0) \rangle + \sum_{i=2}^{n} q_{1i} \left(\frac{\lambda_i}{\lambda_1} \right)^t \langle \boldsymbol{q}_i, \mathbf{x}(0) \rangle}{c \langle \boldsymbol{q}_1,\, x(0) \rangle + \sum_{i=2}^{n} \langle \mathbf{1}, \boldsymbol{q}_i \rangle \left(\frac{\lambda_i}{\lambda_1} \right)^t \langle \boldsymbol{q}_i, \mathbf{x}(0) \rangle}$$

Substituting the above into (4.6), setting $t = P_2(\log(n))$, and rearranging, we obtain the condition:

$$\begin{aligned} &[P_1(\log(n))\, q_{11} - c] \langle \boldsymbol{q}_1, \mathbf{x}(0) \rangle \geq \\ &\sum_{i=2}^{n} \left(\frac{\lambda_i}{\lambda_1} \right)^{P_2(\log(n))} [\langle \mathbf{1}, \boldsymbol{q}_i \rangle - P_1(\log(n))\, q_{1i}] \langle \boldsymbol{q}_i, \mathbf{x}(0) \rangle \end{aligned} \tag{4.9}$$

Since $\boldsymbol{q}_1 = c\,\boldsymbol{\pi}$, we substitute $P_1(\log(n))\,q_{11} - c = c\,[P_1(\log(n))\pi_1 - 1]$, and $\langle \boldsymbol{q}_1, \mathbf{x}(0)\rangle = c\langle \boldsymbol{\pi}, \mathbf{x}(0)\rangle$, to get:

$$c^2\,[P_1(\log(n))\pi_1 - 1]\langle \boldsymbol{\pi}, \mathbf{x}(0)\rangle \geq \sum_{i=2}^{n} \left(\frac{\lambda_i}{\lambda_1}\right)^{P_2(\log(n))} (\langle \mathbf{1}, \boldsymbol{q}_i\rangle - P_1(\log(n))\,q_{1i})\,\langle \boldsymbol{q}_i, \mathbf{x}(0)\rangle, \tag{4.10}$$

$\forall\, \mathbf{x}(0) \in \Delta_n$.

At this point, we take interest in the second-largest eigenvalue λ_2. Let us define

$$r = \lambda_2/\lambda_1. \tag{4.11}$$

For any $\delta > 0$, if r is small enough, then

$$\begin{aligned}\delta \;&\geq\; \left|\sum_{i=2}^{n} r^{P_2(\log(n))}\,(\langle \mathbf{1}, \boldsymbol{q}_i\rangle - P_1(\log(n))\,q_{1i})\,\langle \boldsymbol{q}_i, \mathbf{x}(0)\rangle\right| \\ &\geq \left|\sum_{i=2}^{n} \left(\frac{\lambda_i}{\lambda_1}\right)^{P_2(\log(n))} (\langle \mathbf{1}, \boldsymbol{q}_i\rangle - P_1(\log(n))\,q_{1i})\,\langle \boldsymbol{q}_i, \mathbf{x}(0)\rangle\right| \geq 0.\end{aligned}$$

In this case, condition (4.10) is met provided

$$[P_1(\log(n))\pi_1 - 1]\langle \boldsymbol{\pi}, \mathbf{x}(0)\rangle \geq \delta/c^2$$

or

$$\pi_1 \geq \frac{1 + \frac{\delta}{c^2\,\langle \boldsymbol{\pi}, \mathbf{x}(0)\rangle}}{P_1(\log(n))} > \frac{1}{P_1(\log(n))}. \tag{4.12}$$

Hence, for small enough r, the only condition for rapid first hitting time is that the frequency of the optimum at equilibrium be on the order of $P_1(\log(n))^{-1}$. We know that selection is required in order for $\pi_1 \geq \frac{1}{P_1(\log(n))}$ since the principle eigenvector of $\boldsymbol{T}$ has $\pi_1 = \frac{1}{n}$ by the fairness assumption. Thus:

Theorem 1 *If the system* $\mathbf{x}(t) = \frac{1}{\nu(t)}(\boldsymbol{TW})^t\,\mathbf{x}(0)$ *exhibits rapid first hitting time, then there exists a critical value* $\sigma* \in [0, 1)$ *such that the system* $\mathbf{x}(t) = \frac{1}{\nu(t)}(\boldsymbol{TW}^{\sigma})^t\,\mathbf{x}(0)$ *no longer exhibits rapid first hitting time for all* $\sigma \leq \sigma*$.

Characterizing the dependence of σ on $\boldsymbol{T}$ and $\boldsymbol{W}$ remains an open question.

Now, it remains to be asked, what transmission matrices $\boldsymbol{T}$ minimize $r = \frac{\lambda_2}{\lambda_1}$?

1.5 TRANSMISSION MATRICES MINIMIZING λ_2/λ_1

If we find a transmission matrix that gives $r = \lambda_2/\lambda_1 = 0$, then the only condition we require for rapid first hitting time is (4.12). The rank- 1 matrix yields $r = 0$:

$$\boldsymbol{T} = \boldsymbol{U} = \frac{1}{n}\mathbf{1}\,\mathbf{1}^{\top} = \frac{1}{n}\begin{bmatrix} 1 & 1 & \cdots & 1 \\ 1 & 1 & \cdots & 1 \\ \vdots & & & \vdots \\ 1 & 1 & \cdots & 1 \end{bmatrix}.$$

We have $\lambda_1(\boldsymbol{U}) = 1$, and $\lambda_2(\boldsymbol{U}) = \cdots = \lambda_n(\boldsymbol{U}) = 0$. When we include selection:

$$\boldsymbol{UW} = \frac{1}{n}\mathbf{1}\,\mathbf{1}^{\top}\boldsymbol{W} = \frac{1}{n}\mathbf{1}\,[w_1\ w_2 \cdots w_n] = \frac{1}{n}\begin{bmatrix} w_1 & w_2 & \cdots & w_n \\ w_1 & w_2 & \cdots & w_n \\ \vdots & & & \vdots \\ w_1 & w_2 & \cdots & w_n \end{bmatrix}$$

is also a rank-1 matrix, with eigenvalues $\lambda_1(\boldsymbol{UW}) = \frac{1}{n}\sum_{i=1}^{n} w_i$, and $\lambda_2(\boldsymbol{UW}) = \cdots = \lambda_n(\boldsymbol{UW}) = 0$.

Thus, it would appear that the rank-1 matrix would be a candidate transmission matrix to achieve rapid first hitting times. However, this hope is instantly dashed by noting that for $\boldsymbol{UW}$, $\pi_1 = 1/n$, which is not greater than $1/P_1(\log(n))$. We might ask if we can find another rank-1 matrix where $\pi_1 \geq 1/P_1(\log(n))$, but the is precluded by the condition that $\boldsymbol{T}$ be 'fair', and thus doubly-stochastic, requiring that $\pi_i = 1/n$ for all i. This result is not unexpected, when we consider that the rank-1 matrix corresponds to random search.

So, we are left with the following:

Open Question: *For a given set of fitnesses, $\boldsymbol{W}$, what classes of fair transmission matrices maximize π_1 while minimizing $r = \lambda_2/\lambda_1$ so as to satisfy the conditions for rapid first hitting time ?* $\diamondsuit$

One step we may take in defining the notion of classes of transmission matrices is to note that the *topology* of transmission may be separated from the *operator intensity* by the following parameterization:

$$\boldsymbol{T}(\mu) = (1-\mu)\boldsymbol{I} + \mu\,\boldsymbol{P}, \tag{4.13}$$

where $\mu \in [0,1]$ is the mutation rate, and $\boldsymbol{P}$ is a transmission matrix in which at least one value $P_{ii} = 0$ (Altenberg and Feldman, 1987). For those genetic

operators that can be represented as graphs, where a vertex represents a type, and an edge represents an operator transformation from one type to another, then $\boldsymbol{P}$ naturally corresponds to a normalized adjacency matrix for the graph.

We can see immediately that if $\mu = 0$, the matrix $\boldsymbol{T}(\mu)\boldsymbol{W}$ becomes reducible, so if $x_1(0) = 0$, then $x_1(t) = 0$ for all t. For small μ, the following should be readily shown:

Conjecture *If the system* $\mathbf{x}(t) = \frac{1}{\nu(t)}(\boldsymbol{T}\boldsymbol{W})^t\,\mathbf{x}(0)$ *exhibits rapid first hitting time, then the system*

$$\mathbf{x}(t) = \frac{1}{\nu(t)}([(1-\mu)\boldsymbol{I} + \mu\,\boldsymbol{T}]\boldsymbol{W})^t\,\mathbf{x}(0)$$

will exhibit rapid first hitting time for $\mu \in [\frac{1}{P(\log(n))}, 1]$, *for some polynomial in* $\log(n)$, $P(\log(n))$, *and will not exhibit rapid first hitting time for* $\mu \in [0, \frac{1}{n}]$.

Let us return to the example of the ONEMAX problem as the paradigmatic EA-easy problem. The transmission matrix for the ONEMAX problem is simple bit-flip mutation, which produces an L-dimensional binary hypercube when represented as a graph between genotypes that mutate to one another. When fitnesses are permuted to the proper order (which Liepins et al. (1990) prove can always be done), the problem becomes the ONEMAX problem. Hence, one can conjecture that a transmission matrix representing the binary hypercube would be a primary candidate for rapid first hitting time. However, it is clear that $\boldsymbol{W}$ can be designed for which no rapid first hitting time can be achieved:

Conjecture *It is possible to choose* ϵ *small enough so that if*

$$|\{i : \epsilon > w_1 - w_i > 0\}| \approx \mathcal{O}(n),$$

then there exists no fair transmission matrix that can produce rapid first hitting time.

With the proper constraints on $\boldsymbol{W}$, however, we may find the following:

Conjecture *Consider a search space,* $\mathcal{S}$, *with* $|\mathcal{S}| = n = 2^L$. *Let the fitness values be* $w_i = e^{-\sigma i}$. *Consider a binary encoding of the indices,* $B(i)$, *such that* $w_i > w_j$ *if and only if the Hamming distances,* $H[\ ,\ ]$, *between the binary encodings satisfies* $H[B(1), B(i)] < H[B(1), B(j)]$. *Let* $\boldsymbol{T} = (1-\mu)\boldsymbol{I} + \mu\,\boldsymbol{P}$, $0 \le \mu \le 1$, *where* $\boldsymbol{P}$ *is the normalized adjacency matrix for the L-dimensional binary hypercube* Q_L *under this encoding. Then for some* $\sigma* > 0$, *if* $\sigma \ge \sigma*$, *then there exists* $\mu(\sigma)$ *such that the system* $\mathbf{x}(t) = (\boldsymbol{T}\boldsymbol{W})^t\,\mathbf{x}(0)/\nu(t)$ *has rapid first hitting time.*

Other examples of evolutionary systems that attain rapid first hitting times can be found in Vitanyi (2000).

We may also consider a class of transmission matrices which can never achieve rapid first hitting time for any set of fitnesses, namely, the "long path?? (Horn et al., 1994) matrices:

Conjecture *Let* $\boldsymbol{T} = (1-\mu)\boldsymbol{I} + \mu\boldsymbol{P}$, *where* $P_{ij} = P_{1n} = P_{n1} = 1/2$ *for* $|i-j| = 1$, $P_{ij} = 0$ *otherwise.*

Then, there are no fitnesses $\boldsymbol{W}$, *nor values* μ, *such that the system*

$$\mathbf{x}(t) = \frac{1}{\nu(t)}(\boldsymbol{TW})^t \mathbf{x}(0)$$

has rapid first hitting time.

1.6 RAPID FIRST HITTING TIME AND NO FREE LUNCH THEOREMS

It should be noted that the concept of rapid first hitting times allows us to distinguish between transmission matrices in a way that the No Free Lunch Theorem (Wolpert and Macready, 1995; Wolpert and Macready, 1997) can.

The No Free Lunch Theorem states, in the current context, that all transmission matrices have the same performance when averaged over all permutations of a set of fitnesses. However, Wolpert and Macready (1995) point out that minimax properties can be used to distinguish search algorithms. In this case, an example of a minimax property is the existence, or lack of existence, of a permutation of fitnesses for a given transmission matrix such that rapid first hitting time occurs, as discussed above.

So, while a long path operator and a binary hypercube operator will have the same average performance in locating the global optimum over all permutations of fitness, there will be some permutations that we expect will give the binary hypercube a rapid first hitting time, while none that will give the long path operator a rapid first hitting time. In this way, we can make a definite judgement that the binary hypercube is superior to the long path operator for optimization.

At this juncture, I will refrain from pursuing the numerous possibilities that exist for investigating these open questions, and leave them for forthcoming work.

2. SPECTRA FOR FINITE POPULATION DYNAMICS

One of the important open questions in evolutionary computation is the relationship between the dynamics of the infinite and the finite population models.

The Wright-Fisher model of finite populations[1] is derived from the canonical model of an infinite population by the addition of only one free parameter—the population size. It thus provides the ideal model in which pose this question.

2.1 WRIGHT-FISHER MODEL OF FINITE POPULATIONS

In the Wright-Fisher model of a finite population, the action of selection and genetic operators on the current members of the population produces a probability distribution from which each member of the population in the next generation is drawn independently. It is as if an infinite zygote pool was created from which only finite many can survive with equal probability.

The elements of the Wright-Fisher model are mostly the same as for the infinite population model. Let:

N be the population size;

x be the vector of frequencies of each type i in the population, corresponding to $N x_i$ individuals of type i;

$\mathbf{x}'$ be the vector of the frequencies of each type i in the population in the next generation, corresponding to $N x_i'$ individuals of type i, produced by taking N independent samples from the distribution y(x);

$\mathbf{y}(\mathbf{x}) = \frac{1}{w} TW\mathbf{x}$ be the vector representing the probability distribution for drawing an individual of type i to compose the population in the next generation. T and W again represent the transmission matrix and fitness matrix, respectively.

Since the population consists of discrete individuals, the frequency vectors are now restricted to a lattice of discrete points on the simplex Δ_n, namely

$$\Delta_n(N) = \{\mathbf{x} : N x_i \in \{0, 1, \ldots, N\}, \sum_{i=1}^{n} x_i = 1\}.$$

The Wright-Fisher model forms a Markov chain, whose transition matrix on frequency vectors is:

$$M = \left[M_{\mathbf{x}',\mathbf{x}}\right]_{\mathbf{x},\mathbf{x}' \in \Delta_n(N)}$$

[1]The Markov theory of Wright and Fisher is known in the genetic algorithms community as the "Nix and Vose model" (Nix and Vose, 1991) since this community developed largely without awareness of prior work in mathematical population genetics. Other work on Markov chains in genetic algorithms includes (Goldberg and Segrest, 1987), and (Davis and Principe, 1993).

with entries

$$M_{\mathbf{x}',\mathbf{x}} = N! \prod_{i=1}^{n} \frac{y_i^{Nx_i'}}{(Nx_i')!} = \frac{N!}{\prod_{i=1}^{n}(Nx_i')!} \prod_{i=1}^{n} \left(\frac{\boldsymbol{e}_i^{\top} \boldsymbol{T} \boldsymbol{W} \mathbf{x}}{\mathbf{1}^{\top} \boldsymbol{W} \mathbf{x}} \right)^{Nx_i'}$$

where $\boldsymbol{e}_i^{\top} = [0\ 0\ \cdots 1 \cdots 0\ 0]$ has the 1 in the ith position.

If we make the assumption that $\boldsymbol{T}$ is primitive, and $T_{ij} = T_{ji}$, then we may employ the Jordan form (4.7):

$$M_{\mathbf{x}',\mathbf{x}} = \frac{N!}{\prod_{i=1}^{n}(Nx_i')!} \prod_{i=1}^{n} \left(\frac{\sum_{j=1}^{n} q_{ij} \lambda_j \langle \boldsymbol{q}_j, \mathbf{x} \rangle}{\sum_{j=1}^{n} w_j x_j} \right)^{Nx_i'}. \tag{4.14}$$

Wright and Fisher analyzed some simple cases for this Markov system and derived a number of their properties, including rates of convergence, probabilities of fixation, time to fixation, and stationary distributions of allele frequencies.

In the special case of $\boldsymbol{T} = \boldsymbol{W} = \boldsymbol{I}$, and $n = 2$, the solution for all the eigenvalues of $\boldsymbol{M}$ was found by Feller (1951), and by Cannings (1974) through his method of "exchangeable processes" (see Ewens 1979, pp. 77-79). The solution is:

$$\overline{\lambda}_1 = 1 \text{ and } \overline{\lambda}_i = \prod_{j=2}^{i} \left(1 - \frac{j-2}{N} \right),\ i \in \{2, \ldots, N+1\},$$

where $\overline{\lambda}_i$ refers to the eigenvalues of $\boldsymbol{M}$, not of $\boldsymbol{TW}$.

Regrettably, the method of exchangeable processes can not be applied when different individuals have different offspring probability distributions. We are therefore left with the following:

Open Question: *What is the relationship between the eigenvalues and eigenvectors of* $\boldsymbol{TW}$ *and those of* $\boldsymbol{M}$*?* ◇

Since $\boldsymbol{M}$ is defined explicitly in terms of the eigenvalues and eigenvectors of $\boldsymbol{TW}$ in (4.14), establishing their relationship with the eigenvalues of $\boldsymbol{M}$ is simply a matter of algebra. The complexity of the algebra, however, obscures the relationship. One may be able to simplify the sums in (4.14) by making assumptions that cause one term to dominate the sum, for example, if $\lambda_2, \cdots, \lambda_n = 0$. But the utility of such an approach has yet to be demonstrated.

One can nevertheless make the following observations. Because the state space of the system is restricted to the lattice $\Delta_n(N) \subset \Delta_n$, and the situation of interest is when $N \approx \mathcal{O}(\log(n)) \ll n$, the vast majority of the entries of

any $\mathbf{x} \in \Delta_n(N)$ must be 0. Thus, $\Delta_n(N)$ has no points on the interior of Δ_n, and is in fact restricted to the low-dimensional boundaries of Δ_n.

Thus, the indices of the non-zero components of x make up a sparse set. Let us define the sparse set:

$$\Psi(\mathbf{x}) = \{i : x_i > 0\} \tag{4.15}$$

Then we may rewrite (4.14) as:

$$M_{\mathbf{x}',\mathbf{x}} = \frac{N!}{\prod_{i\in\Psi(\mathbf{x}')}(Nx_i')!} \prod_{i\in\Psi(\mathbf{x}')} \left(\frac{\sum_{j=1}^{n} q_{ij}\lambda_j \sum_{k\in\Psi(\mathbf{x})} q_{jk}x_k}{\sum_{j\in\Psi(\mathbf{x})} w_j x_j} \right)^{Nx_i'} \tag{4.16}$$

The trajectory of points in the finite population model will be radically different from the trajectory in the infinite population model. In the finite population model, a probability distribution will move over the surface of Δ_n, while in the infinite population model, the system will immediately enter the interior of Δ_m since $TW > 0$. Evolution in the finite population model can be views as transitions between one k-dimensional ($k \leq N$) edge of Δ_n and another, with the probability of transition being highest for types i where the terms $\sum_{j=1}^{n} q_{ij}\lambda_j \langle \boldsymbol{q}_j, \mathbf{x} \rangle$ are the largest.

My earlier discussion of the transmission matrix representing the binary hypercube took place in the context of the infinite population model. I conjecture that it will exhibit rapid first hitting time properties in the infinite population dynamics. However, it seems apparent that the binary hypercube mutation will be especially advantageous in traversing the low-dimensional boundaries of the simplex in the finite population model.

I suspect that methods which can analyze (4.14) as a flow along the low-dimensional boundaries of the simplex may prove to be most helpful in understanding finite population dynamics. In the work of van Nimwegen (1999) we find this approach applied to specific models of mutation and selection, with a nice harvest of analytical results. Answers to the general spectral problem, however, await discovery.

2.2 RAPID FIRST HITTING TIME IN A FINITE POPULATION

For a Wright-Fisher model, we can define the criteria for rapid first hitting time in terms of the actual first hitting time for the Markov chain. Here I depart only slightly from Vitanyi (2000).

Let us refer to the set of populations that contain the global optimum as:

$$\mathcal{B}^+ = \{\mathbf{x} \in \Delta_n(N) : x_1 > 0\},$$

and conversely,

$$\mathcal{B}^- = \{\mathbf{x} \in \Delta_n(N) : x_1 = 0\},$$

Suppose that the population always begins fixed on one type other than the optimum, so $\mathbf{x}(0) = \boldsymbol{e}_i, i \in \{2, \ldots, n\}$.

Define $\overline{\boldsymbol{M}}$ to be the restriction of $\boldsymbol{M}$ to $\mathcal{B}^+$. Then the probability that the global optimum first appears after generation τ is:

$$G_i(t) = \sum_{\mathbf{x} \in \mathcal{B}^-} \left[\overline{\boldsymbol{M}}^t\right]_{\mathbf{x}, \boldsymbol{e}_i} = \langle \mathbf{1}, \overline{\boldsymbol{M}}^t \boldsymbol{e}_i \rangle$$

We can define the criteria for rapid first hitting time in terms of the speed at which $G_i(\tau)$ declines with τ.

As a basis for comparison, we can consider how $G_i(\tau)$ behaves for random search, i.e. $\boldsymbol{M} = \frac{1}{\hat{n}} \mathbf{1}\mathbf{1}^\top$, where $\hat{n} = |\Delta_n(N)|$. Then $G_i(t) = \sum_{\mathbf{x} \in \mathcal{B}^-} \frac{1}{\hat{n}} = \left(1 - \frac{1}{n}\right)^N$ for all i. So

$$G(\tau) = \left(1 - \frac{1}{n}\right)^{\tau N}.$$

In order for $G(\tau)$ to be reduced to $\mathcal{O}(1)$, to be specific, say $G(\tau) \leq \frac{1}{e}$, we have:

$$\log[G(\tau)] = \tau N \log\left(1 - \frac{1}{n}\right) \leq -1,$$

thus for large n, $\log\left(1 - \frac{1}{n}\right) \approx -\frac{1}{n}$, hence $\tau N \geq n$, which is what we expect. The essential idea for rapid first hitting time is that we would like $\tau N \leq P(\log(n))$. The the obvious candidate for a condition to define rapid first hitting time would be:

Definition: Rapid First Hitting Time in a Finite Population. *The evolutionary algorithm is said to possess a rapid first hitting time if there exist polynomials in* $\log(n)$, $P_1(\log(n))$ *and* $P_2(\log(n))$, *such that* $N \leq P_1(\log(n))$, *and*

$$\max_{i \in \mathcal{B}^-} \langle \mathbf{1}, \overline{\boldsymbol{M}}^\tau \boldsymbol{e}_i \rangle \leq \frac{1}{e}, \quad \textit{for all } \tau \geq P_2(\log(n)), \tag{4.17}$$

Clearly, the smaller the spectral radius of $\overline{\boldsymbol{M}}$, the more easily that rapid first hitting time is achieved.

We are then ready to pose the main open question regarding the spectrum of evolutionary systems:

Open Question: What conditions on the eigenvalues and eigenvectors of $\boldsymbol{TW}$ satisfy condition (4.17) for rapid first hitting time? ♢

3. KARLIN'S SPECTRAL THEOREM FOR GENETIC OPERATOR INTENSITY

Samuel Karlin derived a quite fundamental theorem for evolutionary dynamics in a paper that examines the role of population subdivision in maintaining genetic diversity (Karlin, 1982). The problem at hand was to understand whether migration would enhance or inhibit the maintenance of genetic diversity. Karlin took the approach of finding the conditions that would prevent an allele from becoming extinct, i.e. which would cause it to increase when rare. To this end, he proved a general theorem on the spectral radius of the stability matrix which solved his problem for the case of any number of demes, any migration pattern, and any selection regime—quite extraordinary in its generality. I present theorem below. The square matrix $\boldsymbol{P}$ represents the migration pattern, ξ represents the overall migration rate, and the diagonal matrix $\boldsymbol{W}$ represents the average (or 'marginal') selection coefficients of an allele when rare.

Theorem 2 (Karlin, 1982)
Let

$$\boldsymbol{M}(\xi) = (1 - \xi)\boldsymbol{I} + \xi\boldsymbol{P},$$

where $\boldsymbol{P}$ *is an irreducible Markov matrix, and let* $\boldsymbol{W}$ *be a diagonal matrix with strictly positive diagonal elements, where* $\boldsymbol{W} \neq a\boldsymbol{I}$, *for any scalar* a. *Then the spectral radius* $\rho(\boldsymbol{M}(\xi)\boldsymbol{W})$ *is strictly decreasing in* ξ:

$$\frac{d}{d\xi}\rho(\boldsymbol{M}(\xi)\boldsymbol{W}) < 0, \quad \textit{for } 0 \leq \xi \leq 1.$$

An allele goes extinct when rare if $\rho(\boldsymbol{M}(\xi)\boldsymbol{W}) < 1$, and is protected from extinction if $\rho(\boldsymbol{M}(\xi)\boldsymbol{W}) > 1$. Since $\rho(\boldsymbol{M}(\xi)\boldsymbol{W})$ decreases with ξ, the consequence of this result is that more migration makes it more difficult to maintain genetic diversity.

In the process of answering the question about migration, Karlin produced a result far more fundamental. Its first application outside of that specific question was its use by Altenberg (Altenberg, 1984; Altenberg and Feldman, 1987) to unify the Reduction Principle result for the evolution of genetic systems. In several earlier studies that separately modeled the evolution of mutation, recombination, and migration rates, the same result kept arising: new alleles which reduced these rates could always invade a population. By applying Karlin's theorem, I showed that each of these results were in fact special cases of the same result: that when an allele that modifies rates of mutation, recombination, migration, or any other transformation of type is introduced into a population near equilibrium, it will increase in frequency if it uniformly reduces the rates of transformation, and go extinct if it uniformly increases the rates of transformation.

Another immediate result from Karlin's theorem regards the mean fitness under a mutation-selection balance. The mean fitness of haploid system decreases with increasing mutation rates:

Corollary 1 *Consider an evolutionary system consisting of*

- *constant selection,*
- *asexual genetic operators, and*
- *discrete, non-overlapping generations.*

The mean fitness of the population at an attractor is a decreasing function of the probability of the genetic operator acting.

Proof. Let the asexual genetic operator be represented by the Markov matrix $\boldsymbol{M}$, and let μ is the probability of applying the operator. Then the transmission matrix for the algorithm is:

$$\boldsymbol{T} = (1 - \mu)\boldsymbol{I} + \mu\boldsymbol{M},$$

and the recursion for discrete, non-overlapping generations is:

$$\overline{w}\,\mathbf{x}' = [(1 - \mu)\boldsymbol{I} + \mu\boldsymbol{M}]\,\boldsymbol{W}\,\mathbf{x}$$

For the global attractor, $\hat{\mathbf{x}}$, which is the leading eigenvector of $\boldsymbol{TW}$ (whose existence and positive value are established by the Perron-Frobenius Theorem (Gantmacher, 1959), we have:

$$[(1 - \mu)\boldsymbol{I} + \mu\boldsymbol{M}]\,\boldsymbol{W}\,\hat{\mathbf{x}} = \boldsymbol{T}\,\boldsymbol{W}\,\hat{\mathbf{x}} = \hat{\mathbf{x}}\,\rho(\boldsymbol{TW}) = \hat{\mathbf{x}}\,\overline{w}.$$

Hence the mean fitness of the global attractor,

$$\widehat{\overline{w}} = \rho(\boldsymbol{T}(\mu)\boldsymbol{W}),$$

is a decreasing function of the operator probability μ. ■

3.1 KARLIN'S THEOREM ILLUSTRATED WITH THE DECEPTIVE TRAP FUNCTION

Suppose a mutation operator is ergodic: i.e. repeated application of the operator can mutate any genotype into any other genotype. Then, under an algorithm of constant selection and mutation, the Perron-Frobenius Theorem shows that there is only one domain of attraction of the system—i.e. one 'fitness peak', as discussed in Section 1.1. This may seem contradictory to

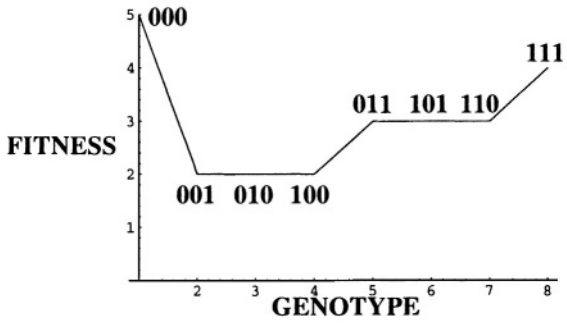

Figure 4.1. The Deceptive Trap fitness landscape for three loci with two alleles.

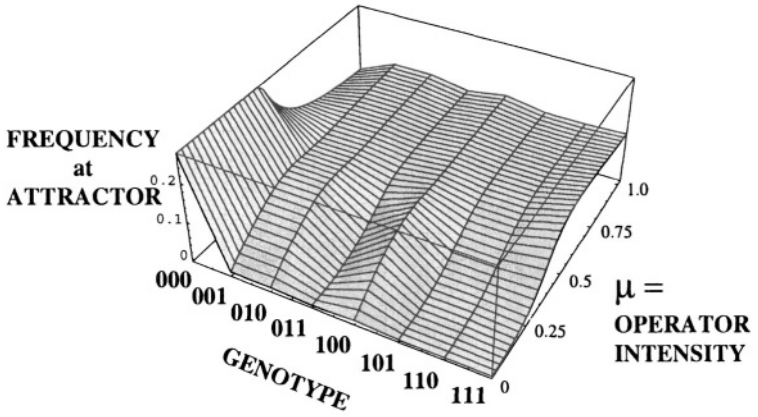

Figure 4.2. There is only one attractor at each value μ, but an 'error catastrophe' is evident for $\mu \approx 0.5$.

intuition about 'multi-modal' fitness landscapes, in which one would expect multiple domains of attraction. But multiple domains do not occur in haploid, infinite population models under ergodic mutation; finite populations are required to produce quasi-stability of multiple attractors. The global nature of the attractor for ergodic mutation under infinite population size is illustrated with the Deceptive Trap fitness landscape (Ackley, 1987), shown in Figure 4.1. In terms of the hypercube topology, this is a bimodal fitness function. The frequency vector of the global attractor is shown as a function of the mutation rate, for a simple point mutation model, in Figure 4.2. The mean fitness of the attractor is seen to decrease as a function of the mutation rate, as the Karlin theorem proves. This is shown in Figure 4.3.

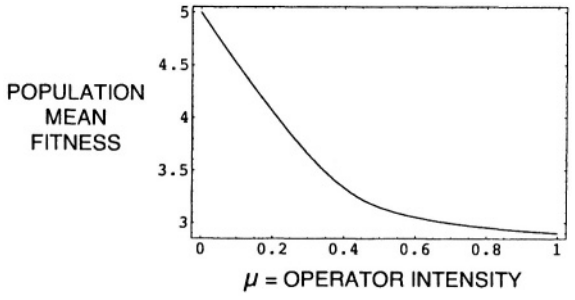

Figure 4.3. The mean fitness of the population at the global attractor as a function of mutation rate. It decreases in accord with Karlin's theorem.

3.2 APPLICATIONS FOR AN EXTENDED KARLIN THEOREM

Several problems are encountered for which an extended Karlin theorem would allow solution, but which are currently unsolved. One of these is in modifier theory. This has been called 'self-adaptation' (Schwefel, 1987; Bäck, 1996) in the Evolutionary Computation literature. In Altenberg (1984) and Altenberg et al. (1987) it is proven that the Reduction Principle for linear variation in transmission holds for modifiers that are tightly linked to haplotypes under viability selection (Altenberg and Feldman, 1987, Result 3, p. 565). It is conjectured that the result would also hold for looser linkage to the modifier locus. The analysis requires that we show for $r > 0$ that the spectral radius of $\boldsymbol{M}(\mu, r)\boldsymbol{W}$ decreases in μ, where:

$$\boldsymbol{M}(\mu, r) = (1-\mu)[(1-r)\boldsymbol{I} + r\boldsymbol{Q}] + \mu[(1-r)\boldsymbol{S} + r\tilde{\boldsymbol{S}}],$$

with $\boldsymbol{Q}$, $\boldsymbol{S}$, and $\tilde{\boldsymbol{S}}$ being Markov matrices (see (Altenberg and Feldman, 1987) for details). The proof awaits an extension of Karlin's theorem for $r > 0$.

The other context of unsolved problems occurs when several transformation processes act on types in the population, such as the simultaneous action of mutation, recombination, and migration. This can result in recursions of the form:

$$\begin{aligned} \overline{w}\mathbf{x}' &= \boldsymbol{M}(\mu, \beta, \gamma)\boldsymbol{W}\mathbf{x} \\ &= [(1-\mu)\boldsymbol{I} + \mu\boldsymbol{A}][(1-\beta)\boldsymbol{I} + \beta\boldsymbol{B}][(1-\gamma)\boldsymbol{I} + \gamma\boldsymbol{C}]\boldsymbol{W}\mathbf{x}, \end{aligned}$$

where $\boldsymbol{A}$, $\boldsymbol{B}$, and $\boldsymbol{C}$ are Markov matrices representing different transformation processes, and μ, β, and γ are the overall rates of those processes. We wish to know how the spectral radius of $\boldsymbol{M}(\mu, \beta, \gamma)\boldsymbol{W}$ changes as a function of each parameter μ, β, and γ. Such a result would allow understanding of genetic recombination can evolve in the presence of mutation under certain circumstances (Altenberg, 1984; Kondrashov, 1988). It is clear that for certain cases

of A, B, C, and W, the spectral radius is not monotonically decreasing in each of μ, β, and γ. However, specifying the conditions that produce an increase in the spectral radius with respect to μ, β, γ, etc. requires an extension of Karlin's theorem. The existence of cases of increase led to the "Principle of Partial Control" for the evolution of genetic modifiers:

Conjecture *(Altenberg, 1984, p. 149) When a modifier gene has only partial control over the transformations occurring at selected loci, then it is possible for this part of the transformation to evolve an increase.*

3.3 EXTENDING KARLIN'S THEOREM

We need to extend Karlin's theorem on linear variation from products of the form

$$[(1-\mu)\boldsymbol{I} + \mu\boldsymbol{B}]\boldsymbol{W}$$

to products of the more general form

$$[(1-\mu)\boldsymbol{A} + \mu\boldsymbol{B}]\boldsymbol{W}.$$

Open Question: *Let*

$$\boldsymbol{M}(\mu) = [(1-\mu)\boldsymbol{A} + \mu\boldsymbol{B}],$$

where $\boldsymbol{A}$ and $\boldsymbol{B}$ are irreducible Markov matrices, and $\boldsymbol{W}$ is a diagonal matrix with strictly positive diagonal elements. For what conditions on $\boldsymbol{A}$, $\boldsymbol{B}$, and $\boldsymbol{W}$ is the spectral radius $\rho(\boldsymbol{M}(\mu)\boldsymbol{W})$ strictly decreasing in μ:

$$\frac{d}{d\mu}\rho(\boldsymbol{M}(\mu)\boldsymbol{W}) < 0, \ \textit{for}\ 0 \le \mu \le 1?$$

$\diamondsuit$

Karlin proved that the spectral radius $\rho([(1-\mu)\boldsymbol{I} + \mu\boldsymbol{B}]\boldsymbol{W})$ is decreasing in μ. Clearly each matrix pair $\{\boldsymbol{B},\boldsymbol{W}\}$ determines a class of matrices $\boldsymbol{A}$ for which the spectral radius $\rho([(1-\mu)\boldsymbol{A} + \mu\boldsymbol{B}]\boldsymbol{W})$ is decreasing in μ. Explicit characterization of this class is not immediately obvious. However, one can follow Karlin's proof to produce a condition which would provide the answer if it could be solved.

Suppose that

$$\boldsymbol{M}(\mu, r) = [(1-\mu)\boldsymbol{I} + \mu\boldsymbol{A}][(1-r)\boldsymbol{I} + r\boldsymbol{B}],$$

where $\boldsymbol{A}$ and $\boldsymbol{B}$ are Markov matrices.

I retrace the analysis of Karlin (1982, pp. 195-196). Define

$$\phi(\mathbf{p},\mu,r)=\sup_{\mathbf{x}>0}\sum_i p_i\log\left(\frac{x_i}{[M(\mu,r)\mathbf{x}]_i}\right). \tag{4.18}$$

Let $\mathbf{x}(\mu,r)$ be the vector for which the supremum is attained. The Donsker-Varadhan (1975) variational formula for the spectral radius gives:

$$\log\rho\left(M(\mu,r)D\right)=\sup_{\mathbf{p}>0}\left[\langle\mathbf{p},\log(D\,\mathbf{1})\rangle-\phi(\mathbf{p},\mu,r)\right] \tag{4.19}$$

where $D=\mathbf{diag}\left[w_i/\overline{w}\right]$, 1 is the vector of ones, $\log(\boldsymbol{v})$ stands for the vector of components $\log(v_i)$,

$$\log(D\,\mathbf{1})=\left[(\log w_i-\log\overline{w})\right],$$

and we set $\sum_i p_i=1$. Let $\mathbf{p}(\mu,r)$ be the vector at which this supremum is attained.

Since both $\mathbf{x}(\mu,r)$ and $\mathbf{p}(\mu,r)$ are unique critical points as implicitly defined.

$$\frac{\partial x(\mu,r)_i}{\partial\mu}=\frac{\partial p(\mu,r)_i}{\partial\mu}=0$$

for all i. Hence

$$\frac{\partial\rho}{\partial\mu}=-\rho\frac{\partial}{\partial\mu}\phi(\mathbf{p},\mu,r)$$

with $\mathbf{p}=\mathbf{p}(\mu,r)$ fixed. Further evaluation paralleling Karlin (1982) yields the condition

$$\frac{\partial\rho}{\partial\mu}\ \leq\ 0\iff \tag{4.20}$$

$$1\leq\sum_i p_i(\mu,r)\frac{[\mathbf{x}(\mu,r)]_i}{[M(\mu,r)\mathbf{x}(\mu,r)]_i}.$$

For $r=0$, Karlin uses Jensen's inequality to give:

$$\log\sum_i p_i(\mu,0)\frac{x_i(\mu,0)}{[M(\mu,0)\mathbf{x}(\mu,0)]_i} \tag{4.21}$$

$$\geq\ \phi(\mathbf{p},\mu,0)=\sum_i p_i(\mu,0)\log\frac{x_i(\mu,0)}{[M(\mu,0)\mathbf{x}(\mu,0)]_i}.$$

By using the principal eigenvector

$$\tilde{\mathbf{x}}=M(\mu,0)\,\tilde{\mathbf{x}},$$

the supremum definition of ϕ gives:

$$\phi(\mathbf{p}, \mu, 0) \geq \sum_i p_i(\mu, 0) \log \frac{\tilde{x}_i}{[\boldsymbol{M}(\mu, 0)\tilde{\mathbf{x}}]_i} = 0.$$

Thus was it is proved that for $r = 0$, $\frac{\partial \rho}{\partial \mu} \leq 0$. The analysis of (4.20) for $r > 0$ does not allow us to use (4.21), and is unsolved. This leaves us with:

Open Question: *What conditions on the matrices $\boldsymbol{A}$ and $\boldsymbol{B}$, and scalars r and μ, produce*

$$1 \leq \sum_i p_i(\mu, r) \frac{[\mathbf{x}(\mu, r)]_i}{[[(1-\mu)\boldsymbol{I} + \mu\boldsymbol{A}][(1-r)\boldsymbol{I} + r\boldsymbol{B}]\, \mathbf{x}(\mu, r)]_i},$$

where $\mathbf{x}(\mu, r)$ and $\mathbf{p}(\mu, r)$ are the vectors producing the suprema of expressions (4.18) and (4,19)? $\diamondsuit$

Another direction to extend Karlin's theorem, which would be quite relevant to the issue of rapidly mixing Markov chains and rapid first hitting times, is to say something about the second-largest eigenvalue. I would offer (without claiming undue certitude) the following:

Conjecture
Let

$$\boldsymbol{M}(\xi) = (1-\xi)\boldsymbol{I} + \xi\boldsymbol{P},$$

where $\boldsymbol{P}$ is an irreducible Markov matrix, and let $\boldsymbol{W}$ be a diagonal matrix with strictly positive diagonal elements. Then the ratio of the second-largest eigenvalue $\lambda_2(\boldsymbol{M}(\xi)\boldsymbol{W})$, to the spectral radius $\rho(\boldsymbol{M}(\xi)\boldsymbol{W})$, is strictly increasing in ξ:

$$\frac{d}{d\xi} \frac{\lambda_2(\boldsymbol{M}(\xi)\boldsymbol{W})}{\rho(\boldsymbol{M}(\xi)\boldsymbol{W})} > 0, \quad \textit{for } 0 \leq \xi \leq 1.$$

3.4 DISCUSSION

Karlin's theorem, because it holds for arbitrary Markov and fitness matrices, captures a fundamental property of Darwinian dynamics, the interaction of selection and transformation caused by genetic operators. What is not generally understood is how multiple genetic operators interact with one another. The difficulty of analyzing Wright's Shifting Balance Theory (Wright, 1931), which is about the interaction of recombination, mutation, migration, selection, and drift, exemplifies the mathematical difficulties. Attempting to understand the interaction of multiple genetic operators brings us to the need to extend Karlin's theorem.

4. CONCLUSION

I hope that the reader, having followed the lines of discussion through this essay, may come away with the conclusion that the spectra of evolutionary systems provide a useful means to pose, and occasionally to solve, problems in evolutionary dynamics. I have used the spectral representation of the generalized mutation-selection system to address the question of when an evolutionary algorithm is useful for function optimization. I have described an analog to "rapidly mixing Markov chains" (Sinclair, 1992) that is appropriate for optimization, "rapid first hitting time". The conditions needed for an evolutionary algorithm to exhibit rapid first hitting time can be described in terms of the spectra of the linear systems that represent them.

I have also posed, questions on the dynamics of finite populations in terms of the spectra of the underlying operators. Tying together the spectra of infinite population models with the spectra of the finite population models into which they are embedded remains a major open question in the theory of evolutionary dynamics. Progress may result if flows over the low-dimensional boundaries of the simplex can be modeled.

Lastly, I have reviewed an important theorem by Karlin (1982) on the spectral properties of genetic operator intensity. Extensions of this theorem would find immediate application.

Since these are spectral problems, there may indeed already be analytic techniques that could be applied to their solution. It is hoped that this essay may bring attention to these problems and thus hasten their solution.

REFERENCES

Ackley, D. H. (1987). *A Connectionist Machine for Genetic Hillclimbing,* volume SECS28 of *The Kluwer International Series in Engineering and Computer Science.* Kluwer Academic Publishers, Boston.

Altenberg, L. (1984). *A Generalization of Theory on the Evolution of Modifier Genes.* PhD thesis, Stanford University. Available from University Microfilms, Ann Arbor, MI.

Altenberg, L. (1995). The Schema Theorem and Price's Theorem. In Whitley, D. and Vose, M. D., editors, *Foundations of Genetic Algorithms 3,* pages 23–49. Morgan Kaufmann, San Mateo, CA.

Altenberg, L. and Feldman, M. W. (1987). Selection, generalized transmission and the evolution of modifier genes. I. The reduction principle. *Genetics,* 117:559–572.

Arora, S., Rabani, Y., and Vazirani, U. (1994). Simulating quadratic dynamical systems is PSPACE-complete. In *Proceedings of the 26th Annual ACM Symposium on Theory of Computing,* pages 459–467.

Bäck, T. (1996). *Evolutionary Algorithms in Theory and Practice: Evolutionary Strategies, Evolutionary Programming and Genetic Programming.* Oxford University Press, Oxford.

Cannings, C. (1974). The latent roots of certain Markov chains arising in genetics: a new approach, I. haploid models. *Advances in Applied Probability,* 6:260–290.

Christiansen, F. B. (2000). *Population Genetics of Multiple Loci.* John Wiley and Sons, LTD, Chichester.

Davis, T. E. and Principe, J. C. (1993). A Markov chain framework for the simple genetic algorithm. *Evolutionary Computation,* 1(3):269–288.

Donsker, M. D. and Varadhan, S. R. S. (1975). On a variational formula for the principal eigenvalue for operators with maximum principle. *Proceedings of the National Academy of Science, USA,* 72:780–783.

Ewens, W. J. (1979). *Mathematical Population Genetics.* Springer-Verlag, Berlin.

Feller, W. (1951). Diffusion processes in genetics. In Neyman, J., editor, *Proceedings of the Second Berkeley Symposium on Mathematical Statistics and Probability,* pages 227–246. University of California Press, Berkeley.

Fisher, R. A. (1930). *The Genetical Theory of Natural Selection.* Clarendon Press, Oxford.

Gantmacher, F. R. (1959). *The Theory of Matrices, volume 2.* Chelsea Publishing Company, New York.

Goldberg, David E. and Deb, Kalyanmoy (1991). A comparative analysis of selection schemes used in genetic algorithms. In Rawlins, G., editor, *Foundations of Genetic Algorithms,* pages 69–93. Morgan Kaufmann, San Mateo, CA.

Goldberg, David E. and Segrest, P. (1987). Finite Markov chain analysis of genetic algorithms. In *Proceedings of the Second International Conference on Genetic Algorithms,* pages 1–8.

Horn, J., Goldberg, David E., and Deb, Kalyanmoy (1994). Long path problems. In Schwefel, H. P. and R. Männer, editors, *Parallel Problem Solving from Nature—PPSN III,* volume 866, Berlin. Springer-Verlag.

Karlin, S. (1982). Classification of selection-migration structures and conditions for a protected polymorphism. In Hecht, M. K., Wallace, B., and Prance, G. T., editors, *Evolutionary Biology,* volume 14, pages 61–204. Plenum Publishing Corporation.

Kondrashov, A. S. (1988). Deleterious mutations and the evolution of sexual reproduction. *Nature(London),* 336:435–440.

Liepins, G. and Vose, M. D. (1990). Representational issues in genetic optimization. *Journal of Experimental and Theoretical Artificial Intelligence,* 2(2):101–115.

Nix, A. E. and Vose, M. D. (1991). Modeling genetic algorithms with Markov chains *Annals of Mathematics and Artificial Intelligence* , 5:79–88.

Palmer, R. G. (1982). Broken ergodicity. *Advances in Physics,* 31:669–735.

Rabani, Y., Rabinovich, Y., and Sinclair, A. (1995). A computational view of population genetics. In *Annual ACM Symposium on the Theory of Computing,* pages 83–92.

Rabinovich, Y., Sinclair, A., and Wigderson, A. (1992). Quadratic dynamical systems. In *IEEE Symposium on Foundations of Computer Science,* pages 304–313.

Rabinovich, Y. and Wigderson, A. (1999). Techniques for bounding the convergence rate of genetic algorithms. *Random Structures Algorithms,* 14:111–138.

Rudolph, G. (1997). *Convergence properties of evolutionary algorithms.* **Verlag Kovač,** Hamburg.

Schmitt, F. and Rothlauf, F. (200la). On the importance of the second largest eigenvalue on the convergence rate of genetic algorithms. In Spector, L., Goodman, E. D., Wu, A., Langdon, W. B., Voigt, H.-M., Gen, M., Sen, S., Dorigo, M., Pezeshk, S., Garzon, M. H., and Burke, E., editors, *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2001)*, pages 559–564, San Francisco, California, USA. Morgan Kaufmann.

Schmitt, F. and Rothlauf, F. (2001b). On the mean of the second largest eigenvalue on the convergence rate of genetic algorithms. Technical Report Working Paper 1/2001, University of Bayreuth, Department of Information Systems, Universitaetsstrasse 30, D-95440 Bayreuth, Germany. Working Papers in Information Systems.

Schwefel, H.-P. (1987). Collective phenomena in evolutionary systems. *Preprints of the 31st Annual Meeting of the International Society for General Systems Research, Budapest,* 2:1025–1033.

Sinclair, A. (1992). *Algorithms for random generation and counting: A Markov chain approach.* Birkhäuser, Boston.

Suzuki, J. (1995). A Markov chain analysis on simple genetic algorithms. *IEE Transactions on Systems, Man and Cybernetics*, 25(4):655–659.

van Nimwegen, E. J. (1999). *The Statistical Dynamics of Epochal Evolution.* PhD thesis, Universiteit Utrecht, Amsterdam.

van Nimwegen, E. J., Crutchfield, J. P., and Huynen, M. (1999). Metastable evolutionary dynamics: Crossing fitness barriers or escaping via neutral paths? *Bulletin of Mathematical Biology,* 62:799–848.

Vitanyi, P. (2000). A discipline of evolutionary programming. *Theoretical Computer Science,* 241(l–2):3–23.

Wolpert, D. H. and Macready, W. G. (1995). No free lunch theorems for search. Technical Report SFI-TR-95-02-010, Santa Fe Institute, Santa Fe, NM.

Wolpert, D. H. and Macready, W. G. (1997). No free lunch theorems for optimization. *IEEE Transactions on Evolutionary Computation*, l(l):67–82.
Wright, S. (1931). Evolution in Mendelian populations. *Genetics,* 16:97–159.

Chapter 5

SOLVING COMBINATORIAL OPTIMIZATION PROBLEMS VIA REFORMULATION AND ADAPTIVE MEMORY METAHEURISTICS

Gary A. Kochenberger
School of Business
University of Colorado at Denver
Gary.Kochenberger@cudenver.edu

Fred Glover
School of Business
University of Colorado at Denver
Fred.Glover@Colorado.edu

Bahram Alidaee
Hearin Center for Enterprise Science
University of Mississippi
Balidaee@bus.olemiss.edu

Cesar Rego
Hearin Center for Enterprise Science
University of Mississippi
Crego@bus.olemiss.edu

Abstract Metaheuristics - general search procedures whose principles allow them to escape the trap of local optimality using heuristic designs - have been successfully employed to address a variety of important optimization problems over the past few years. Particular gains have been achieved in obtaining high quality solutions to problems that classical exact methods (which guarantee convergence) have found too complex to handle effectively. Typically a metaheuristic method is crafted to suit the particular characteristics of the problem at hand, exploiting to the extent possible the structure available to enable a fruitful and efficient search process. An alternative to this problem specific solution approach is a more general methodology that recasts a given problem into a common modeling

format, permitting solutions to be derived by a common, rather than tailor-made, heuristic method.

The optimization folklore strongly emphasizes the unproductive consequences of converting problems from a specific class to a more general representation, since the "domain-specific structure" of the original setting then becomes invisible and can not be exploited by a method for the more general problem representation. Nevertheless, there is a strong motivation to attempt such a conversion in many applications to avoid the necessity to develop a new method for each new class. We demonstrate the existence of a general problem representation that frequently overcomes the limitation commonly ascribed to such models. Contrary to expectation, when a specially structured problem is translated into this general form, it often does not become much harder to solve, and sometimes becomes even easier to solve provided the right type of solution approach is applied. The model with this appealing property is the Quadratic Unconstrained Integer Programming (QUIP) problem in binary variables, accompanied by the device of introducing quadratic infeasibility penalty functions to handle constraints. Not only is the model capable of representing a wide range of "special case" problem classes, but it can be advantageously exploited by adaptive memory (tabu search) metaheuristics and associated evolutionary (scatter search) methods. Computational outcomes disclose the effectiveness of this combined modeling and solution approach for problems from a diverse collection of challenging settings.

1. INTRODUCTION

The Quadratic Unconstrained Integer Program (QUIP[1]) can be written in the form:

$$\text{QUIP:} \quad \min \ f(\mathbf{x}) = \mathbf{x}^t Q \mathbf{x}, \tag{5.1}$$

$$\text{subject to:} \quad \mathbf{x} \text{ is binary}, \tag{5.2}$$

where Q is an n by n matrix of constants and x is an n-vector of zero-one variables. QUIP is notable for its ability to represent a significant variety of important problems. The applicability of this representation has been reported in diverse settings such as social psychology (Harary, 1954), financial analysis (Laughunn, 1970), (McBride and Yormack, 1980), computer aided design (Krarup and Pruzan, 1978), traffic management (Gallo et al., 1980), (Witsgall, 1975), machine scheduling (Alidaee et al., 1994), cellular radio channel allocation (Chardaire and Sutler, 1994), and molecular conformation (Phillips and Rosen, 1994). Moreover, many combinatorial optimization problems pertaining to graphs such as determining maximum cliques, maximum cuts, maximum vertex packing, minimum coverings, maximum independent sets, and maximum independent weighted sets are known to be capable of being

[1] We are indebted to Anil Menon for suggesting this name and acronym.

formulated by the QUIP problem as documented in papers of (Pardalos and Rodgers, 1992), and (Pardalos and Xue, 1994).

The application potential of QUIP is yet substantially greater than this, however, due to reformulation methods that enable certain constrained models to be re-cast in the form of QUIP. (Hammer and Rudeanu, 1968) and (Hansen, 1979) show that any quadratic (or linear) objective in bounded integer variables and constrained by linear equations can be reformulated as a QUIP model. Our purpose is to report results that disclose this wide array of potential reformulations into the QUIP format is not merely a representational novelty, but is a source of practical consequences. The following material draws upon recent findings in by (Kochenberger et al., 1998) and in (Glover et al., 1999b).

2. TRANSFORMATIONS

We take as our starting point the constrained problem

$$\min x_o = \mathbf{x}^t Q\mathbf{x}, \tag{5.3}$$

$$\text{subject to: } A\mathbf{x} = \boldsymbol{b},\ \mathbf{x} \text{ is binary.} \tag{5.4}$$

This model describes both the quadratic and linear case since the linear case results when Q is a diagonal matrix. Problems with inequality constraints can also be put into this form by representing their bounded slack variables by a binary expansion. These constrained quadratic optimization models are converted into equivalent QUIP models by adding a quadratic infeasibility penalty function to the objective function in place of explicitly imposing the constraints $A\mathbf{x} = \boldsymbol{b}$.

$$x_o = \mathbf{x}^t Q\mathbf{x} + P(A\mathbf{x} - \boldsymbol{b})^t(A\mathbf{x} - \boldsymbol{b}), \tag{5.5}$$

$$= \mathbf{x}^t Q\mathbf{x} + \mathbf{x}^t D\mathbf{x} + \boldsymbol{c}, \tag{5.6}$$

$$= \mathbf{x}^t \hat{Q}\mathbf{x} + \boldsymbol{c}, \tag{5.7}$$

where the matrix D and the additive constant $\boldsymbol{c}$ result directly from the matrix multiplication indicated. Dropping the additive constant, the equivalent unconstrained version of our constrained problem becomes,

$$\text{QUIP(PEN): } \min \mathbf{x}^t \hat{Q}\mathbf{x},\ \mathbf{x} \text{ is binary.} \tag{5.8}$$

From a theoretical standpoint, a suitable choice of the penalty scalar P can always be chosen such that the optimal solution to QUIP(PEN) is the optimal solution to the original constrained problem (Hammer and Rudeanu, 1968). Similar theoretical outcomes apply to many types of representations other than the QUIP model, of course, and the issue of interest is whether there is any practical merit in undertaking such a transformation in the QUIP case. The same question arises by reference to another transformation, which likewise

falls within the context of the QUIP model. We refer to the preceding general transformation as transformation # 1. A very important special class of constraints that arise in many applications can be handled by an alternative approach, given below, which we call transformation #2. Many problems have considerations that isolate two specific alternatives and prohibit both from being chosen. That is, for a given pair of alternatives, one or the other but not both may be chosen. If x_j and x_k are binary variables denoting whether or not alternatives j and k are chosen, the standard constraint that allows one choice but precludes both is:

$$x_j + x_k \leq 1. \tag{5.9}$$

Then, for a positive scalar P, adding the penalty function Px_jx_k to the objective function is a simple alternative to imposing the constraint is a traditional manner. This penalty function has sometimes been used by to convert certain optimization problems on graphs (e.g., the maximum clique problem) into an equivalent QUIP model. Its potential application, however, goes far beyond graph problems as we demonstrate in later sections of this paper.

3. EXAMPLES

Before highlighting a variety of problem classes to which we have successfully applied the foregoing transformation approaches, we give two small examples from classical problem settings to provide concrete illustrations:

Example 1 Set Partitioning:

$$\min \quad x_o = 3x_i + 2x_2 + x_3 + x_4 + 3x_5 + 2x_6,$$

subject to:

$$\begin{aligned} x_1 + x_3 + x_6 &= 1, \\ x_2 + x_3 + x_5 + x_6 &= 1, \\ x_3 + x_4 + x_5 &= 1, \\ x_1 + x_2 + x_4 + x_6 &= 1. \end{aligned}$$

and x binary. Applying transformation #1 with $P = 10$ gives the equivalent QUIP model:

$$\text{QUIP(PEN):} \quad \min \mathbf{x}^t \hat{Q} \mathbf{x}, \ \mathbf{x} \text{ is binary},$$

where the additive constant $\boldsymbol{c}$, is 40 and,

$$\hat{Q} = \begin{pmatrix} -17 & 10 & 10 & 10 & 0 & 20 \\ 10 & -18 & 10 & 10 & 10 & 20 \\ 10 & 10 & -29 & 10 & 20 & 20 \\ 10 & 10 & 10 & -19 & 10 & 10 \\ 0 & 10 & 20 & 10 & -17 & 10 \\ 20 & 20 & 20 & 10 & 10 & -28 \end{pmatrix}.$$

Solving QUIP(PEN) by the method[2] of (Glover et al, 1999c) we obtain an optimal solution $x_1 = x_5 = 1$ for which $x_o = 6$. In the straightforward application of transformation #1 to this example, it is to be noted that the replacement of the original problem formulation by the QUIP(PEN) model did not involve the introduction of new variables. In many applications, transformation #1 and transformation #2 can be used in concert to produce an equivalent QUIP model, as demonstrated next.

Example 2 P-Median Problem:
The P-Median problem can be modeled as: : can be modeled as:

$$\min \; x_o = \sum_{i=1}^{m} \sum_{j=1}^{n} c_{ij} x_{ij},$$

subject to:

$$\begin{aligned} \sum_{j=1}^{n} x_{ij} &= 1, \quad i = 1 \ldots, m, \\ \sum_{j=1}^{n} y_j &= p, \\ x_{ij} &\leq y_j, \quad \forall\, (i,j) \text{ pairs.} \end{aligned}$$

where c_{ij} is the weighted distance from facility i to demand node j, $y_j = 1$ if a facility is located at location j, and $x_{ij} = 1$ if demand node i is assigned to the facility at location j.

The first two sets of constraints can clearly be accommodated by transformation #1. The last set of constraints can be handled by transformation #2 by a "trick" of replacing the y variables by their compliments. (This same approach can be employed to model many fixed charge problems.)

To illustrate, consider the 12 variable example with $m = n = 3, p = 2$ and the C matrix:

$$C = \begin{pmatrix} 0 & 2 & 3 \\ 2 & 0 & 1 \\ 3 & 1 & 0 \end{pmatrix}.$$

[2]Almost any method will work for this example.

For $P = 20$, the additive constant $\boldsymbol{c}$ is 80 and the matrix $\hat{Q}$ for the equivalent QUIP model is:

$$\begin{pmatrix}
-20 & 20 & 20 & 0 & 0 & 0 & 0 & 0 & 0 & 10 & 0 & 0 \\
20 & -18 & 20 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 10 & 0 \\
20 & 20 & -17 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 10 \\
0 & 0 & 0 & -18 & 20 & 20 & 0 & 0 & 0 & 10 & 0 & 0 \\
0 & 0 & 0 & 20 & -20 & 20 & 0 & 0 & 0 & 0 & 10 & 0 \\
0 & 0 & 0 & 20 & 20 & -19 & 0 & 0 & 0 & 0 & 0 & 10 \\
0 & 0 & 0 & 0 & 0 & 0 & -17 & 20 & 20 & 10 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 20 & -19 & 20 & 0 & 10 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 20 & 20 & -20 & 0 & 0 & 10 \\
10 & 0 & 0 & 10 & 0 & 0 & 10 & 0 & 0 & -20 & 20 & 20 \\
0 & 10 & 0 & 0 & 10 & 0 & 0 & 10 & 0 & 20 & -20 & 20 \\
0 & 0 & 10 & 0 & 0 & 10 & 0 & 0 & 10 & 20 & 20 & -20
\end{pmatrix}.$$

Solving QUIP(PEN) gives $x_1 = x_6 = x_9 = y_1 = y_3 = 1$ for which $x_o = 1$, which is optimal for the original problem.

4. SOLUTION APPROACHES

Due to its computational challenge and application potential, QUIP has been the focus of a considerable number of research studies in recent years, including both exact and heuristic solution approaches. Notable recent studies addressing QUIP are those by (Williams, 1985), (Pardalos and Rodgers, 1990), (Boros et al., 1989), (Chardaire and Sutter, 1994), (Glover et al., 1998),(Glover et al., 1999a), (Alkhamis et al., 1998), (Beasley, 1999), (Lodi et al., 1997), (Amini et al., 1999), and (Glover et al., 1999a). Other promising work is reported by (Katayama et al., 2000) and (Merz and Freisleben, 1999). These various studies approach the problem by branch and bound, decomposition, tabu search, simulated annealing, and evolutionary methods such as genetic algorithms and scatter search. Each of these approaches exhibits some degree of success. However, the exact methods degrade rapidly with problem size, and have meaningful application to general QUIP problems with no more than 100 variables. For larger problems, heuristic methods are required. Two methods we have found to be particularly successful for a wide variety of problems are based on tabu search and on the related evolutionary strategy scatter search (Amini et al., 1999). In the following we highlight our tabu search approach.

4.1 TABU SEARCH OVERVIEW

Our TS method for QUIP is centered around the use of strategic oscillation, which constitutes one of the primary strategies of tabu search. The variant of strategic oscillation we employ may be sketched in overview as follows.

The method alternates between constructive phases that progressively set variables to 1 (whose steps we call "add moves") and destructive phases that progressively set variables to 0 (whose steps we call "drops moves"). To control the underlying search process, we use a memory structure that is updated at critical events, identified by conditions that generate a subclass of locally optimal solutions. Solutions corresponding to critical events are called critical solutions. A parameter span is used to indicate the amplitude of oscillation about a critical event. We begin with span equal to 1 and gradually increase it to some limiting value. For each value of span, a series of alternating constructive and destructive phases is executed before progressing to the next value. At the limiting point, span is gradually decreased, allowing again for a series of alternating constructive and destructive phases. When span reaches a value of 1, a complete span cycle has been completed and the next cycle is launched.

Information stored at critical events is used to influence the search process by penalizing potentially attractive add moves (during a constructive phase) and inducing drop moves (during a destructive phase) associated with assignments of values to variables in recent critical solutions. Cumulative critical event information is used to introduce a subtle long term bias into the search process by means of additional penalties and inducements similar to those discussed above. A complete description of the framework for the method is given in (Glover et al., 1999c).

5. COMPUTATIONAL EXPERIENCE

Our results of applying the tabu search and associated scatter search metaheuristics to combinatorial problems recast in QUIP form have uniformly attractive in terms of both solution quality and computation times. As intimated earlier, although our methods are designed for the completely general form of QUIP, without any specialization to take advantage of particular types of problems reformulated in this general representation, our outcomes have typically proved competitive with or even superior to those of specialized methods designed for the specific problem structure at hand. Our broad base of experience with QUIP as a modeling and solution framework includes a substantial range of problem classes including:

- Quadratic Assignment Problems
  Capital Budgeting Problems
- Multiple Knapsack Problems
- Task Allocation Problems (distributed computer systems)
- Maximum Diversity Problems
- P-Median Problems

- Asymmetric Assignment Problems
- Symmetric Assignment Problems
- Side Constrained Assignment Problems
- Quadratic Knapsack Problems
- Constraint Satisfaction Problems (CSPs)
- Set Partitioning Problems
- Fixed Charge Warehouse Location Problems
- Maximum Clique Problems
- Maximum Independent Set Problems
- Maximum Cut Problems
- Graph Coloring Problems
- Graph Partitioning Problems

Details of our experience with these and other problems are documented in the paper by (Kochenberger et al., 1998). We are currently solving problems via QUIP with more than 10,000 variables in the quadratic representation. The significance of this is underscored by that fact that the well-known transformation of the binary quadratic representation into a binary linear programming representation produces problems containing more than 50,000,000 zero-one variables. Currently we are working on enhancements that will permit larger instances to be solved.

6. SUMMARY

We have demonstrated how a variety of disparate combinatorial problems can be solved by first re-casting them into the common modeling framework of the unconstrained quadratic binary program. Once in this unified form, the problems can be solved effectively by adaptive memory tabu search metaheuristics and associated evolutionary (scatter search) procedures.

Our findings challenge the conventional wisdom that places high priority on preserving linearity and exploiting specific structure. Although the merits of such a priority are well-founded in many cases, the QUIP domain appears to offer a partial exception. In forming QUIP(PEN), we destroy any linearity that the original problem may have exhibited. Moreover, any exploitable structure that may have existed originally is "folded" into the $\hat{Q}$ matrix, and the general solution procedure we apply takes no advantage of it. Nonetheless, our solution outcomes have been remarkably successful, yielding results that rival the effectiveness of the best specialized methods.

This combined modeling/solution approach provides a unifying theme that can be applied in principle to all linearly constrained quadratic and linear programs in bounded integer variables, and the computational findings for a broad spectrum of problem classes raises the possibility that similarly successful results may be obtained for even wider ranges of problems. As the research community continues to provide improved solution methodologies for the QUIP model, the unified framework that QUIP represents for modeling and solving combinatorial problems via reformulation will become an increasingly attractive alternative to traditional specialized representations. These developments, with their apparent promise, open up a new set of research challenges and opportunities for the optimization community.

ACKNOWLEDGEMENTS

The authors would like to give credit to Anil Menon for his comments that led to an improved paper (as well as his coining of the "QUIP" terminology). We would also like to acknowledge the support we have received for our work via ONR grants N000140210151 and N000140010598.

REFERENCES

Alidaee, B., Kochenberger, G., and Ahmadian, A. (1994). 0-1 quadratic programming approach for the optimal solution of two scheduling problems. *International Journal of Systems Science,* 25:1–408.

Alkhamis, T. M., Hasan, M., and Ahmed, M. A. (1998). Simulated annealing for the unconstrained binary quadratic pseudo-boolean function. *European Journal of Operational Research,* 108:641–652.

Amini, M., Alidaee, B., and Kochenberger, G. (1999). A scatter search approach to unconstrained quadratic binary programs. In Come, Dorigo, and Glover, F., editors, *To appear in New Methods in Optimization.* McGraw-Hill Publishers.

Beasley, J. E. (1999). Heuristic algorithms for the unconstrained binary quadratic programming problem. Working Paper, Imperial College.

Boros, E., Hammer, P., and Sun, X. (1989). The ddt method for quadratic 0-1 minimization. Technical Report RRR 39-89, RUTCOR Research Center.

Chardaire, P. and Sutter, A. (1994). A decomposition method for quadratic zero-one programming. *Management Science*, 4:704–712.

Gallo, G., Hammer, P., and Simeone, B. (1980). Quadratic knapsack problems. *Mathematical Programming,* 12:132–149.

Glover, F., Amini, M., Kochenberger, G., and Alidaee, B. (1999a). A new evolutionary metaheuristic for the unconstrained binary quadratic programming: A case study of the scatter search. Technical report, School of Business, University of Colorado, Boulder.

Glover, F., Kochenberger, G., and Alidaee, B. (1998). Adaptive memory tabu search for binary quadratic programs. *Management Science*, 44:336–345.

Glover, F., Kochenberger, G., Alidaee, B., and Amini, M. (1999b). Unconstrained quadratic binary program approach to quadratic knapsack problems. Working Paper, Hearin Center for Enterprise Science, University of Mississippi.

Glover, F., Kochenberger, G., Alidaee, B., and Amini, M. (1999c). Tabu with search critical event memory: An enhanced application for binary quadratic programs. In Voss, S., Martello, S., Osman, I., and Roucairol, C., editors, *Meta-Heuristics: Advances and Trends in Local Search Paradigms for Optimization.* Kluwer Academic Publisher, Boston.

Hammer, P. and Rudeanu, S. (1968). *Boolean Methods in Operations Research.* Springer-Verlag, New York.

Hansen, P. (1979). Methods of nonlinear 0-1 programming. *Annals Discrete Math*, 5:53–70.

Harary, F. (1953/54). On the notion of balanced of a signed graph. *Michigan Mathematical Journal,* 2:143–146.

Katayama, K., Tani, M., and Narihisa, H. (2000). Solving large binary quadratic programming problems by an effective genetic local search algorithm. In *Proceedings of the 2002 Genetic and Evolutionary Computation Conference,* San Francisco, CA. Morgan Kaufmann.

Kochenberger, G., Glover, F., Alidaee, B., and Rego, C. (1998). Applications of the unconstrained quadratic binary program. Working Paper, University of Colorado.

Krarup, J. and Pruzan, A. (1978). Computer aided layout design. *Mathematical Programming Study*, 9:75–94.

Laughunn, D. J. (1970). Quadratic binary programming. *Operations Research,* 14:454–461.

Lodi, A., Allemand, K., and Liebling, T. M. (1997). An evolutionary heuristic for quadratic 0-1 programming. Technical Report OR-97-12, D.E.I.S., University of Bologna.

McBride, R. D. and Yormack, J. S. (1980). An implicit enumeration algorithm for quadratic integer programming. *Management Science,* 26:282–296.

Merz, P. and Freisleben, B. (1999). Genetic algorithms for binary quadratic programming. In *Proceedings of the 1999 International Genetic and Evolutionary Computation Conference (GECCO '99),* pages 417–424. Morgan Kaufmann.

Pardalos, F. and Xue, J. (1994). The maximum clique problem. *The Journal of Global Optimization,* 4:301–328.

Pardalos, P. and Rodgers, G. P. (1990). Computational aspects of a branch and bound algorithm for quadratic zero-one programming. *Computing,* 45:131–144.

Pardalos , P. and Rodgers, G. P. (1992). A branch and bound algorithm for maximum clique problem. *Computer & OR*, 19:363–375.

Phillips, A. T. and Rosen, J. B. (1994). A quadratic assignment formulation of the molecular conformation problem. *The Journal of Global Optimization*, 4:229–241.

Williams, A. C. (1985). Quadratic 0-1 programming using the roof duality with computational results. Technical Report Rutcor Research Report 8-85, Rutgers University, New Brunswick, NJ.

Witsgall, C. (1975). Mathematical methods of site selection for electronic system (ems). NBS Internal Report.

This page intentionally left blank

Chapter 6

PROBLEMS IN OPTIMIZATION

William G. Macready
Research Institute for Advanced Computer Science
Computational Sciences Division
NASA Ames Research Center
Mail Stop 269-4
Moffett Field, CA 94035-1000
wgm@email.arc.nasa.gov

Abstract A series of problems and challenges is posed to help guide future work in optimization.

Keywords: optimization foundations, optimization connections, optimization applications

1. INTRODUCTION

At the beginning of the last century, David Hilbert presented an invited paper to the Second International Congress of Mathematicians, framing what he saw as the future of mathematics (Hilbert, David, 1902). In his paper, Hilbert posed 23 important and then unsolved mathematical problems. These problems formed the backbone of much of 20th century mathematics, and today most of these problems have been at least partially solved.

Our scope here is more modest, though the problems are no less difficult. *What are the problems and challenges that must be addressed in the next century of work in optimization?*[1] Like mathematics in 1900, optimization has a solid foundation upon which to build, and it has no shortage of deep unanswered questions. Thus, I expect significant progress in both the short and long terms.

Since there is no shortage of problems that researchers in optimization can address, it is important to enumerate desiderata to limit the selection of problems. Our most important criterion is that the problem should be rich enough

[1]See (CONDOR, 1988) for an operations research perspective on a similar question.

to lead to new problems and insights. As Hilbert himself said, "As long as a branch of science offers an abundance of problems, so long is it alive; a lack of problems foreshadows extinction..." (Hilbert, David, 1902). Indeed I expect that the answers to the questions posed here have answers with implications beyond optimization itself. A second requirement reflects on the utility of the question. Since optimization is preeminently a practical field, the solution to any problem should be of practical utility. Theory should ultimately support the development of algorithms. We should not, however, use this requirement to limit ourselves to purely immediate engineering concerns. Who could have foreseen that the theory of computational complexity in the 1960s would ultimately result in polynomial-time algorithms for linear programming in the 1980s (Karmarkar, 1984), and general polynomial-time interior-point methods for convex problems in the 1990s (Nesterov, Yurii and Nemirovskii, Arkadii, 1994)? Finally, we require problems that are difficult, and therefore interesting, but will also allow for progress. Problems may in fact be *very* difficult so long as they suggest a course of action with tractable stepwise challenges.

The problems posed here are divided into three categories: foundations, connections and applications. In 1900 Hilbert was interested in axiomatizing all of mathematics (and physics!). Though we now know through the work of Godel, Turing, and others, that Hilbert's grand hopes cannot be fulfilled, it is essential to examine the theoretical foundations of optimization. It is exciting for present-day researchers that at the core of optimization there are mostly unanswered questions. Another category of questions focuses on the connections of optimization to other fields and the discovery of synergies between these fields. We focus on the relation between work in machine learning and optimization. Finally, we conclude with a series of future novel applications of optimization.

2. FOUNDATIONS

If there is any established principle in optimization (whether discrete and/or continuous, constrained or unconstrained, single- or multi-objective), it is the need to tradeoff between exploration and exploitation. For our purposes we express this principle as the problem of discovering and exploiting the structure inherent in any particular optimization task. Recent work has formalized the futility of optimization without assumptions about the structure of the problem at hand (Wolpert and Macready, 1997). The first problem concerns a formalization of this most basic tenet.

Problem 1: *Investigate the essence of the optimization problem through a general formalization of: (1) the notions of problem structure, (2) a broadly applicable definition of a search algorithm, (3) measures of algorithm efficiency,*

and (4) the means whereby the search algorithm exploits the problem structure efficiently.

This formalization should be general enough to capture a broad range of real-world optimization tasks but not so abstract as to yield no concrete insight. Consequently, it is probably best to initially limit the scope to single-objective cost functions $f : \mathcal{X} \mapsto \mathbb{R}$ which map discrete and/or continuous configurations $x \in \mathcal{X}$ to real cost values $y \in \mathbb{R}$. We make no explicit mention of constraints, $\mathcal{C}$, since these can be incorporated through an appropriate definition of the search space $\mathcal{X}$.[2] Likewise it is probably best to begin this program focusing on a smaller class of possible algorithms a. A natural class of commonly applied algorithms is one which determines new populations of points (x, y) from previously sampled populations.[3] Generally then, we consider an optimization algorithm to be an iterated application of two basics steps: an infer step which makes inferences about the structure of a problem, and an act step which uses the inference to determine where next to sample. We represent these basic steps by

$$\textbf{infer -} \qquad P(y|x, I, \mathcal{D}, \mathcal{G}) \tag{6.1}$$

$$\textbf{act -} \qquad a : \left(P(y|x, I, \mathcal{D}, \mathcal{G}), \mathcal{D}, \mathcal{G}, \mathcal{C}\right) \mapsto \mathcal{P}_{\mathcal{X}\setminus\mathcal{D}_x}(x) \tag{6.2}$$

where I indicates any prior information that may be available, $\mathcal{G}$ indicates any neighborhood structure imposed on the search space $\mathcal{X}$, $\mathcal{D} = \{\mathcal{D}_x, \mathcal{D}_y\}$ is list of points, $\mathcal{D}_x$, and their fitnesses, $\mathcal{D}_y$, sampled from the problem ordered according to the time at which they were sampled, and $\mathcal{C}$ indicates any associated constraints. Structure is represented by a probability distribution over the y values at sampled and unsampled points x, and the action step uses this inference, the location of previously sampled points, the neighborhood structure, and constraints to sample at new configurations. To be general, we write this sample as coming from a probability distribution $\mathcal{P}_{\mathcal{X}\setminus\mathcal{D}_x}(x)$ over unsampled $\mathcal{X}$ values.

What can we expect if this endeavor is carried out? Most importantly we will better understand what it means for an algorithm to be well-suited to solving a particular class of problems. It would also be surprising if this program did not suggest new types of algorithms which differ in the manner in which they represent and exploit problem structure. We would better understand the differences between the myriad algorithms and heuristics currently in use. Hopefully, this would lead to insight into the construction of efficient

[2]Of course, in a practical sense this solves nothing since $\mathcal{X}$ may be sufficiently complex that even finding a feasible $x \in \mathcal{X}$ may be NP complete. However, the present problem is concerned entirely with understanding the relationship a problem, it's structure, and algorithms for exploiting that structure.

[3]Populations may contain only a single point.

(according to some criteria) algorithms. This itself suggests the next fundamental problem.

Problem 2: *Given a characterization of an optimization task through its structure as defined in Problem 1, develop a procedure to programmatically generate an efficient algorithm.*

Both of these problems represent formidable challenges, yet I believe that they still satisfy our requirement of stepwise tractability. How might a program of research begin to attack these problems? A first step is the realization that a characterization of structure will likely not completely describe the objective function or class of objective functions. Of necessity the formalization must operate without complete knowledge of all facets of the optimization task. Certainly we don't know the maxima! Consequently, the formalization will be based on probability theory and structure will likely be represented (either explicitly or implicitly) through probability densities of the form $P_{\mathcal{X},\mathbb{R}}(x, y)$. Even these small steps towards recognizing what the formalization must ultimately look like suggest interesting avenues. The geometric structure that automatically accompanies the probabilistic setting (Amari, 2000) allows us to speak of nearby structures. It may not be unrealistic to expect a geometric interpretation of the coupling between problems, their structures, and efficient algorithms.

Without doubt, these first two problems are the most important I shall describe, and their correct formulation and solution will require input from less ambitious programs. I describe two concrete problems, the first of which should provide insight into the characterization of structure, and the second of which should provide guidance into the construction of efficient algorithms from structure specification.

Problem 3: *Given a pair of search algorithms a_1 and a_2, construct optimization tasks (either mappings $f : \mathcal{X} \mapsto \mathbb{R}$, or structure characterizations) for which a_1 is efficient and a_2 is inefficient. Then, construct optimization tasks for which a_1 is inefficient while a_2 is efficient. Characterize the differences between these optimization tasks and relate them back to differences in a_1 and a_2.*

The first part of this problem which asks for the optimization problems is straightforward (at least for small optimization problems). We expand the objective function in some convenient basis (e.g. a Fourier basis for continuous spaces or a Walsh basis for sequence spaces): $f(x) = \sum_i w_i \phi_i(x)$. The basis may by undercomplete, complete, or overcomplete. The objective function is defined by the expansion coefficients w_i. Note that we can limit the objective functions to a smaller class of objective functions, say for example symmetric travelling salesperson problems, through a suitable choice of an undercomplete basis.

If $M(a_1, a_2|f)$ is a measure of the efficiency of algorithm a_1 over a_2[4] on objective f, then we can find f's for which a_1 performs well but a_2 does not by maximizing $M(a_1, a_2|f)$ with respect to the expansion coefficients w_i which define f. We label the set of such objectives $\{f\}_{a_1>a_2}$. Similarly, we can construct problems $\{f\}_{a_1<a_2}$ for which a_1 performs much more poorly than a_2 by maximizing $M(a_2, a_1|f)$.

Given the problem sets $\{f\}_{a_1>a_2}$ and $\{f\}_{a_1<a_2}$ a characterization of the important differences between the sets of functions is difficult, but there are many machine learning algorithms that could be applied to the task. We might first explore simple clustering algorithms to see if the coefficients from each class tend to cluster near each other. Alternatively, classification algorithms could attempt to predict the class $a_1 > a_2$ or $a_1 < a_2$ based on the expansion coefficients. Whatever the form of the characterization of the two sets of objectives, we would then attempt to relate this back to characteristics of the two search algorithms a_1 and a_2.

This procedure, if successfully carried out for many pairs of commonly used algorithms, should yield valuable insight into the differences between common algorithms. An interesting related question may follow from such studies:

Problem 4: *Develop a framework in which to describe existing optimization algorithms and which is general enough to encompass new algorithms.*

There are multiple tasks buried within this problem. For the infer step:

Problem 5: *Is there a broadly applicable manner in which to represent and exploit problem structure including structure specified a priori or inferred through sampling from the objective?*

For the act step:

Problem 6: *Is there a general way to structure and parameterize the manner in which new points are sampled based on the inferences of Problem 5?*

The proper framework in which to consider act mappings will be informed by many examples of the ways in which algorithms exploit problem structure. I pose an instance of this exploitation problem for a particular problem structure which may be exactly solvable and is easily generalizable to more complex objective function structures.

Problem 7: *Given that a one dimensional objective function was generated by a simple Brownian random walk, determine an algorithm which is efficient in quickly locating the maximum of the objective.*

This problem is appealing in its apparent simplicity and much is known about the distribution of peaks and valleys on such surfaces (Mansour, 2002). The solution to this problem should be of interest to the optimization practitioner since the solution is likely to be generalizable to many more interesting

[4]Examples of such measures can be found in (Macready and Wolpert, 1996).

problems. The surface generated by the random walk may be generalized to n dimensions and/or biased in various ways. Such surfaces are easily generated (Massopust, 1995) so that extensive empirical tests may be performed.

The solution to problem 7 may be applied to a more realistic (and accordingly difficult) problem:

Problem 8: *Given a detailed knowledge of problem structure through a Gaussian process, construct an efficient algorithm tailored to this structure.*

Gaussian processes are simple ways to specify the relationships between objective values at different points in the search space (MacKay, 2003). They are adaptable to many different types of search spaces whether continuous or discrete. Any problem structure implicitly contains a relationship between the objective values at different x values. For example, a very smooth objective function has very similar objective values for nearby x values. Gaussian processes build directly on this notion by specifying a probability distribution over objective values $y_1 = f(x_1)$ and $y_2 = f(x_2)$ at two different x values. To keep things as simple as possible imagine that our objective function has been scaled and shifted so that it has an average value of 0 and a variance of 1. As the name implies, Gaussian processes describe the probability distribution with a Gaussian:

$$P(y_1, y_2) \sim \exp\left(-\frac{1}{2}\begin{bmatrix} y_1 & y_2 \end{bmatrix}\begin{bmatrix} C_{x_1,x_1} & C_{x_1,x_2} \\ C_{x_2,x_1} & C_{x_2,x_2} \end{bmatrix}^{-1}\begin{bmatrix} y_1 \\ y_2 \end{bmatrix}\right).$$

The elements in the covariance matrix $\mathbf{C}_{x_1,x_2}$ are given by the expected value: $\mathrm{E}(y(x_1)y(x_2))$. If the search space is continuous and the objective function is relatively smooth, a common form for the covariance matrix elements is:

$$C_{x_i,x_j} \sim \exp[-(x_i - x_j)^2].$$

This form expresses the property that the correlation in fitness at two different x values decays exponentially with the distance between the points. As hinted at in this simple example, Gaussian processes can compactly describe a wide range of structure that may be present in the fitness function. In fact, a family of landscapes based on this idea can be formed, many of whose properties can be determined analytically. If we could design algorithms to exploit this structure we would be a long way towards having efficient and practical new search algorithms.

3. CONNECTIONS

In this section we move from questions concerning the foundations of optimization to ideas from other fields that optimization may draw upon. Given our perspective that optimization is an exercise in inferring and capitalizing on problem structure, it is not surprising that machine learning offers many

insights into better optimization. While the bulk of this section explores connections with machine learning, we also pose questions seeking to integrate different approaches to optimization.

The field of machine learning seeks to embody the learning process in computers. A complete introduction to machine learning may be found in (Duda et al., 2001). Machine learning methods may usefully be categorized into three types of learning, all of which are applicable to optimization.

Unsupervised learning seeks to uncover structure in unlabelled data. A prototypical problem is density estimation, whereby given a set of data $\{x_i\}$ we try to determine the probability density that may have generated the data. As a limiting case, clustering algorithms which group the data into like aggregations are a common tool. As alluded to in Problem 3, unsupervised learning algorithms may help us to uncover exploitable structure in problems.

Supervised learning is another branch of machine learning which tries to learn relationships between factors. The prototypical problem here is to learn the mapping from x to y given a set of labelled data $\{x_i, y_i\}$. This is an important problem allowing for the prediction of output (y) values at new input (x) points. The output may either be continuous or discrete resulting in either regression or classification tasks. Supervised learning algorithms may be used to model the structure in a problem and to suggest means of exploiting the structure.

A third branch of machine learning is called reinforcement learning. Reinforcement learning is concerned with maximizing a reward function in an unknown and potentially noisy environment. Dynamic programming is a well-studied kind of reinforcement learning which is useful when the search space is small and discrete. Of these three tasks, reinforcement learning problems are generally the most difficult. Reinforcement learning problems share much in common with optimization through the need to balance exploration with exploitation. The k-armed bandit problem, which many researchers in evolutionary optimization are familiar with, may be readily approached through reinforcement learning methods.

The modern probabilistic approach to machine learning has provided researchers a common language in which to describe a wide variety of learning algorithms. Graphical models (Jordan, M., 1999) allow for efficient probabilistic inference by exploiting the dependency structure between variables. Perhaps this same machinery can also be exploited for optimization.

Problem 9: *Can new algorithms be developed by exploiting a probabilistic framework for optimization ? In particular, what novel insights are obtained by viewing optimization as an inverse problem?*

At present there is increasing interest in using graphical models to encode dependencies between optimization variables. See (Larrañaga et al., 1999; Mühlenbein et al., 1998) for a summary of recent work. This work is promis-

ing in that it explicitly represents aspects of problem structure as computationally efficient probabilistic models. While this avenue of work is likely to continue to grow and yield important results, I will focus on another aspect of the application of probability theory to optimization.

Admitting the possibility of noise contaminating an objective function, it is convenient to write the objective function as the probability distribution $P(y|x)$. This gives the distribution of objective values at a given x value.[5] We can view optimization as an inverse problem if we take the perspective that the goal of any optimization algorithm is to determine $P(x|y)$ given the objective $P(y|x)$. This is not a common view of optimization, but if we did have access to $P(x|y)$ we could sample from this distribution at whatever y value we wanted to obtain a solution for, and obtain an x value likely to generate this y. In essence, determining the "inverse" distribution would solve our problem. Of course, the above argument is merely suggestive and leaves much unsaid.[6] Nevertheless I believe it will be fruitful to look at optimization from this novel perspective.

This view of optimization suggests that probabilistic inference should play an important role in optimization. In fact, we are already beginning to see this through the nascent unification of constraint programming and integer programming. In (Hooker, 2000) it is shown that important improvements to optimization algorithms can be gained through logical inference.[7] Constraint programming (Tsang, 1993) is concerned with inferring the values that variables may take on if they are required to satisfy constraints. For example, if binary variables x_1 and x_2 are related by the constraint $x_1 \vee x_2$ and we know that $x_1 = 0$ then we can infer in any feasible solution that $x_2 = 1$. In such ways the search space can be trimmed dramatically. Such inferences are logical in that they determine the allowed values other variables may take precisely. These efforts may be generalized to probabilistic inference where we refine the distribution over $P(x|y, \mathcal{C})$ as information obtained during the search becomes available.

We leave probability theory and turn to an important theoretical development in supervised learning that inspired a new class of successful algorithms – the bias-variance tradeoff. The bias-variance tradeoff decomposes the error of any supervised learning algorithm into two contributions - a bias contribution which measures the match between an algorithm and the true target, and a variance contribution which measures the specificity of the match (Geman et al., 1992). The two contributions are not independent, but the result is

[5]If the objective is not noisy then $P(y|x) = \delta(y - f(x))$ where $\delta(\cdot)$ is Dirac's delta function.

[6]In fact the density $P(x|y)$ is not even strictly defined since it is not normalizable for y values outside the range of objective values. However, such problems are easily addressed.

[7]There have been conferences on the integration of AI and OR techniques since 1999. See the website www.crt.umontreal.ca/cpaior/ for the 2003 installment.

important because it leads to new algorithms. Researchers realized that performance could be improved by combining learning algorithms which drive the variance term down without significantly affecting the bias term. Many ways of combining learning algorithms have arisen in recent times (stacking, bagging, boosting, etc.) based on understanding of the bias-variance tradeoff. If a similar procedure could be carried out for optimization algorithms, results would likely also be dramatic. Thus we are led to another important problem.

Problem 10: *Develop theory and algorithms which allow for the combining of different optimization algorithms into a single algorithm which has better performance than all of its constituents.*

Using existing work in machine learning, it is clear how to combine algorithms for improved inference of the structure in a problem (the infer step). However, even given a common framework in which to represent problem structure (see Problem 4), it is not at all clear how to combine optimization procedures which exploit this structure (the act step). This should be an area of significant future effort.

We close this section of connections between machine learning and optimization by examining the application of reinforcement learning (RL) to a particular class of optimization tasks. Due to the similar nature of the tasks, there are many opportunities for cross-fertilization of optimization and RL including the rewarding of valuable stage-setting activities (e.g. crossing a valley of low fitness) or through rewarding important components of any good solution. Some researchers are beginning to explore such connections either directly using RL (Boyan and Moore, 2003) or indirectly through heuristics (Dorigo, M. et al., 1999). While such activities are rife with new questions, I would like to propose another currently unstudied problem related to reinforcement learning.

Accompanying the development of inexpensive computing power has been a rise in the application and sophistication of simulation models. From weather prediction and galaxy formation to consumer modelling and crowd dynamics, simulations are ubiquitous and will only become more prevalent with time. Often, however, the simulation itself is only a means to an end. Once a calibrated simulation model has been constructed, some of its input parameters need to be tuned in order to understand how to effect some desired outcome (e.g. what should the layout of a store be in order to maximize the time consumers spend in the store?). A common approach to this problem is to define an objective function and recast the problem as a minimization. For any given run of the simulation, the objective measures the distance of the obtained end state from the desired end state. By applying the minimization algorithm, the outer level optimization hopefully finds parameters which result in small differences from the desired simulation outcome. This approach, while conceptually straightforward, is horribly inefficient. In most cases the simulation itself is both computationally expensive and stochastic making this direct approach to the

problem infeasible. Since this is likely to become an increasingly common use of optimization, I have posed this as an important future problem.

Problem 11: *Many optimization problems require tuning the parameters of a simulation or dynamical system so that the system exhibits desired behavior. Increase the efficiency of this process by using reinforcement-learning-like methods to exploit the information available in the dynamics of the simulation.*

The naive optimization approach outlined above is wasteful of the information available to tune the simulation parameters. The dynamics of the simulation are not taken into account; only its final attractor which is compared with the desired result is considered. Surely, improved performance could be obtained by monitoring the behavior of the simulation before it attains its end state. There is reason to believe that ideas from reinforcement-learning could prove helpful, since there is a direct mapping of this problem onto the problem solved by reinforcement learning. The value of the objective function can serve as input into the reward function with valuable stage-setting dynamics being rewarded prior to the end of the algorithm.

To close this section, I end with an important connection between optimization and the scientific method itself. A disturbing cultural trend within the optimization community is the preponderance of head-to-head competitions of algorithms against one another. No paper which proposes a new algorithm is complete without a comparison against other algorithms on a tiny set of problems. Often these problems are not even benchmark problems, but rather problems the author has defined to show the strengths of the new algorithm. This is all the more distressing since we know that algorithms behave the same when averaged over all possible problems (Wolpert and Macready, 1997). A comparison of two algorithms merely showing which one is better is almost useless without understanding *why* it is better than the other. Some other less obvious but equally pernicious aspects of the current comparative culture are convincingly discussed in (Hooker, 1996). Thus, researchers need to seriously consider the following problem.

Problem 12: *Develop test problems and methodologies for a more "scientific" establishment of the strengths and weakness of different algorithms.*

Algorithm comparison is not unimportant, but we desperately need new classes of test problems. The goal of these test problems is to provide insight as to why one algorithm performed better than the other on these problems. Ideally, these problems should have features that are deemed important in determining the success of algorithms run on these problems, and these features should be easily tunable. The types of features and the form of the test functions can be informed from the results of Problem 4 for a unified modelling and testing framework.

However, to determine why an algorithm performed well or poorly requires a change in research emphasis. After an algorithm has been developed, more

time should be spent on hypotheses concerning its efficacy on different problems and less on comparing it with other algorithms. Empirical tests of these hypotheses should be performed and hypotheses adjusted accordingly. In short, we should apply the scientific method itself when it comes to the comparison of algorithms.

4. APPLICATIONS

In this final section, I pose problems in the applications of optimization rather than problems in optimization itself. The first two of these problems are driven by the growth of the internet.

Problem 13: *Design distributed, bottom-up optimizing agents which optimize globally using predominantly local information.*

Interest in multi-agent systems has exploded in recent times. Researchers are trying to engineer distributed systems solving a wide variety of different problems including optimization (Wolpert et al., 2003). I expect that such efforts will become more important over time due to the practical value in solving this problem. As an example, one of the most important applications of optimization from a financial perspective is the optimization of the supply chains which link businesses together. With the internet and electronic commerce, businesses are now linked more closely than ever. Past supply chain gains made by operational researchers which optimized locally, improving a single step within a single business, have been exhausted. The future of additional efficiency improvements is in optimizing across an entire supply chain. Unfortunately, much of the global optimization must be accomplished using local information because companies are unwilling to divulge proprietary information. Moreover, even if the information is available, it rapidly becomes obsolete in the modern volatile business climate.

The internet has also enabled the development of avatars or human surrogates which act to fulfill the requests of humans. There are two steps in accomplishing this: 1) eliciting the preferences from the human and 2) acting to maximize those preferences. Preferences can conveniently be captured in one or more utility functions.[8] The second step is likely to be an important future application for optimization methods. Despite significant previous effort (Keeney and Raiffa, 1993) elicitation of preferences remains problematic and no entirely satisfactory solution is available.

Problem 14: *Develop algorithms and procedures whereby qualitative objective functions (in particular utility functions) can be quantified or otherwise represented so that they may be optimized.*

[8]Though admittedly, not without some drawbacks.

As suggested in the problem statement, it may not be necessary to formally represent the objective function as an algorithm. We have seen many interesting examples where evolutionary algorithms have been used to construct aesthetically appealing designs utilizing user-defined fitness functions. Other iterative approaches might allow the human to guide the search as it progresses. If such techniques can be extended to easily capture human desires into computationally-convenient forms, new applications will abound.

I close with an open-ended suggestion rather than a specific problem. The development of genetic programming (Koza, 1992) in the early 1990s was important because it brought attention to optimization on a much larger class of problems. Today, we are no longer confined to optimize over traditional search spaces like $\mathbb{R}^n$ but can also consider problems defined over bizarre search spaces like fragments of computer code. In spite of these new applications I believe we have only begun to explore new applications of optimization.[9]

Problem 15: *Explore novel applications of optimization by considering more general notions of search spaces and constraints.* As a concrete example of new applications, we can look to mathematics. Formal systems in mathematics provide fertile opportunities for novel optimization applications. In most cases, the formal systems[10] defined by mathematicians and computer scientists used to study particular problems can easily be encoded in an algorithmically convenient manner. Defining an objective function over these formal systems will result in entirely new classes of optimization problems. We might, for example, want to build a formal system having certain characteristics. If we can define an objective quantifying the difference between any formal system and one with the attributes we desire then we can optimize to find such structures. Of course, the optimization will only be successful if our representation of the problem results in a relatively smooth objective landscape. As a concrete example of the power of this approach, we might use some of the algebras that have been developed to model concurrency in computer systems (Milner, 1999). There is no reason why, in principle, we cannot optimize over representations of mobile concurrent processes[11] to construct processes that perform certain tasks in as parallel a manner as possible.

New applications like these will surely bring new questions. How do we even begin to think theoretically of the properties of such spaces when the configurations represent complex formal systems? How do we combat the

[9]Interestingly, the field of machine learning is now exploding with new applications by using kernel methods which map many different input spaces (e.g. text documents, bioinformatic sequences, phylogenetic trees, rankings, etc.) to linear vector spaces where the learning algorithms operate. See www.kernel-machines.org for more information.

[10]These formal systems may be either discrete (discrete groups, algebras, graphs, logics, calculi) or continuous (continuous groups, algebras, vector spaces).

[11]Mobile processes can change their patterns of interaction over time.

entropic force resulting in ever larger configurations? Work in this direction has begun (a recent example of some theoretical properties of the search space of trees is found in (Bastert et al., 2001)), and will become more important.

5. CONCLUSIONS

Optimization has a bright and interesting future if the problems posed here, and the countless others not considered, are any indication. If, in a hundred years, researchers can look back on these problems as we do on David Hilbert's problems seeing most of them solved, optimization will be a vastly more influential field than it is today. The answers to these questions will have spawned myriad new applications and perhaps will have shaped the way we view other disciplines and the natural world itself.

Acknowledgements I would like to thank BiosGroup Inc. and RIACS/NASA for support, and Mohammed El-Beltagy, Stuart Kauffman, Jose Lobo, and Michele Shouse for input.

REFERENCES

Amari, S. (2000). *Methods of Information Geometry,* volume 191 of *AMS Translations of Mathematical Monographs.* AMS and Oxford University Press.

Bastert, O., Rockmore, D., Stadler, P. F., and Tinhofer, G. (2001). Landscapes on spaces of trees. Technical Report SFI preprint 01-01-006, Sante Fe Institute.

Boyan, J. and Moore, A. (2003). Learning evaluation functions to improve local search. Journal of Machine Learning Research, to appear.

CONDOR (1988). Operations research: the next decade. *Operations Research,* 36:619–637. (Committee On the Next Decade in Operations Research).

Dorigo, M., Di Caro, G., and Gambardella, L. (1999). Ant algorithms for discrete optimization. *Artificial Life,* 5:137–172.

Duda, R., Hart, P., and Stork, D. (2001). *Pattern classification.* Wiley, New York.

Geman, S., Bienenstock, E., and Doursat, R. (1992). Neural networks and the bias/variance dilemma. *Neural Networks,* 4:1–58.

Hilbert, David (1902). Mathematical Problems, Paris, 1900. Translation by Mary Winston published in the Bulletin of the American Mathematical Society.

Hooker, J. (1996). Testing heuristics: we have it all wrong. *Journal of Heuristics,* 1:33–42.

Hooker, J. (2000). *Logic-based methods for optimization.* Wiley, New York.

Jordan, M. ed. (1999). *Learning in graphical models.* MIT Press, Cambridge.

Karmarkar, N. (1984). A new polynomial-time algorithm for linear programming. *Combinatorica,* 4:373–395.

Keeney, R. and Raiffa, H. (1993). *Decisions with Multiple Objectives.* Cambridge University Press, Cambridge.

Koza, John R. (1992). *Genetic programming: on the programming of computers by means of natural selection.* MIT Press, Cambridge.

Larrañaga, P., Etxechabarria, R., Lozano, J. A., Sierra, B., Inza, I., and Penna, J. M. (1999). A review of the cooperation between evolutionary computation and probabilistic graphical models. In *Proceedings of the II Second Symposium on Artificial Intelligence,* pages 314–324.

MacKay, D. (2003). Gaussian processes - a replacement for supervised neural networks? available at http://wol.ra.phy.cam.ac.uk/mackay/abstracts/gp.html.

Macready, W. G. and Wolpert, D. H. (1996). What makes an optimization problem hard? *Complexity,* 5.

Mansour, T. (2002). Counting peaks at height k in a Dyck path. unpublished, available at http://arxiv.org/abs/math.CO/0203222.

Massopust, P. (1995). *Fractal functions, fractal surfaces, and wavelets.* Academic Press.

Milner, R. (1999). *Communicating and mobile systems: the π calculus.* Cambridge University Press, Cambridge.

Mühlenbein, H., Mahnig, T., and Ochoa, A. R. (1998). Schemata, distributions and graphical models in evolutionary optimization. *Journal of Heuristics,* 5.

Nesterov, Yurii and Nemirovskii, Arkadii (1994). *Interior-Point Polynomial Algorithms in Convex Programming.* Society for Industrial and Applied Mathematics, Philapelphia.

Tsang, E. (1993). *Foundations of constraint satisfaction.* Academic Press.

Wolpert, D. H., Bandari, E., and Tumer, K. (2003). Improving simulated annealing by recasting it as a non-cooperative game. In preparation.

Wolpert, D. H. and Macready, W. G. (1997). No Free Lunch Theorems for Search. *Transactions on Evolutionary Computation*, 1.

Chapter 7

EC THEORY - "IN THEORY"

Towards a Unification of Evolutionary Computation Theory

Christopher R. Stephens
Instituto de Ciencias Nucleares
UNAM, Mexico
stephens@nuclecu.unam.mx

Riccardo Poli
Department of Computer Science
University of Essex, UK
rpoli@essex.ac.uk

Abstract We present a personal overview of EC theory. In particular, we try to show that recent theoretical developments have pointed the way to a grand unification of different branches of EC, such as Genetic Algorithms and Genetic Programming, and also different theoretical models, such as the Vose model and Holland's Schema theorem. We give a broad outline of this unification program showing how the different elements above are related to each other via changes of representation on the space of EC models. Based on our work we pose a series of challenges which if met, we believe, will lead to a much deeper understanding of EC and the various types of evolutionary algorithm.

Keywords: Schema theory, Vose model, Unification, Genetic Algorithms, Genetic Programming, Evolutionary strategies, Building Blocks

INTRODUCTION

Relatively speaking, Evolutionary Computation (EC) is a fairly immature subject. It exhibits many different facets without a high degree of intellectual consensus. It sometimes seems that it is all things to all people. It is a subject that is principally empirical and phenomenological. Moreover, it is empirical and phenomenological within a very ill defined framework, in distinction to the

world as seen through the eyes of physics and biology. This is because in EC the only real limit to what can be studied is the imagination. In its more "scientific" guise it is related to an older, more mature field - population genetics - but without the constraints that nature imposes and without the same degree of intellectual coherence that comes with maturity. Mathematically speaking it is the study of certain classes of heuristic algorithms based on populations of objects (Vose, 1999), though as we shall see what these classes are is far from clear. Seen from the "engineering" point of view it is an area where the analogical use of "natural selection" appears as a moulding force in the creation of more "competent" problem-solvers (Goldberg, 2002).

One salient characteristic of EC theory is that it is difficult. It is also very exciting. It is relatively simple to write an evolutionary algorithm (EA). It is exceedingly difficult to understand its behavior other than at a superficial level. Even fairly simple EAs, such as a Genetic Algorithm (GA) with selection and mutation only, present formidable difficulties.[1] This stark contrast between the ease with which an EA can be written and the complexity of understanding its behavior leads to a very large expectation gap between EC "practitioners", who focus on the empirical aspects, and often seem capable of thinking of five new genetic operators before breakfast, and the theorists who, to the practitioners, seem fixated on no selection or "counting ones".

Unlike more mature areas of science there is not even a clear consensus on what should be the task of EC theory. Is it to provide recipes for practitioners, to provide exact computational models, to allow a deeper understanding of a complex system, all of these, none of these, or what? Having established what theory should do, it is then important to ask ourselves - "What has it done?" and "Where is it going?" A goal of this article is to give, albeit briefly, our personal view on this.

Different approaches to EC theory have been proposed in the past. These include schema theories (Holland, 1975), the Vose model (Nix and Vose, 1992), the statistical mechanics approach (Prügel-Bennett and Shapiro, 1994) and more. Is there a model that is superior to all others? Often, models are judged by their clarity, simplicity, and ability to explain and predict. Is there a framework that does this best? Once we have established what the task of EC theory should be, it will be easier to answer these questions.

A theoretical model is also often judged by how well it unifies a range of phenomena. As there are many different flavors of EA - GAs, Genetic Programming (GP), Evolution Strategies (ES) etc. - one may ask if there is a

[1] These problems can, in fact, be mapped to problems familiar in very well established branches of science, such as statistical mechanics. Even there, however, where there is vast experience, they remain an enormous challenge.

theoretical framework that encompasses them all? If not, then which is the framework with the broadest applicability?

The framework with the broadest applicability is inhomogeneous Markov chain theory. However, describing EC as a subset of such a theory means very little. So, what are the essential elements common to different EAs? These are: a choice of genotype-phenotype map, a choice of fitness function and a set of evolution operators. Here, our first objective is to present a unified theoretical framework applicable to virtually any type of fitness function, any type of genotype-phenotype map, any type of selection and any type of mutation and crossover.[2] Our second objective is to demonstrate that all current and past theoretical models of EAs are in fact simply mathematical transformations of one another.

In no way do we want to give the impression that we have totally achieved these objectives. Rather, we are indicating in which direction we believe EC theory should move and what we see as the principal challenges ahead.

THE ROLE OF THEORY IN EC

We begin by asking - what should be the task of EC theory? Is it reasonable, for instance, to think that a theoretician should be able to deduce from first principles after exactly how many generations there's a 95% chance that a better optimal individual will not be found in the following 50 generations for a particular 555-job job-shop scheduling problem with three point crossover with probability 0.9 and mutation probability 0.02? We would categorically deny that this is the principal task of the theory. In fact, we will probably never be able to answer questions such as this. Equally, we may ask if it is the exclusive task of EC theory to consider only very general, global results such as the "No Free Lunch" theorem. Once again, we would say no, rather, theory is at its most powerful when between the very detailed and the very general - but just where?

In the EC community there is a strong distinction between "scientific" theory and "engineering" theory (Goldberg, 2002). These differ both in terms of methodology and motivation. The role of theory in science is to explain and understand phenomena (often results of controlled experiments) within a formal, well defined framework. The role of theory in engineering is to "build better bridges". Sometimes the theory used is rooted in an underlying scientific theory, but often uses rules of thumb that are far removed from the scientific roots. We emphasize that it is not a question of which is the superior approach. They address very different concerns. Here, though, we will be very much

[2]We believe a unified framework can be given for many other classes of EA, including new ones like ant systems, artificial immune systems, etc. but this is beyond the scope of this article.

more concerned with EC theory from the scientific perspective with, however, one eye always on the engineering point of view wherein we may also examine things from a practitioner's standpoint.

In many sciences a large part of theory is associated with taxonomy - classification with respect to natural relationships. In EC, various high-level taxonomic labels are at our disposal, such as GP, GAs, ES etc. Whether these labels are optimal, or even useful other than in an historic sense, however, is a debatable point, as we shall see. Taxonomy allows us to understand commonality between different things. Subsequently we must understand why such commonality exists. For instance, the periodic table was initially an empirical and phenomenological construct until the atomic theory gave it a firm "microscopic" foundation. What is the "periodic table" for EC? Does such a construct exist? If nothing of this nature existed it would be deeply worrying as it would mean that a theoretical treatment of each and every EA and/or problem would be different. It is clear however that there is commonality. The question is more - can it be suitably formalized? At the other extreme one could claim a type of "hyperuniversality", such as was present in the original version of the Building Block Hypothesis (Goldberg, 1989c; Grefenstette, 1993), which claimed that all GAs behaved in the same way in finding an optimum - via fit, short schemata. We now know that this, in its strict form, is wrong, being rather an engineering rule-of-thumb with only limited validity, and that such a degree of hyperuniversality does not exist. Nevertheless, a prime job of EC theory should be to tell us what EAs and problems, or classes of EAs and problems, are likely to lead to similar outcomes or behavior. It does not need to be elaborated on that a deeper understanding of this would be of great use to practitioners.

Passing beyond the taxonomic component of EC theory we should also ask that the theory be able to predict, at least within some more or less approximate scheme, the dynamical evolution of an EA. To address this one needs to start with a framework that at least formally captures the behavior of an EA. This can be at the level of a theory or model which is exact or approximate from the outset. All else being equal an exact model is preferable. Great progress has been made in the last decade in exact formulations of EA dynamics. For instance, the work of Michael Vose and collaborators (Vose, 1999; Nix and Vose, 1992) in the context of the simple GA, where the transition probability matrix for the population evolution is iterated as for a Markov chain, and the work of Stephens, Poli and collaborators (Stephens and Waelbroeck, 1997; Stephens and Waelbroeck, 1999; Stephens, 2001; Poli and McPhee, 2001a; Poli, 2000a; Poli, 2001 a), where an exact dynamics is modelled in terms of schemata, thus leading to a generalized and exact form of Holland's original Schema theorem (Holland, 1975), are two such approaches.

Beyond the mathematical representation of EC theory one should also require that the theory give some intuitive framework within which an EA, or class of EAs, can be understood. The concept of a fitness landscape from population biology (Wright, 1932; Wright, 1967; Reidys and Stadler, 2002) is a prime example of a construct that offers a framework to do just that. The original Schema theorem of Holland and associated Building Block Hypothesis are another very important example. In the seventies and eighties, and to a lesser extent later, they, in fact, seemed to provide a perfectly valid and sufficient theoretical foundation for GAs. So much so that, in the early to late nineties, developing a schema theorem like Holland's became the target for GP theorists too.[3]

Finally, it would be useful to better understand the relationship of EC theory to other more well-established areas in computer science, mathematics, biology and physics. This, for example, would allow us to know whether what has been done in EC is novel. More generally it would make it possible to remove barriers between disciplines and allow for an easier exchange of ideas and results.

Having established some criteria by which we may judge a theory to be "good" or not we may ask: Out of the many possible approaches to EC theory and motivations to develop it, under what circumstances is one better than another, or is there one which is superior to all others under all these roles? We will provide our answer to this in the rest of the chapter.

EC THEORY - THE "BARE NECESSITIES"

In this section we wish to give a brief, non-rigorous exposition of the fundamentals of EC theory, as we see them, that essentially could be applied to any EA. Thus, we try to maintain as much generality as possible, in particular to show how a unified theoretical framework, encompassing most, if not all, standard EAs, can be developed. Formally, an EA is an algorithm that takes

[3] It is worth pointing out that for many years there has been a hot debate in EC as to the strengths and weaknesses of the notion of schema and of Holland's Schema theorem, their usefulness having been widely criticised (see for example (Chung and Perez, 1994; Altenberg, 1995; Fogel and Ghozeil, 1997; Fogel and Ghozeil, 1998)), as has the Building Block Hypothesis (Grefenstette, 1993; Stephens et al., 1999). While some criticisms are really not justified, as discussed in (Radcliffe, 1997; Poli, 2000b; Holland, 2000), others are reasonable. The debate has certainly led to some degree of confusion in the field, with most EC practitioners being divided into two different camps: those who still think Holland's Schema theorem provides a satisfactory theoretical foundation for GAs, not having heard, or not caring, about the debate about its weaknesses, and those who believe there is nothing good, not just in Holland's Schema theorem, but in the notion of schema itself and any theory built on it. Until very recently most EC theoreticians belonged to this second category (see for example (Vose, 1999, preface) and (Bäck and Fogel, 2000, Page xxxiv)). Many of them thought that only Vose's model (Nix and Vose, 1992; Vose, 1999) could provide a serious and mathematically sound way of modelling GAs. Both types of practitioners and theorists are wrong.

as input a population of "objects" (strings, trees etc.[4]) and a fitness function, at a given time, and gives as output the population at a later time. Canonically the evolution is a Markov process generated by a set of genetic operators that act stochastically. The fact that an EA is a stochastic process has the important ramification that we may only expect to make statistical predictions in terms of the dynamics. The objects live on a configuration[5] space *X,* of dimensionality N_X, with elements $i \in X$, where the index $i \in \{1, 2, ..., N_X\}$. As EAs are population based one needs to consider sets of elements, some of which may be repeated multiply. Thus, we denote a population by $\mathbf{P} = (n_1, n_2, ..., n_{N_X})$, where n_i represents the proportion of objects of type i in the population. Each object is assigned a quality or fitness, via a fitness function $f_X : X \rightarrow R^+$. This notion leads one to the important concept of a fitness landscape, $\mathcal{F}$, where one thinks of a topography wherein f_X represents a height function "above" the space *X.* The intuition behind the landscape concept is that populations will seek the peaks in the landscape and move away from the valleys. Often, always in the case of population biology, the fitness landscape possesses a degeneracy, i.e. many genotypes have the same fitness (i.e. corresponding to the same phenotype). One can speak of a "symmetry", strictly speaking an equivalence relation, in this case and ask if this symmetry is preserved by genetic operators other than selection. A concrete example of this is the equivalence under selection of those genotypes that correspond to phenotypes of the same fitness. A dynamics is imposed via an evolution operator, $\mathcal{H}$, such that in the infinite population limit, where $\mathbf{P}(t)$ is the probability distribution at time t, $\mathbf{P}(t+1) = \mathcal{H}\mathbf{P}(t)$. The specific form of $\mathcal{H}$ depends on the specific set of genetic operators used, which in their turn depend on families of parameters. We will generically focus on the standard ones: selection, mutation and recombination. Selection is an operator that depends on the fitness values of the objects. The number of parameters necessary depends on the type of fitness function and the amount of degeneracy of f_X. For instance, for a counting-ones GA problem only *N* fitness values are needed, while for the Eigen model (Eigen et al., 1989) ("needle-in-a-haystack" fitness function) only two, in both cases the genotype-phenotype map being highly degenerate. Mutation, a one-body operator, usually only depends on one parameter - the mutation probability - that is applied uniformly to each locus, though more general operators can easily be considered. Two-parent recombination generically depends on the set of recombination distributions, $\{\lambda_{ijk}(m)\}$, that characterize the transferral of genetic material from parents to offspring, where $\lambda_{ijk}(m)$ is the probability to form an offspring object, i, given two parent objects, j and k, and a crossover

[4]We believe that this generality extends to even more complex objects such as Neural or Bayesian networks etc.

[5]Configurations will most usually be thought of as genotypes.

"mode", m, i.e. a rule for redistributing genetic material between parent and offspring objects. The complexity inherent in this representation can be appreciated by writing down the exact string evolution equation for the simple case of three-bit strings as in (Whitley, 1992). We mentioned previously that taxonomy is important without being specific as to what exactly should be classified. One may think that EAs themselves should be classified. An EA alone however, is in some sense a "black box" which takes a "problem" (usually a fitness landscape and an initial population) as input and then gives an output (the population at a later time). A given EA, though, may have very different characteristics with respect to a given measure on one problem versus another. Another way to see this is that an EA does not fully specify the dynamics of the system, whereas an EA and a problem together do. Hence, we are led to consider a taxonomy of EA/problem pairs. We will call an EA/problem pair a "model". In this context we may characterize a particular model, α, by a set of parameters $\{\{f_\alpha\}, \{\mu_\alpha\}, \{\lambda_\alpha\}\}$, where $\{f_\alpha\}$ represents the fitness landscape and selection mechanism, $\{\mu_\alpha\}$ mutation and $\{\lambda_\alpha\}$ recombination. With these three in hand we can specify a very large class of models. We will denote this space of models, $\mathcal{E}$. We believe that a better understanding of the taxonomy of EAs and fitness landscapes can be achieved by studying $\mathcal{E}$. In principle one could put a metric on $\mathcal{E}$ and talk about how close one model is to another. A less rigorous, but more pragmatic, approach is to think of two models as being "close" if they lead to "similar" behavior. Of course, to do this one must define "similarity measures". At any rate, continuity on $\mathcal{E}$ would lead one to believe that models with similar parameter values should behave similarly, except, of course, in the neighborhood of a singularity.

As a simple example of this approach, consider a GA without mutation and selection but with m-point crossover acting on N-bit strings. In this case we can think of $\mathcal{E}$ as containing only m distinct models. If we chose as similarity measure, $t_c(m)$, the number of generations needed for the correlation function $\langle a_x a_y \rangle$ to decrease by a factor c, where $\langle\ \rangle$ denotes population average and a_x and a_y are the allele values at loci x and y, then one would find, for example for $N = 20$, that $t_c(1) < t_c(2) < \ldots < t_c(N-1) < t_c(N)$. With the specific values one may determine, for example, that 2-point crossover is closer to 1-point crossover than 15-point crossover.

One can think of population flows as taking place on X, the configuration space, or on $\mathcal{F}$, the fitness landscape. All the main branches of EC - GP, GAs, ES etc. - fall into this general framework. The chief differences lie more in what objects are being represented in X and what specific operators constitute $\mathcal{H}$. For instance, in GAs the i represent fixed length strings. In GP they are program trees and in machine code GP (Nordin and Banzhaf, 1995a; Nordin, 1997) or Grammatical evolution (O'Neill and Ryan, 2001) they are variable length strings. We shall also see that "coarse grained" representations,

such as schemata, or particular sets of schemata - Building Block Schemata - also offer very useful basis representations. Interestingly, in nature, genotypes are variable length due to phenomena such as gene duplication and deletion. Additionally, a non-linear structure can also be more appropriate, when for instance modelling protein secondary or tertiary structure.

Rather than considering one type of basis as being more "general" or fundamental than another it is useful to think of passing between different basis representations via coordinate transformations on X, via embeddings of X in a larger or higher dimensional space, or, in the case of true coarse grainings, via projections. Probably the best known alternative coordinate system is the Walsh basis (Goldberg, 1989a; Goldberg, 1989b). Another example of a coordinate transformation, whose importance and utility we will examine shortly, is the following: take fixed length binary strings of length N. In this case X is the N-dimensional Boolean hypercube, the N string loci forming a complete orthonormal basis for the hypercube. Now change to an alternative basis, which we term the Building Block Basis (BBB) (Stephens, 2003), which consists of all schemata corresponding to a given string, where the choice of string is arbitrary. Formally, $i = \sum_j \Lambda_{ij} j$, where j are strings, $\Lambda_{ij} = 1$ iff j is a member of Building Block i and is zero otherwise. The coordinate transformation engendered by $\mathbf{\Lambda}$ yields a basis which is not orthonormal. We will consider the BBB more extensively later. An example of an embedding transformation, at least in principle, would be that of passing from variable length strings of up to maximum size N_m with binary alleles to a fixed length basis representation of size N_m by including a third allele value that specifies that there was no corresponding bit in the variable length case. Of course, for these more general transformations development of the operators and the corresponding theory necessary to maintain syntactic correctness of the offspring is a totally open issue. In this case, one might be better off using the theory for variable length structures already developed in GP.[6] Finally, a simple projective coarse graining would be that of passing between genotype and phenotype.

The above types of map give us flexibility in terms of what particular representation we may find most suitable for a problem and also give a more unified framework within which we may view different elements of EC, such as GP and GAs, in a more coherent light. In fact, our lack of understanding and consideration of such transformations is one of the reasons why EC theory has been, and continues to be, fragmentary.

An even more important reason for considering general classes of transformation associated with X is that it facilitates an understanding of the dy-

[6] Of course, using this basis representation and developing appropriate operators for it would just lead to a form of GP which is isomorphic to current forms, and therefore the theory for such a GA-type of GP would just be isomorphic to the theory already developed there.

namical equations associated with the true effective degrees of freedom of the model. These effective degrees of freedom will more often than not be aggregations of the underlying "microscopic" degrees of freedom and may be made more manifest via a coordinate transformation, embedding or coarse-graining/projection. Additionally, it may be the case that effective degrees of freedom most naturally emerge in an approximation to the dynamics rather than the exact dynamics.

As the model dynamics moves a population composed of individual objects around in X an important precondition for understanding the dynamics is a notion, $\mathcal{N}$, of neighborhood, nearness, distance, or accessibility on X. In some settings, such as binary GAs, a natural neighborhood relation is associated with the Hamming metric. In more complicated cases, such as GP, where one requires a metric on the space of trees of variable size and shape, this is a much more subtle question. Additionally, different genetic operators are often most naturally associated with different notions of nearness. For instance, mutation is very naturally associated with Hamming distance. This is not the case for recombination however. One may be led in this way to consider a different metric for every operator (Jones, 1995). However, given that the dynamics of the model is due to a single composition of different genetic operators, it is questionable as to what extent this picture is useful.

GENERIC GENETIC DYNAMICS

The space of models, $\mathcal{E}$, is of very high dimensionality if one thinks of all possible genetic operators. Selection, mutation and recombination form a very important subset and we will now restrict attention to them. For transparency we will also consider the dynamics in the infinite population limit, writing evolution equations for the probability distribution of objects, $\{P_i(t)\}$, where $P_i(t)$ is the probability for finding object i at time t. The extension to finite populations is relatively straightforward.

Formally at least, the following also applies to GP as well as GAs:

$$P_i(t+1) = \sum_j \mathcal{P}_{ij} P_j^c(t) \tag{7.1}$$

where $P_i^c(t)$ is the probability to find objects of type i after selection and crossover. The matrix elements of the mutation matrix, $\mathcal{P}$, give the probability to mutate object j to object i. In the simple case of fixed length GAs for instance, $\mathcal{P}_{ij} = p_m^{d^H(i,j)}(1-p_m)^{N-d^H(i,j)}$, where d_{ij}^H is the Hamming distance between the two strings and N is the strings' length. For mutation Hamming distance is clearly a very natural metric. Explicitly $P_i^c(t)$ is given by

$$P_i^c(t) = (1-p_c)P_i^I(t) + \sum_m \sum_j \sum_k \lambda_{ijk}(m) P_j^I(t) P_k^I(t) \tag{7.2}$$

where $P_i'(t)$ is the probability to select i. In the case of proportional selection $P_i' = (f_i/\bar{f}(t))P_i$, where $\bar{f}$ is the average population fitness. $\lambda_{ijk}(m)$ is an interaction term between objects, i.e. objects j and k are selected and crossed over ("interact") to potentially form an object i. $\lambda_{ijk}(m)$ depends not only on the objects j and k but also on the particular recombination mode. In the case of homologous crossover the recombination modes are just crossover masks with $\sum_m$ being the sum over all possible recombination masks. In the case of non-homologous crossover the modes are more general than masks.

Equations (7.1) and (7.2), as an exact representation of the dynamics in terms of incidence vectors for objects, in the case of fixed-length GAs, where a crossover mode is simply a mask, are equivalent to the Vose model or, indeed, to earlier formulations in population biology (see (Bürger, 2000) and references therein). It looks quite different because we are using a less condensed notation in order to bring the different roles that each operator play to the fore. These equations however are also valid for objects other than fixed-length strings. A particular criticism of the Vose model has been that although elegant it looks all but hopeless to get other than very general information from the equations. Also, the equations are far removed from older elements of GA theory such as the Schema theorem and Building Block Hypothesis. This has led proponents of the Vose approach to question both the validity and the utility of the latter. As the above equations are equivalent to the Vose equations we may understand the enormity of the task of trying to obtain either quantitative or qualitative results from them. They represent N_X coupled, simultaneous non-linear difference equations. At the level of mutation and selection the problem is linear hence, conceptually at least, the problem is easily addressed - one must find the eigenvalues and eigenvectors of the mutation/selection matrix. The introduction of recombination at first sight leads to a degree of complexity far beyond that of selection and mutation.

To write the interaction constants more explicitly we would have to consider a more definite model. However, we may make some generic comments. First of all, $\lambda_{ijk}(m) = 0$ unless the mode m creates object i from j and k. Generically, this is very unlikely and hence the vast majority of interactions are zero. For instance, in GAs with binary alleles for a given i and m, $\lambda_{ijk}(m)$ is a 2^N-dimensional square matrix. However, only of the order of 2^N matrix elements are non-zero. Thus, the microscopic representation is very inefficient, there being very few ways of creating a given target by recombination of strings. The vast majority of string recombination events are neutral in that they lead to no non-trivial interaction. These comments also hold for more complicated types of object.

UNDERSTANDING GENETIC DYNAMICS: FITNESS, IS IT "EFFECTIVE"?

Having written down a generic dynamics how do we understand it? In population biology the concept of a fitness landscape has played an important role. Standard intuition views a fitness landscape as a rugged terrain where populations flow towards fitness peaks. Thus, natural selection can be viewed as a type of "hill climbing" on this topography.

The classical fitness concept, and associated fitness landscape, however, do not take into account the important effect the mixing genetic operators may have in determining the complete reproductive success of an individual. In particular, the effect of these genetic operators can be such that population flows on the standard fitness landscape cannot be understood with any degree of intuition. In fact, the flows can be quite counterintuitive, leading to situations where populations flow *against* the fitness gradient. A simple concrete example is, once again, the Eigen model where the fitness landscape is just one isolated fitness peak in an otherwise flat landscape. In the absence of mutation the entire population will eventually climb to the top of the fitness peak. In the presence of mutation the proportion of the population associated with the peak is less than one. However, above a certain critical mutation rate (Eigen et al., 1989), p_{crit}, the peak proportion is what it would be in a completely random population on a flat fitness landscape. The landscape remains the same, i.e. with a single peak, yet selection does not act, in the sense that there is no preference for the peak. Clearly hill climbing is not a very useful analogy here.

The mixing operators can also lead to directed flows on neutral networks due to an "induced" breaking of the genotype-phenotype symmetry (Stephens, 1999a; Angeles et al., 1998; Stephens et al., 1998; Mora et al., 1999). Such phenomenon, unlike the case of population flow due to positive selection cannot be naturally understood in terms of hill climbing on a standard fitness landscape either. However, all these phenomena can be intuitively understood within the framework of a different paradigm - *effective fitness*[7] (Stephens and Waelbroeck, 1998; Stephens, 1999b; Stephens and Vargas, 2000; Stephens and Vargas, 2001; Poli, 2000a; Stadler and Stephens, 2003), albeit with the consequence that effective fitness is not a constant quantity but rather depends on the state of the entire system and hence is intrinsically time dependent. We define the effective fitness in the case of objects as

$$P_i(t+1) = \frac{f_i^{\text{eff}}(t)}{\bar{f}(t)} P_i(t) \tag{7.3}$$

[7]Effective (or adjusted) fitness was first introduced in (Nordin and Banzhaf, 1995b; Goldberg, 1989a) in the context of accounting for the destructive effect of crossover in the framework of Holland's Schema Theorem where a simple constant factor multiplies the landscape fitness.

One may think of the effective fitness as representing the effect of all genetic operators in a single reproductive selection factor. Here, we have assumed proportional selection. Effective fitness can easily be generalized for other selection mechanisms however. $f_i^{\text{eff}}(t)$ is the fitness value at time t required to increase or decrease $P_i(t)$ by pure reproductive selection by the same amount as all the genetic operators combined in the context of a reproductive fitness f_i. If $f_i^{\text{eff}}(t) > f_i(t)$ the effect of the genetic operators other than selection is to enhance the reproductive success of object i. Obviously, the converse is true when $f_i^{\text{eff}}(t) < f_i(t)$.

The exact functional form of the effective fitness obviously depends on the set of genetic operators involved. For the fairly general case of equation (7.1) we have

$$f_i^{\text{eff}}(t) = \frac{\bar{f}(t)}{P_i(t)} \sum_j \mathcal{P}_{ij} P_j^c(t) \tag{7.4}$$

In the limit $p_m \to 0$, $\lambda_{ijk} \to 0$ (or in more general circumstances when the strengths of operators other than reproductive selection $\to 0$) $f_i^{\text{eff}}(t) \to f_i$.

The key element behind effective fitness, irrespective of its mathematical definition, is that population flows in the presence of operators other than pure reproductive selection are much more readily understood in terms of it. In fact, to go further, even in the case of pure reproductive selection, if one performs any sort of coarse graining and considers schemata rather than strings, then population flow is more readily understood in terms of an effective fitness landscape rather than the reproductive one. As an example, for the evolution of a particular order-1 schema in a population of N-bit strings it is more natural to consider the *time dependent* one-dimensional landscape associated with the schema than the collective dynamics of the $2^{(N-1)}$ string types that go up to make the 1-schema. The job of evolution at the end of the day is to produce fit offspring which in their turn produce fit offspring which in their turn... It is no use having an individual with high reproductive fitness that is associated with a high probability to mutate to a very low fitness individual.

In the case of the Eigen model the effective fitness of the master sequence or needle, under selection and mutation only, is

$$\begin{aligned} f_{\text{needle}}^{\text{eff}}(t) &= f_{\text{needle}}(1-p_m)^N \\ &+ f_{\text{hay}} \sum_{i \neq \text{needle}} \frac{P_i(t)}{P_{\text{needle}}(t)} p_m^{d^H_{\text{needle},i}} (1-p_m)^{N-d^H_{\text{needle},i}} \end{aligned} \tag{7.5}$$

where f_{needle} and f_{hay} are the fitnesses of the needle and the "hay" respectively. In the limit $p_m \to p_{cr}$, $f_{\text{needle}}^{\text{eff}}(t) \to \bar{f}(t)$ and we see that the effective fitness landscape becomes flat thereby giving an intuitive explanation for the behavior in the vicinity of the critical mutation rate. We can thus think of evolution as a hill-climbing process on an effective fitness landscape (which is time

dependent). In this model mutation breaks the genotype-phenotype symmetry among the non-needle strings in such a way that those strings that are closer in Hamming distance to the needle have more reproductive success. Once again, this cannot be understood in terms of the fitness landscape as it is flat for the non-needle strings. The analog of equation (7.5) for non-needle strings shows us however that the effective fitness of strings that are close to the needle is higher than that of distant strings. Effective fitness in this sense is a direct measure of the strength of the breaking of the genotype-phenotype symmetry and hence offers both a qualitative and quantitative framework within which phenomena such as GP bloat and evolutionary robustness may be understood.

UNDERSTANDING GENETIC DYNAMICS: WHAT ARE THE RIGHT EFFECTIVE DEGREES OF FREEDOM?

All genetic operators affect what are the appropriate effective degrees of freedom[8] for a particular model, although in potentially very different ways. For selection, almost by definition, the principal effective degree of freedom is the phenotype. For pure mutation they are the eigenvectors of the mutation matrix, the most relevant ones being those with the largest eigenvalues. When combining selection and mutation it becomes much more difficult to determine the correct effective degrees of freedom. As a simple example, consider again the needle-in-a-haystack landscape for N-bit strings. In this case there are 2^N genotypes but only two phenotypes - the "needle" and the "hay". For selection only, due to the strong genotype-phenotype symmetry the dynamics is much more simply considered in terms of fitness equivalence classes, as there are only two of them. However, as we pointed out in the previous section mutation breaks this symmetry. In this case the more appropriate effective degrees of freedom are the error classes (sets of strings a fixed Hamming distance from the needle.)

Schemata offer another class of effective degree of freedom, where one coarse grains to a smaller number of fixed loci than in the original model. This has been familiar in population biology for a long time, where reduction to a small number of loci is ubiquitous. There, however, traditionally the coarse graining has been posited and a reduced model for the schemata dynamics directly written down rather than, more correctly, *deriving* the schemata dynamics from the underlying microscopic dynamics. Interestingly, any schemata coarse graining, except in exceptional cases will lead to schema fitnesses that

[8]By number of degrees of freedom we mean the number of variables needed to describe the state of an "object". In many cases those variables can actually be dependent on one another. In these cases, it is often possible to identify a smaller set of *independent* variables to describe the system in an exact or approximate, but, sufficient way. We call these the effective degrees of freedom of the system.

are time dependent as they depend on the dynamics of the population. Thus, if one thinks of a fitness landscape for schemata, it will inevitably be time dependent. It is obviously of great interest to then ask when and under what approximation can the time dependence be ignored? It is natural to imagine that if fitness is defined with respect to a certain phenotypic character that depends principally on a small number of genotypic loci then the resultant landscape should be approximately time independent. Although a particular schemata-type coarse graining might suggest itself, the space of schemata-type coarse grainings has huge dimensionality (e.g. $(k+1)^N$ for N-bit strings and a cardinality k alphabet). Hence, the search for a set of schemata that capture the effective degrees of freedom is in an even larger space than the original problem! The question is under what circumstances does a particular set of schemata suggest itself? The chief cornerstones of early GA theory - Holland's Schema theorem and the Building Block Hypothesis - gave an apparent answer to this question - that it is fit, short schemata that are the effective degrees of freedom, the associated intuition being intimately linked to selection and the destructive effect of crossover. At that time no exact equations that took into account schema creation were known and the apparent contradiction between an hypothesis that posited the existence of building blocks and a theorem that did not take into account how building blocks formed higher order schemata was overlooked.

Holland's Schema theorem and the Building Block Hypothesis strongly asserted that crossover plays a privileged role in the utility of EAs. The fundamental dynamical equations (7.1) and (7.2), as in the Vose model, and as in older exact population biology models, both for the case of fixed-length strings, are written in terms of the microscopic degrees of freedom, i.e. the strings themselves. However, the simple structure hides the large scale redundancy inherent in this representation (i.e. the vast majority of the λ_{ijk} are zero) and the complication associated with the sums over the j and k. Recently, it has become possible to write these equations in a form that extends and generalizes Holland's Schema theorem, allows for a critical and rigorous analysis of the Building Block Hypothesis and is still intimately linked to a microscopic formulation that is equivalent to the Vose model. In its original formulation it encompassed fixed-length, linear genomes. Importantly, however, it has now also been extended to variable-length linear and tree-like representations (Poli, 2000a; Poli and McPhee, 2001a; Poli, 2001a).

In the fixed-length case[9] one may understand the relationship between the two formulations in terms of a linear coordinate transformation $\Lambda_i : \mathcal{G} \rightarrow \tilde{\mathcal{G}}$

[9] We conjecture that it is also true in the case of more general objects. The very fact that the exact schema equations for GP are so close to their GA counterparts in the BBB lend weight to this conjecture. Obviously, coordinate transformations on these more complex spaces need to be better understood.

to the BBB mentioned previously. A simple example is in the case of two bits where $\mathcal{G} = \{11, 10, 01, 00\}$, while $\tilde{\mathcal{G}} = \{11, 1*, *1, **\}$. The invertible matrix Λ, the inversion leading back to the original string basis, is such that $\Lambda_{ij} = 1 \iff j \in \xi_i$, where ξ_i is any schema associated with the string i. Note that as the choice of vertex is arbitrary there are 2^N totally equivalent BBBs.

The BBB is complete but clearly not orthonormal. By construction i is a fixed point of this transformation. Apart from the vertex i, points in $\tilde{\mathcal{G}}$ correspond to higher dimensional objects in $\mathcal{G}$. For instance, `1*` and `*1` are one-planes in $\mathcal{G}$ while `**` is the whole space. Note that the BBs here are not necessarily short or fit and are certainly not static when considered in the context of the evolution equation. Note as well that we are not working here in the space of all schemata. The BBB, in fact, forms a very small subset of the latter. Nevertheless, these are what are being processed by recombination. In the BBB, in the case of homologous crossover, one finds

$$\tilde{P}_i^c(t+1) = (1-p_c)\tilde{P}_i'(t) + \sum_{m=1}^{2^N} \sum_j \sum_k \tilde{\lambda}_{ijk}(m)\tilde{P}_j'(t)\tilde{P}_k'(t) \qquad (7.6)$$

where $\tilde{\lambda}_{ijk}(m) = \Lambda_{ii'}\lambda_{i'j'k'}\Lambda^{-1}_{j'j}\Lambda^{-1}_{k'k}$. One may ask what is the advantage of going to this new basis? In the original string basis the properties and symmetries of $\lambda_{ijk}(m)$ are very hidden. However, this is not the case for $\tilde{\lambda}_{ijk}(m)$, which has the property that for a given mask only interactions between BBs that construct the target schema are non-zero, i.e. $\tilde{\lambda}_{ijk}(m) = 0$, unless k corresponds to a schema which is the complement of j with respect to i. Furthermore, $\tilde{\lambda}_{ijk}(m) = 0$ unless j is equivalent to m, where by equivalent we mean that for any `1` in the mask we have a `1` at the corresponding locus in j and for any `0` we have a `*`.[10] These two important properties mean that the two summations over j and k in (7.6) both disappear and we are left with only the sum over masks with an "interaction" constant $p_c(m)$ which depends only on the mask. For example, for three bits, mask `100` and target string `111` recombination of `011` with `110`, `100` or `101` all lead to the desired target. However, in the BBB the mask `100` specifies as first BB parent the schema `1**`. The second BB parent `*11` follows naturally by complementarity.

In $\mathcal{G}$ this has the interesting interpretation that for a target schema ξ of dimensionality $(N-d)$ only geometric objects "dual" in the d-**dimensional** subspace of $\mathcal{G}$ that corresponds to ξ may interact. I.e. a k-**dimensional** object recombines only with a $(N-d-k)$-**dimensional** object. Additionally, a $(N-d)$-**dimensional** object may only be formed by the interaction of higher

[10]This happens because a particular mask projects out a particular element of the BBB, while the other building block is specified purely by complementarity.

dimensional objects. In this sense interaction is via the geometric intersection of higher dimensional objects. For example, the point `11` can be formed by the intersection of the two lines `1*` and `*1`. Similarly, `1111` can be formed via intersection of the three-plane `1***` with the line `*111`, or via the intersection of the two two-planes `11**` and `**11`.

As mentioned, one of the primary advantages of the BBB representation is that the sums over j and k disappear. One obtains, for an arbitrary string i

$$P_i^c(t) = (1 - p_c)P_i'(t) + \sum_{m=1}^{2^N} p_c(m)P_{i_m}'(t)P_{i_{\bar{m}}}'(t) \tag{7.7}$$

where $p_c = \sum_m p_c(m)$. $P_{i_m}'(t)$ is the probability to select the BB i_m (note that the mask uniquely specifies which element, i_m, of the BBB to choose) and $P_{i_{\bar{m}}}'(t)$ the probability to select the BB $i_{\bar{m}}$ which is uniquely specified as the complement of i_m in i. Both i_m and $i_{\bar{m}}$ are elements of the BBB associated with i. The above equation clearly shows that recombination is most naturally considered in terms of the BBB. In the string basis there were of the order of 2^{2N} elements of λ_{ijk} to be taken into account for a fixed i. In the BBB there is only one term. Of course, we must remember that the coarse grained averages of i_m and $i_{\bar{m}}$ contain 2^N terms, still, the reduction in complication is enormous. Thus, we see that recombination as an operator naturally introduces the idea of a coarse graining, the natural effective degrees of freedom associated with crossover being the BBs we have defined.

Inserting (7.7) in (7.1) we can try to solve for the dynamics. However, in order to do that we must know the time dependence of i_m and $i_{\bar{m}}$. Although the number of BB basis elements is 2^N we may generalize and consider the evolution of an arbitrary schema, ξ. To do this we need to sum with $\sum_{i\in\xi}$ on both sides of the equation (7.1). This can simply be done to obtain again the form (7.1), where this time the index i runs only over the 2^{N_2} elements of the schema partition and where again $\mathcal{P}_{ij} = p_m^{d^H(i,j)}(1 - p_m)^{N-d^H(i,j)}$. In this case however $d^H(i,j)$ is the Hamming distance between the two schemata. For instance, for three bit strings the schema partition associated with the first and third bits is $\{1*1, 1*0, 0*1, 0*0\}$. In this case $d^H(1,2) = 1$ and $d^H(1,4) = 2$. $P_\xi^c(t) = \sum_{i\in\xi} P_i^c(t)$ is the probability of finding the schema ξ after selection and crossover. Note the form invariance of the equation after coarse graining. To complete the transformation to schema dynamics we need the schema analog of (7.7). This also can be obtained by acting with $\sum_{i\in\xi}$ on both sides of the equation. One obtains

$$P_\xi^c(t) = (1 - p_c N_\xi)P_\xi'(t) + \sum_{m\in\mathcal{M}_r} p_c(m)P_{\xi_m}'(t)P_{\xi_{\bar{m}}}'(t) \tag{7.8}$$

where ξ_m represents the part of the schema ξ inherited from the first parent and $\xi_{\bar{m}}$ that part inherited from the second and $N_\xi = \sum_{m\in\mathcal{M}_r} p_c(m)$, where $\mathcal{M}_r$

is the set of masks that affect ξ. Obviously, these quantities depend on the type of crossover implemented and on properties of the schema such as defining length. Note that the BBB naturally coarse grains here to the BBB appropriate for the schema ξ as opposed to the string i.

Thus, we see that the evolution equation for schemata has exactly the same form as (7.7), there being only a simple multiplicative renormalization (redefinition) of the crossover probability $p_c \rightarrow p_c N_\xi$. This form invariance, first shown in (Stephens and Waelbroeck, 1997; Stephens and Waelbroeck, 1998), demonstrates that BB schemata in general are a preferred set of coarse grained variables and more particularly the BBB is a preferred basis in the presence of recombination. It has also been shown (Vose, 1999) that schemata, more generally, are the only coarse graining that leads to invariance in the presence of mutation and recombination. Considering again the structure of (7.7) and (7.8) we see that variables associated with a certain degree of coarse graining are related to BB "precursors" at an earlier time, which in their turn ... etc. This hierarchical structure terminates at order-one BBs as these are unaffected by crossover. Thus, for example, the level-one BB combinations of `111`, i.e. BBs that lead directly upon recombination to `111`, are: `11*:**1`, `1*1:*1*` and `1**:*11`. The level-two BBs are `1**`, `*1*` and `**1`. Thus, a typical construction process is that BBs `1**` and `*1*` recombine at $t = t_1$ to form the BB `11*` which at some later time t_2 recombines with the BB `**1` to form the string `111`.

In this basis the validity of the Building Block Hypothesis can be examined. From the structure of (7.7) we see, in fact, that *in a certain sense* the Building Block Hypothesis emerges as a logical consequence of the equations. The hierarchical structure of the equation and its solution show unequivocally how fine grained schemata are built up from more coarse grained BBs. However, the supposition that BBs are fit and short is not generally true. The BBs that are important are those of high effective fitness. These may be short or long, fit or unfit depending on the particular characteristics of the fitness landscape and the other operators. Thus we can construct an *Effective Building Block Hypothesis* (but note that this is not a conjecture, but a mathematically provable consequence of the equations) which applies not only to GAs but to other EAs, such as GP, that fall within our unified framework:

Effective BBH: *an EA with crossover works by repeatedly combining low-order schemata of above average* effective *fitness to form higher-order ones.*

In the above, simply to be more concrete, we have used the fixed-length representation characteristic of GAs. However, it is important to emphasize that almost everything we have said has a natural generalization, with basically exactly the same intuition, at the level of variable-length or tree-like representations, a subject we will now consider further.

THE TWISTED ROAD TO UNIFICATION: GP

To this point we have tried to present as unified a view as possible of EC theory. However, when we have passed from the abstract to the concrete we have up to now used standard GAs as a point of contact. It is important to emphasize however that much of what we have previously described has been rigorously generalized to the case of GP, which presents a host of very difficult and challenging problems not present in GAs. So, if developing a theory for GP and similar evolutionary paradigms is so difficult, why do it? That is, what do we gain from moving away from fixed-length representations into a world of variable size structures?

The first obvious answer to this is that we will understand GP itself better, one motivation being to use theory in GP to achieve analogous things to those achieved using GA theory, such as explanations, predictions, engineering design principles, etc. In the wider context of EC theory in general, however, the first thing we gain is a better understanding of what we were doing before. Work on GP theory has shown us that the evolution of fixed-length strings is in fact a special case of a much broader space of algorithms which include the evolution of: non-binary strings, strings where different loci can have different numbers of alleles, strings whose alleles can take a countably or uncountably large number of different values (like in ESs), strings whose length can be changed by the operators, trees with fixed or variable shape and size, trees built with countably or uncountably large primitive sets (e.g. imagine an ES-GP hybrid), graphs with and without labelled links, and so on.

Secondly, because of the previous point, we have a unique opportunity to completely unify EC theory. In fact, any piece of theory one can produce which is applicable in general to this larger space of algorithms, automatically leads to corresponding results for all the subclasses, and, conversely, any specific result available in one of the subclasses will indicate the possibility that there could be a corresponding, undiscovered result for the general class. Naturally, because ES and GA theory is more developed than GP theory, we should expect that, initially, GP theory will aim at extending pre-existing results, but eventually, as the unification progresses, the biggest rewards should come from working directly in the broader space.

Thirdly, until now EC theory has only borrowed from theoretical population genetics, it has never exported results. There are many reasons for this. Partly, there is a communication problem between computer scientists and geneticists. This is not just due to the different languages we use to describe our evolutionary systems: mostly it is due to the different types of systems we study. Geneticists study diploid representations and consider recombination operators where homologous strands are aligned (by content) and may have variable lengths due to gene deletion and gene duplication events. EC theo-

rists have almost always limited their studies only to haploid representations of fixed length undergoing position-preserving recombination operators. Moving away from fixed-length representations and position-preserving operators by embracing GP theory is a good step in the direction of being able to export our theoretical results to population genetics.

For many years GP appeared to be completely unrelated to GAs, or other fixed-length representation EAs. The differences in the representation adopted and in the semantics of the structures being evolved have been two major obstacles in bridging the gap between them. However, the characteristics of the operators adopted in GP w.r.t. those of other EAs have been one other major obstacle. Mainstream GP used crossover operators that transfer genetic material without necessarily respecting its original position in the parents. Fixed-length EAs typically did the opposite, and most EA theory was based on this very assumption. So, although people felt that there had to be some way to extend GA theory to incorporate GP, in practice that was impossible until two stepping stones became available: the notion of one-point crossover in GP and a natural extension of the notion of GA schema to GP (Poli and Langdon, 1997). GP one-point crossover is an operator where the parents are first aligned starting from their root node and recursively traversing the two trees in parallel, stopping the exploration of each branch when an arity mismatch occurs. Then a *common* crossover point is chosen among the matching nodes and the subtrees rooted in that node in the parents are swapped to produce the offspring. Despite the useful concepts of GP one-point crossover and GP schema, however, other notions like, for example, defining length, what constitutes a building block, and so on, required a good deal of trial and error to get right. (By "getting right" here we mean that those definitions represent proper generalizations of equivalent GA notions.)

Indeed, after the GP-one-point-crossover breakthrough it has required around half a decade for homologous-GP theory to generalise some of the most fundamental results in GA theory, such as Stephens' exact schema theory for crossover (Poli, 2000a; Poli and McPhee, 2001b; Langdon and Poli, 2002), Vose's model for crossover (Poli et al., 2001) and Geiringer's theorem (Poli et al., 2002b; Poli et al., 2002c; Poli et al., 2002a). Only last year, thanks to some of the tools developed in this endeavor and to some good luck, it has finally been possible to write an exact schema-based model for GP with its more standard forms of crossover (Poli, 2001b).

EC THEORY: THE CHALLENGES AHEAD

As mentioned in the introduction: EC theory is a very exciting field with a large number of challenging problems worthy of the attention of any energetic

scientist dedicated enough and with enough passion to attack them. Here we give our own personal view of some of the main ones.

C1: How do we classify EC models so that we can answer the fundamental question: when do we expect the dynamics of two different models to be qualitatively (and maybe at some point quantitatively) similar? (This is also of vital importance for practitioners). We need to understand much better the space $\mathcal{E}$ in which EC models live and the use of metrics and similarity measures. Obviously, to answer the above we need to have a formalism within which we can work - ours in just one possibility.

C2: A deeper understanding of the various transformations - coordinate transformations, embeddings, coarse-grainings/projections and others - that change basis is, we believe, crucial for obtaining a truly unified picture of different models. It is also crucial for identifying and manipulating the appropriate effective degrees of freedom of a model. We have discussed various basis representations: the genotypic (in terms of the "microscopic" objects) and phenotypic representations, and the BB representation. Each has its advantages and disadvantages. The genotypic representation is fundamental but rarely, if ever, are the real effective degrees of freedom directly related to the genotype. The phenotypic representation would be most appropriate in a strong selection regime. BB schemata in GAs and their extension to GP appear to be perfect at capturing the regularities present in homologous crossover operators and are most obviously the appropriate effective degrees of freedom in the case of weak selection and strong crossover. The question remains though, which coarse graining and, more generally, which basis is most appropriate for a given model. We have answers only for a small set of special cases. However, the answer here is very much related to the question of model classification, i.e. find a good basis or coarse graining for a given member of a class and the same basis should be useful for other members of the class. To distinguish one basis or coarse graining from another a quality measure (which would, of course, be parametrized by the particular problem and search algorithm at hand), would be useful in order to rank them.

C3: A more general understanding of the different bases themselves is we believe also of great importance. The BBB is, effectively, a very recent development and much remains to be understood about it. The space of trees of variable size and shape is also not well understood. Furthermore, a major challenge is to move beyond trees to a world of more general graphs. Graphs are maybe the most powerful representation available in computing. Anything from linear structures, to parallel systems, to neural networks, to organizations, etc. can be represented with graphs of one type or another. Extending the EC theory to this type of structures would give it an almost all-encompassing scope. Only a tiny amount of progress has been made here (Greene, 2000).

C4: How rugged or smooth is $\mathcal{E}$? This is very important for being able to estimate the degree of validity of various exact models and approximations. Too often practitioners scoff at theoreticians working with simple model fitness landscapes, such as flat landscapes, counting ones or needle-in-a-haystack believing that they have no relevance to "real-world" problems. However, these simple models are representatives of classes of models. For example, if we consider dynamics on a linear landscape with 1-pt crossover and then add a small amount of epistasis do we expect to see big qualitative changes in the resulting model? Alternatively, starting with a genepool GA on a linear landscape and then adding weak epistasis and changing to three point crossover do we expect qualitatively different behavior? The structure of $\mathcal{E}$ can be studied in this sense empirically.

C5: Currently, exact schema-based models only exist for homologous type of crossover operators and subtree-swapping type of operators. However, these two classes of operators are two extremes of a continuum: one, the case in which perfect alignment of structure is imposed on the trees undergoing crossover, the other, the case in which alignment between the parent trees is not even attempted. However, this continuum is full of interesting alternatives. For example, in nature, the alignment of DNA strands is based on a matching process between bases (and, consequently, between genes). It would be interesting to extend this notion to tree-like structures and be able to model theoretically this type of process. So, being able to categorize, characterize and model the operators in this continuum is an important challenge ahead. The same kind of thing should be done for unary operators, where GP theory so far is limited to subtree-type of mutations.

C6: Effective fitness seems to offer a generalization of fitness that preserves the "hill-climbing" intuition of the latter even in the presence of operators other than selection. How general is its utility in explaining phenomena, both qualitatively and quantitatively, that do not fit into the selection/hill-climbing paradigm such as bloat (Langdon and Poli, 1997), evolution on neutral networks (Reidys and Stadler, 2001), evolutionary robustness (van Nimwegen et al., 1999) etc.?

C7: Although the development of exact evolution equations has been rapid there remains a disquieting lack of tools with which solutions to the equations may be found. In particular, we know of no systematic approximation schemes that have been studied, though several come to mind, such as an expansion around the strong selection limit, perturbing in the mutation or crossover rate. Alternatively, in the strong-crossover, weak-selection limit an expansion in principle should be possible in terms of the deviation away from flatness of the landscape. Often, expansions require an exact solution around which to expand. Hence, it is important to find as many exact solutions as possible which may serve as starting points of an expansion. All this remains to be done.

C8: All of the above we would classify as challenges primarily associated with the scientific point of view of EC. Lest we forget the engineering perspective: the developed theory should be tested to see if it can provide theoretically-valid recipes for practitioners: Which genotype-phenotype map, operators, fitness function, search algorithm, population size, number of generations, number of runs, crossover probability, anti-bloat method, etc. should one use for a given problem or a given class of problems? Perhaps a simple step in this direction is to find approximations a-la Goldberg (Goldberg, 2002) which can really help a designer, but are based on something better and more rigorous than Holland's version of the schema theorem. Or maybe we need to find ways of characterizing the sampling behavior of different operators and defining whether this behavior matches the shape of a particular fitness landscape and to which degree? In all this we have just started scratching the surface (McPhee and Poli, 2002).

Of course, the above list is by no means exhaustive, though we believe there is enough there to keep very many EC theoreticians busy for many years to come. Hopefully, it may help to stimulate a new generation of theoreticians as it is currently stimulating both ourselves, our collaborators and our students. We strongly believe that EC benefits strongly from an interdisciplinary approach and we would hope that more talented researchers from other fields will enter the fray bringing with them their own points of view and toolboxes. In particular, EC was inspired by evolution in nature. It would be more than fitting if EC theory could offer something back to its "older, bigger brother" - population biology.

ACKNOWLEDGEMENTS

CRS is grateful for financial support from DGAPA project ES100201 and Conacyt project 30422-E. RP would like to thank the members of the NEC (Natural and Evolutionary Computation) group at Essex for helpful comments and discussion. The authors would like to thank the editor Anil Menon for exceptionally helpful comments and suggestions.

REFERENCES

Altenberg, L. (1995). The Schema Theorem and Price's Theorem. In Whitley, L. D. and Vose, M. D., editors, *Foundations of Genetic Algorithms 3,* pages 23–49, Estes Park, Colorado, USA. Morgan Kaufmann.

Angeles, O., Stephens, C. R., and Waelbroeck, H. (1998). Emergence of algorithmic language in genetic systems. *Biosystems*, 47:129–147.

Bäck, T. and Fogel, D. B. (2000). Glossary. In Bäck, T., Fogel, D. B., and Michalewicz, T., editors, *Evolutionary Computation 1: Basic Algorithms and Operators.* Institute of Physics Publishing.

Bürger, R. (2000). *The Mathematical Theory of Selection, Recombination, and Mutation.* Wiley, Chichester, UK.

Chung, S. W. and Perez, R. A. (1994). The schema theorem considered insufficient. In *Proceedings of the Sixth IEEE International Conference on Tools with Artificial Intelligence,* pages 748–751, New Orleans.

Eigen, M., McCaskill, J., and Schuster, P. (1989). The molecular Quasispecies. *Adv. Chem. Phys*., 75:149–263.

Fogel, D. B. and Ghozeil, A. (1997). Schema processing under proportional selection in the presence of random effects. *IEEE Transactions on Evolutionary Computation*, l(4):290–293.

Fogel, D. B. and Ghozeil, A. (1998). The schema theorem and the misallocation of trials in the presence of stochastic effects. In Porto, V. W., Saravanan, N., Waagen, D., and Eiben, A. E., editors, *Evolutionary Programming VII: Proc. of the 7th Ann. Conf. on Evolutionary Programming,* pages 313–321, Berlin. Springer.

Goldberg, D. E. (1989a). Genetic algorithms and Walsh functions: I. A gentle introduction. *Complex Systems,* 3(2): 129–152.

Goldberg, D. E. (1989b). Genetic algorithms and Walsh functions: II. Deception and its analysis. *Complex Systems*, 3(2): 153–171.

Goldberg, D. E. (1989c). *Genetic Algorithms in Search, Optimization, and Machine Learning.* Addison-Wesley, Reading, Massachusetts.

Goldberg, D. E. (2002). *The Design of Innovation.* Kluwer Academic Publishers, Boston.

Greene, W. A. (2000). A non-linear schema theorem for genetic algorithms. In Whitley, D., Goldberg, D. E., Cantu-Paz, E., Spector, L., Parmee, I., and Beyer, H.-G., editors, *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2000)*, pages 189–194, Las Vegas, Nevada, USA. Morgan Kaufmann.

Grefenstette, J. J. (1993). Deception considered harmful. In Whitley, L. D., editor, *Foundations of Genetic Algorithms 2*, San Mateo, CA. Morgan Kaufman.

Holland, J. H. (1975). *Adaptation in Natural and Artificial Systems.* University of Michigan Press, Ann Arbor, USA.

Holland, J. H. (2000). Building blocks, cohort genetic algorithms, and hyperplane-defined functions. *Evolutionary Computation*, 8(4):373–391.

Jones, T. (1995). *Evolutionary Algorithms, Fitness Landscapes and Search.* PhD thesis, The University of New Mexico, Albuquerque, NM.

Langdon, W. B. and Poli, R. (1997). Fitness causes bloat. In Chawdhry, P. K., Roy, R., and Pant, R. K., editors, *Soft Computing in Engineering Design and Manufacturing*, pages 13–22. Springer-Verlag London.

Langdon, W. B. and Poli, R. (2002). *Foundations of Genetic Programming.* Springer-Verlag.

McPhee, N. F. and Poli, R. (2002). Using schema theory to explore interactions of multiple operators. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2002)*, New York, USA. Morgan Kaufmann. (accepted as full paper).

Mora, J., Stephens, C. R., Waelbroeck, H., and Zertuche, F. (1999). Symmetry breaking and adaptation: Evidence from a simple toy model of a viral neutralization epitope. *Biosystems*, 51:1–14.

Nix, A. E. and Vose, M. D. (1992). Modeling genetic algorithms with Markov chains. *Annals ofMathematics and Artificial Intelligence*, 5(1):79–88.

Nordin, P. (1997). *Evolutionary Program Induction of Binary Machine Code and its Applications.* PhD thesis, der Universitat Dortmund am Fachereich Informatik.

Nordin, P. and Banzhaf, W. (1995a). Complexity compression and evolution. In Eshelman, L., editor, *Genetic Algorithms: Proceedings of the Sixth International Conference (ICGA95)*, pages 310–317, Pittsburgh, PA, USA. Morgan Kaufmann.

Nordin, P. and Banzhaf, W. (1995b). Complexity compression and evolution. In Eshelman, L., editor, *Genetic Algorithms: Proceedings of the Sixth International Conference (ICGA95)*, pages 310–317, Pittsburgh, PA. Morgan Kaufmann.

O'Neill, M. and Ryan, C. (2001). Grammatical evolution. *IEEE Transaction on Evolutionary Compuation.* Forthcomming.

Poli, R. (2000a). Exact schema theorem and effective fitness for GP with one-point crossover. In Whitley, D., Goldberg, D. E., Cantu-Paz, E., Spector, L., Parmee, I., and Beyer, H.-G., editors, *Proceedings of the Genetic and Evolutionary Computation Conference*, pages 469–476, Las Vegas. Morgan Kaufmann.

Poli, R. (2000b). Why the schema theorem is correct also in the presence of stochastic effects. In *Proceedings of the Congress on Evolutionary Computation (CEC 2000)*, pages 487–492, San Diego, USA.

Poli, R. (200la). Exact schema theory for genetic programming and variable-length genetic algorithms with one-point crossover. *Genetic Programming and Evolvable Machines,* 2(2): 123–163.

Poli, R. (2001b). General schema theory for genetic programming with subtree-swapping crossover. In *Genetic Programming, Proceedings of EuroGP 2001,* LNCS, Milan. Springer-Verlag.

Poli, R. and Langdon, W. B. (1997). A new schema theory for genetic programming with one-point crossover and point mutation. In Koza, J. R., Deb, K., Dorigo, M., Fogel, D. B., Garzon, M., Iba, H., and Riolo, R. L., editors, *Genetic Programming 1997: Proceedings of the Second Annual Conference,* pages 278–285, Stanford University, CA, USA. Morgan Kaufmann.

Poli, R. and McPhee, N. F. (200la). Exact schema theory for GP and variable-length GAs with homologous crossover. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2001),* San Francisco, California, USA. Morgan Kaufmann.

Poli, R. and McPhee, N. F. (2001b). Exact schema theory for GP and variable-length GAs with homologous crossover. Technical Report CSRP-01-4, University of Birmingham, School of Computer Science.

Poli, R., Rowe, J. E., and McPhee, N. F. (2001). Markov models for gp and variable-length gas with homologous crossover. Technical Report CSRP-01-6, University of Birmingham, School of Computer Science.

Poli, R., Rowe, J. E., Stephens, C. R., and Wright, A. H. (2002a). Allele diffusion in linear genetic programming and variable-length genetic algorithms with subtree crossover. In *Proceedings of EuroGP 2002.*

Poli, R., Stephens, C. R., Wright, A. H., and Rowe, J. E. (2002b). On the search biases of homologuous crossover in linear genetic programming and variable-length genetic algorithms. In Langdon, W. B., Cantú-Paz, E., Mathias, K., Roy, R., Davis, D., Poli, R., Balakrishnan, K., Honavar, V., Rudolph, G., Wegener, J., Bull, L., Potter, M. A., Schultz, A. C., Miller, J. F., Burke, E., and Jonoska, N., editors, *GECCO 2002: Proceedings of the Genetic and Evolutionary Computation Conference*, pages 868–876, New York. Morgan Kaufmann Publishers.

Poli, R., Stephens, C. R., Wright, A. H., and Rowe, J. E. (2002c). A schema-theory-based extension of Geiringer's theorem for linear GP and variable-length GAs under homologous crossover. In De Jong, K., Poli, R., and Rowe, J., editors, *Proceedings of the Foundations of Genetic Algorithm (FOGA-VII) Workshop,* Torremolinos.

Prügel-Bennett, A. and Shapiro, J. L. (1994). An analysis of genetic algorithms using statistical mechanics. *Physical Review Letters,* 72:1305–1309.

Radcliffe, N. J. (1997). Schema processing. In Baeck, T., Fogel, D. B., and Michalewicz, Z., editors, *Handbook of Evolutionary Computation,* pages B2.5–1–10. Oxford University Press.

Reidys, C. M. and Stadler, P. F. (2001). Neutrality in fitness landscapes. *Appl. Math. & Comput.,* 117:321–350.

Reidys, C. M. and Stadler, P. F. (2002). Combinatorial landscapes. *SIAM Review,* 44:3–54.

Stadler, P. F. and Stephens, C. R. (2003). Landscapes and effective fitness. *Comm. Theor. Biol,* to be published. Santa Fe Insitute Working Paper: 02-11–062.

Stephens, C. R. (1999a). Effect of mutation and recombination on the genotype-phenotype map. In Banzhaf, W., Daida, J., Eiben, A. E., Garzon, M. H., Honavar, V., Jakiela, M., and Smith, R. E., editors, *Proceedings of the Ge-*

netic and Evolutionary Computation Conference, volume 2, pages 1382–1389, Orlando, Florida, USA. Morgan Kaufmann.

Stephens, C. R. (1999b). Effective fitness landscapes for evolutionary systems. In Angeline, P. J., Michalewicz, Z., Schoenauer, M., Yao, X., and Zalzala, A., editors, *Proceedings of the Congress on Evolutionary Computation,* volume 1, pages 703–714, Mayflower Hotel, Washington D.C., USA. IEEE Press.

Stephens, C. R. (2001). Some exact results from a coarse grained formulation of genetic dynamics. In Spector, L., Goodman, E. D., Wu, A., Langdon, W. B., Voigt, H.-M., Gen, M., Sen, S., Dorigo, M., Pezeshk, S., Garzon, M. H., and Burke, E., editors, *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2001),* pages 631–638, San Francisco, California, USA. Morgan Kaufmann.

Stephens, C. R. (2003). The renormalization group and the dynamics of genetic systems. *Acta Phys. Slov.,* to be published. Preprint: cond-mat/0210217.

Stephens, C. R., García Olmedo, I., Mora Vargas, J., and Waelbroeck, H. (1998). Self-adaptation in evolving systems. *Artificial Life,* 4:183–201.

Stephens, C. R. and Vargas, J. M. (2000). Effective fitness as an alternative paradigm for evolutionary computation I: general formalism. *Genetic programming and evolvable machines,* l(4):363–378.

Stephens, C. R. and Vargas, J. M. (2001). Effective fitness as an alternative paradigm for evolutionary computation II: examples and applications. *Genetic programming and evolvable machines.* Forthcoming.

Stephens, C. R. and Waelbroeck, H. (1997). Effective degrees of freedom in genetic algorithms and the block hypothesis. In Bäck, T., editor, *Proceedings of the Seventh International Conference on Genetic Algorithms (ICGA97),* pages 34–40, East Lansing. Morgan Kaufmann.

Stephens, C. R. and Waelbroeck, H. (1998). Effective degrees of freedom in genetic algorithms. *Phys. Rev.,* 57:3251–3264.

Stephens, C. R. and Waelbroeck, H. (1999). Schemata evolution and building blocks. *Evolutionary Computation,* 7(2): 109–124.

Stephens, C. R., Waelbroeck, H., and Aguirre, R. (1999). Schemata as building blocks: Does size matter? In Banzhaf, W. and Reeves, C., editors, *Foundations of Genetic Algorithms 5,* pages 117–133. Morgan Kaufmann, San Francisco, CA.

van Nimwegen, E., Crutchfield, J. P., and Huynen, M. A. (1999). Neutral evolution of mutational robustness. *Proc. Natl. Acad. Sci. USA,* 96:9716–9720.

Vose, M. D. (1999). *The simple genetic algorithm: Foundations and theory.* MIT Press, Cambridge, MA.

Whitley, D. (1992). An executable model of a simple genetic algorithm. In Whitley, D., editor, *Foundations of Genetic Algorithms Workshop (FOGA-92),* Vail, Colorado.

Wright, S. (1932). The roles of mutation, inbreeding, crossbreeeding and selection in evolution. In Jones, D. F., editor, *Proceedings of the Sixth International Congress on Genetics*, volume 1, pages 356–366.

Wright, S. (1967). "Surfaces" of selective value. *Proc. Nat. Acad. Sci. USA,* 58:165–172.

This page intentionally left blank

Chapter 8

ASYMPTOTIC CONVERGENCE OF SCALED GENETIC ALGORITHMS TO GLOBAL OPTIMA

A gentle introduction to the theory

Lothar M. Schmitt
School of Computer Science and Engineering, The University of Aizu
Aizu- Wakamatsu City, Fukushima Prefecture 965-8580, Japan
lothar@u-aizu.ac.jp

THIS WORK IS DEDICATED TO JOANNE WATARI AND MAKOTO YOSHIDA IN AIZU-WAKAMATSU.

Abstract We present a self-contained theoretical framework for a scaled genetic algorithm over the alphabet {0,1} which converges asymptotically to global optima as anticipated by Davis and Principe in analogy to the simulated annealing algorithm. The algorithm employs multiple-bit mutation, single-cut-point crossover and power-law scaled proportional fitness selection based upon an arbitrary fitness function. In order to achieve asymptotic convergence to global optima, the mutation and crossover rates have to be annealed to zero in proper fashion, and power-law scaling is used with logarithmic growth in the exponent. Our analysis shows that a large population size allows for a particularly slow annealing schedule for crossover. For the foremost described setting, a detailed listing of theoretical aspects is presented including prerequisites on inhomogeneous Markov chains. In particular, we focus on: **(i)** The drive towards uniform populations in a genetic algorithm. **(ii)** Weak and strong ergodicity of the inhomogeneous Markov chain describing the probabilistic model for the scaled algorithm. **(iii)** Convergence to globally optimal solutions. We discuss various generalizations and extensions of the core framework presented in this exposition such as larger alphabets or other versions of the mutation-crossover operator, in particular, the Vose-Liepins version of mutation-crossover. This refers to recent work by the author in [Theoretical Computer Science 259 (2001), 1–61] and [Technical Report 2002-2-002, Aizu University] where similar types of algorithms are considered over an arbitrary-size alphabet and convergence for arbitrary fitness function under more general conditions is shown. Finally, we present an outlook on further developments of the theory.

Keywords: asymptotic convergence of genetic algorithms, multiple-bit mutation, single-cutpoint crossover, unbounded power-law scaled proportional fitness selection, simulated annealing, inhomogeneous Markov chains.

CONTENT

INDEX OF SYMBOLS AND KEYWORDS IN ORDER OF APPEARANCE

1.1: $\mathbf{Z}$, $\mathbf{R}$, $\mathbf{R}^+$, $\mathbf{C}$, $\overline{\lim}$, $\underline{\lim}$, Ω_*, $\mathbf{R}_*^+$, $\mathbf{N}_0$, $\mathbf{N}$, $\delta_{n,m}$, b_κ, $\wp$, $< \cdot >$, $\| \cdot \|_1$, vector in free vector space, $\mathrm{Re}(v)$, $\mathrm{Im}(v)$, $\mathcal{S}_\wp$, compact, (e. 1.2: $\mathbf{M}_k(\Omega)$, fully positive, $\mathbf{M}_k$, sp(.), $\mathbf{1}$, $P(\cdot)$, $P(e)$, $X^{[\cdot]}$, $\mathbf{f}$, $\prod_{\tau=t}^{s} X_\tau$, operator norm $\| \cdot \|_1$. 1.3: stochastic. 1.4: A, $a(\iota)$, ℓ, $\mathcal{C}$, $\wp$, s, L, $p \wedge J$, $\mathrm{set}(p)$, $c \in p$, spot, Δ, $\mathcal{V}_\wp$, $\mathcal{U}$, $\wp \cap \mathcal{U}$, $P_\mathcal{U}$. 2.1: multiple-spot mutation M_μ. 2.2: $C(\sigma; \lambda)$, $\bar{C}(\sigma)$, single-cutpoint regular crossover C_χ. 2.3: f, D_f, selector mask J, $<_f$, $\leq_f$, $\mathcal{C}_{\max}$, $\rho_2(f)$, unbounded power-law scaling f_t, logarithmic exponentiation schedule g, B, proportional fitness selection operator S_t, θ. 3.1: mutation flow inequality, β_μ. 3.2: $G_t = S_t C_{\chi(t)} M_{\mu(t)}$, $H_t^{t'}$, trajectories, annealing schedule for mutation $\mu(t) = (t+1)^{-1/L}/2$. 3.3: annealing schedules for crossover $\chi(t) = \mu(t)^{1/m}$, v_∞. 3.4: steady-state flow inequality (line 26). 3.5: r, VLGA.

INTRODUCTION

Every endeavor such as the quest for new frontiers in theory of genetic algorithms must have a port of embarkation established on firm ground. This work tries, conservatively, to describe a theoretical basis for research on scaled ge-

netic algorithms that use unbounded power-law scaling for the fitness function and otherwise standard operations for mutation and crossover. However, at the end of the exposition, we shall reach a point where the reader is challenged to engage in research on a number of quite non-trivial, theoretical, open problems but with a solid foundation in mind. The target audience of this exposition are, in general, scholars interested in the field of evolutionary computation. The author has tailored the presentation in such a way that it should be conveniently accessible for advanced undergraduate or beginning graduate students in computer science with a solid, standard mathematical background, or a lecturer who wants to include the topic of scaled genetic algorithms into a course and searches for a mostly self-contained framework.

As primary goal of this work, we shall rigorously establish the Global Optimization Theorem 3.4.1 which shows that a properly scaled genetic algorithm converges *for arbitrary fitness function* to a probability distribution over uniform populations containing only elements of maximal fitness.

There are a number of excellent surveys on genetic algorithms: for example, essays by Beyer, Schwefel & Wegener (Beyer et al., 2002), Mühlenbein (Mühlenbein, 1997), and the monographs by Goldberg (Goldberg, 1989), Mitchell (Mitchell, 1996), and Vose (Vose, 1999b). As Mühlenbein points out in the introduction to (Mühlenbein, 1997), evolutionary algorithms based upon "mutation, mating, and selection" were already introduced in the 1960's as a tool for optimization. See the paper by Bremermann, Rogson & Salaff (Bremermann et al., 1966). Genetic algorithms, a particular case of evolutionary algorithms, were invented by Holland (Holland, 1975) and are by now a well-established tool for search and optimization. A given optimization task is encoded in such a way that candidate solutions are understood as elements in a finite collection $\mathcal{C}$ of creatures in a model "world", and a fitness function $f : \mathcal{C} \to \mathbf{R}^+$ exists which has to be maximized. In the model for genetic algorithms presented in this exposition, creatures (candidate solutions) are identified with their genetic information which consists of an ordered string of letters selected from {0, 1}. The collection of creatures in the current, fixed-size population p is subject to three operations: mutation, crossover, and selection which are applied cyclically and iteratively until a termination condition is satisfied. A genetic algorithm is called *simple,* if the three operations mutation, crossover, and selection stay constant over the course of the algorithm.

In this work, we shall be interested in the asymptotic behavior of the genetic algorithm, *i.e.,* the probabilistic behavior of the algorithm if never halted. Asymptotic behavior of genetic algorithm has been investigated by many authors: Agapie (Agapie, 2001), Aytug & Koehler (Aytug and Koehler, 1996), Cerf (Cerf, 1996; Cerf, 1998) Davis&Principe (Davis, 1991; Davis and Principe, 1991; Davis and Principe, 1993), Fogel (Fogel, 1994), Goldberg (Goldberg, 1990), He & Kang (He and Kang, 1999), Holland (Holland, 1975) Leung,

Chen, Xu & Leung (Leung et al., 1998), Liepins & Vose (Vose and Liepins, 1991), Lozano, Larrañaga, Graña & Albizuri (Lozano et al., 1999), Mahfoud & Goldberg (Mahfoud, 1993; Mahfoud and Goldberg, 1992; Mahfoud and Goldberg, 1995), Nix & Vose (Nix and Vose, 1992), Poli & Langdon (Poli, 2001; Poli and Langdon, 1998), Rodolph (Rudolph, 1994), Suzuki (Suzuki, 1997; Suzuki, 1998), and Vose (Vose, 1999b). However, a proof of asymptotic convergence for a genetic algorithm of fixed, relatively small population size using scaled proportional fitness selection has only recently been obtained in (Schmitt, 2001, Thm. 8.6, Rem. 8.7). Other work that claims a result of convergence to global optima such as (Rudolph, 1994; He and Kang, 1999; Greenwood and Zhu, 2001; Agapie, 2001; Cerf, 1996; Cerf, 1998) or (Vose, 1999b, Ch. 3, p. 147) require usually some auxiliary, special condition, limit or setting to achieve their goal such as the trivial elitist selection strategy or infinite population limit. The analysis presented here and in (Schmitt, 2001; Schmitt, 2002) sets boundary conditions for proper design and implementation of genetic algorithms that actually do stop after a finite but large number t of cycles. Let G_t denote the stochastic matrix describing the individual step of the scaled genetic algorithm at time t, and let $v_t = G_t v_t$ be the steady-state probability distribution of G_t. Theorem 3.3.2 establishes strong ergodicity of the inhomogeneous Markov chain $(G_t)_{t\in\mathbf{N}}$. It shows that for large t, the probability distribution describing the state of the algorithm after t steps is close to the limit v_∞ of the v_t. This allows for development of a stopping-criterion for a genetic algorithm which is scaled as in the Global Optimization Theorem 3.4.1. These aspects are discussed after the proof of Theorem 3.3.2. Compare work by Aytug & Koehler (Aytug and Koehler, 1996) and Leung, Chen, Xu& Leung (Leung et al., 1998) in this regard.

The concept of ergodicity plays an important role in the behavior of the scaled genetic algorithm considered in the Global Optimization Theorem 3.4.1. Weak ergodicity as discussed in section 3.2 assures that the (probabilistic) trajectories of the algorithm are independent from initial populations. If the algorithm is started at population p, then the probability distribution describing the state of the algorithm after t steps is given by

$$w(p)_t = G_t \cdot G_{t-1} \cdots G_1 \cdot p.$$

Weak ergodicity essentially says that for any initial populations p or q, the states $w(p)_t$ and $w(q)_t$, *i.e.,* probabilistic trajectories will be arbitrary close for $t \to \infty$. The shrinking of the distance $||w(p)_t - w(q)_t||_1 \to 0$ as $t \to \infty$ is due to a shrinking property of scaled mutation which is established in Proposition 1.3.1 combined with Proposition 2.1.2.2.

Strong ergodicity as discussed in section 3.3essentially assures that $\lim_{t\to\infty} w(p)_t$ exists and is *independent* of the initial population p. Theorem 3.3.2 actually determines that $\lim_{t\to\infty} w(p)_t = v_\infty$ with v_∞ as defined above.

As a consequence, we can discuss v_∞ as limit of the v_t via the steady-state flow inequality established in the proof of the Global Optimization Theorem 3.4.1 (line *26*) to obtain convergence to global optima. Strong ergodicity is in our case a somewhat immediate technical consequence of weak ergodicity due to the nature of the entries of the stochastic matrices describing the scaled genetic algorithm. See the proof of Theorem 3.3.2 for details.

The mathematical model presented in this exposition uses an inhomogeneous Markov chain over a finite population space $\wp$. Thus, the state space of the genetic algorithm consists of probability distributions over populations. Most authors model populations as multi-sets following the work of Davis & Principe (Davis, 1991; Davis and Principe, 1991; Davis and Principe, 1993), Liepins&Vose (Vose and Liepins, 1991), and Nix & Vose (Nix and Vose, 1992). A more general Markovian framework for stochastic search methods using multi-sets is given by Vose's theory of random heuristic search (Vose, 1999a; Vose, 1999b). As in (Schmitt et al., 1998; Schmitt and Nehaniv, 1999; Schmitt, 2001; Schmitt, 2002), the model used in this exposition considers populations as strings of letters in the underlying alphabet and not as multi-sets. As outlined in (Schmitt, 2001, Sec. 2.9), the multi-set model can easily be embedded into the tensor-string model considered here. Rudolph (Rudolph, 1994) developed his Markov chain model for genetic algorithms in the tensor-string model approximately around the same time as (Schmitt et al., 1998).

What makes our approach different is that we do not attempt to unite the genetic operators mutation, crossover and selection to *one* operator which is subsequently analyzed. We rather analyze the genetic operators separately to isolate key properties: Crossover plays a dual role enhancing mutation in the mixing phase of the algorithm (Schmitt, 2001, Thm. 6.1) as well as enhancing selection in the contraction-phase of the algorithm in some cases (Schmitt, 2002, Sec. 2.5, Prop. 2.6.2). Mutation is responsible for weak ergodicity (see section 3.2) and the probabilistic flow away from uniform populations (Proposition 3.1.1.2). Fitness-selection is responsible for contraction towards uniform populations (Proposition 2.3.2.4). Mutation-selection is responsible for convergence to uniform populations in the zero mutation-rate limit (Theorem 3.1.2.3). All three genetic operators act together to obtain the steady-state flow inequality which shows convergence to global optima (proof of the Global Optimization Theorem 3.4.1, line *26*).

Research by Davis & Principe (Davis and Principe, 1993) advocates to establish a framework for scaled genetic algorithms in analogy to the simulated annealing algorithm (Aarts and van Laarhoven, 1989). The main result of this exposition, the Global Optimization Theorem 3.4.1, achieves the following goals: A general-purpose, scaled, genetic algorithm is described that converges to global optima. The setup is quite similar to that of the simulated annealing algorithm. There are no special requirements in regard to the fitness function or

the fitness-landscape. The number of creatures s in populations can stay small and can be set as low as $s = 2\ell + 1$ where ℓ is the length of the genome of creatures (candidate solutions) as strings over $\{0,1\}$. Explicit cooling schedules for mutation and crossover, and exponentiation schedules for fitness-selection are given. The genetic algorithm presented in the Global Optimization Theorem 3.4.1 consequently satisfies *all* goals formulated by Davis & Principe (Davis and Principe, 1993, p. 270).

At the end of this exposition, we shall discuss future extensions of genetic algorithm theory as seen by this author. It is emphasized that the theory should focus on analysis of finite length algorithms on finite state machines following, *e.g.,* work by Catoni (Catoni, 1990; Catoni, 1991b; Catoni, 1991a; Catoni, 1992) on the simulated annealing algorithm. Other interesting aspects of the theory of simulated annealing such as adding sampling noise following work by Kushner (Kushner, 1987) and considering sparse mutation matrices are also discussed. In addition, a case is made for a systematic analysis of custom-designed genetic algorithms in regard to specific problem instances which in a way inverts the point of view put forward in this exhibition of genetic algorithms as a *black-box all-purpose method.*

1. NOTATION AND PRELIMINARIES

Before we describe the proposed scaled genetic algorithm, investigate its components, and prove its asymptotic convergence, we need to collect a number of definitions and elementary facts in this section. The notation used here is essentially the same as in (Schmitt, 2001; Schmitt, 2002). We shall assume that the reader is fluent in linear algebra, calculus and basic probability theory. For a reference on linear algebra, see the introductory monographs by Lang (Lang, 1970) and Greub (Greub, 1975). For a reference on calculus and analysis including some Banach space theory, see the introductory monograph by Lang (Lang, 1970). In regard to probability theory, we first refer the reader to the treatise by Feller (Feller, 1968; Feller, 1971) as well as to the books by Chung (Chung, 1974), Isaacson & Madsen (Isaacson and Madsen, 1961) and Seneta (Seneta, 1981).

1.1 SCALARS AND VECTORS

Let $\mathbf{Z}$, $\mathbf{R}$, $\mathbf{R}^+$, and $\mathbf{C}$ denote the integers, the real numbers, the non-negative real numbers, and the complex numbers, respectively. Let $\overline{\lim}$ denote limes superior, and $\underline{\lim}$ denote limes inferior. For any subset Ω of a vector space, let $\Omega_* = \Omega \setminus \{0\}$. Set $\mathbf{N}_0 = \mathbf{Z} \cap \mathbf{R}^+$, and $\mathbf{N} = (\mathbf{N}_0)_*$. For elements κ, κ' of a set, let $\delta_{\kappa,\kappa'} = 1$, if $\kappa = \kappa'$, and let $\delta_{\kappa,\kappa'} = 0$ otherwise, *i.e.,* δ is the

Kronecker delta. Let $b_\kappa = (\delta_{\kappa,\kappa'})_{\kappa'=0}^{k-1}$ denote the standard unit vectors[1] in $\mathbf{C}^k$, $0 \le \kappa \le k-1$, $k \in \mathbf{N}$. Let $\wp = \{b_\kappa : 0 \le \kappa \le k-1\}$ be the set of unit vectors in $\mathbf{C}^k$. Recall that for $v = (v_\kappa), w = (w_\kappa) \in \mathbf{C}^k$ the canonical inner product of v and w, and the usual Hamming-norm or ℓ^1-norm of v are given by

$$<w, v> = \sum_{\kappa=0}^{k-1} \bar{w}_\kappa v_\kappa, \quad \text{and} \quad ||v||_1 = \sum_{\kappa=0}^{k-1} |v_\kappa|. \tag{1}$$

Note that we can express any $v \in \mathbf{C}^k$ uniquely in the form $v = \sum_{b \in \wp} <b, v> \cdot b$, *i.e.*, v is expressed in the format of a vector in the free vector space over the $b \in \wp$ (sum over coefficients times symbols b). Let $\mathrm{Re}(v) = (\mathrm{Re}(v_\kappa)) \in \mathbf{R}^k$, and $\mathrm{Im}(v) = (\mathrm{Im}(v_\kappa)) \in \mathbf{R}^k$ be the vectors of real resp. imaginary parts of entries of $v = (v_\kappa) \in \mathbf{C}^k$.

Let $\mathcal{S}_\wp = \{v \in (\mathbf{R}^+)^k : ||v||_1 = 1\}$ be the set of probability distributions over the set of "pure states" $\{0, \ldots, k-1\}$, or equivalently, let $\mathcal{S}_\wp$ be the set of all convex combinations of elements of $\wp$. $\mathcal{S}_\wp$ is closed (under limit-taking) (Lang, 1968, p. 111: Thm. VI.6) and bounded in $\mathbf{C}^k$. Hence, $\mathcal{S}_\wp$ is compact, and any sequence of vectors in $\mathcal{S}_\wp$ has a convergent subsequence (Lang, 1968, p. 140: Thm. VIII.5). $\mathcal{S}_\wp$ is the relevant state space of our investigation.

We shall use the notation x^* to denote the adjoint of a vector or matrix x. Let $e = k^{-1} \cdot (1, 1, \ldots, 1)^* \in \mathcal{S}_\wp$. The notation for e differs by a factor 2^{-L} from the notation introduced in (Schmitt et al., 1998, p. 104), but coincides with the notation used in (Schmitt and Nehaniv, 1999; Schmitt, 2001; Schmitt, 2002).

1.2 MATRICES AND OPERATOR NORMS

Let $\mathbf{M}_k(\Omega)$, $k \in \mathbf{N}$, denote the set of $k \times k$ matrices with entries in a set Ω. A matrix in $\mathbf{M}_k(\mathbf{R}_*^+)$ will be called *fully positive*. Let $\mathbf{M}_k = \mathbf{M}_k(\mathbf{C})$. A matrix $X = (X_{\kappa',\kappa}) \in \mathbf{M}_k$ will operate by matrix multiplication from the left on column vectors in $\mathbf{C}^k$. Note that a number of authors including Isaacson &Madsen (Isaacson and Madsen, 1961) and Seneta (Seneta, 1981) use row vectors and matrix multiplication from the right. Realizing that $X_{\kappa',\kappa} = <b_{\kappa'}, X b_\kappa>$ allows to express the coefficients of X in such a way that larger numbers of subscripts attached to a particular symbol are avoided. The set of eigenvalues of X, *i.e.*, the spectrum of X will be denoted as $\mathrm{sp}(X)$.

The matrix associated with the identity map $\mathbf{C}^k \to \mathbf{C}^k$ will be denoted by $\mathbf{1}$. For any $v \in \mathcal{S}_\wp$, let $P(v) = (v, v, \ldots, v) \in \mathbf{M}_k$. $P(v)$ is a projection, *i.e.*, $P(v) = P(v)^2$, but only for $v = e$ an orthogonal projection, *i.e.*, $P(e) = P(e)^*$. If $X \in \mathbf{M}_k$, and $v \in \mathbf{C}^k$ is a row vector, then we shall write $X^{[v]}$ for

[1]Coefficients of vectors or matrices are enumerated with indices starting from 0 since this establishes a natural correspondence of indices with the canonical order on populations as discussed in section 1.4 as well as the canonical tensor product construction for the free vector space $\mathcal{V}_\wp$ over populations which is discussed in (Schmitt. 2001, Sec. 2.6).

the matrix obtained from X by replacing the first row of X with v. Let the flip-matrix $\mathbf{f} \in \mathbf{M}_2(\{0,1\})$ be defined as follows:

$$\mathbf{f}_{0,0} = 0 = \mathbf{f}_{1,1}, \qquad \mathbf{f}_{0,1} = 1 = \mathbf{f}_{1,0}, \qquad \mathbf{f} \in \mathbf{M}_2(\{0,1\}). \tag{2}$$

We shall use the $\prod$-**symbol** to denote products of possibly *non-commuting* matrices as follows ($t, t' \in \mathbf{Z}, t \neq t'$):

$$\prod_{\tau=t}^{t'} X_\tau = X_t \cdot X_{t+\mathrm{sign}(t'-t)} \cdots X_{t'}, \quad \prod_{\tau=t}^{t} X_\tau = X_t, \quad X_\tau \in \mathbf{M}_k. \tag{3}$$

The operator norm of $X = (X_{\kappa',\kappa}) \in \mathbf{M}_k$ with respect to the Hamming norm on $\mathbf{C}^k$ is given by

$$||X||_1 = \sup\{||Xv||_1 : v \in \mathbf{C}^k, ||v||_1 = 1\} \tag{4}$$

as in (Schaefer, 1974, p. 5: eq. (5)). The **proofs** of the following simple facts listed in lines (*5–7*) are given in section 5:

$$||X + Y||_1 \leq ||X||_1 + ||Y||_1 \text{ for every } X, Y \in \mathbf{M}_k \text{ (triangle ineq.).} \tag{5}$$

$$||X \cdot Y||_1 \leq ||X||_1 \cdot ||Y||_1 \text{ for every } X, Y \in \mathbf{M}_k. \tag{6}$$

$$||X||_1 = \max\{\textstyle\sum_{\kappa'} |X_{\kappa',\kappa}| : \kappa\} = \max\{||(X_{\kappa',\kappa})_{\kappa'}||_1 : \kappa\}, \tag{7}$$

Line (*7*) is identical to (Schaefer, 1974, p. 5: eq. (7'))- It says that $||X||_1$ is the maximum over the ℓ^1-norms of the columns of X, and it shows that component-wise convergence of matrices is exactly the same as $||\cdot||_1$-wise convergence. As a direct consequence of line (*7*), we also obtain the following results:

$$\text{If } P \in \mathbf{M}_k(\{0,1\})_* \text{ is a diagonal matrix, then } ||P||_1 = 1. \tag{8}$$

$$\text{If } v, w \in \mathcal{S}_\wp \text{ then } ||P(v) - P(w)||_1 = ||v - w||_1. \tag{9}$$

1.3 STOCHASTIC MATRICES

In this section, we shall take advantage of the fact that the mutation operator contributes fully positive matrices to the inhomogeneous Markov chain describing the probabilistic model of the scaled genetic algorithm considered in this exposition (see Proposition 2.1.2.1). As a benefit, we can circumvent most of the more general theory of stochastic matrices and establish key ingredients of our mathematical framework for scaled genetic algorithms such as (1) "weak ergodicity of the inhomogeneous Markov chain describing the algorithm" and (2) "the existence of a uniquely determined steady state distribution of a single step of the genetic algorithm" in a short, simple and mostly self-contained way.

A matrix in $X \in \mathbf{M}_k(\mathbf{R}^+)$ is called column-stochastic or for short *stochastic,* if each of its columns sums to 1. The next line lists several basic facts about stochastic matrices. The **proof** is given in section 5.

If X, Y are stochastic, then: $||X||_1 = 1$, $XS_\wp \subset S_\wp$, and
XY is stochastic. *(10)*

If we assume that a probabilistic algorithm such as a genetic algorithm acts on a state space $\{0, \ldots, k-1\}$, then a single step at time $t \in \mathbf{N}$ of such an algorithm shall be described by a stochastic matrix $X_t \in \mathbf{M}_k$. In that case, $<b_{\kappa'}, X_t b_\kappa>$ equals the transition probability from state κ to state κ' in step t of the algorithm. If we consider the stochastic matrices X_t and X_{t+1} associated with two consecutive steps and ask for the transition probability from state κ to state κ' under both steps combined, then we have to consider all possible paths and disjoint events $\kappa \mapsto \nu \mapsto \kappa'$ for $\nu \in \{0, \ldots, k-1\}$. For a particular $\nu \in \{0, \ldots, k-1\}$, we know that transition $\kappa \mapsto \nu$ occurs with probability $<b_\nu, X_t b_\kappa>$. The $<b_{\kappa'}, X_{t+1} b_\nu>^{\text{th}}$ portion of the latter probability is then the probability for passage $\kappa \mapsto \nu \mapsto \kappa'$. Thus, the total probability for passage from state κ to state κ' under steps t and $t+1$ combined is given by

$$\sum_{\nu=0}^{k-1} <b'_\kappa, X_{t+1} b_\nu> \cdot <b_\nu, X_t b_\kappa> = <b'_\kappa, (X_{t+1} \cdot X_t)\, b_\kappa>. \qquad (11)$$

Hence, transition probabilities for combined steps of a probabilistic algorithm are given by the combined matrix product of the associated stochastic matrices.

See the first chapter of Schaefer's book (Schaefer, 1974) for a good and short introduction to theoretical aspects of stochastic matrices. One may be inclined to develop a theory for stochastic matrices and genetic algorithms over real vector spaces. However, it is useful to develop such theory over complex vector spaces since certain aspects such as spectral calculus (Rudin, 1973, p. 243: eq. (2)) and Frobenius' Theorem (Schaefer, 1974, p. 22: Thm. 6.5) are inherently complex theory. This is demonstrated in (Schaefer, 1974) and (Schmitt et al., 1998; Schmitt and Nehaniv, 1999; Schmitt, 2001; Schmitt, 2002). In particular, see (Schmitt and Nehaniv, 1999, Lemma 7.1.3) for a connection between the steady-state distribution of a fully positive stochastic matrix and complex spectral calculus as well as (Schmitt, 2002, section 2.1) for a discussion of the spectrum of mutation in relation to the unitary cyclic-shift operator acting on the free vector space over the underlying alphabet $\mathcal{A}$.

Section 3.2 discusses weak ergodicity of the inhomogeneous Markov chain underlying the model for the scaled genetic algorithm considered in this exposition. The following result shortens the discussion of weak ergodicity considerably compared to, *e.g.,* (Isaacson and Madsen, 1961; Seneta, 1981). The **proof** of Proposition 1.3.1 is listed in section 5.

1.3.1. Proposition. Let $\epsilon_t \in (0, 1/k]$, $t \in \mathbf{N}$ such that $\sum_{t \in \mathbf{N}} \epsilon_t = \infty$. Let $M_t \in \mathbf{M}_k([\epsilon_t, 1])$ and $X_t \in \mathbf{M}_k$ be stochastic matrices for $t \in \mathbf{N}$. Let $v, w \in S_\wp$. Then we have

1. $||M_t(v-w)||_1 \leq (1 - \epsilon_t k)\, ||v-w||_1$

2. $\lim_{t\to\infty}(\prod_{\tau=t}^{1} X_\tau \cdot M_\tau)(v - w) = 0.$

In many papers (*e.g.,* (Davis and Principe, 1991; Davis and Principe, 1993; Schmitt et al., 1998; Schmitt and Nehaniv, 1999; Schmitt, 2001; Schmitt, 2002; Suzuki, 1997; Suzuki, 1998; van Nimwegen et al., 1999)), the machinery of Perron-Frobenius theory (Schaefer, 1974, p. 22: Thm. 6.5, p. 23: Cor. 1) is invoked to obtain the uniquely determined steady-state distribution of the stochastic matrix describing a single step of a genetic algorithm. The **proofs** of Proposition 1.3.2, Corollary 1.3.3 and Corollary 1.3.4 which are listed in section 5 develop a simple framework based essentially on the compactness of $\mathcal{S}_\wp$ to achieve the same objective. We start with listing the relevant result for fully positive, stochastic matrices.

1.3.2. Proposition. Let $X \in \mathbf{M}_k$ be a fully positive, stochastic matrix. Then there exists $v \in (\mathbf{R}_*^+)^k \cap \mathcal{S}_\wp$ such that $Xv = v$, *i.e.,* v is a fully positive, normalized right eigenvector of X to eigenvalue 1. If $w \in \mathbf{C}^k$ is such that $Xw = w$, then $w = \zeta v$ for some $\zeta \in \mathbf{C}$, *i.e.,* the eigenspace pertaining to eigenvalue 1 of X is one-dimensional.

Perron-Frobenius theory, in particular, (Schaefer, 1974, p. 23: Cor. 2) shows that 1 is actually the *only* eigenvalue of absolute value 1 of a fully positive stochastic matrix X. A simple discussion of stretching the norm of eigenvectors under application of a stochastic matrix X shows that there are no eigenvalues with absolute value strictly greater than 1 in view of line (*10*), *i.e.,* $|\mathrm{sp}(X)| \leq 1$.

1.3.3. Corollary. Let $X \in \mathbf{M}_k$ be a stochastic matrix. Then there exists $v \in \mathcal{S}_\wp$ such that $Xv = v$, *i.e.,* v is an right eigenvector of X to eigenvalue 1 and a probability distribution.

The proof of Corollary 1.3.3 listed in the appendix is simpler than (Schaefer, 1974, p. 7: Prop. 2.3). The following result will be used to show that the stochastic matrix associated with an individual step of a scaled genetic algorithm has a positive, invariant right eigenvector which is uniquely determined up to scalar multiples.

1.3.4. Corollary. Let $M \in \mathbf{M}_k(\mathbf{R}_*^+)$ and $X \in \mathbf{M}_k$ be stochastic matrices. Then MX is fully positive and possesses an invariant right eigenvector $v = MXv \in \mathbf{C}^k$ which as such is uniquely determined up to scalar multiples. In addition, $v \in (\mathbf{R}_*^+)^k \cap \mathcal{S}_\wp$. If M is invertible, then, $M^{-1}v$ is an invariant right eigenvector of XM which as such is uniquely determined up to scalar multiples. In addition, $M^{-1}v \in \mathcal{S}_\wp$.

1.4 CREATURES AND POPULATIONS

In this introductory exposition, we shall only consider binary genetic algorithms which use the underlying alphabet $\mathcal{A} = \{a(0), a(1)\}$. However, in applications where real parameters are optimized in a compact domain of $\mathbf{R}^\ell$, $2 \leq \ell \in \mathbf{N}$, it may be advantageous to consider a larger, discrete alphabet representing a finite, equidistant set of real numbers[2]. Such an approach is advocated and used, *e.g.,* in work by Markus, Renner, & Vanza (Márkus et al., 1997, p. 48), Kondoh & Schmitt (Schmitt and Kondoh, 2000), and Savchenko& Schmitt (Savchenko and Schmitt, 2001). See also work by Nomura & Shimohara (Nomura and Shimohara, 2001). One task in future work is certainly to generalize the approach taken here and in (Márkus et al., 1997; Savchenko and Schmitt, 2001; Schmitt and Kondoh, 2000; Schmitt, 2001; Schmitt, 2002) to the case of a continuous alphabet.

We shall consider (the genome of) creatures or candidate solutions in the model world to which the genetic algorithm is applied as strings of length ℓ over the alphabet $\mathcal{A}$ where usually $2 \leq \ell \in \mathbf{N}$. Let $\mathcal{C} = \mathcal{A}^\ell$ denote the set of creatures.

The set of populations $\wp$, to which the genetic algorithm is applied, is the set of **s-tuples** of creatures, $s \in \mathbf{N}$. We shall assume that s is even, and $s \geq 4$, if not explicitly stated otherwise. Set $L = \ell \cdot s$. Then every population is (canonically identified with) a string of length L over $\mathcal{A}$. Let $J \subset \{1, \ldots, s\}$. If $p = (c_1, c_2, \ldots, c_s)$ is a population, $c_\sigma \in \mathcal{C}$, $1 \leq \sigma \leq s$, then we define $p \wedge J = (c_\sigma)_{\sigma \in J}$ and $\mathrm{set}(p) = \{c_\sigma : 1 \leq \sigma \leq s\}$. If $c \in \mathcal{C}$, then we shall write $c \in p$, if $c \in \mathrm{set}(p)$. A *spot* in the genome is, by definition, the position of one of the letters in a word over $\mathcal{A}$ representing a creature or population. For $p, q \in \wp$, we define the Hamming distance $\Delta(p, q)$ as the number of spots in the genome where p and q differ.

The vector space $\mathcal{V}_\wp$ underlying our model for genetic algorithms is the free complex vector space[3] over $\wp$. Thus, $\wp$ becomes the basis of $\mathcal{V}_\wp$ which is consistent with the notation introduced in section 1.1. Every population $p \in \wp$ can be identified canonically with an integer $\tilde{p}$ in $[0, 2^L - 1]$, *i.e.*, the letters comprising p are used as digits to define the integer $\tilde{p}$ in binary representation. This induces a natural order on $\wp$. Now, we identify $p \in \wp \subset \mathcal{V}_\wp$ with $b_{\tilde{p}} \in \mathbf{C}^{2^L}$,

[2] If a regular programming language such as Fortran or C is employed for the Implementation of the genetic algorithm, then only a finite set $R_o \subset \mathbf{R}$ of real numbers is used. In many cases, the search space can be restricted further by a rough analysis of the given optimization problem to a finite interval $\mathcal{A} = \{a(\iota) = x_o + \iota\delta : 0 \leq \iota < \alpha \in \mathbf{N}\} \subset R_o, x_o, \delta \in \mathbf{R}$.

[3] (Schmitt, 2001, Sec. 2.6) discusses the identification of $\mathcal{V}_\wp$ with the L-fold tensor product of the free vector space over $\mathcal{A}$. This can be used for analysis of mutation as in (Schmitt, 2001, Prop. 3.3, Prop. 3.6) and various crossover operators as in (Schmitt, 2002, Sec. 2.4–5).

and by linear extension, this defines an isomorphism $\mathcal{V}_{\wp} \to \mathbf{C}^{2^L}$. Let $p, q \in \wp$. By suppressing notation for the particular isomorphism $\mathcal{V}_{\wp} \to \mathbf{C}^{2^L}$ just defined, we can consistently denote the q-component of Xp where X stands for a linear operator acting on $\mathcal{V}_{\wp}$ *and* the corresponding matrix acting on $\mathbf{C}^{2^L}$ as

$$X_{q,p} = <q, Xp> = <b_{\tilde{q}}, Xb_{\tilde{p}}> = X_{\tilde{q},\tilde{p}}. \tag{12}$$

Let $\mathcal{U} \subset \mathcal{V}_{\wp}$ be the free vector space over all populations which are uniform, *i.e.*, which consist of s copies of a single creature. Consequently, $\wp \cap \mathcal{U}$ shall denote the set of uniform populations. In addition, $P_{\mathcal{U}}$ shall denote the orthogonal projection onto $\mathcal{U}$. The following simple result whose **proof** is listed in section 5 allows for compact notation in some of the subsequent results and proofs.

1.4.1. Proposition. Let $X : \mathcal{V}_{\wp} \to \mathcal{V}_{\wp}$ be a linear map such that $Xp = p$ for every $p \in \wp \cap \mathcal{U}$. Then X satisfies $XP_{\mathcal{U}} = P_{\mathcal{U}}$, and $(1 - P_{\mathcal{U}})X = (1 - P_{\mathcal{U}})X(1 - P_{\mathcal{U}})$.

2. THE GENETIC OPERATORS

The genetic operators can be categorized into two groups: the mixing operators mutation and crossover which are used concurrently in a genetic algorithm, and the selection operators such as proportional fitness selection, tournament selection or simulated annealing type selection which provide alternatives for implementation of a genetic algorithm. However, crossover has some common features with selection such as leaving uniform populations invariant.

Mutation-crossover has been investigated by many researchers. The most common and simple framework investigated is multiple-spot mutation combined with single-cutpoint crossover in the multi-set model for populations over a binary alphabet. Earlier references include the work of Davis & Principe (Davis and Principe, 1991; Davis and Principe, 1993), Vose & Liepins (Vose and Liepins, 1991), and Nix & Vose (Nix and Vose, 1992). Other work which has significance to the present work are the papers by Suzuki (Suzuki, 1997; Suzuki, 1998) (compare (Schmitt, 2001, Sec. 8.3)). In (Vose and Wright, 1998a; Vose and Wright, 1998b), Vose & Wright discuss the mutation-crossover matrix via the Walsh-transform. Note that the Vose-Liepins version of mutation-crossover is different than the mutation-crossover operation discussed here. Section 3.5 discusses how to embed the Vose-Liepins version of mutation-crossover into the model presented here.

In contrast to popular belief, one must observe that mutation and not crossover is the main thriving force for mixing in a genetic algorithm. Mutation assures weak ergodicity, and as an immediate consequence strong ergodicity of the Markov chain describing the mathematical model for the genetic al-

gorithm (see section 3.2 and Theorem 3.3.2). In the generic situation of a blind search with a fitness function of largely unknown behavior, it is mutation and not crossover that drives the algorithm. In particular, mutation creates the noise that destroys uniform populations containing suboptimal solutions which is something crossover cannot do. Note that Banzhaf, Francone & Nordin (Banzhaf et al., 1996) report experimental results that favor larger mutation rates, *i.e.,* strong mixing by mutation. There are quite natural situations where crossover asymptotically plays no role in the probabilistic outcome of a genetic algorithm (*cf.* (Schmitt, 2001, Thm. 8.3.3, Thm. 8.5.2–3)). Crossover assists mutation in accelerating the mixing process towards the uniquely determined fix-point e of the fully positive, symmetric mutation-crossover operator $M_\mu C_\chi = C_\chi M_\mu$ (*cf.* Proposition 2.2.3.4). This statement is made very precise in (Schmitt et al., 1998, Prop. 10) and (Schmitt, 2001, Thm. 6.1). However, there are "royal road cases" where by the design of the fitness landscape the acceleration by crossover is significant (*cf.* work by Jansen & Wegener (Jansen and Wegener, 2001)). In regard to the family of selection operators, we shall restrict us here to the case of proportional fitness selection which is discussed in detail in section 2.3. With respect to tournament selection, we refer the reader to work by Goldberg (Goldberg, 1990), (Goldberg,) as well as Goldberg & Deb (Goldberg and Deb, 1991), and to the monographs by Mitchell (Mitchell, 1996, p. 170) and Michalewicz (Michalewicz, 1994, p. 59). In regard to simulated annealing type selection, we refer the reader to the introductory paper by Aarts & Van Laarhoven (Aarts and van Laarhoven, 1989), work by Lozano, Larrañaga, Graña & Albizuri (Lozano et al., 1999) and Mahfoud & Goldberg (Mahfoud and Goldberg, 1992; Mahfoud and Goldberg, 1995), (Schmitt et al., 1998, p. 124: Rem. on Simulated Annealing) and (Schmitt, 2001, Sec. 6.2).

2.1 MULTIPLE-SPOT MUTATION

Mutation models random change in the genetic information of creatures, and is inspired by random change of genetic information in living organisms, *e.g.,* through the effects of radiation or chemical mismatch. Multiple-spot mutation M_μ has been studied theoretically by many authors as discussed in the introductory paragraphs to section 2. See also (Schmitt et al., 1998, Sec. 2.1, p. 110 *ff.,* "multiple-bit mutation"), (Schmitt, 2001, Sec. 3.3), (Schmitt, 2002, Sec. 2.2). In this section, we shall repeat some of the analysis in (Schmitt et al., 1998). However, our discussion here will be limited to the absolute minimum. Multiple-spot mutation is the most commonly used procedure for mutation in implementations of genetic algorithms.

2.1.1. Definition (multiple-spot mutation M_μ). Let $\mu \in [0,1]$ denote the mutation rate, and for $\hat{\lambda} = 1 \ldots L$ execute the following two steps:

(STEP 1) Decide probabilistically whether or not to change the letter at spot $\hat{\lambda}$ in the current population. The decision for change is made positively with probability μ. (STEP 2) If the decision has been made positively in step 1, then the letter at spot $\hat{\lambda}$ is altered, *i.e.,* the bit at spot $\hat{\lambda}$ is flipped.

Let M_μ also denote the stochastic matrix associated with multiple-spot mutation. M_μ acts on $\mathcal{V}_\wp$ in the sense of line (*12*) and describes transition probabilities for entire populations. • • •

If $\mu = 0$, then mutation is the identity operation. If $\mu = 1$, then the current population is bit-wise complemented. In what follows, we shall usually exclude these two trivial cases, if we discuss mutation.

2.1.2. Proposition. Let M_μ denote multiple-spot mutation as in Definition 2.1.1 with mutation rate $\mu \in (0, 1)$. Suppose that $p, q \in \wp$. Then we have:

1 The coefficients of M_μ are given as follows:
$<q, M_\mu p> = \mu^{\Delta(p,q)} \cdot (1-\mu)^{L-\Delta(p,q)} > 0.$
In particular, M_μ is a fully positive and symmetric.

2 If $\mu \in (0, 1/2]$, then $2^{-L}\mu^L \leq <q, M_\mu p>$.

3 If $p \neq q$, then $\lim_{\mu\to 0} <q, M_\mu p> = 0$. In addition, $\lim_{\mu\to 0} <p, M_\mu p> = 1$.

4 If μ is sufficiently small, then M_μ is an invertible matrix.

5 e is the uniquely determined invariant probability distribution of M_μ.

PROOF: In order to pass from p to q, one has to make the decision to change one of p's bits $\Delta(p, q)$ times, and one has to retain p's bits at $L - \Delta(p, q)$ spots. Independently from the order of such steps, the combined probability for the required procedure is given by $\mu^{\Delta(p,q)} \cdot (1-\mu)^{L-\Delta(p,q)}$. One has $<q, M_\mu p> = <p, M_\mu q>$ since $\Delta(p, q) = \Delta(q, p)$. This shows statement (1). If $\mu \leq 1/2$, then $(1-\mu)^{L-\Delta(p,q)} \geq 2^{-L+\Delta(p,q)} \geq 2^{-L}$. This combined with statement (1) shows statement (2). By statement (1), one has $\lim_{\mu\to 0} <p, M_\mu p> = \lim_{\mu\to 0}(1-\mu)^L = 1$. Observing that M_μ is stochastic then yields statement (3). The determinant $\det(M_\mu)$ is a continuous (polynomial) function in the coefficients of M_μ. Statement (3) shows that $\det(M_\mu) \to \det(\mathbf{1}) = 1$ as $\mu \to 0$. This implies that $\det(M_\mu) \neq 0$ for sufficiently small μ. Consequently, M_μ is invertible (Lang, 1970, p. 108: Thm. 8) for such μ. This shows statement (4). Since M_μ is symmetric, it follows that e is an invariant vector of M_μ. Proposition 1.3.2 now shows the remainder of statement (5). Q.E.D.

Proposition 2.1.2.4 is contained in (Schmitt et al., 1998, Prop. 3.4) or the slightly stronger (Schmitt, 2001, Prop. 3.6.3) which show that M_μ is invertible[4] for $\mu \neq 1/2$.

2.2 SINGLE-CUTPOINT REGULAR CROSSOVER

Crossover models the exchange of genetic information of creatures and is inspired by exchange of genetic information in living organisms, *e.g.,* during the process of sexual reproduction. The crossover operator is treated here only in a very short manner: the definition of regular single-cutpoint crossover is given and some basic consequences are derived. Nevertheless, the analysis presented here shall improve some of the results in (Schmitt et al., 1998, Sec. 2.2, "simple crossover").

Recall that the size $s \geq 4$ of populations is supposed to be an even integer. Regular crossover shall refer to a procedure where the creatures $c_1, \ldots, c_s$ in the population are sequentially paired, and a specific crossover operation is then applied to each of the pairs $(c_1, c_2), \ldots (c_{s-1}, c_s)$ with probability χ. This follows, *e.g.,* Goldberg's approach (Goldberg, 1989, p. 16–17). Single-cutpoint regular crossover has previously been studied in the tensor-string model for populations in (Schmitt et al., 1998, Sec. 2.2), (Schmitt, 2001, Sec. 5.2) and (Schmitt, 2002, Sec. 2.4).

2.2.1. Definition (elementary single-cutpoint crossover). Let $1 \leq \sigma \leq s/2$ and $1 \leq \lambda \leq \ell_o$, $\ell_o \in \{\ell - 1, \ell\}$. λ is called the cutpoint[5]. Let $p = (c_1, \ldots, c_s) \in \wp$ be the current population, $c_{\sigma'} \in \mathcal{C}$, $1 \leq \sigma' \leq s$. Then the elementary single-cutpoint crossover operation $C(\sigma; \lambda)$ is defined by the following three steps: (STEP 1) Pick creatures $c_{2\sigma-1} = (a(\iota_1), \ldots, a(\iota_\ell))$ and $c_{2\sigma} = (a(\iota'_1), \ldots, a(\iota'_\ell))$ from p where $a(\iota_{\lambda'}), a(\iota'_{\lambda'}) \in \mathcal{A}$, $1 \leq \lambda' \leq \ell$. (STEP 2) For $\nu = 1, \ldots, \ell$ do: ((If $\nu \leq \lambda$, then switch letters by setting $\bar{a}_\nu = a(\iota'_\nu)$, and $\bar{a}'_\nu = a(\iota_\nu)$. If $\nu > \lambda$, then copy letters by setting $\bar{a}_\nu = a(\iota_\nu)$, and $\bar{a}'_\nu = a(\iota'_\nu)$.)) (STEP 3) Replace $c_{2\sigma-1}$ by $(\bar{a}_1, \ldots, \bar{a}_\ell)$, and replace $c_{2\sigma}$ by $(\bar{a}'_1, \ldots, \bar{a}'_\ell)$ in p.

---

[4] Based upon the tensor product description of $\mathcal{V}_\wp$ (Schmitt, 2001, Sec. 2.6), one represents M_μ as a corresponding tensor product of "spot mutation matrices" $(1 - \mu)\,\mathbf{1} + \mu\,\mathbf{f}$. Then, one obtains $\mathbf{sp}(M_\mu) = \{(1 - 2\mu)^n : n \in [0, L] \cap \mathbf{N}\}$. $1 - 2\mu$ equals the factor contributed by mutation to the spectrum of the combined mutation-crossover matrix in Koehler's Theorem (Koehler, 1994, p. 419), *i.e.,* the Vose-Liepins conjecture. A tensor-product description of mutation is known in Theoretical Biology. See, *e.g.,* work by Griffiths & Taveré (Griffiths and Taveré, 1997).

[5] The case $\ell_o = \ell$ is considered in (Schmitt et al., 1998, Sec. 2.2, p. 113: footnote) mainly for mathematical convenience. See also the discussion of Geiringer's Theorem in (Schmitt, 2001, Sec. 5.4). Some readers may find the case $\ell_o = \ell - 1$ a more natural setting.

We shall also denote the stochastic matrix associated with the elementary single-cutpoint crossover operation by $C(\sigma;\lambda)$. $C(\sigma;\lambda)$ acts on $\mathcal{V}_{\wp}$ in the sense of line *(12)* and describes transition probabilities for entire populations.

• • •

It is easy to see from Definition 2.2.1 that any two operators $C(\sigma;\lambda)$ and $C(\sigma';\lambda')$ commute. Clearly, one has

$$C(\sigma;\lambda)p = p \text{ for every } p \in \wp \cap \mathcal{U}. \tag{13}$$

$C(\sigma;\lambda)$ is a map $\wp \to \wp$ that is its own inverse. This shows that up to a rearrangement of the basis $\wp$ of $\mathcal{V}_{\wp}$, $C(\sigma;\lambda)$ is a block diagonal matrix consisting of 1 of proper dimension corresponding to the populations invariant under $C(\sigma;\lambda)$, and flip matrices f as defined in equation (2). Hence $C(\sigma;\lambda)$ is symmetric, $C(\sigma;\lambda)^2 = \mathbf{1}$, and $\mathrm{sp}(C(\sigma;\lambda)) = \mathrm{sp}(\mathbf{f}) = \{\pm 1\}$.

$C(\sigma;\lambda)$ commutes with M_{μ} since $C(\sigma;\lambda)$ moves letters around but does not alter them; and it does not matter whether the entire collection of letters in a population is mutated spot-wise before or after being rearranged.

One then randomizes the choice of the cut-cutpoint λ giving every possible value for λ equal probability. This yields the *averaged single-cutpoint crossover operation* $\bar{C}(\sigma)$ which is given by the following "averaged" stochastic matrix:

$$\bar{C}(\sigma) = \ell_o^{-1} \sum_{\lambda=1}^{\ell_o} C(\sigma;\lambda). \tag{14}$$

Simply observe that for $p, q \in \wp$ the expression $\sum_{\lambda=1}^{\ell_o}(\ell_o^{-1} <q, C(\sigma;\lambda)p>)$ equals the combined probability for a transfer $p \mapsto q$ under disjoint events $C(\sigma;\lambda)$ where the specific action determined by λ is chosen with probability ℓ_o^{-1}. $\bar{C}(\sigma)$ is symmetric and commutes with M_{μ} as an $\mathbf{R}$-linear (convex) combination of symmetric matrices $C(\sigma;\lambda)$ that commute with M_{μ}. Any two $\bar{C}(\sigma)$ and $\bar{C}(\sigma')$ commute since all $C(\sigma;\lambda)$ commute, *i.e.,* we have

$$\begin{aligned}\bar{C}(\sigma)\bar{C}(\sigma') &= \ell_o^{-2} \sum_{\lambda,\lambda'=1}^{\ell_o} C(\sigma;\lambda)C(\sigma';\lambda') \\ &= \ell_o^{-2} \sum_{\lambda,\lambda'=1}^{\ell_o} C(\sigma';\lambda')C(\sigma;\lambda) = \bar{C}(\sigma')\bar{C}(\sigma).\end{aligned} \tag{15}$$

Line *(13)* shows that $\bar{C}(\sigma)p = p$ for $p \in \wp \cap \mathcal{U}$.

If one decides with positive probability $\chi \in [0, 1]$ whether or not to apply averaged single-cutpoint crossover to a population, then the stochastic matrix associated with this operation is given by $((1-\chi)\mathbf{1} + \chi\bar{C}(\sigma))$. In fact, the probability to activate transition under $\bar{C}(\sigma)$ equals χ, and $\chi <q, \bar{C}(\sigma)p>$ then is the probability for transition $p \mapsto q$, $p, q \in \wp$. On the other hand $1 - \chi = <p, (1-\chi)\mathbf{1}p>$ is the probability for not invoking $\bar{C}(\sigma)$. These considerations shall be useful in the proof of Proposition 2.2.3.

2.2.2.Definition (single-cutpoint regular crossover C_χ). Let $\chi \in [0,1]$ be the crossover rate. For $\sigma = 1 \ldots s/2$ do the next two steps: (STEP 1) Decide probabilistically whether or not crossover takes place in the current population involving parent creatures $c_{2\sigma-1}$ and $c_{2\sigma}$. The decision for crossover to take place is made positively with probability χ. (STEP 2) If the decision for crossover involving creatures $c_{2\sigma-1}$ and $c_{2\sigma}$ has been made positively in step 1, then execute $\bar{C}(\sigma)$, *i.e.,* chose a random cutpoint λ and switch letters in spots as determined by $C(\sigma;\lambda)$.

Let C_χ also denote the stochastic matrix associated with single-cutpoint regular crossover. C_χ acts on $\mathcal{V}_\wp$ in the sense of line *(12)* and describes transition probabilities for entire populations. •••

The case of a negative decision in step 1 above is referred to as *cloning* in (Vose, 1999b, p. 43). We summarize basic properties[6] of C_χ in the next result.

2.2.3. Proposition. Let M_μ denote multiple-spot mutation as in Definition 2.1.1 with mutation rate $\mu \in (0,1)$. Let C_χ denote single-cutpoint regular crossover as in Definition 2.2.2. Then we have:

1 $C_\chi = \prod_{\sigma=1}^{s/2} ((1-\chi)\mathbf{1} + \chi\bar{C}(\sigma))$.

2 C_χ is a symmetric matrix whose coefficients are polynomials in χ.

3 $C_\chi p = p$ for every $p \in \wp \cap \mathcal{U}$.

4 $C_\chi M_\mu = M_\mu C_\chi$ is symmetric with uniquely determined invariant probability distribution $\boldsymbol{e}$.

PROOF: Taking into account the discussion shortly before Definition 2.2.2, we see that C_χ is defined as the sequential application of stochastic processes/matrices $((1-\chi)\mathbf{1} + \chi\bar{C}(\sigma))$ for $\sigma = 1 \ldots s/2$. As discussed in section 1.3, line *(11),* the action of the whole of C_χ then corresponds to the product of the matrices for the individual steps. Since all $\bar{C}(\sigma))$ commute by line (*15*), the order in the matrix product can be reversed. This shows statement (1). Now, we have:

---

[6]One has $\mathbf{sp}(\bar{C}(\sigma)) \subset \{-1\} \cup [-1+2/\ell_o, 1-2/\ell_o] \cup \{1\}$ using line (*14*), the fact that all $C(\sigma;\lambda)$ commute, (Rudin, 1973, Thm. 11.23), and $\mathbf{sp}(C(\sigma;\lambda)) = \{\pm 1\}$. This, Proposition 2.2.3.1 and (Rudin, 1973, Thm. 11.23) imply for sufficiently small χ an estimate $1 - 2\chi/\ell_o$ for the second largest modulus in $\mathbf{sp}(C_\chi)$. This improves (Schmitt et al., 1998, Prop. 10) and (Schmitt, 2001, Thm. 6.1). The estimate $1-2\chi/\ell_o$ corresponds to (but does not equal) the factor $1-\chi/(\gamma-1)$ contributed by crossover in the third largest eigenvalue obtained in Koehler's Theorem (Koehler, 1994, p. 419), *i.e.,* the Vose-Liepins conjecture. The reason for the factor 2 above and a leading factor 1/2 in Koehler's Theorem will become apparent in section 3.5. See (Schmitt, 2001, Thm. 6.2) for a related result where the spectrum of mutation-crossover (but not the Vose-Liepins version) is computed in the multi-set model as a projection of the mutation-crossover matrix in the tensor-string model.

$$C_\chi = \prod_{\sigma=s/2}^{1}((1-\chi)\mathbf{1} + \chi\bar{C}(\sigma)) = \prod_{\sigma=s/2}^{1}((1-\chi)\mathbf{1} + \chi\bar{C}(\sigma))^*$$

$$= (\prod_{\sigma=1}^{s/2}((1-\chi)\mathbf{1} + \chi\bar{C}(\sigma)))^* = C_\chi^*.$$

Hence, C_χ is symmetric. The fact that C_χ has entries that are polynomials in χ follows directly from statement (1). This shows statement (2). Observe that $p \in \wp \cap \mathcal{U}$ is invariant under every $\bar{C}(\sigma)$ as discussed above in consequence of line *(13)*. This shows statement (3). The discussion of $\bar{C}(\sigma)$ also shows that every $((1-\chi)\mathbf{1} + \chi\bar{C}(\sigma))$ and, consequently, their product commutes with M_μ. Now, statement (2) and Proposition 2.1.2.1 show that $M_\mu C_\chi = C_\chi M_\mu = C_\chi^* M_\mu^* = (M_\mu C_\chi)^*$. This shows that $M_\mu C_\chi$ is symmetric. Thus, e is an invariant vector of $M_\mu C_\chi$. The remainder of statement (4) now follows from Proposition 2.1.2.1 and Corollary 1.3.4. Q.E.D.

Regular single-cutpoint crossover also commutes with population-wise single-spot mutation as discussed in (Schmitt, 2001, Sec. 5.2.1.2) but not with creature-wise single-spot mutation.

There are *other canonical crossover operators* to chose from: (1) (Schmitt et al., 1998, p. 117) discusses *unrestricted crossover* where the positions of creatures to be mated are chosen at random in $\{1, \ldots, s\}$. Unrestricted crossover also commutes with population-wise single/multiple-spot mutation, *cf.* (Schmitt, 2001, Sec. 5.3.1.2). (2) (Schmitt, 2002, Sec. 2.4) discusses *regular multiple-cutpoint* and *uniform crossover* which also commute with the latter two mutation operators. (3) (Schmitt, 2002, Sec. 2.5) discusses *gene-lottery crossover* which does not commute with mutation.

Observe that in every example for crossover discussed here, the given crossover operation may alter every creature in the population. Thus, the population before crossover and the population after crossover may be disjoint, if they are seen as sets of creatures.

2.3 THE FITNESS FUNCTION AND SELECTION

Fitness selection models reproductive success of adapted organisms in their environment and, usually, includes a random rearrangement of the creatures (individuals) in a population. In this work, we shall restrict the analysis to scaled proportional fitness selection based upon a given fitness function $f : D_f \to \mathbb{R}^+$ (consult, *e.g.,* Goldberg's book (Goldberg, 1989, p. 16), (Schmitt et al., 1998, Sec. 2.3), or (Schmitt, 2001, Sec. 7.1)) which is used in standard applications of genetic algorithms to select the creatures in the future population from the creatures in the present population after the crossover-mutation operation. $D_f \subset \mathcal{C} \times \wp$ is the set of all pairs (c, p) such that $c \in p$. Let $J \subset \{1, \ldots, s\}$ have $\#(J) \geq 2$ elements. J will be called the *selector mask.*

Suppose that f is non-trivial in that $\max\{f(c,p) : c \in p \wedge J\} > 0$ for every $p \in \wp$ (to assure that the selection operator is well-defined). In addition, we shall assume that for any two creatures c, d and for any two populations p, q such that $c, d \in p$ and $c, d \in q$, one has:

$$f(c,p) < f(d,p) \quad \Rightarrow \quad f(c,q) < f(d,q). \tag{16}$$

This induces a quasi order $<_f$ on $\mathcal{C}$ (recall that $s \geq 4$). We shall write $c \leq_f d$, if $f(c,p) \leq f(d,p)$, $c, d \in p$. Typical examples for fitness functions satisfying the above are a fitness function whose values $f(c,p) \in \mathbf{R}_*^+$ are independent of the population p, and rank based upon a function $\mathcal{C} \to \mathbf{R}$. The reader may give a suitable definition of rank or consult (Schmitt, 2001, Sec. 7.3). Using a selection method based upon rank induced by a given raw fitness function $\mathcal{C} \to \mathbf{R}$ was proposed by Baker (Baker, 1987). Let

$$\mathcal{C}_{\max} = \{d \in \mathcal{C} : c \leq_f d \text{ for every } c \in \mathcal{C}\} \tag{17}$$

be the set of maximal elements in $\mathcal{C}$ in regard to the quasi order induced by the fitness function. The optimization algorithm is supposed to *maximize* f in the sense of finding an element of $\mathcal{C}_{\max}$. Let

$$\rho_2(f) = \min\{f(c,p)/f(d,p) : p \in \wp,\ c \in (p \wedge J) \cap \mathcal{C}_{\max} \neq \emptyset,\ d \in (p \wedge J) \setminus \mathcal{C}_{\max} \neq \emptyset\} > 1. \tag{18}$$

$\rho_2(f)$ measures the "strength" of second-to-best creatures d in populations p containing elements $c \in \mathcal{C}_{\max}$ where both creatures c and d are sitting on components/spots of p corresponding to J. $\rho_2(f)$ is easy to determine, if the fitness function f is given by rank. We shall suppose that f is non-trivial in that $\rho_2(f) < \infty$.

Next, we define *power-law scaling* of the fitness function in accordance with, *e.g.,* (Goldberg, 1989, p. 124), (Schmitt et al., 1998, Sec. 2.3), (Schmitt, 2001, Sec. 7.1), (Suzuki, 1997, p. 65), (Suzuki, 1998, p. 100). In fact, we set for $(c,p) \in D_f$, $t \in \mathbf{N}$, $B \in \mathbf{R}_*^+$:

$$f_t(c,p) = (f(c,p))^{g(t)} \quad \text{with} \quad g(t) = B \cdot \log(t+1) \tag{19}$$

In addition, define $f_t(c,p) = 0$, if $(c,p) \in \mathcal{C} \times \wp \setminus D_f$. In this exposition, we shall only consider *logarithmic scalings* $g(t)$ as listed above which are unbounded. It has been shown in (Schmitt, 2001, Thm. 8.5), that faster scalings with, *e.g.,* linear growth $g(t) = at + b$ in the exponent are of limited value, in particular, in regard to the use of a crossover operation. In fact, such algorithms are asymptotically equivalent to a "take-the-best" algorithm (Schmitt, 2001, Def. 8.4) where one cycle of the algorithm consists of the mutation-step and picking maximal creatures in the current population. Finally, scaled proportional fitness selection is defined as follows:

2.3.1. Definition (scaled proportional fitness selection S_t). Let $J \subset \{1, \ldots, s\}$ be the selector mask as defined above. Let $t \in \mathbf{N}$ and f_t be as in line (*19*). Suppose that $p = (c_1, c_2, \ldots, c_s) \in \wp$ is the current population with $c_\sigma \in \mathcal{C}$, $1 \leq \sigma \leq s$. For $c \in \mathcal{C}$ let $n(c, p \wedge J)$ denote the number of copies of c in $p \wedge J$. Now, the new population $q = (d_1, d_2, \ldots, d_s)$ is assembled as follows: for $\sigma = 1, \ldots, s$ execute the following step: ((Select creature $d_\sigma \in q$ probabilisticly among the creatures in $p \wedge J$ such that a particular $c \in p \wedge J$ has relative probability $(\sum_{\sigma' \in J} f_t(c_{\sigma'}, p))^{-1} \cdot n(c, p \wedge J)\, f_t(c, p)$ for being selected as d_σ.))

Let S_t also denote the stochastic matrix associated with scaled proportional fitness selection. S_t acts on $\mathcal{V}_\wp$ in the sense of line (*12*) and describes transition probabilities for entire populations. • • •

Definition 2.3.1 generalizes (Schmitt, 2001, Sec. 7.1, eq. (23)). The following Proposition collects basic properties of the scaled fitness selection operator S_t.

2.3.2. Proposition. Let $\theta = 1 - \#(J)^{-s}$ where $\#(J) \geq 2$ is the size of the selector mask $J \subset \{1, \ldots, s\}$. Let S_t denote scaled proportional fitness selection as in Definition 2.3.1 with scaling $g(t)$ as in line (*19*). Suppose that $p = (c_1, c_2, \ldots, c_s), q = (d_1, d_2, \ldots, d_s) \in \wp$, $c_\sigma, d_\sigma \in \mathcal{C}$, $1 \leq \sigma \leq s$. Then we have

1 $<q, S_t p> = (\sum_{\sigma' \in J} f_t(c_{\sigma'}, p))^{-s} \cdot \prod_{\sigma=1}^{s} n(d_\sigma, p \wedge J)\, f_t(d_\sigma, p)$.

2 If $p \in \wp \cap \mathcal{U}$, then $S_t p = p$.

3 $||P_{\mathcal{U}} S_t p||_1 \geq 1 - \theta$.

4 If $v \in \mathcal{S}_\wp$, then $||(\mathbf{1} - P_{\mathcal{U}}) S_t v||_1 \leq \theta \cdot ||(\mathbf{1} - P_{\mathcal{U}}) v||_1$.

PROOF: If $<q, S_t p> \neq 0$, then one must have $\mathrm{set}(q) \subset \mathrm{set}(p \wedge J)$. In this situation, $(\sum_{\sigma' \in J} f_t(c_{\sigma'}, p))^{-1} \cdot n(d_\sigma, p \wedge J)\, f_t(d_\sigma, p)$ is the proportional strength and selection probability for $d_\sigma \in q$ among the $c_{\sigma'} \in p \wedge J$. These probabilities for the independent selection-events $\sigma = 1, \ldots, s$ are then multiplied together. If $<q, S_t p> = 0$, then one of the products $n(d_\sigma, p \wedge J))\, f_t(d_\sigma, p)$ must equal 0. This shows statement (1). If $p \in \wp \cap \mathcal{U}$, then only one creature is available for selection from $p \wedge J$ which shall reproduce p again. This shows statement (2). If a particular $d \in \mathrm{set}(p \wedge J)$ has maximal fitness-value in this set, then the probability to select d in one step of the for-loop in Definition 2.3.1 is greater or equal $\#(J)^{-1}$. Hence, the probability to generate the uniform population $(d, d, \ldots, d)$ is greater or equal $1 - \theta$. Consequently, $||P_{\mathcal{U}} S_t p||_1 \geq 1 - \theta$. This shows statement (3). Let us finally show statement (4). Using Proposition 1.4.1, statements (2) and (3), we obtain for $v \in \mathcal{S}_\wp$:

$$||(1 - P_{\mathcal{U}})S_t v||_1 = ||(1 - P_{\mathcal{U}})S_t(1 - P_{\mathcal{U}})v||_1$$
$$= \sum_{p,q \in \mathcal{P} \setminus \mathcal{U}} <q, S_t p> <p, v> = \sum_{p \in \mathcal{P} \setminus \mathcal{U}} (1 - ||P_{\mathcal{U}} S_t p||_1) <p, v>$$
$$\leq \theta \cdot ||(1 - P_{\mathcal{U}})v||_1. \qquad \text{Q.E.D.}$$

3. CONVERGENCE OF SCALED GENETIC ALGORITHMS TO GLOBAL OPTIMA

3.1 THE DRIVE TOWARDS UNIFORM POPULATIONS

In this section, we shall investigate the convergence of a scaled genetic algorithm towards uniform populations. This is a result of the tendency of the selection operator S_t to produce uniform populations (*cf.* Proposition 2.3.2.3) and the fact that the mutation rate μ is scheduled to converge to zero. Results related to the present discussion have previously been obtained in (Schmitt et al., 1998, Thm. 15.4–5) (Schmitt, 2001, Thm. 8.1.3, Thm. 8.2.3–4) and (Schmitt, 2002, Thm. 3.1.1.3).

The results (Schmitt et al., 1998, Thm. 17), (Schmitt, 2001, Thm. 8.1, Thm. 8.2), and quite drastically (Schmitt, 2001, Thm. 8.3) show that a genetic algorithm with strictly positive mutation rate limit cannot asymptotically converge to a probability distribution over populations containing only globally optimal creatures. This includes the case of the simple genetic algorithm. Consequently, in order to obtain asymptotic convergence to global optima, the mutation rate has to be annealed to zero. Theorem 3.1.2 shows that in this situation the algorithm must converge to a probability distribution over uniform populations only. Thus, even though the goal of an optimization algorithm should be to find just one copy of an optimal creature, the fabric of the algorithm will asymptotically deliver a uniform population containing optimal creatures. We point out that a properly designed, scaled, asymptotically converging genetic algorithm as in the Global Optimization Theorem 3.4.1 allows for probabilistic estimates in regard to running the algorithm only a finite but larger number of cycles and approaching the limit probability distribution over uniform populations containing globally optimal creatures, *cf.* the discussion after the proof of Theorem 3.3.2 and (Isaacson and Madsen, 1961, p. 160: proof of Thm. V.4.3).

The next result simplifies and generalizes the approach taken in (Schmitt et al., 1998, Prop. 2.4, Prop. 4.4), (Schmitt, 2001, Prop. 3.4.4, Prop. 3.7.4) and (Schmitt, 2002, Prop. 2.2.3) by showing a mutation flow inequality for the combined crossover-mutation operator without referring much to the particular action of mutation.

3.1.1. Proposition (mutation flow inequality). Let M_μ denote multiple-spot mutation as in Definition 2.1.1 with mutation rate $\mu \in (0, 1)$.

Let C_χ denote single-cutpoint regular crossover as in Definition 2.2.2. Then we have:

1 $\beta_\mu = \beta(M_\mu) = \min\{||P_{\mathcal{U}} M_\mu p||_1 : p \in \wp \cap \mathcal{U}\} \in (0,1)$, and $\lim_{\mu \to 0} \beta_\mu = 1$.

2 $||(\mathbf{1} - P_{\mathcal{U}}) C_\chi M_\mu v||_1 \leq 1 - \beta_\mu + \beta_\mu ||(\mathbf{1} - P_{\mathcal{U}}) v||_1$, for every $v \in \mathcal{S}_\wp$.

PROOF: If $p \in \wp \cap \mathcal{U}$, then changing a single spot in the genome under multiple-spot mutation makes p non-uniform. By Proposition 2.1.2.1 the probability for this to occur is strictly positive as well as the probability for retaining p under multiple-spot mutation. Hence, the combined probability for obtaining a uniform population from p under multiple-spot mutation, *i.e.*, $||P_{\mathcal{U}} M_\mu p||_1$ is an element of (0, 1). If $\mu \to 0$, then by Proposition 2.1.2.3 we have: $1 \geq ||P_{\mathcal{U}} M_\mu p||_1 \geq \langle p, M_\mu p \rangle \to 1$. Taking into account that $\wp \cap \mathcal{U}$ is finite shows that $\beta_\mu \in (0,1)$, and $\lim_{\mu \to 0} \beta_\mu = 1$. This shows statement (1).

To show statement (2), observe first that $||\mathbf{1} - P_{\mathcal{U}}||_1 = ||C_\chi||_1 = 1$ applying lines (*8*) and (*10*). Using Proposition 2.2.3.3 and Proposition 1.4.1, we obtain:

$$
\begin{aligned}
&||(\mathbf{1} - P_{\mathcal{U}}) C_\chi M_\mu v||_1 = ||(\mathbf{1} - P_{\mathcal{U}}) C_\chi (\mathbf{1} - P_{\mathcal{U}}) M_\mu v||_1 \\
&\leq ||(\mathbf{1} - P_{\mathcal{U}}) M_\mu v||_1 = \textstyle\sum_{q \in \wp \setminus \mathcal{U}} \sum_{p \in \wp} \langle q, M_\mu p \rangle \langle p, v \rangle \\
&= \textstyle\sum_{p \in \wp \setminus \mathcal{U}} ||(\mathbf{1} - P_{\mathcal{U}}) M p||_1 \langle p, v \rangle + \sum_{p \in \wp \cap \mathcal{U}} (1 - ||P_{\mathcal{U}} M_\mu p||_1) \langle p, v \rangle \\
&\leq \textstyle\sum_{p \in \wp \setminus \mathcal{U}} \langle p, v \rangle + \sum_{p \in \wp \cap \mathcal{U}} (1 - \beta_\mu) \langle p, v \rangle \\
&= ||(\mathbf{1} - P_{\mathcal{U}}) v||_1 + (1 - \beta_\mu)(1 - ||(\mathbf{1} - P_{\mathcal{U}}) v||_1) = 1 - \beta_\mu + \beta_\mu ||(\mathbf{1} - P_{\mathcal{U}}) v||_1.
\end{aligned}
$$

Q.E.D.

The constant β_μ in Proposition 3.1.1 has been explicitly computed in (Schmitt et al., 1998, Prop. 4.4) and the more general (Schmitt, 2001, Prop. 3.7.4). The mutation flow inequality shows how the mutation operation controls the balance between uniform and non-uniform populations in a genetic algorithm. If the mutation flow inequality is combined in a proper way with the contraction of the selection operator towards uniform populations established in Proposition 2.3.2.4, then this ensures that the combined probability over non-uniform populations in the steady-state distribution of a simple genetic algorithm becomes small for small mutation rates. This fact is shown in statement (3) of the next Theorem taking into account Proposition 3.1.1.1.

3.1.2. Theorem. Let M_μ denote multiple-spot mutation as in Definition 2.1.1 with mutation rate $\mu \in (0,1)$. Suppose that $\beta_\mu \in (0,1)$ is given as in Proposition 3.1.1.1. Let C_χ denote single-cutpoint regular crossover as in Definition 2.2.2. Let $\theta = 1 - \#(J)^{-s}$ where $\#(J) \geq 2$ is the size of the selector mask $J \subset \{1, \ldots, s\}$. Let S_t denote scaled proportional fitness selec-

tion as in Definition 2.3.1 with scaling $g(t)$ as in line (*19*). Then we have for $v \in \mathcal{S}_\wp$:

1 $||(\mathbf{1} - P_\mathcal{U})S_t C_\chi M_\mu v||_1 \leq \theta \cdot (1 - \beta_\mu + \beta_\mu ||(\mathbf{1} - P_\mathcal{U})v||_1)$.

2 $||(\mathbf{1} - P_\mathcal{U})(S_t C_\chi M_\mu)^k v||_1$
$\leq \theta(1 - \beta_\mu)/(1 - \theta\beta_\mu) + (\theta\beta_\mu)^k ||(\mathbf{1} - P_\mathcal{U})v||_1$ for $k \in \mathbf{N}$.

3 If v is an invariant vector of $S_t C_\chi M_\mu$, then
$||(\mathbf{1} - P_\mathcal{U})v||_1 \leq \theta(1 - \beta_\mu)/(1 - \theta\beta_\mu)$.

PROOF: Using Proposition 2.3.2.2, Proposition 1.4.1, Proposition 2.3.2.4 and Proposition 3.1.1.2, one has

$$||(\mathbf{1} - P_\mathcal{U})S_t C_\chi M_\mu v||_1 = ||(\mathbf{1} - P_\mathcal{U})S_t(\mathbf{1} - P_\mathcal{U})C_\chi M_\mu v||_1$$
$$\leq \theta \cdot ||(\mathbf{1} - P_\mathcal{U})C_\chi M_\mu v||_1 \leq \theta \cdot (1 - \beta_\mu + \beta_\mu ||(\mathbf{1} - P_\mathcal{U})v||_1).$$

This shows statement (1). Statement (1) shows statement (2) for $k = 1$ since $1/(1 - \beta\theta) \geq 1$. To complete the proof of statement (2), we proceed by induction:

$$||(\mathbf{1} - P_\mathcal{U})(S_t C_\chi M_\mu)^{k+1} v||_1 \leq \theta(1 - \beta_\mu + \beta_\mu ||(\mathbf{1} - P_\mathcal{U})(S_t C_\chi M_\mu)^k v||_1)$$
$$\leq \theta \cdot (1 - \beta_\mu + \beta_\mu(\theta(1 - \beta_\mu)/(1 - \theta\beta_\mu) + (\theta\beta_\mu)^k ||(\mathbf{1} - P_\mathcal{U})v||_1))$$
$$= \theta(1 - \beta_\mu)/(1 - \theta\beta_\mu) + (\theta\beta_\mu)^{k+1} ||(\mathbf{1} - P_\mathcal{U})v||_1.$$

Statement (3) is now obtained as follows:

$$||(\mathbf{1} - P_\mathcal{U})v||_1 = \lim_{k \to \infty} ||(\mathbf{1} - P_\mathcal{U})(S_t C_\chi M_\mu)^k v||_1$$
$$\leq \lim_{k \to \infty}(\theta(1 - \beta_\mu)/(1 - \theta\beta_\mu) + (\theta\beta_\mu)^k ||(\mathbf{1} - P_\mathcal{U})v||_1)$$
$$= \theta(1 - \beta_\mu)/(1 - \theta\beta_\mu).$$ Q.E.D.

3.2 WEAK ERGODICITY

From now on, we shall set $G_t = S_t C_{\chi(t)} M_{\mu(t)}$. In accordance with the discussion in section 1.3, we see that G_t describes the probabilistic passage in one cycle **mutation→crossover→selection** of the genetic algorithm. For $t > t' \in \mathbf{N}$, define $H_t^{t'} = \prod_{\tau=t}^{t'} G_\tau$. Let $v_o, w_o \in \mathcal{S}_\wp$. Weak ergodicity of the inhomogeneous Markov chain $(G_t)_{t \in \mathbf{N}}$ as discussed in the books by Isaacson & Madsen (Isaacson and Madsen, 1961, p. 142–151, p. 151: Thm. V.3.2) or Seneta (Seneta, 1981, p. 85–86, p. 134–142, p. 137: Thm. 4.8, p. 141: Thm. 4.9) refers to the phenomenon that the trajectories $H_t^{t'} v_o$ and $H_t^{t'} w_o$ of sequential application of the G_t to initial probability distributions v_o and w_o become arbitrary close as $t \to \infty$ for any initial offset $t' \in \mathbf{N}$. However, weak ergodicity does not imply that $\lim_{t \to \infty} H_t^{t'} v_o$ exists.

If we set $\mu(t) = (t+1)^{-1/L}/2$, $t \in \mathbf{N}$, then Proposition 2.1.2.2 shows that M_μ has entries that are bounded below by $\epsilon_t = 2^{-2L}(t+1)^{-1}$. Proposition 1.3.1.2 applied with $X_t = S_t C_{\chi(t)}$ then shows that $\lim_{t\to\infty} H_t^{t'}(v_o - w_o) = 0$, *i.e.,* weak ergodicity.

Note that $\lim_{t\to\infty} H_t^{t'}(A - B) = 0$ where A and B are stochastic matrices acting on $\mathcal{V}_\wp$ since each of the columns of $A - B$ is a difference of elements of $\mathcal{S}_\wp$.

3.3 STRONG ERGODICITY

Let G_t, $H_t^{t'}$, and v_o, w_o be as in section 3.2. Strong ergodicity as discussed in (Isaacson and Madsen, 1961, p. 157: Sec. V.4) or (Seneta, 1981, p. 92: Sec. 3.3, p. 149: Sec. 4.5) refers to the existence of $v_\infty = \lim_{t\to\infty} H_t^{t'} v_o$ for any initial offset $t' \in \mathbf{N}$. Using that $\lim_{t\to\infty} H_t^{t'}(v_o - w_o) = 0$, we see that $v_\infty = \lim_{t\to\infty} H_t^{t'} w_o$, *i.e.,* the asymptotic probabilistic outcome (limit) of applying the $(G_t)_{t\in\mathbf{N}}$ sequentially to v_o is independent from the initial probability distribution v_o or distribution sequence. This holds, in particular, if $p = v_o \in \wp$, *i.e.,* the algorithm starts at a user-selected population.

To establish strong ergodicity of $G_t = S_t C_{\chi(t)} M_{\mu(t)}$, we need the following technical but rather powerful result:

3.3.1. Proposition. Let M_μ denote multiple-spot mutation as in Definition 2.1.1 with mutation rate $\mu = (t+1)^{-1/L}/2$, $t \in \mathbf{N}$. Let C_χ denote single-cutpoint regular crossover as in Definition 2.2.2 with crossover rate $\chi(t) = \mu(t)^{1/m}$, $m \in \mathbf{N}$ fixed. Let S_t denote scaled proportional fitness selection as in Definition 2.3.1 with scaling $g(t)$ as in line (*19*). Let $G_t = S_t C_{\chi(t)} M_{\mu(t)}$. By Corollary 1.3.3, G_t has an invariant probability distribution $v_t \in \mathcal{S}_\wp$. If t is sufficiently large enough, then v_t is uniquely determined. In addition,

1. $\sum_{t=1}^{\infty} ||v_{t+1} - v_t||_1 < \infty$ 2. $\lim_{t\to\infty} v_t = v_\infty$ exists.

The **proof** of Proposition 3.3.1 which is listed in section 5 runs along the lines of the following idea: The coefficients of the v_t are essentially something like rational functions in $\mu = (t+1)^{-1/L}/2$ which must take values in [0, 1] since $v_t \in \mathcal{S}_\wp$. Consequently, these functions cannot have a pole at $\mu = 0$ and consequently must even be (almost) differentiable in $\mu = 0$. This implies monotone behavior and summability.

3.3.2. Theorem. Let M_μ denote multiple-spot mutation as in Definition 2.1.1 with mutation rate $\mu = (t+1)^{-1/L}/2$, $t \in \mathbf{N}$. Let C_χ denote single-cutpoint regular crossover as in Definition 2.2.2 with crossover rate $\chi(t) =$

$\mu(t)^{1/m}$, $m \in \mathbf{N}$ fixed. Let S_t denote scaled proportional fitness selection as in Definition 2.3.1 with scaling $g(t)$ as in line (*19*). Let $G_t = S_t C_{\chi(t)} M_{\mu(t)}$.

1 Then the inhomogeneous Markov chain $(G_t)_{t\in\mathbf{N}}$ is strongly ergodic.

2 If $v_o \in \mathcal{S}_\wp$ is arbitrary and $v_\infty \in \mathcal{S}_\wp$ is as in Proposition 3.3.1, then $\lim_{t\to\infty} \prod_{\tau=t}^{1} G_\tau v_o = v_\infty$.

PROOF: Let $v_t, v_\infty \in \mathcal{S}_\wp$ be as in Proposition 3.3.1. Let $t_o \in \mathbf{N}$ be fixed. For $t \geq t' \in \mathbf{N}$, let $H_t^{t'} = \prod_{\tau=t}^{t'} G_\tau$. Using the remark in the last paragraph of section 3.2, we know that $\lim_{t\to\infty} ||H_t^{t'}(H_{t'}^{t_o} - P(v_{t'+1}))|| = 0$. In addition, $H_{t'}^{t'} P(v_{t'}) = G_{t'} P(v_{t'}) = P(v_{t'})$ since $v_{t'}$ is an invariant vector of $G_{t'}$. Now, using the triangle inequality repeatedly, using $H_t^{t_o} = H_t^{t'} H_{t'}^{t_o}$, lines (*9*) and (*10*), we have:

$$
\begin{aligned}
&\overline{\lim}_{t\to\infty} ||H_t^{t_o} - P(v_\infty)||_1 = \lim_{t'\to\infty} \overline{\lim}_{t\to\infty} ||H_t^{t_o} - P(v_\infty)||_1 \\
&\leq \lim_{t'\to\infty} (\lim_{t\to\infty} ||H_t^{t'}(H_{t'}^{t_o} - P(v_{t'}))|| + \\
&\qquad + \overline{\lim}_{t\to\infty} ||H_t^{t'} P(v_{t'}) - P(v_{t'})||_1 + ||P(v_{t'}) - P(v_\infty)||_1) \\
&\leq \lim_{t'\to\infty} \overline{\lim}_{t\to\infty} (||H_t^{t'+1}(P(v_{t'}) - P(v_{t'+1}))||_1 + \\
&\qquad + ||H_t^{t'+1} P(v_{t'+1}) - P(v_{t'+1})||_1 + ||P(v_{t'+1}) - P(v_{t'})||_1) \\
&\leq \lim_{t'\to\infty} \overline{\lim}_{t\to\infty} (||H_t^{t'+1}||_1 \cdot ||P(v_{t'+1}) - P(v_{t'})||_1 + \\
&\qquad + ||H_t^{t'+1} P(v_{t'+1}) - P(v_{t'+1})||_1 + ||v_{t'+1} - v_{t'}||_1) \\
&\leq \lim_{t'\to\infty} \overline{\lim}_{t\to\infty} (2||v_{t'+1} - v_{t'}||_1 + 2||v_{t'+2} - v_{t'+1}||_1 + \\
&\qquad + ||H_t^{t'+2} P(v_{t'+2}) - P(v_{t'+2})||_1) \\
&\leq \lim_{t'\to\infty} \overline{\lim}_{t\to\infty} 2 \sum_{\tau=t'}^{t-1} ||v_{\tau+1} - v_\tau||_1 \\
&= \lim_{t'\to\infty} 2 \sum_{\tau=t'}^{\infty} ||v_{\tau+1} - v_\tau||_1 = 0 \qquad (20)
\end{aligned}
$$

The last conclusion in the line (*20*) uses Proposition 3.3.1.1. Now, we have $0 \leq \underline{\lim}_{t\to\infty} ||H_t^{t_o} - P(v_\infty)||_1 \leq \overline{\lim}_{t\to\infty} ||H_t^{t_o} - P(v_\infty)||_1 = 0$ which shows the existence of $\lim_{t\to\infty} ||H_t^{t_o} - P(v_\infty)||_1 = 0$. Thus, $\lim_{t\to\infty} ||H_t^{t_o} w - v_\infty||_1 = \lim_{t\to\infty} ||(H_t^{t_o} - P(v_\infty))w||_1 \leq \lim_{t\to\infty} ||(H_t^{t_o} - P(v_\infty))||_1 \cdot ||w||_1 = 0$ for every $w \in \mathcal{S}_\wp$ and $t_o \in \mathbf{N}$. This completes the proof of both statements of the Theorem. Q.E.D.

Theorem 3.3.2 is a short, simplified variant of (Isaacson and Madsen, 1961, p. 160: Thm. V.4.3). It is included here for the sake of completeness and the convenience of the reader. The proof of Theorem 3.3.2 illustrates the asymptotic behavior of ergodic, scaled genetic algorithms. The reader should realize in which manner the interplay of weak ergodicity and the convergence behav-

ior of the v_t, in particular, in regard to the summability of $(||v_{t+1} - v_t||_1)_{t\in\mathbf{N}}$ or coordinate-wise monotone behavior of $(v_t)_{t\in\mathbf{N}}$ contributes to the asymptotic behavior of the ergodic, scaled genetic algorithm established above. In principle, one can say that "*The genetic algorithm follows the trajectory of the steady-state distributions of the individual steps of the algorithm.*" since the quantity $||H_t^{t_0} - P(v_t)||$ becomes small for large t as established above. Using the explicit form of the mutation and crossover schedules, Proposition 1.3.1, the inequalities in the above proof and the steady-state flow inequality established in the proof of the Global Optimization Theorem 3.4.1 (line *26*), then allow to develop stopping criteria similar to results by Aytug & Koehler (Aytug and Koehler, 1996).

3.4 CONVERGENCE TO GLOBAL OPTIMA.

The following Theorem 3.4.1 is the main result of this exposition. It shows that a carefully scaled genetic algorithm converges *for arbitrary fitness function* to a probability distribution over uniform populations containing only elements of $\mathcal{C}_{\max}$ (see definition (*17*)). Thus in the case of a binary alphabet, (Schmitt, 2001, Thm. 8.6, Rem. 8.7) are simplified and strengthened considerably in regard to applicability and implementation. However, there is a price to pay in that we require the crossover-rate being annealed to 0 for the algorithm described below. Such a condition is not needed in (Schmitt, 2001, Thm. 8.6, Rem. 8.7). Note that our analysis in Theorem 3.4.1 superseeds the approach taken in Vose's book (Vose, 1999b) where it is always assumed that the fitness function is *injective* (Vose, 1999b, p. 25: footnote).

3.4.1. Theorem (Global Optimization). Let M_μ denote multiple-spot mutation as in Definition 2.1.1 with mutation rate $\mu = (t+1)^{-1/L}/2$, $t \in \mathbf{N}$. Let C_χ denote single-cutpoint regular crossover as in Definition 2.2.2 with crossover rate $\chi(t) = \mu(t)^{1/m}$, $m \in \mathbf{N}$ fixed. Let the selector mask satisfy either $J = [0, s] \cap (2\mathbf{N})$ or $J = [0, s] \cap \mathbf{N}$. Suppose $s > 2m\ell$. Let $\rho_2(f) < \infty$ be given by definition (*18*). Let S_t denote scaled proportional fitness selection as in Definition 2.3.1 with scaling $g(t) = B\cdot\log(t+1)$, $B \in \mathbf{R}_*^+$. Suppose that B satisfies: $\ell < LB\log(\rho_2(f)) + 1/m$. Let $G_t = S_t C_{\chi(t)} M_{\mu(t)}$. Then we have:

1 The inhomogeneous Markov chain $(G_t)_{t\in\mathbf{N}}$ describing the scaled genetic algorithm is strongly ergodic, *cf.* Theorem 3.3.2.1.

2 Let $v_t = G_t v_t \in \mathcal{S}_\wp$ denote a steady-state distribution of an individual steps G_t of the scaled genetic algorithm. For sufficiently large $t \in \mathbf{N}$, v_t is uniquely determined up to scalar multiples as invariant right eigenvector of G_t, *cf.* Proposition 3.3.1. Let $v_\infty = \lim_{t\to\infty} v_t$, *cf.* Proposition

3.3.1.2. Then v_∞ is strictly positive only over uniform populations generated by creatures in $\mathcal{C}_{\max}$.

3 Let $v_o \in \mathcal{S}_\wp$ be the probability distribution for the selection of the initial population. Then, $\prod_{\tau=t}^{1} G_\tau v_o \in \mathcal{S}_\wp$ describes the state of the algorithm after step t and we have: $\lim_{t\to\infty} \prod_{\tau=t}^{1} G_\tau v_o = v_\infty$, *cf.* Theorem 3.3.2.2. Consequently, the states of the scaled genetic algorithm converge to (a probability distribution over) uniform populations generated by globally optimal creatures.

PROOF: We only have to establish that "v_∞ is strictly positive only over uniform populations generated by creatures in $\mathcal{C}_{\max}$." All other claims in Theorem 3.4.1 have already been established in previous results.

Part 1: Convergence towards uniform populations. Let $\theta = 1 - \#(J)^{-s}$. Theorem 3.1.2.3 implies:

$$||(\mathbf{1} - P_{\mathcal{U}})v_\infty||_1 = \lim_{t\to\infty} ||(\mathbf{1} - P_{\mathcal{U}})v_t||_1$$
$$\leq \lim_{t\to\infty} (1 - \beta(\mu(t)))\theta/(1-\theta) = 0$$

where $\beta(\mu(t))$ is given as in Proposition 3.1.1.1. This shows that v_∞ is nonzero only over uniform populations.

Part 2: Convergence towards populations containing maximal creatures. To complete the proof, we show that v_∞ is strictly positive only over populations in $\Omega = \mathcal{C}_{\max}^s \subset \wp$, *i.e.,* populations that contain only globally optimal creatures. The idea for the following argument is to derive an estimate for the probabilistic flow between Ω and $\Omega' = \wp \setminus \mathcal{C}_{\max}^s$, if the homogeneous Markov chain defined by G_t is in steady state. This is based upon the fact that $v_t = G_t v_t$. Let P_Ω be the orthogonal projection onto $\mathrm{span}_{\mathbf{C}}(\Omega)$, and let $P_{\Omega'}$ be the orthogonal projection onto $\mathrm{span}_{\mathbf{C}}(\Omega')$. Let $\omega(t) = ||P_\Omega v_t||_1 \to ||P_\Omega v_\infty||_1$ as $t \to \infty$. Let $\Omega^+ = \{p \in \wp : \mathrm{set}(p \wedge J) \cap \mathcal{C}_{\max} \neq \emptyset\}$.

Part 2a: The flow towards Ω. In order to make a transition under mutation from $q' \in \Omega'$ to a population $q^+ \in \Omega_+$, one has to change at most the letters in the spots corresponding to a single creature in q'. If $t \to \infty$ and, consequently, $\chi(t) \to 0$, then Proposition 2.2.3.1 shows that $C_{\chi(t)} \to \mathbf{1}$. Thus, for t sufficiently large $<p, C_{\chi(t)}p> \geq 1/2$ for every $p \in \mathcal{S}_\wp$. A transition from $q^+ \in \Omega_+$ to a population $p \in \Omega \cap \mathcal{U}$ under fitness selection occurs with probability bounded below by $\#(J)^{-s}$, *cf.,* Proposition 2.3.2.3. Hence, we have for $K_1 \in \mathbf{R}_*^+$ and summations over $p \in \Omega \cap \mathcal{U}$, $q^+ \in \Omega^+$, $q' \in \Omega'$:

$$||P_\Omega G_t P_{\Omega'} v_t||_1 \geq \sum_{p,q'} <p, S_t C_{\chi(t)} M_{\mu(t)} q'> <q', v_t>$$
$$\geq \sum_{p,q^+,q'} <p, S_t q^+> <q^+, C_{\chi(t)} q^+> <q^+, M_{\mu(t)} q'> <q', v_t>$$
$$\geq \sum_{q^+,q'} \#(J)^{-s}/2 <q^+, M_{\mu(t)} q'> <q', v_t>$$

$$\geq (\#(J)^{-s}/2)\mu^{\ell} \sum_{q'} <q', v_t> \geq K_1 \mu^{\ell} \cdot ||P_{\Omega'} v_t||_1. \tag{21}$$

Hence, we have for t such that $P_{\Omega'} v_t \neq 0$:

$$||P_{\Omega'} G_t P_{\Omega'} v_t||_1 = ||P_{\Omega'} v_t||_1 \cdot ||P_{\Omega'} G_t(||P_{\Omega'} v_t||_1^{-1}) P_{\Omega'} v_t||_1 \tag{22}$$

$$= ||P_{\Omega'} v_t||_1 \cdot (1 - ||P_{\Omega'} v_t||_1^{-1} \cdot ||P_{\Omega} G_t P_{\Omega'} w_t||_1) \leq ||P_{\Omega'} v_t||_1 \cdot (1 - K_1 \mu^{\ell}).$$

Inequality (22) is trivial, if $P_{\Omega'} v_t = 0$. Let $q^+ = (c_1, c_2, \ldots, c_s) \in \Omega^+$, $c_\sigma \in \mathcal{C}$, $1 \leq \sigma \leq s$. Let $\nu_{\max} \in [1, \#(J)] \cap \mathbf{N}$ denote the number of components in $p \wedge J$ that are occupied by elements in $\mathcal{C}_{\max}$. The probability for selecting an arbitrary element $d \in \mathrm{set}(q^+ \wedge J) \cap \mathcal{C}_{\max}$ in the process of the scaled proportional fitness selection operation is then given by

$$\nu_{\max} f_t(d, q^+) / (\textstyle\sum_{\sigma \in J} f_t(c_\sigma, q^+)). \tag{23}$$

The expression in line (23) is bounded below by $(1 + (\#(J) - 1)(t+1)^{-B \log(\rho_2(f))})^{-1}$. Hence we have:

$$||P_{\Omega} S_t q^+||_1 \geq (1 + (\#(J) - 1)(2\mu)^{LB \log(\rho_2(f))})^{-s} =: x(\mu).$$

Hence, there exists $K_2 \in \mathbf{R}_*^+$ such that for sufficiently small μ:

$$||P_{\Omega'} S_t q^+||_1 = 1 - ||P_{\Omega} S_t q^+||_1 \leq 1 - x(\mu) \leq K_2 \mu^{LB \log(\rho_2(f))}. \tag{24}$$

Part 2b: The flow towards Ω'. In order to estimate the probabilistic flow from Ω to Ω' in application of G_t to w_t, we distinguish two cases:

CASE 1: *Initial mutation-crossover step destroys all globally optimal creatures with positions in J.* In order to make a transition from $p \in \Omega$ to a population $q^c \in \wp \setminus \Omega^+$ via mutation-crossover, one has to change every creature with position in J to a creature in $\mathcal{C} \setminus \mathcal{C}_{\max}$. In that case, a subsequent selection operation cannot generate an element of Ω^+. Since crossover and mutation commute by Proposition 2.2.3.4, we may assume that crossover is applied first. Then the crossover operation alone may achieve changing every creature with position in J to a creature in $\mathcal{C} \setminus \mathcal{C}_{\max}$. By Definition 2.2.2 or Proposition 2.2.3.1 and the choices for J, the combined probability for this to happen is bounded from above for small μ by $K_3 \chi(t)^{s/2} = K_3 \mu(t)^{s/(2m)}$, $K_3 \in \mathbf{R}_*^+$. Mutation may then keep the resulting $q^c \in \wp \setminus \Omega^+$. Let $\sigma \in [0, \#(J)] \cap \mathbf{N}$. Suppose that crossover only changed σ creatures with position in J in $p \in \Omega$. If $J = [0, s] \cap (2\mathbf{N})$, then the probability for this to happen is bounded from above by terms in the order of $o(t) = \mu(t)^{\sigma/m}$. If $J = [0, s] \cap \mathbf{N}$, then the probability for this to happen is bounded from above by terms in the order of $o(t) = \mu(t)^{\sigma/(2m)}$. Then mutation has to alter at least one spot in the unchanged $\#(J) - \sigma$ creatures in p with position in J. By Proposition 2.1.2.1, the combined probability for the latter to happen is bounded by $K_4 \mu^{\#(J) - \sigma}$, $K_4 \in \mathbf{R}_*^+$ which is then multiplied by terms in the order of $o(t)$ to obtain an upper bound for the combined probability of the anticipated transition. The

asymptotically largest estimate (*i.e.*, for sufficiently small μ) obtained in this discussion is the term $K_3\mu(t)^{s/(2m)}$.

CASE 2: *Initial crossover-mutation step retains globally optimal creatures with positions in J.* An initial application of $C_{\chi(t)}M_{\mu(t)}$ to $p \in \Omega$ yields elements $q^+ \in \Omega^+ \setminus \Omega$ with probability bounded from above by $K_5\mu^{1/m}$, $K_5 \in \mathbf{R}_*^+$, for small μ, since at least one spot in a creature with position in J in p must be changed by mutation, or crossover must be applied. If selection is then applied to q^+, then the combined probability to generate elements of Ω' is bounded from above by $K_2\mu^{LB\log(\rho_2(f))}$ as was shown in line (*24*).

Hence, we have for $K_6 \in \mathbf{R}_*^+$ and summations over $q' \in \Omega'$, $q^c \in \wp \setminus \Omega^+$, $q^+ \in \Omega^+ \setminus \Omega$, $p \in \Omega$:

$$
\begin{aligned}
&||P_{\Omega'}G_tP_\Omega v_t||_1 = \\
&= \textstyle\sum_{q',q^c,p} \langle q', S_tq^c\rangle \langle q^c, C_{\chi(t)}M_{\mu(t)}p\rangle \langle p, v_t\rangle \\
&\qquad + \textstyle\sum_{q',q^+,p} \langle q', S_tq^+\rangle \langle q^+, C_{\chi(t)}M_{\mu(t)}p\rangle \langle p, v_t\rangle \\
&\le \textstyle\sum_{q^c,p} \langle q^c, C_{\chi(t)}M_{\mu(t)}p\rangle \langle p, v_t\rangle \\
&\qquad + \textstyle\sum_{q^+,p} K_2\mu^{LB\log(\rho_2(f))} \langle q^+, C_{\chi(t)}M_{\mu(t)}p\rangle \langle p, v_t\rangle \\
&\le \textstyle\sum_p K_6\mu^{s/(2m)} \langle p, v_t\rangle + \sum_p K_2K_5\mu^{LB\log(\rho_2(f))+1/m} \langle p, v_t\rangle \\
&= (K_6\mu^{s/(2m)} + K_2K_5\mu^{LB\log(\rho_2(f))+1})\omega(t). \qquad (25)
\end{aligned}
$$

Part 2c: The steady-state flow inequality. Combining inequalities (*22*) and (*25*) yields the steady-state flow inequality as follows:

$$
\begin{aligned}
&1-\omega(t) = ||P_{\Omega'}v_t||_1 = ||P_{\Omega'}G_tv_t||_1 = ||P_{\Omega'}G_tP_\Omega v_t||_1 + ||P_{\Omega'}G_tP_{\Omega'}v_t||_1 \\
&\le (K_6\mu^{s/(2m)} + K_2K_5\mu^{LB\log(\rho_2(f))+1/m})\omega(t) + (1-K_1\mu^\ell)(1-\omega(t)) \quad (26)
\end{aligned}
$$

Inequality (*26*) shows that
$K_1(1-\omega(t))/\omega(t) \le K_6\mu^{s/(2m)-\ell} + K_2K_5\mu^{LB\log(\rho_2(f))+1/m-\ell}$. This shows that $\lim_{t\to\infty}\omega(t) = 1$ and completes the proof. Q.E.D.

Theorem 3.3.2 is shown in more general versions in (Schmitt, 2002, Thm. 3.3.2, Cor. 3.3.3, Cor. 3.3.4). In particular, other more general crossover operators are allowed that need not commute with mutation. It shows the quite remarkable effect that with increasing population size, one is allowed to use a more relaxed cooling schedule for crossover. Thus for larger population size, the algorithm-design, *i.e.,* definition of data-structures (creatures), which is exploited by crossover plays a more important role. Overall, crossover has more time and opportunity to perform its enhancement of the mixing phase of the genetic algorithm. See (Schmitt et al., 1998, Prop. 10) and (Schmitt, 2001, Thm. 6.1) where this statement is given a precise meaning in terms of contraction properties of the combined mutation-crossover operator.

3.5 THE VOSE-LIEPINS VERSION OF MUTATION-CROSSOVER

Vose (Vose, 1999b, Sec. 5.4: p. 44) describes one cycle of the (simple) genetic algorithm as follows: (a) Obtain two parents by the selection function, (b) Mutate the parents by the mutation function. (c) Produce the mutated parent's child by the crossover function. (d) Put the child into the next generation. (e) If the next generation contains less than r members, go to step (a). r is the population size. Consult also the analysis by Vose&Liepins (Vose and Liepins, 1991). In what follows, VLGA shall refer to a genetic algorithm whose cycle is described by steps (a)–(e) as above.

The procedure of pairing creatures to produce offspring in the VLGA produces one child at a time while regular crossover as defined in Definition 2.2.2, in (Schmitt, 2001, Sec. 5.2.1), (Schmitt et al., 1998, Sec. 2.2) following, *e.g.,* Goldberg's book (Goldberg, 1989, p. 16–17) produces 2 offspring from 2 parents in a single crossover-step.

If $r \in \mathbf{N}$ denotes the population size of a VLGA, then we set $s = 2r$ and use the selector mask $J = [0, s] \cap (2\mathbf{N})$. The reader will easily check that a cycle $C_{\chi(t)} M_{\mu(t)} S_t$ in the setting described in this exposition corresponds exactly to the VLGA as listed above. This embeds the VLGA into the model developed here and in (Schmitt et al., 1998; Schmitt and Nehaniv, 1999; Schmitt, 2001; Schmitt, 2002). After every cycle, the next selection step will disregard the offspring $c_{2\sigma-1}$ for $1 \leq \sigma \leq r$ obtained through mutation-crossover in the previous cycle and randomly arrange the chosen new parents. Thus, we do not have to perform a "selection-step from the two offspring" as in (Vose, 1999b, p. 43: line 25). We need not assume that the given fitness function $f_v : \mathcal{C} \to \mathbf{R}_*^+$ in the sense of (Vose, 1999b, p. 25) is injective as in all of (Vose, 1999b).

Besides application of the Global Optimization Theorem 3.4.1 given above, the new model for the VLGA allows for application of the results in (Schmitt, 2001), their extensions as discussed in (Schmitt, 2002, Sec. 4.1), and the main results (Schmitt, 2002, Thm. 3.3.2, Cor. 3.3.3, Cor. 3.3.4). In particular, (Schmitt, 2001, Thm. 8.2, Thm. 8.3) and their extension discussed in (Schmitt, 2002, Sec. 4.1) show ergodicity but non-convergence to global optima for the VLGA with strictly positive mutation limit which includes the case of the simple VLGA. On the other hand, (Schmitt, 2001, Thm. 8.5, Thm. 8.6) and their extensions as well as (Schmitt, 2002, Thm. 3.3.2, Cor. 3.3.3, Cor. 3.3.4) show convergence to global optima of the scaled VLGA.

4. FUTURE EXTENSIONS OF THE THEORY

4.1 TOWARDS FINITE-LENGTH ANALYSIS ON FINITE-STATE MACHINES

In order to channel future development of theory of genetic algorithms, let us for a brief moment turn to the past. Theoretical description of genetic algorithms can be roughly classified in two categories or —considering a time line— overlapping phases: The first phase is characterized by schema-theory following (Holland, 1975) (including the variant of building block hypothesis, *cf.* (Goldberg, 1989, p. 41–45)), the second phase is characterized by Markov-chain analysis. Schema-theory has overall failed to produce any significant general convergence results (to global optima) be it for the genetic algorithm or the more elaborate setting of genetic programming. This does not say that schema-theory may not be useful to explain one-step behavior of a genetic algorithm in an environment where the fitness function is changing over time. See work by Lux and Schornstein in that regard (Lux and Schornstein, 2002). In the personal opinion of this author, schema-theory is a pleasant heuristic tool but should be abandoned as means to analyze genetic algorithms as optimization procedure. This admittedly blunt point of view is cautiously shared by Vose (Vose, 1999b, p. 211: lines 1–4). There are schema-theorems in existence where "*mutation is ignored* ($\mu=0$) *and the interplay crossover-selection manages optimization.*" Such work considers nothing but non-ergodic genetic drift as described and analyzed in (Schmitt and Nehaniv, 1999, Sec. 6) and (Schmitt, 2001, Sec. 7.5). This setting implies that the probabilistic outcome of the zero-mutation-rate genetic algorithm strongly depends upon the initial population or distribution of populations. This fact is rather obvious by considering the extreme case of an initial, suboptimal, uniform population. There are other schema-theorems in existence where "*mutation is supposed to work constantly in the background and the interplay crossover-selection manages optimization.*" Such algorithms must essentially fail by virtue of (Schmitt et al., 1998, Thm. 17), (Schmitt, 2001, Thm. 8.1–3). See also Rudolph's book (Rudolph, 1997).

Markov chain analysis of genetic algorithms is, in the opinion of this author, still in its infancy. Such analysis was initiated notably through work by Liepins & Vose (Vose and Liepins, 1991), Nix & Vose (Nix and Vose, 1992), and Davis &Principe (Davis, 1991; Davis and Principe, 1991; Davis and Principe, 1993). However, even though it is fairly simple (as shown in this exhibition) to set up a mathematical model for genetic algorithms based upon Markov chains, it has taken quite some time that non-elementary results with correct proof appeared in the literature. In important contributions, Davis&Principe (Davis and Principe, 1991; Davis and Principe, 1993) found that annealing the mutation rate to zero alone does not imply convergence of the genetic algorithm to global

optima. (Schmitt et al., 1998, Thm. 17) and (Schmitt, 2001, Thm. 8.1–3) show that increasing the selection pressure alone must fail as well. Thus, annealing the mutation rate to zero and increasing the selection pressure properly are required to assure at least asymptotic convergence to global optima. A significant contribution is then Cerf's work (Cerf, 1996; Cerf, 1998) which, however, assumes the auxiliary condition of a larger population-size that strongly depends upon the problem instance. Cerf's work seems to indicate that an "*infinite population limit*" may be a mathematical solution to achieve a comprehensive theoretical model for genetic algorithms. In the personal opinion of this author, such an approach is also of limited value mainly for the following reason: if optimization is performed with a computer, then we are dealing with something quite small and finite, *i.e.,* a small number of candidate solutions with a small number of genes. Genetic optimization in computers does not deal with large ensembles in the sense of statistical physics (Landau and Lifschitz, 1975) where an infinite number-of-particle limit may be appropriate. Genetic algorithms should be analyzed probabilisticly but in the spirit of Knuth (Knuth, 1997b; Knuth, 1997c; Knuth, 1997a): an algorithm that runs on a finite-state machine for a finite period of time. This point of view forces one to rethink even any asymptotic result such as the Global Optimization Theorem 3.4.1 which must be seen together with the discussion of stopping criteria after the proof of Theorem 3.3.2. See also section 4.2 in this regard.

To summarize the discussion in this section, let it be stated that future theoretical research on genetic algorithms and genetic programming should primarily deal with finite-length algorithms on finite-state machines and estimates in regard to approaching infinite-length asymptotics and global optima using a probabilistic framework. Thus, theory is at its beginning.

4.2 ESTIMATES FOR FINITE-LENGTH GENETIC ALGORITHMS À LA CATONI

Similar to the work presented in this exhibition, the simulated annealing algorithm was initially investigated in regard to its asymptotic behavior. See, *e.g.,* the essay by Aarts, & van Laarhoven for an excellent introduction and overview in regard to the simulated annealing algorithm. Asymptotic analysis of simulated annealing probably reached a peak in work by Hajek (Hajek, 1988).

At this point, let us mention that Lozano, Larrañaga, Graña & Albizuri (Lozano et al., 1999) have developed a genetic algorithm with a simulated-annealing-type selection strategy which converges asymptotically to global optima. This provides an alternative to the selection mechanism developed here. See also work by Mahfoud & Goldberg (Mahfoud and Goldberg, 1992; Mahfoud and Goldberg, 1995).

Work by Catoni (Catoni, 1990; Catoni, 1991b; Catoni, 1991a; Catoni, 1992) based upon large deviation estimates took analysis of the simulated annealing algorithm to a higher level. It constitutes a major advance in regard to developing probabilistic estimates for finite-length simulated annealing algorithms, *i.e.*, stopping criteria for these algorithms.

Future theoretical research on genetic algorithms and genetic programming should apply Catoni's work to the setting of scaled genetic algorithms with a simulated annealing type selection strategy or scaled proportional fitness selection.

4.3 ADDING SAMPLING NOISE

Kushner (Kushner, 1987) has extended the study of simulated annealing in yet another direction: it is assumed that the fitness function is sampled via Monte-Carlo simulation (see Binder's book (Binder, 1978)) and therefore the setting of the optimization algorithm is perturbed by sampling noise. Kushner presents an analysis of this setting via the theory of large deviations and discusses applications to global optimization via Monte Carlo methods. This study should be extended to the case of the genetic algorithm in *all* its standard incarnations. See also the discussion in (Beyer et al., 2002, Sec. 2).

4.4 FURTHER ANALOGY WITH SIMULATED ANNEALING: PARALLELISM AND SPARSE MUTATION

There are many similarities between the genetic algorithm as presented here and the simulated annealing algorithm. However, there are also fundamental differences. Genetic algorithms have an inherent parallelism, simulated annealing does not. Simulated annealing corresponds to population-size $s = 1$. Consequently, if one searches for an enveloping concept for simulated annealing and genetic algorithms, then one has to consider *parallel simulated annealing* (see, *e.g.*, (Azencott, 1992) for an overview).

A more mathematical/thermodynamic distinction between the two types of probabilistic algorithms is that the genetic algorithm presented here is weakly ergodic because the fully positive, scaled stochastic matrices associated with the generator-phase of the algorithm assure weak ergodicity (see section 3.2). In the simulated annealing procedure, weak ergodicity is obtained through carefully scaling the selection operator such that repeated combination of a constant, non-fully-positive generator-matrix and selection retains enough shrinking ability similar to the combined shrinking by operators $X_\tau M_\tau$ in Proposition 1.3.1. In both cases, increasing selection pressure assures convergence to global optima. If one considers the simulated annealing algorithm

with a fully positive generator matrix, then for any cooling schedule a globally optimal solution must be found eventually. One could therefore argue that the Global Optimization Theorem 3.4.1 has a certain weakness in that fully positive mutation matrices are used (*i.e.*, ergodicity is obtained too easily) even though the setting is *standard* in genetic algorithm applications.

Population-wise single-spot mutation as discussed in (Schmitt et al., 1998, Sec. 2.1: "one-bit mutation", Thm. 15) and (Schmitt, 2001, Sec. 3.2, Thm. 8.1) constitutes a non-fully-positive (sparse) mutation matrix which may implement a stronger analogue to the simulated-annealing type setting. The quoted Theorems deal with the applications of population-wise single-spot mutation in genetic algorithms with strictly positive limit mutation rate which includes the case of the simple genetic algorithm.

In order to strengthen the analogy between the genetic algorithm presented here and the simulated annealing algorithm in regard to a neighborhood-based search, future theoretical research should generalize (Schmitt, 2001, Thm. 8.5, Thm. 8.6) and their extensions discussed in (Schmitt, 2002, Sec. 4.1) as well as (Schmitt, 2002, Thm. 3.3.2, Cor. 3.3.3, Cor. 3.3.4), Theorem 3.4.1 and finite-length estimates to a setting where sparse mutation operators such as population-wise single-spot mutation are used.

4.5 ANALYSIS FROM INSIDE-OUT AND OUTSIDE-IN

There are, in principle, two major ways to analyze genetic algorithms: from *inside-out* and *outside-in.* The approach taken in this exhibition is to analyze the genetic algorithm from *outside-in.* The algorithm is understood as an all-purpose tool which is used in a black-box scenario, *i.e.*, on a fitness function of largely unknown behavior and characteristics. Thus, this analysis could be characterized as "finding a least upper bound" for suitable implementation of genetic algorithms. They are seen here as an ergodic "cooling procedure" similar to the simulated annealing algorithm setting which is inspired by the real-world process that is used to generate large crystals such as rubies by carefully cooling heated material.

The opposite way to analyze genetic algorithms is from *inside-out*, *i.e.*, to systematically analyze the behavior of possibly different, specially designed genetic algorithms for specific classes of problem instances. Such a way of analysis is even more in the spirit of Knuth (Knuth, 1997b; Knuth, 1997c; Knuth, 1997a). It could be characterized as "finding greatest lower bounds" for specific classes of problem instances and corresponding suitable implementations of genetic algorithms. The survey by Beyer, Schwefel & Wegener (Beyer et al., 2002, Sec. 3) advocates and illustrates this approach to theoretical analysis of genetic or evolutionary algorithms. It is pointed out that this

direction of research leads to a vast, unknown territory waiting for exploration. In regard to this point of view, consult, *e.g.*, work by Droste, Jansen, Tinnefeld & Wegener (Droste et al., 2002; Droste et al., 2003; Jansen and Wegener, 2001).

To summarize the discussion in this section, let it be stated that future theoretical research on genetic algorithms and genetic programming should also deal with detailed analysis of specific classes of problem instances and the behavior of corresponding suitable genetic algorithm implementations.

4.6 NON-MONOTONE AND SELF-ADAPTING ANNEALING SEQUENCES

In practical applications of genetic algorithms, one may be interested in annealing the mutation-rate (noise) in a rapid manner for a period of time in order to give the algorithm time to "thoroughly explore the neighborhood" of the current population and/or let crossover dominate as mixing operator, and then to increase the mutation rate again if the algorithm has "settled," *i.e.*, the fitness values in the population have become close to uniform, or the population became close to uniform itself. Concurrently with the mutation rate, the fitness selection scaling and the crossover rate may be altered in a non-monotone way as well.

Such scheduling of the mutation-rate and other parameters may be steered by an external schedule where an estimate for the "settling-time" has been obtained through experiments, or may be self-adaptive. Anily & Federgruen (Anily and Federgruen, 1987, Thm. 2) have shown for the simulated annealing algorithm that one can use certain non-monotone sequences for the cooling parameter and still obtain an asymptotically converging simulated annealing algorithm. The reader may adapt Anily & Federgruen's work to the situation of the Global Optimization Theorem 3.4.1 to obtain asymptotically converging genetic algorithms with more general annealing schedules for mutation, crossover and selection than presented in this work.

The author conjectures that the Global Optimization Theorem 3.4.1 can be generalized to the following situation with self-adapting, random-nature annealing schedules for mutation, crossover and selection: The mutation rate, the crossover rate and the exponentiation for the fitness-scaling are chosen at runtime depending upon the state of the algorithm. If the algorithm repeatedly returns to the same state (*i.e.*, population), then the mutation rate and crossover rate are increased by a certain magnitude and the selection-pressure is decreased correspondingly. Overall, the parameters follow similar trajectories as proposed in the Global Optimization Theorem 3.4.1. See Agapie's work (Agapie, 2001) in regard to this conjecture.

4.7 DISCRETE *vs.* CONTINUOUS ALPHABETS

Many applications of genetic algorithms, in particular, in engineering are optimization problems where **R**-valued parameters are optimized in a bounded or unbounded domain. As discussed in section 1.4, one pragmatic approach to optimization of **R**-valued parameters is to accept that computer languages such as Fortran or C usually use only a finite subset of the real numbers. Another reason for "discrete" optimization of **R**-valued parameters is the following: a bit-string such as `'o|||oo|||ooo'` may on the one hand be understood as the binary number $0.011100111000 \in [0,1]$ but can on the other hand be seen as the label of a leaf in a binary tree of depth 12. Thus (arguing here in a purely heuristic manner), discrete implementation allows a genetic algorithm to map a "floating compromise between real, fractal and purely discrete structures".

However, there exist a considerable amount of research in evolutionary optimization for **R**-valued parameters as outlined in (Beyer et al., 2002, Sec. 2). One may therefore also pursue the generalization of the approach presented here and in (Schmitt, 2001; Schmitt, 2002) to the case of a "continuous" alphabet such as [0, 1] or **R**.

5. APPENDIX — PROOF OF SOME BASIC OR TECHNICAL RESULTS

This appendix lists a collection of proofs of some of the more basic results used in this exposition for the convenience of the reader.

PROOF of lines (5) and (6): The definition of $||X||_1$ in line (*4*) via the supremum shows that $||X||_1$ is the smallest constant r such that $||Xv||_1 \leq r||v||_1$ for every $v \in \mathbf{C}^k$. If $X, Y \in \mathbf{M}_k$ and $v \in \mathbf{C}^k$, then one has: $||(X+Y)v||_1 \leq ||Xv||_1 + ||Yv||_1 \leq ||X||_1 \cdot ||v||_1 + ||Y||_1 \cdot ||v||_1 = (||X||_1 + ||Y||_1) \cdot ||v||_1 \Rightarrow$ **line (5)**, and $||XYv||_1 \leq ||X||_1 \cdot ||Yv||_1 \leq ||X||_1 \cdot ||Y||_1 \cdot ||v||_1 \Rightarrow$ **line (6)**. Q.E.D.

PROOF of line (*7*): Let $r = \max\{||(X_{\kappa',\kappa})_{\kappa'}||_1 : \kappa\}$. Let $v = (v_\kappa) \in \mathbf{C}^k$ such that $||v||_1 = 1$. Then $||Xv||_1 = \sum_{\kappa'} |\sum_\kappa X_{\kappa',\kappa} v_\kappa| \leq \sum_{\kappa,\kappa'} |X_{\kappa',\kappa} v_\kappa| = \sum_\kappa ||(X_{\kappa',\kappa})_{\kappa'}||_1 |v_\kappa| \leq r \sum_\kappa |v_\kappa| \leq r$. This shows $||X||_1 \leq r$. Conversely, $||X||_1 \geq ||Xb_\kappa||_1 = ||(X_{\kappa',\kappa})_{\kappa'}||_1$ for *every* κ, $0 \leq \kappa \leq k-1$ since $||b_\kappa||_1 = 1$. This shows that $||X||_1 \geq r$. Q.E.D.

PROOF of line (*10*): $||X||_1 = 1$ follows directly from line (*7*). If $X = (X_{\kappa',\kappa})$ and $v = (v_\kappa) \in \mathcal{S}_\wp$, then $||Xv||_1 = \sum_{\kappa'}(\sum_\kappa X_{\kappa',\kappa} v_\kappa) = \sum_\kappa (\sum_{\kappa'} X_{\kappa',\kappa}) v_\kappa = \sum_\kappa v_\kappa = 1$. This shows $X\mathcal{S}_\wp \subset \mathcal{S}_\wp$. Applying the latter to the columns of Y yields that XY is stochastic. Q.E.D.

PROOF of Proposition 1.3.1: We first note that $<e, v - w> = <e, v> - <e, w> = k^{-1}||v||_1 - k^{-1}||w||_1 = 0$. Hence, $P(e)(v - w) = 0$. Since every coefficient of M_t is in $[\epsilon_t, 1]$, one has $M_t = \epsilon_t k\, P(e) + M_t'$ with $M_t' \in \mathbf{M}_k(\mathbf{R}^+)$. Using line (*7*), we conclude that $||M_t'||_1 \leq 1 - \epsilon_t k$. Hence, $||M_t(v - w)||_1 = ||M_t'(v - w)||_1 \leq (1 - \epsilon_t k)||v - w||_1$. This shows statement (1) of Proposition 1.3.1. Using the statements in line (*10*), and applying statement (1) to $M_t, M_{t-1}, \ldots$ inductively, we obtain

$$
\begin{aligned}
&||(\textstyle\prod_{\tau=t}^{1} X_\tau M_\tau)(v - w)||_1 \\
&= ||X_t M_t \cdot ((\textstyle\prod_{\tau=t-1}^{1} X_\tau M_\tau)v - (\prod_{\tau=t-1}^{1} X_\tau M_\tau)w)||_1 \\
&\leq (1 - \epsilon_t k)||(\textstyle\prod_{\tau=t-1}^{1} X_\tau M_\tau)(v - w)||_1 \\
&\leq (\textstyle\prod_{\tau=1}^{t}(1 - \epsilon_\tau k))\, ||v - w||_1. \qquad (27)
\end{aligned}
$$

One also has: $\log(\prod_{\tau=1}^{t}(1 - \epsilon_\tau k)) = \sum_{\tau=1}^{t} \log(1 - \epsilon_\tau k) \leq \sum_{\tau=1}^{t}(-\epsilon_\tau k) \to -\infty$ as $t \to \infty$. This shows $\prod_{\tau=1}^{t}(1 - \epsilon_\tau k) \to 0$ as $t \to \infty$. Inequality (27) then shows statement (2) of Proposition 1.3.1. Q.E.D.

PROOF of Proposition 1.3.2: We have $X \in \mathbf{M}_k([\epsilon, \infty))$ for some $\epsilon \in (0, 1/k]$. Let $v_t = X^t b_0 \in \mathcal{S}_\wp$ for $t \in \mathbf{N}$. Then there exists a subsequence $(v_{t(\tau)})_{\tau \in \mathbf{N}})$ that converges to $v \in \mathcal{S}_\wp$ since $\mathcal{S}_\wp$ is compact. Now we have:

$$
\begin{aligned}
&||Xv - v||_1 = \lim_{\tau \to \infty} ||Xv_{t(\tau)} - v_{t(\tau)}||_1 \\
&= \lim_{\tau \to \infty} ||X(X^{t(\tau)} b_0 - X^{t(\tau)-1} b_0)||_1 \\
&\leq \lim_{\tau \to \infty} (1 - \epsilon\, k)^{t(\tau)} ||Xb_0 - b_0||_1 = 0
\end{aligned}
$$

where Proposition 1.3.1.1 was applied repeatedly in the second-to-last step. This shows $||Xv - v||_1 = 0$. Thus, $v = Xv$ must be fully positive since X is fully positive and, consequently, every component of v is a convex combination of strictly positive entries of X.

If $w \in \mathbf{C}^k$ is such that $Xw = w$, then $X\mathrm{Re}(w) = \mathrm{Re}(Xw) = \mathrm{Re}(w)$. Now, chose $r \in \mathbf{R}^+$ large enough such that $w^+ = \mathrm{Re}(w) + rv \in (\mathbf{R}_*^+)^k$ and let $w' = ||w^+||_1^{-1} w^+ \in \mathcal{S}_\wp$. Then, $w' = Xw'$ since both $\mathrm{Re}(w)$ and v are invariant vectors of X. Applying Proposition 1.3.1.1 again, we get

$$||w' - v||_1 = ||X(w' - v||_1 \leq (1 - \epsilon\, k)||w' - v||_1 = 0$$

since $1 - \epsilon\, k < 1$. This shows $w' = v$. Hence, $\mathrm{Re}(w) = (||w^+||_1 - r)v$ is a scalar multiple of v. Similarly, one obtains that $\mathrm{Im}(w)$ is a scalar multiple of v. This concludes the proof of Proposition 1.3.2. Q.E.D.

PROOF of Corollary 1.3.3: The matrix $X_t = (1 - t^{-1})X + t^{-1}P(e)$ is stochastic and fully positive for $t \in \mathbf{N}$. By Proposition 1.3.2, there exists

$v_t \in \mathcal{S}_\wp$ such that $X_t v_t = v_t$. Since $\mathcal{S}_\wp$ is compact, there exists a subsequence $(v_{t(\tau)})_{\tau \in \mathbf{N}})$ that converges to $v \in \mathcal{S}_\wp$. Then, we have

$$Xv = \lim_{\tau \to \infty} X_{t(\tau)} \cdot \lim_{\tau \to \infty} v_{t(\tau)} = \lim_{\tau \to \infty} X_{t(\tau)} v_{t(\tau)}$$

$$= \lim_{\tau \to \infty} v_{t(\tau)} = v. \qquad \text{Q.E.D.}$$

PROOF of Corollary 1.3.4: MX is fully positive since every column of MX is a convex combination of the strictly positive columns of M. Proposition 1.3.2 shows that a right eigenvector v with the desired properties exists.

Any possible invariant right eigenvector w of XM satisfies $MXMw = Mw$. Thus, $Mw = \zeta v$ for some $\zeta \in \mathbf{C}$. Hence, if M is invertible, then w is uniquely determined up to scalar multiples. Corollary 1.3.3 shows that XM has an invariant right eigenvector $w \in \mathcal{S}_\wp$. Then, $Mw \in \mathcal{S}_\wp$ since M is stochastic. Hence, $Mw = v$. Q.E.D.

PROOF of Proposition 1.4.1: We have $XP_\mathcal{U} p = Xp = p = P_\mathcal{U} p$ for $p \in \wp \cap \mathcal{U}$, and $XP_\mathcal{U} p = 0 = P_\mathcal{U} p$ for $p \in \wp \setminus \mathcal{U}$. Hence, $XP_\mathcal{U} = P_\mathcal{U}$ since the actions of the two linear operators coincide on the basis $\wp$ of $\mathcal{V}_\wp$. In addition, we have $(\mathbf{1} - P_\mathcal{U})X = (\mathbf{1} - P_\mathcal{U})XP_\mathcal{U} + (\mathbf{1} - P_\mathcal{U})X(\mathbf{1} - P_\mathcal{U}) = (\mathbf{1} - P_\mathcal{U})P_\mathcal{U} + (\mathbf{1} - P_\mathcal{U})X(\mathbf{1} - P_\mathcal{U}) = (\mathbf{1} - P_\mathcal{U})X(\mathbf{1} - P_\mathcal{U})$. Q.E.D.

PROOF of Proposition 3.3.1: Proposition 2.1.2.4 shows that for $t > t_o$, M_μ is an invertible matrix, $t_o \in \mathbf{N}$. Corollary 1.3.4 applied with $X = S_t C_{\chi(t)}$ then shows that v_t is uniquely determined for $t > t_o$. In order to show statement (1), we have to show that $\sum_{t=t_1}^{T} ||v_{t+1} - v_t||_1 = \sum_{p \in \wp} \sum_{t=t_1}^{T} |<p, v_{t+1} - v_t>| \leq B_o$ for fixed t_1, B_o and every $T \in \mathbf{N}, T > t_1$. Since $\wp$ is finite, it is enough to show that $\sum_{t=t_p}^{T} |<p, v_{t+1} - v_t>| \leq B_p$ for fixed B_p and every $T \in \mathbf{N}$, $T > t_p, p \in \wp$.

Let $t > t_o$, then v_t is up to scalar multiples the uniquely determined solution of $(G_t - \mathbf{1})v_t = 0$, *i.e.*, v_t generates the one-dimensional kernel of $G_t - \mathbf{1}$. Adding the rows of $G_t - \mathbf{1}$ to the first row (which is an admissible operation under the Gauss elimination algorithm (Greub, 1975, p. 97: 3.17)), we see that the kernel of $(G_t - \mathbf{1})^{[0]}$ is generated by v_t as well. Since $<e, v_t> = \dim(\mathcal{V}_\wp)^{-1} = 2^{-L} > 0$, we see that $(G_t - \mathbf{1})^{[e^*]}$ has kernel $\{0\}$,. Thus, $(G_t - \mathbf{1})^{[e^*]}$ is invertible, and the equation $(G_t - \mathbf{1})^{[e^*]} v_t = (2^{-L}, 0, \ldots, 0)^*$ uniquely determines v_t. In this situation, v_t can be computed using Cramer's Rule (Lang, 1970, p. 182: Thm. 3) which amounts to computing quotients of determinants.

Using Proposition 2.3.2.1 and $a^{B \log(b)} = \exp(B \log(a) \log(b)) = b^{B \log(a)}$, we obtain for the coefficients of the selection operator S_t:

$$<q, S_t p>$$

$$= (\textstyle\sum_{\sigma' \in J}^{s} (t+1)^{B \log(f(c_{\sigma'}, p))})^{-s} \prod_{\sigma=1}^{s} n(d_\sigma, p \wedge J) \, (t+1)^{B \log(f(d_\sigma, p))}$$

$$= (\sum_{\sigma' \in J}^{s} 2^{-L} \mu^{-LB \log(f(c_{\sigma'}, p))})^{-s} \cdot$$

$$\cdot \prod_{\sigma=1}^{s} n(d_\sigma, p \wedge J)\, 2^{-L} \mu^{-LB \log(f(d_\sigma, p))} \tag{28}$$

for $p = (c_1, c_2, \ldots, c_s)$, $q = (d_1, d_2, \ldots, d_s) \in \wp$, $c_\sigma, d_\sigma \in \mathcal{C}$, $n(d_\sigma, p \wedge J) \in \mathbf{N}_0$, $1 \leq \sigma \leq s$. Proposition 2.1.2.1 shows that the coefficients of M_μ are polynomials in μ. Proposition 2.2.3.1 shows that the coefficients of C_χ are polynomials in χ. Combining the latter two statements with line (*28*) and $\mu = \chi^m$, we obtain for $<p, v_t>$, $t = (2\mu)^{-L} - 1$, computed via Cramer's rule:

$$\phi(\mu) = <p, v_t> = \sum_{\nu=1}^{n_1} r_{\nu,1} \cdot \mu^{r_{\nu,2}} / \sum_{\nu=1}^{n_2} r_{\nu,3} \cdot \mu^{r_{\nu,4}},,$$

$$r_{\nu,1}, r_{\nu,3} \in \mathbf{R},\ r_{\nu,2} \in \mathbf{R}^+ \text{ distinct},\ r_{\nu,4} \in \mathbf{R}^+ \text{ distinct},$$

$$n_1, n_2 \in \mathbf{N},\ r_{1,3} = 1,\ r_{1,4} = 0. \tag{29}$$

We can assume W.L.O.G., that $r_{1,3} = 1$ and $r_{1,4} = 0$ since $<p, v_t>$ is the component of a probability distribution and, consequently, stays bounded as $\mu \to 0$. This shows that $\lim_{t \to \infty} <p, v_t> = \lim_{\mu \to 0} \phi(\mu)$ exists. This implies statement (2) of Proposition 3.3.1.

We substitute $\mu = z^K$ for a constant $K \geq 1$ in line (29) to obtain

$$\phi(z^K) = \sum_{\nu=1}^{n_1} r_{\nu,1} \cdot z^{r'_{\nu,2}} / (1 + \sum_{\nu=2}^{n_2} r_{\nu,3} \cdot z^{r'_{\nu,4}}),$$

$$r'_{\nu,2}, \in \{0\} \cup [1, \infty) \text{ distinct}, r'_{\nu,4} \in [1, \infty) \text{ distinct}.$$

Hence, $|(d/dz)\phi(z^K)|$ is bounded by some constant $K' > 0$ for $z \in [0, z_p]$, $z_p > 0$. By the Mean Value Theorem (Lang, 1968, p. 60: Thm. III.1), we have for $t_p \in \mathbf{N}$ such that $2^{-1/K}(t_p + 1)^{-1/KL} \leq z_p$ and every $T > t_p$:

$$\sum_{t=t_p}^{T} |<p, v_{t+1} - v_t>| = \sum_{t=t_p}^{T} |\phi((t+2)^{-1/L}/2) - \phi((t+1)^{-1/L}/2)|$$

$$\leq \sum_{t=t_p}^{T} K' \cdot |(t+2)^{-1/(K \cdot L)} - (t+1)^{-1/(K \cdot L)}|$$

$$= K' \cdot ((t_p + 1)^{-1/(K \cdot L)} - (T+2)^{-1/(K \cdot L)})$$

$$< K' \cdot ((t_p + 1)^{-1/(K \cdot L)} = B_p.$$

This completes the proof of statement (1) of Proposition 3.3.1. Statement (2) follows from statement (1) but was already obtained above in passing. Q.E.D.

REFERENCES

Aarts, E. H. L. and van Laarhoven, P. J. M. (1989). Simulated Annealing: An Introduction. *Statistica Neerlandica,* 43:31–52.

Agapie, A. (2001). Theoretical analysis of mutation-adaptive evolutionary algorithms. *Evolutionary Computation,* 9:127–146.

Anily, S. and Federgruen, A. (1987). Ergodicity in Parametric Non-Stationary Markov Chains: An Application to Simulated Annealing Methods. *Operations Research,* 35:867–874.

Aytug, H. and Koehler, G. J. (1996). Stopping Criteria for Finite Length Genetic Algorithms. *INFORMS Journal on Computing,* 8:183–191.

Azencott, R., editor (1992). *Simulated Annealing — Parallelization Techniques.* John Wiley & Sons Publishers.

Baker, J. E. (1987). Reducing Bias and Inefficiency in the Selection Algorithm. In Grefenstette, J. J., editor, *Genetic Algorithms and Their Applications: Proceedings of the Second International Conference on Genetic Algorithms.* Lawrence Erlbaum Publishers.

Banzhaf, W., Francone, F. D., and Nordin, P. (1996). The Effect of Extensive Use of the Mutation Operator on Generalization in Genetic Programming using Sparse Data Sets. In Ebeling, W., Rechenberg, I., Schwefel, H. P., and Voigt, H. M., editors, *4th International. Conference. on Parallel Problem Solving from Nature (PPSN96),* pages 300–309. Springer Verlag.

Beyer, H.-G., Schwefel, H.-P., and Wegener, I. (2002). How to analyse evolutionary algorithms. *Theoretical Computer Science,* 287:101–130.

Binder, K. (1978). *Monte Carlo methods in Statistical Physics.* Springer Verlag.

Bremermann, H. J., Rogson, J., and Salaff, S. (1966). Global Properties of Evolution Processes. In Pattee, H. H., Edelsack, E., Fein, L., and Callahan, A., editors, *Natural Automata and Useful Simulations,* pages 3–42. Spartan Books, Washington DC.

Catoni, O. (1990). *Large Deviations for Annealing.* PhD thesis, Université Paris XI.

Catoni, O. (1991a). Applications of Sharp Large Deviations Estimates to Optimal Cooling Schedules. *Annales de l'Institute Henri Poincaré. Probabilités ét Statistique,* 27:493–518.

Catoni, O. (1991b). Sharp Large Deviations Estimates for Simulated Annealing Algorithms. *Annales de l'Institute Henri Poincaré Probabilités ét Statistique,* 27:291–383.

Catoni, O. (1992). Rough Large Deviations Estimates for Simulated Annealing — Application to Exponential Schedules. *The Annals of Probability,* 20:1109–1146.

Cerf, R. (1996). An Asymptotic Theory for Genetic Algorithms. In Alliot, J. M., Lutton, E., Ronald, E., Schoenauer, M., and Snyers, D., editors, *Artificial Evolution. European Conference AE 95. Brest, France, September 4–6, 1995. Selected Papers,* volume 1063 of *Lecture Notes in Computer Science,* pages 37–53. Springer Verlag.

Cerf, R. (1998). Asymptotic Convergence of Genetic Algorithms. *Advances in Applied Probability,* 30:521–550.

Chung, K. L. (1974). *A Course in Probability Theory.* Academic Press Publishers. Second edition.

Davis, T. (1991). *Toward an Extrapolation of the Simulated Annealing Convergence Theory onto the Simple Genetic Algorithm.* PhD thesis, University of Florida.

Davis, T. E. and Principe, J. C. (1991). A Simulated Annealing-like Convergence Theory for the Simple Genetic Algorithm. In Belew, R. K. and Booker, L. B., editors, *Proceedings of the Fourth International Conference on Genetic Algorithms '91,* pages 174–181. Morgan Kaufmann Publishers.

Davis, T. E. and Principe, J. C. (1993). A Markov Chain Framework for the Simple Genetic Algorithm. *Evolutionary Computation,* 1:269–288.

Droste, S., Jansen, T., Tinnefeld, K., and Wegener, I. (2003). A new framework for the valuation of algorithms for black-box optimization. In De Jong, K, Poli, R., and Rowe, J., editors, *Foundations of Genetic Algorithms 7*, pages 197–214. Morgan Kaufmann Publishers.

Droste, S., Jansen, T., and Wegener, I. (2002). On the Analysis of the (1 + 1) Evolutionary Algorithm. *Theoretical Computer Science,* 276:51–81.

Feller, W. (1968). *An Introduction to Probability Theory and Its Applications. Volume 1.* John Wiley & Sons Publishers. (third edition).

Feller, W. (1971). *An Introduction to Probability Theory and Its Applications. Volume 2.* John Wiley & Sons Publishers. (second edition).

Fogel, D. B. (1994). Asymptotic Convergence Properties of Genetic Algorithms and Evolutionary Programming: Analysis and Experiments. *Cybernetics and Systems*, 25:389–407.

Goldberg, D. E. Genetic Algorithms Tutorial. Genetic Programming Conference, Stanford University, (July 13, 1997).

Goldberg, D. E. (1989). *Genetic Algorithms, in Search, Optimization & Machine Learning.* Addison-Wesley Publishers.

Goldberg, D. E. (1990). A Note on Boltzmann Tournament Selection for Genetic Algorithms and Population Oriented Simulated Annealing. *Complex Systems*, 4:445–460.

Goldberg, D. E. and Deb, K. (1991). A Comparative Analysis of Selection Schemes used in Genetic Algorithms. In Rawlins, G. J. E., editor, *Foundations of Genetic Algorithms,* pages 69–93. Morgan Kaufmann Publishers.

Greenwood, G. W. and Zhu, Q. J. (2001). Convergence in Evolutionary Programs with Self-Adaptation. *Evolutionary Computation,* 9:147–157.

Greub, W. (1975). *Linear Algebra.* Springer-Verlag.

Griffiths, R. C. and Taveré, S. (1997). Computational Methods for the Coalescent. In Donnelly, P. and Taveré, S., editors, *Progress in Population Genetics and Human Evolution,* volume 87 of *IMA Volumes in Mathematics and its Applications,* pages 165–182. Springer Verlag.

Hajek, B. (1988). Cooling Schedules for Optimal Annealing. *Mathematics of Operations Research,* 13:345–351.

He, J. and Kang, L. (1999). On the Convergence Rates of Genetic Algorithms. *Theoretical Computer Science,* 229:23–39.

Holland, J. H. (1975). *Adaptation in Natural and Artificial Systems.* University of Michigan Press. Extended new edition. MIT Press (1992).

Isaacson, D. L. and Madsen, R. W. (1961). *Markov Chains: Theory and Applications.* Prentice-Hall Publishers.

Jansen, T. and Wegener, I. (2001). Real royal road functions — where crossover provably is essential. In Spector, L. and Goodman, E. D., editors, *GECCO 2001, Proceedings of the Genetic and Evolutionary Computation Conference,* pages 375–382. Morgan Kaufmann Publisher.

Knuth, D. E. (1997a). *The Art of Computer Programming. Volume 3: Sorting and Searching.* Addison-Wesley Publishers, Second Edition edition.

Knuth, D. E. (1997b). *The Art of Computer Programming. Volume 1: Fundamental Algorithms.* Addison-Wesley Publishers, Third Edition edition.

Knuth, D. E. (1997c). *The Art of Computer Programming. Volume 2: Seminumerical Algorithms.* Addison-Wesley Publishers, Third Edition edition.

Koehler, G. J. (1994). A Proof of the Vose-Liepins Conjecture. *Annals of Mathematics and Artificial Intelligence,* 10:408–422.

Kushner, H. J. (1987). Asymptotic Global Behavior for Stochastic Approximation and Diffusions with Slowly Decreasing Noise Effects: Global Minimization via Monte Carlo. *SIAM Journal of Applied Mathematics,* 47:169–185.

Landau, S. D. and Lifschitz, E. M. (1975). *Lehrbuch der thoeretischen Physik V. Statistische Physik.* Berlin: Akademie Verlag.

Lang, S. (1968). *Analysis I.* Addison-Wesley Publishers.

Lang, S. (1970). *Linear Algebra.* Addison-Wesley Publishers, 2nd edition.

Leung, Y., Chen, Z.-P., Xu, Z.-B., and Leung, K.-S. (1998). *Convergence Rate for Non-Binary Genetic Algorithms with Different Crossover Operators.* The Chinese University of Hong Kong. (Preprint).

Lozano, J. A., Larrañaga, P., Graña, M., and Albizuri, F. X. (1999). Genetic Algorithms: bridging the Convergence Gap. *Theoretical Computer Science,* 229:23–39.

Lux, T. and Schornstein, S. (2002). Genetic learning as an Explanation of Stylized Facts of Foreign Exchange Markets. Manuscript, Universität Kiel, London School of Economics. pdf-file, www.bwl.uni-kiel.de/vwlinstitute/gwrp/publications/publications.htm.

Mahfoud, S. W. (1993). Finite Markov Chain Models of an Alternative Selection Strategy for Genetic Algorithms. *Complex Systems,* 7:155–170.

Mahfoud, S. W. and Goldberg, D. E. (1992). A Genetic Algorithm for Parallel Simulated Annealing. In Männer, R. and Manderick, B., editors, *Parallel Problem Solving from Nature, 2*, pages 301–310. Elsevier Publishers.

Mahfoud, S. W. and Goldberg, D. E. (1995). Parallel Recombinative Simulated Annealing: a Genetic Algorithm. *Parallel Computing,* 21:1–28.

Márkus, A., Renner, G., and Vanza, J. (1997). Spline Interpolation with Genetic Algorithms. In Kunii, T. L., Falcidieno, B., Savchenko, V. V., and Pasko, A., editors, *Proceedings of the International Conference on Shape Modeling, 1997, March 3–6, The University of Aizu, Aizu-Wakamatsu, Japan*, pages 47–54. IEEE Computer Society Press.

Michalewicz, Z. (1994). *Genetic Algorithms + Data Structures = Evolution Programs.* Springer Verlag, 2nd edition.

Mitchell, M. (1996). *An Introduction to Genetic Algorithms.* MIT Press.

Mühlenbein, H. (1997). Genetic Algorithms. In Aarts, E. and Lenstra, L., editors, *Local Search in Combinatorial Optimization,* pages 131–171. John Wiley & Sons Publishers.

Nix, A. E. and Vose, M. D. (1992). Modeling Genetic Algorithms with Markov Chains. *Annals of Mathematics and Artificial Intelligence,* 5:79–88.

Nomura, T. and Shimohara, K. (2001). An Analysis of Two-Parent Recombinations for Real-Valued Chromosomes in an Infinite Population. *Evolutionary Computation,* 9:283–308.

Poli, R. and Langdon, M. (1998). Schema Theory for Genetic Programming with One-Point Crossover and Point Mutation. *Evolutionary Computation,* 6(3):231–252.

Poli, R. (2001). Exact Schema Theory for Genetic Programming and Variable-Length Genetic Algorithms with One-Point Crossover. *Genetic Programming and Evolvable Machines,* 2:123–163.

Rudin, W. (1973). *Functional Analysis.* McGraw-Hill Publishers.

Rudolph, G. (1994). Convergence Analysis of Canonical Genetic Algorithms. *IEEE Transactions on Neural Networks,* 5:96–101.

Rudolph, G. (1997). *Convergence Properties of Evolutionary Algorithms.* Hamburg: Verlag Dr. Kovac.

Savchenko, V. and Schmitt, L. M. (2001). Reconstructing occlusal surfaces of teeth using a genetic algorithm with simulated annealing type selection. In Anderson, D. C. and Lee, K., editors, *Proceedings of the Sixth ACM Symposium on Solid Modeling and Applications,* pages 39–46. ACM Press.

Schaefer, H. H. (1974). *Banach Lattices and Positive Operators,* volume 215 of *Die Grundlehren der mathematischen Wissenschaften in Einzeldarstellungen.* Springer-Verlag.

Schmitt, L. M. (2001). Theory of Genetic Algorithms. *Theoretical Computer Science,* 259:1–61.

Schmitt, L. M. (2002). Theory of Genetic Algorithms II. – Convergence to optima for arbitrary fitness function. Technical Report 2002-2-002, The University of Aizu. (Submitted for journal publication).

Schmitt, L. M. and Kondoh, T. (2000). Optimization of Mass Distribution in Articulated Figures with Genetic Algorithms. In Hamza, M. H., editor, *Proceedings of the 1ASTED International Conference "Applied Simulation and Modelling",* pages 191–197. IASTED/ACTA Press.

Schmitt, L. M. and Nehaniv, C. L. (1999). The Linear Geometry of Genetic Operators with Applications to the Analysis of Genetic Drift and Genetic Algorithms using Tournament Selection. In Nehaniv, C. L., editor, *Mathematical and Computational Biology: Computational Morphogenesis, Hierarchical Complexity, and Digital Evolution, An International Workshop, 1997, October 21–25, The University of Aizu, Aizu-Wakamatsu, Japan,* volume 26 of *Lectures on Mathematics in the Life Sciences Series,* pages 147–166. American Mathematical Society.

Schmitt, L. M., Nehaniv, C. L., and Fujii, R. H. (1998). Linear Analysis of Genetic Algorithms. *Theoretical Computer Science,* 200:101–134.

Seneta, E. (1981). *Non-negative Matrices and Markov Chains.* Springer Series in Statistics. Springer Verlag.

Suzuki, J. (1997). A Further Result on the Markov Chain Model of Genetic Algorithms and Its Application to a Simulated Annealing-like Strategy. In Belew, R. K. and Vose, M. D., editors, *Foundations of Genetic Algorithms 4,* pages 53–72. Morgan Kaufmann Publishers.

Suzuki, J. (1998). A Further Result on the Markov Chain Model of Genetic Algorithms and Its Application to a Simulated Annealing-like Strategy. *IEEE Trans. on Systems, Man, and Cybernetics — Part B,* 28:95–102.

van Nimwegen, E., Crutchfield, J. P., and Mitchell, M. (1999). Statistical Dynamics of the Royal Road Genetic Algorithm. *Theoretical Computer Science,* 229:41–102.

Vose, M. D. and Wright, A. H. (1998a). The Simple Genetic Algorithm and the Walsh Transform: Part I, Theory. *Evolutionary Computation,* 6:253–273.

Vose, M. D. and Wright, A. H. (1998b). The Simple Genetic Algorithm and the Walsh Transform: Part II, The Inverse. *Evolutionary Computation*, 6:275–289.

Vose, M. D. (1999a). Random Heuristic Search. *Theoretical Computer Science*, 229:103–142.

Vose, M. D. (1999b). *The Simple Genetic Algorithm: Foundations and Theory.* MIT Press.

Vose, M. D. and Liepins, G. E. (1991). Punctuated Equilibria in Genetic Search. *Complex Systems*, 5:31–44.

Chapter 9

THE CHALLENGE OF PRODUCING HUMAN-COMPETITIVE RESULTS BY MEANS OF GENETIC AND EVOLUTIONARY COMPUTATION

John R. Koza
Stanford University
Stanford, California
koza@stanford.edu

Matthew J. Streeter
Genetic Programming Inc.
Mountain View, California
mjs@tmolp.com

Martin A. Keane
Econometrics Inc.
Chicago, Illinois
martinkeane@ameritech.net

Abstract Human-competitive results include those equivalent to new scientific results published in peer-reviewed scientific journals, solutions to long-standing or indisputably difficult problems, patented inventions, and results that tie or beat human contestants in regulated competitions. We argue that the pursuit of human-competitive results is not only a worthy goal in itself, but a useful compass for guiding the future growth of the field. We say this for reasons of utility, objectivity, complexity, and interminability. We believe that the continuing generation of evermore important human-competitive results relies on progress in three areas of research: multiobjective optimization, parallel computing, and the development and perfection of competent genetic and evolutionary search methods. Addressing the characteristics of human-competitive problems is one way to expand the theoretical underpinnings of the field of genetic and evolutionary computation.

Keywords: Genetic programming, human-competitive results

1. TURING'S PREDICTION CONCERNING GENETIC AND EVOLUTIONARY COMPUTATION

In his seminal 1948 paper "*Intelligent Machinery*," Turing identified three ways by which human-competitive machine intelligence might be achieved. In connection with one of those ways, Turing said:

> "There is the genetical or evolutionary search by which a combination of genes is looked for, the criterion being the survival value." (Turing, 1948)

Turing did not specify how to conduct the "genetical or evolutionary search" for machine intelligence. In particular, he did not mention the idea of a population-based parallel search in conjunction with sexual recombination (crossover) as described in John Holland's 1975 book *Adaptation in Natural and Artificial Systems.* However, in his 1950 paper "*Computing Machinery and Intelligence*," Turing did point out:

> "We cannot expect to find a good child-machine at the first attempt. One must experiment with teaching one such machine and see how well it learns. One can then try another and see if it is better or worse. There is an obvious connection between this process and evolution, by the identifications,
>
> Structure of the child machine = Hereditary material
>
> Changes of the child machine = Mutations
>
> Natural selection = Judgment of the experimenter" (Turing, 1950)

That is, Turing correctly perceived that one possibly productive approach to machine intelligence would involve an evolutionary process in which a description of a computer program (the hereditary material) undergoes progressive modification (mutation) under the guidance of natural selection (i.e., selective pressure in the form of what we now call "fitness").

2. DEFINITION OF HUMAN-COMPETITIVE

We contend that the pursuit of producing human-competitive results is a worthy compass for guiding the future growth of the field of genetic and evolutionary computation.

When we use the term "human-competitive" in connection with evaluating the results of an automated problem-solving method, we mean it in the sense used by machine learning pioneer Arthur Samuel (1983):

> "[T]he aim [is]...to get machines to exhibit behavior, which if done by humans, would be assumed to involve the use of intelligence." (Samuel, 1983)

To make the idea of human-competitiveness concrete, we say an automatically created solution to a problem is human-competitive if it satisfies one or more of the eight criteria in table 9.1.

Table 9.1. Eight criteria for saying that an automatically created result is human-competitive

	Criterion
A	The result was patented as an invention in the past, is an improvement over a patented invention, or would qualify today as a patentable new invention.
B	The result is equal to or better than a result that was accepted as a new scientific result at the time when it was published in a peer-reviewed scientific journal.
C	The result is equal to or better than a result that was placed into a database or archive of results maintained by an internationally recognized panel of scientific experts.
D	The result is publishable in its own right as a new scientific result — *independent* of the fact that the result was mechanically created.
E	The result is equal to or better than the most recent human-created solution to a long-standing problem for which there has been a succession of increasingly better human-created solutions.
F	The result is equal to or better than a result that was considered an achievement in its field at the time it was first discovered.
G	The result solves a problem of indisputable difficulty in its field.
H	The result holds its own or wins a regulated competition involving human contestants (in the form of either live human players or human-written computer programs).

3. DESIRABLE ATTRIBUTES OF THE PURSUIT OF HUMAN-COMPETITIVENESS

The pursuit of producing human-competitive results by means of genetic and evolutionary computation is a worthy goal on the grounds of utility, objectivity, complexity, and interminability.

3.1 UTILITY

Arthur Samuel's vision for the field of machine learning matches that of the founders of the field of artificial intelligence, namely the automatic creation of computational techniques that are able to solve problems in a human-like way. The goal of both of these fields is to augment the capacity of humans to solve problems and to extend the range of complexity of problems that can be solved.

The augmentation of human capacity is especially relevant in fields where massive amounts of primary data requiring examination, classification, and integration is accumulating in computer readable form. Examples include biological DNA and protein sequence data, astronomical observations, geological and petroleum data, financial time series data, satellite observation data, weather data, marketing databases, and the universe of web pages, e-mail messages, news stories, and communication messages.

The extension of problem complexity includes examples where conventional techniques of mathematical analysis are unable to solve evermore complex real-world problems. The field of design provides numerous problems. Design is a major activity of practicing engineers. Engineers are often called upon to design complex structures (e.g., controllers, circuits, antennas, networks of chemical reactions) that satisfy certain prespecified high-level design goals. The design of a complex structure typically involves tradeoffs between a number of competing considerations. The end product of the design process is usually a satisfactory, as opposed to a perfect, design.

3.2 OBJECTIVITY

In attempting to evaluate an automated problem-solving method, the question arises as to whether there is any real substance to the demonstrative problems that are published in connection with the method. Published demonstrative problems are often contrived toy problems that circulate exclusively inside academic groups that study a particular methodology, but have no relevance to any actual work that is being done in any field of science or engineering.

As will seen from table 9.1, the eight criteria for human-competitiveness have the desirable attribute of being at arms-length from the fields of artificial intelligence, machine learning, and genetic and evolutionary computation. That is, a result cannot acquire the rating of "human competitive" merely because it is endorsed by researchers inside the specialized fields that are attempting to automate the problem-solving process. Instead, a result produced by an automated method must earn the rating of "human competitive" independent of the fact that it was mechanically generated. The earning of this rating from the outside confers objectivity on the rating.

Thus, for example, an automated method that produces a proof for a problem in algebraic topology would be considered "human-competitive" if algebraic topologists regard the theorem as a publishable result in their field. On the other hand, an automated method that solves a toy problem (e.g., the towers of Hanoi, block stacking, cannibals and missionaries, exclusive-or) would not be considered "human-competitive" because the solution is not publishable in its own right as a new scientific result (and is, in fact, only of interest because it was mechanically created).

3.3 COMPLEXITY

Pursuing producing human-competitive results necessarily leads one toward addressing complexity (and away from toy problems).

To give one example, systems with which real world engineers and scientists work typically contain massive regularity, symmetry, homogeneity, and modularity. For example, non-trivial analog electrical circuits almost always contain

multiple occurrences of certain subcircuits (e.g., Darlington emitter-follower sections, current mirrors, cascodes, voltage divider subcircuits). At a higher level, analog circuits often also contain multiple occurrences of various more complex entities, such as filters, op amps, oscillators, voltage-controlled current sources, and phase-locked loops. Similarly, digital circuits almost always contain multiple occurrences of certain standard cells. And, digital circuits often also contain multiple occurrences of higher-level entities (e.g., counters, registers, multiplexers). The design of large circuits would be considerably more difficult (and perhaps even impractical) if the designer had to separately think through the design of each subcircuit from the first principles of electronic design on each occasion when it is needed. Reuse enables the designer to solve a particular problem once and, thereafter, simply reuse the already-learned solution.

However, in spite of the manifest importance of reuse in solving problems in many fields, problems exhibiting reuse are historically virtually absent in the in the literature of automated problem-solving methods.

Recalling our own work on genetic programming, the pursuit of human-competitive results forced us to focus very early on the importance of reuse in producing scalable automated problem-solving. This focus on producing human-competitive results led to the development of concepts such as automatically defined functions (a way to implement subroutines in genetically evolved computer programs) as described in a paper entitled "Hierarchical Automatic Function Definition in Genetic Programming" at the 1992 Foundations of Genetic Algorithms Workshop (Koza, John R., 1993). However, given the computational resources available in 1992, this paper merely contained a theoretical discussion about how to implement this mechanism accompanied by a solution to the "toy" even-parity problem. (See also (Koza, John R., 1992) and (Koza, John R., 1994)). Later work, such as the 1999 book *Genetic Programming III: Darwinian Invention and Problem Solving* (Koza, John R. et al., 1999) and the 2003 book *Genetic Programming IV. Routine Human-Competitive Machine Intelligence* (Koza, John R. et al., 2003) demonstrated that this automated mechanism for reuse could actually deliver human-competitive results in a variety of fields.

The point is that, in 1992, it was the (distant) driving goal of producing human-competitive results that motivated our study of the regularity, symmetry, homogeneity, and modularity inherent in many non-trivial problems. This study, in turn, led to the development of mechanisms for automated reuse. Absent the driving goal of producing human-competitive results, mechanisms for automated reuse might not have been pursued at all. Thus, we believe that the pursuit of the complexity inherent in human-competitive results can act as one (but, by no means, the only) compass for future extensions to the theoretical foundations of genetic and evolutionary computation.

3.4 INTERMINABILITY

Pursuing producing human-competitive results is an inherently interminable process.

It is unlikely, in the foreseeable future, that any automated method will either match the subtlety of human-produced results or duplicate the breadth of areas where human intelligence is applied.

Thus, as each human-competitive result is produced, additional evermore challenging human-competitive problems will appear on the horizon.

4. HUMAN-COMPETITIVENESS AS A COMPASS FOR THEORETICAL WORK

A small percentage of theoretical work arises solely from intellectual curiosity. A larger percentage of theoretical work is done simply because a researcher happens to be facile in a certain technique and "looks for the keys near the lamppost."

The significant theoretical work in a field arises from a shrewd assessment of which issues have the capability of advancing the field and which are mere academic curiosities. While it is sometimes fashionable to say that theoretical work requires no justification at all, all theoretical work is not, in fact, equal. The inequality becomes manifest if reviews some of the seminal examples of theoretical work in the field of genetic and evolutionary computation.

For example, Holland shrewdly perceived in *Adaptation in Natural and Artificial Systems* that the genetic algorithm is a massive parallel competition among schemata (the "combination of genes" that Turing spotted in his 1950 paper) and that the genetic algorithm searches for ever-better schemata in an arguably near-optimal way (Holland, J. H., 1975). The emphasis of this entirely theoretical book is on explaining, arguing for the relevance of, and proving theorems supporting this central theme. The book mentions, in passing, dozens of other theoretical issues and curiosities relating to the genetic algorithm. However, instead of proving theorems or otherwise dwelling on these side issues, they are simply noted.

As a more recent example, Goldberg's *The Design of Innovation: Lessons from and for Competent Genetic Algorithms* (Goldberg, 2002) is guided by an overarching goal of making genetic and evolutionary search scalable, more efficient, and "competent." It is a theoretical work (augmented by numerous well-chosen experiments), but it is theory that squarely addresses specific issues that manifestly have the ability to advance the field.

5. RESEARCH AREAS SUPPORTIVE OF HUMAN-COMPETITIVE RESULTS

The continuing generation of evermore important human-competitive results relies on progress in three areas of research: multiobjective optimization, parallel computing, and the development and perfection of competent genetic and evolutionary search methods.

First, because the solution to the vast majority of human-competitive problems involve subtle and complex combinations of competing considerations, the techniques of multiobjective optimization indexOptimization!multiobjective (Zitzler, Eckart et al., 2001) are likely to be increasing important in the pursuit of human-competitive results.

Second, because the solution to human-competitive problems often requires relatively large expenditures of computational resources, techniques for efficient parallelization (Cantu-Paz, 2000) are likely to be increasing important in the pursuit of human-competitive results.

Third, most importantly, the recognized deficiencies in current genetic and evolutionary search methods need to be addressed along the lines set forth in *The Design of Innovation: Lessons from and for Competent Genetic Algorithms* (Goldberg, 2002).

6. PROMISING APPLICATION AREAS FOR GENETIC AND EVOLUTIONARY COMPUTATION

Since its early beginnings, the field of genetic and evolutionary computation has produced a cornucopia of results.

Methods of genetic and evolutionary computation may be especially productive in areas having some or all of the following characteristics:

- where the interrelationships among the relevant variables are unknown or poorly understood (or where it is suspected that the current understanding may possibly be wrong),
- where finding the size and shape of the ultimate solution to the problem is a major part of the problem,
- where large amounts of primary data requiring examination, classification, and integration is accumulating in computer readable form,
- where conventional mathematical analysis does not, or cannot, provide analytic solutions,
- where an approximate solution is acceptable (or is the only result that is ever likely to be obtained), or

- where small improvements in performance are routinely measured (or easily measurable) and highly prized.

7. ACKNOWLEDGEMENTS

The authors especially thank David E. Goldberg of the University of Illinois at Urbana-Champaign for comments that aided in focusing and refining this chapter and to the editor, Anil Menon, for numerous helpful and perceptive comments on earlier drafts of this chapter.

REFERENCES

Cantu-Paz, E. (2000). *Efficient and Accurate Parallel Genetic Algorithms.* Kluwer Academic Publishers, Boston.

Goldberg, D. E. (2002). *The Design of Innovation: Lessons from and for Competent Genetic Algorithms.* Kluwer Academic Publishers, Boston.

Holland, J. H. (1975). *Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control, and Artificial Intelligence.* University of Michigan Press, Ann Arbor, MI. Second edition. Cambridge, MA. The MIT Press 1992.

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

Koza, John R. (1993). Hierarchical automatic function definition in genetic programming. In Whitley, D., editor, *Foundations of Genetic Algorithms,* volume 2, pages 297–318. Morgan Kaufmann Publishers, San Mateo, CA.

Koza, John R. (1994). *Genetic Programming II: Automatic Discovery of Reusable Programs.* MIT Press, Cambridge, MA.

Koza, John R., Bennett III, Forrest, H., Andre, David, and Keane, Martin A. (1999). *Genetic Programming III: Darwinian Invention and Problem Solving.* Morgan Kaufmann, San Francisco, CA.

Koza, John R., Keane, M. A., Streeter, M. J., Mydlowec, W., Yu, J., and Lanza, G. (2003). *Genetic Programming IV: Routine Human-Competitive Machine Intelligence.* Kluwer Academic Publishers.

Samuel, A. L. (1983). AI: Where it has been and where it is going. In *Proceedings of the Eighth International Joint Conference on Artificial Intelligence,* pages 1152–1157, Los Altos, CA. Morgan Kaufmann.

Turing, A. M. (1948). Intelligent machinery. Reprinted in Ince, D. C. (editor). 1992. *Mechanical Intelligence: Collected Works of A. M. Turing.* Amsterdam: North Holland. Pages 107-127.

Turing, A. M. (1950). Computing machinery and intelligence. *Mind,* 59:433–460. Reprinted in Ince, D. C. (editor). 1992. *Mechanical Intelligence: Collected Works of A. M. Turing.* Amsterdam: North Holland. Pages 133-160.

Zitzler, E., Deb, K., Lothar, T., Coello, Coello, C. A., and David, C., editors (2001). *Evolutionary Multi-Criterion Optimization, First International Conference, EMO 2001*, Zurich, Switzerland. Lecture Notes in Computer Science. Volume 1993, Springer-Verlag: Berlin, Germany.

This page intentionally left blank

Chapter 10

CASE BASED REASONING

An Evolutionary Computation Perspective

Vivek Balaraman
Artificial Intelligence Group
Tata Research Development & Design Centre, Pune, India
vivekb@pune.tcs.co.in

Abstract Case-Based Reasoning (CBR) is the model of human problem solving using prior experiences (called cases). A case memory is a learning environment where new cases are being injected, existing cases getting purged and yet others adapted to fit situations in response to environmental pressures. As experience accumulates, case memories ideally progress incrementally towards expertise. At present however there is the lack of a framework to understand the nature of this progression and how various factors influence it. In this essay, using the analogy with natural selection, we cast case memories as evolutionary systems to see whether the perspective affords us any insights. We examine case memory processes in the light of their evolutionary role and enquire into the nature of search in evolutionary case memories (ECM). We show that while there are several unresolved questions, the perspective affords us interesting speculations and observations. As an extended application of the ECM perspective we examine whether the evolution of cases can lead to abstract knowledge levels. There is evidence that experiential knowledge may not suffice to achieve higher levels of task expertise but may require abstract knowledge structures such as schema. However, little is currently known about how schema come into existence. We explore the intriguing possibility that schema may evolve from cases as an evolutionary operation. The essay also raises a number of research problems that can be attempted by both CBR and Evolutionary System communities.

1. INTRODUCTION

Case-Based Reasoning (CBR) is a computational model of instance based human problem solving. Derived from Schank's theory of dynamic memory (Schank, 1982), CBR has been used with some success in a variety of application areas such as call centre resolution (Nguyen et al., 1993), estimation (Bisio and Malabocchia, 1995), (Gonzalez and Laureano-Ortiz, 1992), experi-

ence management (Bergmann et al., 1999a), (Khemani et al., 2002) and cognitive information retrieval (Balaraman et al., 2003). Research has focused on improving the engineering of the technology by examining facets such as retrieval algorithms (Borner, 1998), (Chakraborti and Balaraman, 2003), organization (Burkhard and Lenz, 1996), maintenance (Leake, 1996), (Smyth and McKenna, 1999) and development methodologies (Bergmann et al., 1999b), (TRDDC, 2000).

However, an area which needs to be better understood in a case memory is it's temporal locus in terms of problem solving ability and how this locus comes to be as a combination of internal processes and external environmental effects. Change is a continual presence in case memories. New cases are injected as previously unencountered experiences, unfit cases get purged and existing cases are adapted to fit problem situations. Every problem solving interaction may lead to changes in case memory. The changes may be a mere strengthening of a case's success factor or weakening of another but over time, these changes lead at a case level to survival of some cases at the expense of others and globally to an improvement in the competence of the case memory. Each process in the case memory also biases the evolution of the case-base. Because these changes are incremental and the factors that lead to them implicit in the case memory processes, their cumulative influence is not well understood. A framework is required to understand the nature of change in case memories and how this affects the behavior of the case memory over time. Such a framework can also be used to understand the biases and limitations of current processes and enable the creation of better problem solvers.

In this essay we cast case memories as evolutionary systems to see if this perspective helps us understand the temporal behavior of a case memory. We explore the conventional model of case-based systems, recast it to an evolutionary model (which we call Evolutionary Case Memory or ECM) and view it from this changed perspective. We discuss how this view can help us to understand the evolutionary role played by each case memory process and while there are several unresolved questions, yet lead us to interesting observations and speculations. We also examine the evolution of abstract knowledge structures (called schemas) from cases. There is evidence that experiential knowledge alone is insufficient to lead to higher levels of expertise and that schemas are required. There is however little understanding of the process by which schemas come into existence. We explore schema creation as an evolutionary exercise over cases. The approach borrows ideas from the very different notion of schema in genetic algorithms. This discussion too leads to some interesting insights and open questions that can be taken up for further exploration.

A small caveat. Given the complexity of case memory operation, it would be unrealistic to expect an exact mapping between the concepts and operators of evolutionary algorithms and case memories. The analogies are meant to be interpreted in the spirit of Bunge's cautious optimism:

> "There is no question that analogy can be fruitful in the preliminary exploration of new scientific territory, by suggesting that the new and the unknown is, in some respects, like the old and the known. If B behaves like A in certain regards, then it is worthwhile to hypothesize that it does so in other respects as well. Whether nor not the hypothesis succeeds, we shall have learned something, while nothing will have been learned if no hypothesis at all had been formulated…The question is to decide what to stress at a given stage of research: whether resemblance or difference should be emphasized." (Bunge, 1968).

Some Clarifications: a) While this essay samples existing work in the very large space of CBR and EA, it was not intended to be a comprehensive survey but merely indicative of the work underway. We apologize in advance for any omissions. b) Literature refers to CBR both as a cognitive principle and it's computational implementation. In order to ground the discussion in this essay we view CBR as a computationally implementable model of a cognitive principle. c) While most CBR systems share some characteristics, there are also several exceptions. As referring to exceptions at each point of the discussion would lead to the argument getting lost in a morass of conditionals, we omit such references. Thus, when we refer to CBR systems, it should be taken to mean typical CBR systems.

2. CASE-BASED REASONING

CBR (Schank and Riesbeck, 1987) is derived from a long sequence of work by Roger Schank's group that began with the work on Conceptual Dependency (CD) (Schank, 1972), moved on to the world of scripts (Schank and Abelson, 1977) and culminated in the theory of dynamic memory (Schank, 1982). The overall goal was to build automated systems that would be able to 'understand' and answer non-trivial queries about textual episodes such as newspaper reports or stories. An interesting sidelight of this sequence is due the fact that unusually with AI theories, the progression to a new theory was dictated by experimental evidence that pointed to the cognitive incompleteness of the preceding theory. Thus deficiencies in CD[1] (Schank and Abelson, 1977) prompted the scripts model, whose limitations in turn(Schank, 1982) led to the Dynamic Memory (DM) model. CBR may be considered a subset of the DM model focused on experiential problem solving. Problem solving is modeled as a search for and application of the best fitting prior instance. Such problem solving using cases is typical in ill structured domains such as legal reasoning, disaster management and help desk call resolution (Kolodner, 1993), (ICSR, 1995-2002). However, while CBR may be most useful in such domains, it has also found wide applicability in classical AI domains such as estimation, design, planning and diagnosis (ICSR, 1995-2002).

[1]The implicit knowledge problem, discussed in further detail in section 5.1.

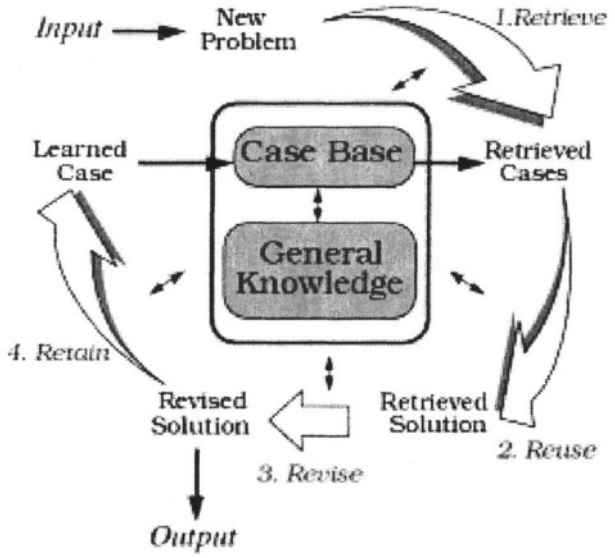

Figure 10.1. CBR problem solving process

The basic components of current case-based reasoning systems are, the case-base, a similarity computation engine that maps and aligns a new situation with the case- base to identify the set of matched cases, the adaptation engine to transform a retrieved case to fit it to the current problem requirements and a learning engine that maintains the case-base.

The case-based process of problem solving works as follows (see Figure 10.1). The new problem (called the input case) is compared to the cases in the case-base by the matching engine. This yields a set of closely matching cases. This is similar to the cognitive notions of *Alignment & Mapping* (Forbus et al., 1991), (Holyoak and Koh, 1987) where the current situation is aligned with prior knowledge and the aligned features are then mapped to each other for correspondence. The adaptation engine adapts the retrieved case(s) to fit the requirements of the input case. The [adapted] cases are then applied to the problem. This is similar to the cognitive notion of *Transfer* (Forbus et al., 1991) where the learning of the past is transferred to understand the current situation. The learning engine then decides whether the input case is worth adding to the case-base or whether an existing case is to be updated. The key engineering aspects of a case-based system are:

1 Case knowledge acquisition: Often the biggest hurdle in developing knowledge based systems is acquiring the knowledge. While this problem is considered to be simpler than with more abstract structures such

as rules, it remains a vexing issue (TRDDC, 2000). Raw case knowledge exists in a variety of unstructured forms and is often incomplete or full of noise. Converting this to a form that can be reasoned over continues to be a big problem especially in multi-expert, multi-user scenarios.

2 Organization of the case-base: The structure of the case-base indicating how the cases are stored with respect to one another. Case-base organization has to solve conflicting requirements. From the point of speed of retrieval, the best organization is each case existing in a separate compartment, which guarantees high Precision. However from the point of view of Recall[2], the best organization is all the cases heaped in one container. Choosing an organization so that these conflicting goals are maximally satisfied is a challenge.

3 Representation and indexing of the cases: Representation is concerned with the structure and content of each case in the case-base. The key issues in representation are: structure wise it must include all significant aspects of the actual case and second, content wise the modeling must be adequate[3]. Failure in either lead to low precision and recall. Indexing is the selection of those features of the case that will be used to retrieve the case. Choice of wrong indexes would lead to the case not being retrieved when it should and being retrieved when it should not. Index selection to ensure high precision and recall is again a challenge.

4 Similarity measure: The similarity measure is the metric that computes the correspondence between the input case and the cases in the case-base. The similarity measure is the primary way of assessing the quality of the case-based system. As with the other parameters, the choice of the wrong metric would have a direct effect on precision and recall. Most similarity measures provide an element of parameterization to enable tuning of the measure to obtain high precision and recall. The most common similarity measures are nearest neighbor/distance measures (Balaraman and Vattam, 1998). For such measures a lot of work has focused on optimal weight learning(Kolodner, 1993), (Kelly and Davis, 1991).

5 Adaptation: The process by which cases are adapted. Adaptation can be *One-case* in that a single case is adapted to fit the new situation or *Many-case* where the case proposed is synthesized from elements drawn from several cases (Kolodner, 1993), (Leake, 1996).

[2] Precision and Recall are metrics borrowed from Information Retrieval. Precision measures the percentage of retrieved cases that are relevant while Recall measures the percentage of relevant cases that are retrieved.
[3] This essentially means choosing the right type.

6 Maintenance: The process by which new cases are added, old cases updated or deleted. Reckless addition of new cases could lead to a large sized case- base with high redundancy and long retrieval times, while the other extreme of no addition would lead to a static system that cannot solve newer problems. The challenge of maintenance therefore is to ensure that redundancy is reduced while ensuring maximal competence of the existent cases.

This section has necessarily simplified the issues in case- based reasoning. Those wanting more information can refer to the introductory books by Schank and Riesbeck (Schank and Riesbeck, 1987), Kolodner (Kolodner, 1993), newer books on the subject by Leake (Leake, 1996) and Lenz (Lenz et al., 1998) as well as the numerous conference proceedings (ICSR, 1995-2002).

3. CASE MEMORY AS AN EVOLUTIONARY SYSTEM

Why should a case memory be viewed as an evolutionary system and what do we hope to gain from it? There are at systems least four reasons:

Viewing learning as evolution: Case memories are learning, adaptive environments which seek over time to maximize their problem solving capabilities which in experiential memories is usually a function of the case- base size. This maximization however has to take into account time response demands which act to limit the growth of the case-base. As a consequence of these opposing forces, cases in a case memory have to compete for inclusion and retention. Since the latter decision is based on task performance, cases are also in competition for retrieval and selection. Thus while case memories are learning systems, the learning of the new might be at the expense of the old. There thus appears to be a similarity to evolutionary processes. A reason therefore for viewing case memories as evolutionary systems is to model learning as evolution. This is by no means an isolated perspective. Earlier speculations such as GA classifiers, Genetic Machine Based Learning systems (Goldberg, 2000) and the Copycat system of Hofstadter have adopted similar perspectives.

To understand the processes of change: As said earlier, there is a lack of a framework to model or understand the nature of change in case memories and how this influences the performance of these systems over time. That there is change is indisputable. But there is no framework within which the relationship between the factors that drive change in the case-based system, the changes in the case- base and the performance of the case memory as a whole can be studied. Casting case memories as evolutionary systems will allow

change to be studied within a framework and with a robust set of formalisms.

To understand biases and limitations: It should not be forgotten that CBR is still a very nascent technology. Proposed in 1987, CBR has seen intensive use only over the last decade. Perhaps naturally, the application domains have tended to be those with an immediate need for such problem solving. Since the technology has followed the need, current case-base processes may have biases built into their functioning that are suited for only a certain class of domains. These as well as limitations may get revealed by casting them in a model where the cumulative influence of each process on the performance of the whole can be studied. As an example, case selection strategy tends to be elitist in current systems. The best seeming case is applied. But an elitist strategy may not be right in all domains. There is perhaps a need for studies on the lines of De Jong's in function optimization to understand the relation between various case-base processes and their influence on case memory both at a snapshot in time and cumulatively over a period.

To open up new possibilities: An improved understanding can lead to improvements in the technology which in turn may lead to opening up of new and interesting application domains.

3.1 A SIMPLE MODEL OF ECM

The model that will be presented below is the case memory process stripped to it's bare essentials. We present each process and entity to understand the plain vanilla process. In the simplest model, there are 3 case memory processes, an external environment and the case-base which contains the population of cases. We view each interaction with the environment as a generation. The environment poses a problem to the ECM. *GenerateSolution* reacts by proposing a solution to the problem by retrieving similar cases and processing them to fit the problem. *Evaluate* gives a decision on whether the solution was acceptable both in terms of quality and time response and finally *Reorganize* modifies the case-base in the light of the decision. We now examine each entity in the model.

3.1.1 Case-Base. The case-base is the set of cases. We assume each case in the case-base is the pair (p, s) where p is the problem and s is the solution. Let P_t^* be the union of the problem parts of all cases in the case-base at generation t and S_t^* the union of all solutions. Then the case-base at generation t is given as $C_t^* = \{C_j \mid C_j = \{p_j, s_j\}, p_j \in P_t^*, s_j \in S_t^*\}$.

3.1.2 Environment. The environment poses a set of problems $\{p_i\}$ to the case memory and evaluates the solutions suggested by the case

memory. We assume that at the end of each problem solving cycle, the en-

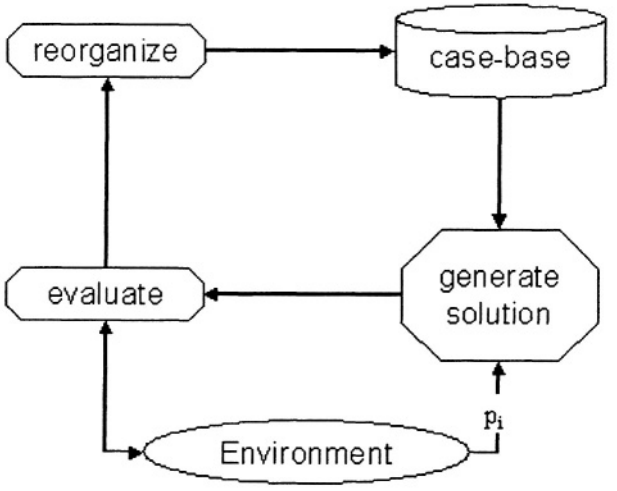

Figure 10.2. Simple model of evolutionary case memory at generation i

vironment gives it's decision on two factors, whether the solution worked and whether the time response was within an assumed performance threshold $T_{\max}$. The decision may be given as the tuple $\{d, o\}$ where d is the decision on solution quality and o is the decision on time performance.

3.1.3 Generate Solution. *GenerateSolution* can be seen as the sequence of processes *Retrieve* $\circ$ [*Select*] $\circ$ [*Adapt*]. The square parentheses indicates that the enclosed process is optional.

The *Retrieve* operator takes a case-base C_t^* and compares it with the new problem p_i to yield a set of relevant[4] cases. *Retrieve* : $C_t^* \times p_i \rightarrow C_R^* \subseteq C_t^*$. Relevance is computed in terms of similarity in problem space. A number of retrieve operators have been proposed among them the nearest neighbor/distance measure, the MAC/FAC model of Forbus (Forbus et al., 1991), feature counting and template matching.

While *Retrieve* retrieves the top R cases on the basis of problem space similarity, there could be other considerations to decide the cases that will actually be selected for application.

Select : $C_R^* \rightarrow C_S^* \subseteq C_R^*$ acts on the retrieved set C_R^* and produces a reordered set that gives the suggested order using one or more selection factors. Sample selection factors are *TaskPer formance* and *SelectionStrategy*.

[4]What relevant means is dictated by the use environment.

Two cases with equivalent similarity scores could vary considerably in their actual solving ability. *TaskPer formance* measures the actual utility of a case in solving problems. A simple task performance measure may be a function over the number of successes and number of failures. *TaskPer formance* takes the cases passed on to it by retrieve and sorts them on problem solving ability. *SelectionStrategy* takes the list of cases passed on to it by the preceding processes and decides the actual case(s) which will be passed on to *Adapt* or applied on the problem. There could be other selection factors.

Adapt takes the set of selected cases, modifies one or more of them for applicability and generates the case(s) that will actually be applied to the problem. *Adapt* is usually driven by the differences between the input problem and the set of selected cases suggested by *Select* and works to modify the cases to bring them in line with the problem. *Adapt* : $p_i \times C_S^* \rightarrow C_A$. So seen as a whole, *GenerateSolution* : $C_t^* \times p_i \rightarrow C_A$, i.e. *GenerateSolution* generates a candidate solution as a response to the problem posed by the environment.

3.1.4 Evaluate. *Evaluate* applies the proposed solution to the problem posed by the environment. The environment decision tuple $\{d, o\}$ reveals whether the case is successful, not successful in solving the problem and whether the time performance was acceptable.

3.2 REORGANIZE

At the end of a problem solving cycle, *Reorganize* : $C_t^* \rightarrow C_{t+1}^*$ decides the actions to be carried out on the case memory based on the performance in terms of success in problem solving and other criteria such as times taken by retrieval and adaptation. *Reorganize* translates the feedback from evaluation to an evolutionary action on the case- base. Some actions based on functions of current case memories are: add new case, add adapted case, delete case, modify case, replicate case and modify, purge case-base, restructure case-base, strengthen case task performance and weaken case task performance. Standard evolutionary operations like crossover and mutation have also been proposed (Soh and Tsatsoulis, 2001).

3.3 DISCUSSION

We presented a simple model of an evolutionary case memory. Some observations based on the model and current case memories.

1 *Retrieve* applies selective pressure both by commission and omission. A case is retrieved only if the problem part of the case matches the problem presented by the environment. A poorly indexed or represented case will get retrieved when it should not and not retrieved when it should.

Secondly, over time, problem distribution might change. Problem distribution is the relation between new problems and the existing problems in the case-base. Even a well indexed case might never get retrieved if no problems arise for which the case is relevant. And finally, if retrieval time is poor, it might signal a need to purge the case- base of unfit or unused cases ((Leake and Wilson, 1999) and (Portinale et al., 1999) may be referred for a discussion on problem-distribution metrics and forgetting strategies).

2 *TaskPer formance* is similar to the notion of Strength discussed in GA classifiers which is based on the contribution of the classifier to successful problem solving. Strength was based on factors like taxes, bid values and receipts. A minimal model of change in *TaskPer formance* would include the current *TaskPer formance* values of the contributing cases and the current decision on success or failure. Let PM_t^S be the performance values of the S cases selected for application in generation t. Then $TaskPer\,formanceChange : PM_t^S \times d \rightarrow PM_{t+1}^S$.

3 *SelectionStrategy* has some similarities to reproduction strategy in GAs. *ReproductionStrategy* decides the individuals that will pass on their genes to the next generation and biases the search process. Similarly, *SelectionStrategy* decides the cases that will actually contribute to solving the current problem and thus biases the problem solving. Current CBR systems use an elitist case selection strategy. The N most similar cases are selected. Since CBR systems can be seen as single generation problem solvers, where a problem is presented in one part of the cycle and a solution proposed in another, using the best available choice may well be the right strategy. Recent work on diversity conscious retrieval (McSherry, 2002) where cases are selected based on solution diversity point to the fact that even in the current spectrum of application domains, different domains require different case selection strategies. There is a need to study what would be optimum case selection strategies in different domains and it's biasing effect on case-base performance.

4 Present case-based systems use a number of adaptation techniques, rules, adaptation formulae, model based adaptation, adaptation cases and even genetic algorithms (Kolodner, 1993), (Leake, 1996), (Gomez et al., 1999).

5 *Adapt* extends the coverage of the case-base but at a cost. With powerful adaptation techniques, even a small sized case-base can handle a large number of problems enabling minimal retrieval times. However, adaptation especially using model based techniques can be time consuming which might negate the time advantage gained by *Retrieve.*

Reorganize has to decide whether the adaptation effort is worth the retrieval time gains or it is better to add the adapted case to the case-base leading to an increase in retrieval time but reduced adaptation effort for a similar problem.

6 At each generation, the solution(s) proposed to *Evaluate* is a single case or a very small set of cases. Thus, only a very small subset of the case-base is evaluated at each cycle.

7 There are 2 processes that lead to change in existing cases, *Adapt* and *Reorganize*. While *Reorganize* has a direct influence, *Adapt* plays a secondary role.

8 *Reorganize* leads to incremental changes on the case-base. Unless triggered by factors like need to purge cases, the fraction of cases affected at each generation are a very small subset of the cases in the case-base. However, since a case memory is used to solve a wide range of problems, the cases affected may be a significant percentage of the cases that are *relevant* in the current problem context.

9 Since only a percentage of the cases in the case-base are affected in each generation ECM may be viewed as evolutionary systems with overlapping populations.

10 In the simplest scenarios, a case may be added without affecting other cases. However, workers such as Portinale (Portinale et al., 1999) propose schemes where for every case to be added, an existing case which is either of low task performance strength or unused is deleted. This appears to have some similarity to the niche crowding strategy suggested by de Jong (Goldberg, 2000). Smyth and McKenna (Smyth and McKenna, 1999) propose coverage checks that will only add cases that have a different solution coverage as compared to existing cases. The intention in both is to have compact yet maximally competent case- bases.

11 *Reorganize* may not just lead to addition, modification and deletion of cases but also to restructuring of the case-base. The case-base may not be a flat structure but have a complex organization. Cases may be clustered around prototypes, organized as a tree or a network. A case indexed under a wrong prototype would need to get correctly classified. A dense uncategorized case cluster may signal the need to evolve a higher level structure. Restructuring can affect both time performance and goodness of retrieval. Restructure is similar to graph transformation.

12 Optimizers can be viewed as evolutionary systems where the question asked by the environment is unchanged at each generation, with the sequence of answers leading incrementally to an optima or near optima.

On the other hand, in ECM, different generations lead to different questions being asked. Thus different segments of the population are evaluated or affected in each generation.

13 While ECM may be evolutionary they are not called on to be optimizers. The solution proposed in each generation to a question is not an optimal but a workable solution. Current use environments do not demand optimality but acceptable solutions within a timeframe.

14 If ECM are evolutionary systems as we assume, then it might be worth asking whether ECM can simulate optimizers. ECM may be viewed as optimizers where the same problem is posed repeatedly over successive generations until the solution is optimal or near optimal (see Figure 10.3). At each generation, *Evaluate* would indicate the goodness of the proposed solution(s).This however calls into question the nature of search in ECM. Since each generation leads to *Adapt* proposing an adapted case C_A to *Evaluate,* exploration of the search space is dependent upon C_A. If we assume that the cases proposed at each generation, $C_{A1}, C_{A2}, \ldots, C_{At}$, are added to the case-base and the case-base reaches optima in generation N, optima can be realized only if the sequence of processes applied successively for N generations on an incrementally expanding case- base lead to it. It should be noted that *Retrieve, Select* and *Adapt* as implemented by current CBR systems are deterministic. It would be interesting to ask what form the four major processes should take to be able to prove that the process will finally lead to optima.

15 Conversely we can also perhaps understand the need for a memory for previous solutions if evolutionary systems are asked to produce a workable solution within a finite time and for a wide range of problem situations. While conventional ES may be capable of producing an optimal solution, the time constraint might force them to do 2 things, cut short the number of generations and examine the seeding and or solution injection strategy. The memory can thus 'kick start' the solution determination process.

16 We assumed in the simple model that the processes themselves are static. I.e. *Retrieve*, *Select*, *Adapt* and *Reorganize* themselves don't change over time. In general, this is not a valid assumption. Each of these processes may also change over time. Adaptation knowledge may grow with time, retrieval parameters such as feature weights and indexes may change (see section 4.2 for a discussion on Type B systems which optimize such parameters) to provide better precision or recall and new selection strategies may be learnt. Evolution may thus not be restricted to the world of cases.

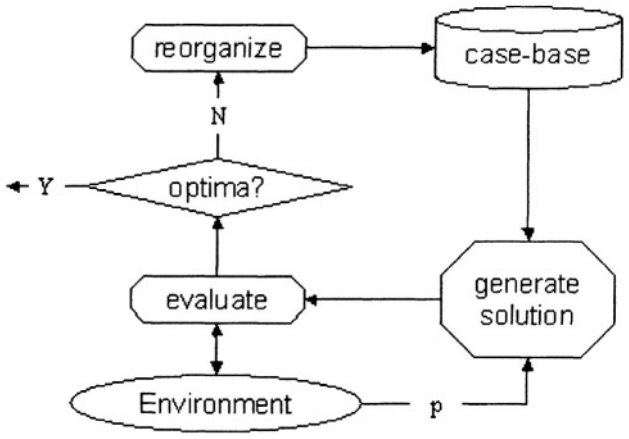

Figure 10.3. ECM as optimizers

Questions and research areas:

[Q1] How do the processes that lead to proposing a solution (*Retrieve, Select* and *Adapt*) individually influence selection pressure and bias the problem solving process?

[Q2] Are there correlates in evolutionary algorithms to the kind of operations carried out by reorganize?

[Q3] Conversely, if we restrict *Reorganize* operations to the set of evolutionary operations in EA, where could such an ECM be used? Section 4 suggests that candidates could be design, configuration and agent based reasoning domains.

[Q4] Current case-based systems use deterministic strategies in their functioning. This may be largely due to the fact that current case-based systems are largely used as decision aids where the user is interested in the *N* 'best' choices. There could however be domains where deterministic policies might be ineffective, indeed even counterproductive. There is a need to explore this possibility from two ends, both by developing non-deterministic correlates to these operators as well as exploring domains which might need more complex policies. An example of the latter could be autonomous systems such as agents which are required to satisfy multiple, possibly conflicting goals by taking a series of decisions in a possibly adversarial and information incomplete world.

[Q5] How would different combinations of *Retrieve*, *Select*, *Adapt* and *Reorganize* operators influence effectiveness of problem solving and the growth in competence of the case-base? Each operation biases the process of problem solving and consequent change in the case-base in a certain way. If $Retrieve*$ is the set of possible retrieve operations and similarly $Select*$, $Adapt*$, $Evaluate*$ and $Reorganize*$ are the set of select, adapt, evaluate and reorganize operators, it would be interesting to study the performance of the case-base and it's progress towards expertise with different combinations of operators. Indeed, probably the biggest gain in viewing case memories as evolutionary systems is to see each process not in isolation but how their cumulative effect over time biases the ECM in a certain direction.

[Q6] How can ECM be made to simulate optimizers? Exploring this question will allow us to understand the different ways an ECM can explore a search space. We may also gain insight into how task competence grows in dense problem distribution spaces. Different choices of arriving at C_A as well as different choice of reorganize operators might realize different search techniques. Which of them are cognitively valid and also computationally useful is a different problem which also needs study.

This simple model of the ECM has provided a number of interesting observations as well as areas requiring further exploration. We now take a look at some previous work on intermingling case-based reasoning and evolutionary algorithms.

4. HYBRID SYSTEMS

There has already been a fair amount of work on hybrid systems that combine case-based and evolutionary systems. These systems can be characterized as Loosely Coupled. In Loosely coupled systems, a case-based system operates on a case-base and carries out case retrieval and incremental maintenance. Coupled to this is an EA system that operates on an optimization space. Each system works on a space suited to it's individual strengths but yet has a complementary problem solver that remedies the weaknesses. We classify loosely coupled systems into two rough categories: Type A systems use the CBR system primarily as a memory and use EA as the optimizer. Given a problem the CBR system retrieves a set of related cases which are used to seed the EA system or inject solutions at intermediate stages of evolution of the EA. Type B systems use the EA mechanism to optimize the parameter settings of a CBR system. We discuss each in turn.

4.1 TYPE A - CBR AS A MEMORY, EA AS THE OPTIMIZER

Evolutionary Algorithms are theoretically robust, i.e. convergence is not dependent on the distribution of the initial or intermediate populations. While there is no explicit theoretical dependence of the convergence quality or rate on the composition of the population, work such as by Louis on combinatorial logic circuit design (Louis, 2002) show that in practice, choice of distribution can play a part in speed and quality of convergence. Studies by Bohm and Geyer-Schulz (Bohm and Geyer-Schulz, 1996) show that there are biases built into the generation of random populations (that is, random initialization may not achieve *uniform* initialization) that affect performance adversely. One solution to improving performance may lie in coupling an EA system with a CBR system containing prior good solutions (see Figure 10.4).

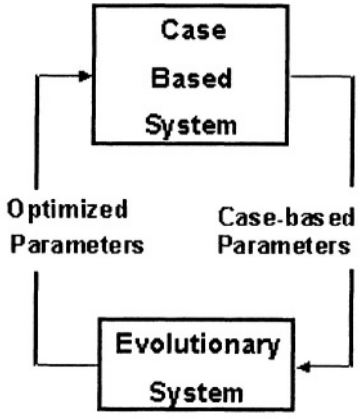

Figure 10.4. Type A: EA Using CBR

Louis has used a CBR system to provide not initial populations but to inject solutions at intermediate stages of evolution. The resultant system CIGAR considerably outperformed a system using random population initialization and injection. Since the CBR system is also used to store solutions generated by the EA system, the CIGAR system as a whole is a learning system whose performance improves over time. As Louis puts it, "*These performance gains imply that fewer evaluations are required to reach a certain design quality and the organization deploying this system builds a knowledge base of cases.*" Perez (Perez et al., 2001) have also used a CBR coupled to a GA for combinatorial logic design. This work's interest lies in their combining CBR and GA in two ways: CBR is used to provide seed populations to the GA mechanism as in CIGAR. Additionally, cases are also mined to extract highly fit building blocks that occur across a set of cases. We will return to this latter aspect in section 5.3.1. Another interesting system is by Soh and Tsatsoulis (Son and Tsatsoulis, 2001) who use a GA to seed and maintain a case-base. The domain is agent based resource scheduling and goal satisfaction. Agents negotiate with one another for access to shared resources to best satisfy global or local goals. Cases are negotiation strategies that an agent uses either for initiating a request or responding to a request. The GA component maintains a case evolution trace in the form of

a hierarchy. Cases evolve in the hierarchy based on intrinsic and environmental fitness. Unique and fit cases are promoted to the case memory. The GA acts like a case nursery and the CBR environment as the test bed to test the utility of cases so produced. Cases found to have low utility are removed from the case memory but retained in the hierarchy since under different environmental situations they might become useful again.

From a CBR perspective, EA are seen as optimizers of solutions in a CBR system. Since CBR systems make no claim to optimality of solutions stored in the system, an EA system can be used to combine one or more sub-optimal cases to create optimal or near optimal solutions. Thus an EA system can also be considered to be a multi-case adaptation mechanism as the final solution could have elements from multiple cases. An illustrative example is the work by De Silva and Maher (Gomez et al., 1999) who use a Genetic Algorithm to optimally design an architecture that conforms to Feng Shui principles. The work can be seen either as a GA system using cases to provide initial populations or a rare case-based system that can use a GA as a multi-case adaptation mechanism. Viewed as the latter, parametric adaptation of cases is achieved through mutation and structural adaptation through crossover.

4.2 TYPE B - EA AS CBR SYSTEM PARAMETER OPTIMIZERS

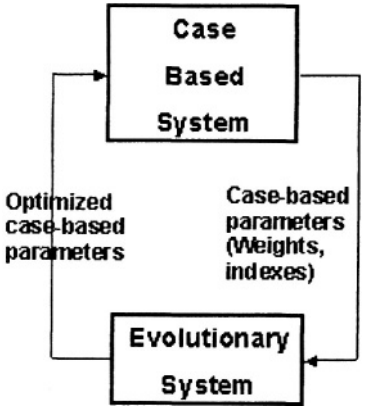

Figure 10.5. Type B: CBR Using EA

CBR systems are typically used in ill structured, ill bounded domains where finding any solution is considered good. Optimality of solution is not a criterion. A domain that illustrates these characteristics is problem resolution in call centers and help-desks (Nguyen et al., 1993). Metrics are often simply whether or not the problem was solved. Solutions also tend to be textual making it difficult to assess goodness without extensive semantic analysis.

EA on the other hand are applied in hard well bounded domains such as optimization in engineering or manufacturing. Such domains have 2 important characteristics, viz., solutions or solution parameters are expressible in enumerated or quantitative terms and second, solutions can be objectively measured for their goodness using formal functions (such as an objective or fitness function). These functions can *a pri-*

ori accurately determine the effectiveness of a solution without actually trying it out. Such functions are difficult to specify in CBR application domains. The exception to this are CBR applications in domains like configuration or design, an example of which we examined earlier (Gomez et al., 1999).

Thus, use of EA by the CBR community has been largely focused on improving the Quantifiable aspects of case- based systems (see Figure 10.5). These are usually the weights or "importances" of case features and the selection of the best indexes. The much referenced Kelly and Davis (Kelly and Davis, 1991) was the first to discuss how a GA could be used to decide on feature weights[5] of a CBR system. Shin and Han (Shin and Han, 1999) use a GA to decide on the best indexing features for a case-based system for stock market prediction. An EA approach is feasible because the parameters to be optimized (presence or absence of a feature as an index, the feature weights for a KNN type similarity measure) can be expressed as an enumeration, a binary value or a numeric.

4.3 DISCUSSION

The following observations can be made:

1 A significant difference between ECM and systems such as CIGAR is the fact that in ECM, the case memory itself is modeled as an evolutionary process while in most Type A systems, the CBR component serves merely as a store house of cases and the GA component does the bulk of the processing. While CIGAR's GA component does store intermediate solutions in the case memory and similarly requests cases for injection, the interaction between the two components is intermittent and not cyclic as in ECM. The extreme situation may be where the CBR component provides merely the initial seed to the GA component and there is no further interaction until the latter has finished cranking out the optima.

2 From an ECM perspective the system proposed by Soh can be viewed as *Retrieve*, *Select*, *Adapt* and *Evaluate* being carried out by the CBR component and *Reorganize* by the GA component. Classical evolutionary operators like crossover and mutation are used to evolve new cases in contrast to most CBR systems. In a sense, the *functionality* of the ECM as proposed in the earlier section may be said to be realized in this implementation. The system as a whole evolves incrementally and grows in competence over time.

3 The Soh system discusses a number of operations that change the case-base in evolutionary terms such as *Evolutionary Incompatibility* that is

[5]Good feature weights lead to good precision and recall.

in effect a case forgetting strategy based on case utility, *Evolutionary Enhancement* that seeks to compensate for changing problem distribution, *Evolutionary Refinement* that is a form of case replication with modification and *Evolutionary Breakthrough* that discusses how new cases are handled and propagated. It is interesting that many of the operations briefly listed under *Reorganize* can be recast as evolutionary operations.

4 The system proposed by de Silva and Maher can be viewed as *Retrieve* being carried out by the CBR component and *Adapt* by the GA component. However, *Adapt* is in effect an optimizer which bundles *Evaluate* within it's cycle. There is no indication whether the system has a *Reorganize* component to learn new designs.

5 If the case-base stores the optimized or near optimized solutions generated by the EA solver in Type A systems, an issue may be lack of divergence in the stored cases for a class of problems. The cases may be clustered in a single region which used as seeds might lead to quick but sub-optimal convergence. While this problem has been recognized and handled by Louis (Louis, 2002) it may be beneficial to refer to CBR work on a similar theme. There has been a lot of recent attention on Diversity-conscious retrieval(McSherry, 2002), (Bradley and Smyth, 2001). These approaches lead to the CBR system recommending cases that are maximally diverse but similarity preserving. Type A systems might benefit from experimenting with these approaches. Alternately, at the time of adding cases to the case-base for a given problem context, a diversity check among the solutions of the candidates for addition may be made. Only cases whose solutions have a minimum diversity may be added.

Based on our discussion on ECM, some questions and areas requiring further exploration:

[Q7] While we have given rough mappings between different Type A systems and ECM, it will be interesting to study how hybrid CBR and EA operations as seen in Type A systems can be equivalently realized within the framework of ECM.

[Q8] What changes would need to be made to the ECM model to realize Type B operations? We assumed in the discussion on ECM that the processes themselves are static and don't change over time.

5. EVOLVING HIGHER LEVELS

The ECM model presented in section 3 only discussed experiential knowledge or cases. But there is considerable evidence that human problem solving uses a number of non-instance knowledge structures such as rules or schema. But where do rules or schema originate? While there could well be hard coded schema templates for certain cognitive tasks like language acquisition, it is too much to expect such templates to exist for tasks like eating at a restaurant or opening a bank account. Occam's Razor indicates that experiences being the most common knowledge currency, higher level structures could be abstractions over cases. In this section, we explore how higher level structures can be viewed as an evolution over cases. But before we delve into that, we take a look at what higher level structures are and the role they play in cognition and human problem solving.

5.1 SCHEMAS

How is human knowledge organized? Early workers such as Bartlett conjectured knowledge only as abstract prototypical structures called schema and disallowed instance based problem solving(Brewer, 2000). However, as reported by Smith (Smith et al., 1992), a number of current cognitive theories organize their functioning purely on the basis of experiences with the hypothesis that abstract structures are implicit in a collection of experiences and no explicit higher level structures are required. However, neither extreme of pure schema or pure instance appears tenable in the face of cognitive evidence.

Medin and Ross (Medin and Ross, 1989) have surveyed and presented evidence of instance based human problem solving. Similarly Smith, Langston and Nisbett (Smith et al., 1992) presented 8 criteria to decide whether abstract structures such as rules are used in human reasoning and showed that these criteria are met by cognitive evidence. Others such as Neisser (Neisser, 1976), Mero (Laszlo, 1990) have presented arguments to demonstrate that understanding is closely related to having abstract cognitive structures which they have labeled schema.

Correspondingly, in Artificial Intelligence, computational modes of problem solving consist in essence of case-based reasoning (Schank and Riesbeck, 1987), (Kolodner, 1993), the ubiquitous rule based reasoning and script/frame based/model based reasoning (Schank and Abelson, 1977), (Minsky, 1981).

There is thus support from direct cognitive evidence as well as supporting evidence in the form of computational intelligence theories that problem solving exists at multiple levels of reasoning and representation. So what are schema? A schema can be defined as an abstract structure that contains knowledge of prototypical objects or processes encountered in common experience. The notion of schema was first discussed by Bartlett in his classic, *Remem-*

bering (Bartlett, 1932). Bartlett hypothesized that understanding was based on existing knowledge structures possessed by humans which he labeled Schema. While the pure schema theory favored by Bartlett conjectured that knowledge existed **only** in the form of schema, more relaxed theories allow individual experiences to be stored under a more general schema (Brewer, 2000). Example schemas are: Racing-Car-Schema, Fast-Food-Restaurant-Schema, Withdraw-Money-From-ATM-Schema, Traveling-By-Bus-Schema. Each of these would describe the structure of the object or the process of which they are prototypes.

The idea of schema has subsequently found support among numerous workers in cognitive science and artificial intelligence. Though there are slight differences Schema are also called cognitive schemata, scripts and frames (Minsky, 1981),(Carlos, 1997). It is interesting to note that Rich and Waters in their work on automating program development (Rich and Waters, 1990) conjecture template knowledge structures called Clichés. Given a problem, they hypothesized that experienced programmers do not code from first principles but instead compose the solution using 'chunks' of commonly used program fragments labeled Clichés. Clichés seem to be merely another form of schema.

Schank's work provides a good illustration both of the nature of episodic schema (called scripts) and the need for such structures. Schank's earlier theory of Conceptual Dependency (Schank, 1972) proved unable to handle the problem of implicit knowledge. Conceptual Dependency or CD was based on the idea of having a semantic representation of text removed from the syntax. CD modeled sentences using a core set of primitives called Actions. Each action was a graphical frame with slots. Understanding was viewed as a slot filling exercise with the unfilled slots of one sentence leading to expectations of the information to come. While CD was successful in certain classes of language to language translation, the main goal namely, understanding proved elusive.

To explain the problem in CD, let us look at the following sentence: *John got on a bus and alighted at his destination.*

A natural language understanding system must on the basis of this one sentence be able to answer questions such as: What is a bus?, Why did John get into the bus?, What did John do in the bus? and Why did John get off the bus?

These questions can be answered only by the use of knowledge outside the information communicated in the sentence. Since CD lacked such external knowledge, it was unable to simulate the understanding process. This lacunae in the CD theory led Schank to conjecture about knowledge structures called scripts (Schank and Abelson, 1977). A script was a pattern describing a commonly occurring episode. Thus a script Traveling-by-bus might have the following ordered sub-events: Wait for the bus, Get on the bus, Calculate fare from entry point to destination, Pay fare, Sit if seats are available and Alight at destination. In addition to the sub-events in the script, the script also contained

information such as pre-conditions for script application, the roles of different entities in the script, the objects (called props) that figure in the script and the expected post-condition after script application.

With such a script, the understanding system could now answer some of the questions listed above. In essence, the schema, script theory and their derivations argue that understanding and task expertise is based on a wealth of knowledge structures at different levels. At the bottom are the individual experiences or episodes and at higher levels are schema like structures.

5.2 A BRIEF ASIDE ON LEVELS OF HIGHER EXPERTISE

Studies of chess players (Charness, 1981), (Reynolds, 1982) and nursing professionals (Dreyfus and Dreyfus, 1986) show that there appear to be a well graded path to expertise. The accumulation of mere experience does not lead to expertise. While some argue that there are five levels of expertise and others four, there appears to be agreement that expertise acquisition is not a linear process but can be quantized into levels. Mero (Laszlo, 1990) for example, characterizes expertise into 4 levels:

1 Beginner: Recently introduced to the task. Equivalent to a chess player who just knows the legal moves.

2 Advanced student: Has some task competence. Equivalent to 1 year of experience and learning in the task.

3 Candidate master: Has graduated to a level of excellence. Equivalent to anywhere between 3 to 5 years of experience and learning in the task. It also appears from studies of chess players that the level of candidate master is the maximum level to which knowledge explication is possible. Knowledge beyond this level has been difficult to extract and encode.

4 Grandmaster: The highest level and while competence at this level is not time dependent, might require around 10 years.

The path from beginner to grandmaster also appears to involve two processes (Laszlo, 1990):

1 Increase in the number of schemata: As expertise builds up, the number of schemata increase. Estimates indicate that a beginner has less than 100 schemata, an advanced learner less than 1000, a candidate master about 5000 and a grandmaster about 50,000.

2 Evolution of very complex schemata: In addition to the increase in the schemata, the schemata possessed also appear to grow tremendously in

complexity. Thus a schema of a beginner in chess might have an interpretation of a few positions. A schema at the level of a grandmaster might involve numerous interpretations, reference hundreds of actual games (especially master and higher level games) and winning strategies. Grandmaster schemata also appear to possess intriguing properties such as tangled hierarchies which Mero dramatically likens to the seeming impossibility of Eicher's pictures and Bach's fugues.

Thus not only do the number of abstract structures increase but so does their structural and content complexity. The point of this digression has been to show both the existence of quantized higher levels as well as their complexity. In this essay, we assume there is a single higher level that indexes or subsumes a set of experiences.

5.3 TOWARDS MEMORY BASED REASONING

The most common knowledge currency are experiences, the events encountered in the course of navigating the world. CBR as we have seen is problem solving purely through use of this world. However, the preceding sections indicate that expertise requires a combination of experiences and schema like structures (except possibly in chaotic domains where all knowledge is experiential). The question is, "How do abstract knowledge levels come to be?." While the computational world has solutions in the form of induction algorithms such as ID3 (Quinlan, 1986) and C4.5 (Quinlan, 1993) which given a set of cases would churn out a corresponding set of rules, our concern is to remain true where possible to the cognitive continuum where cases might lead to schema and new cases are in turn indexed under existing schema with both processes leading to incremental modifications of each level (see Figure 10.6). In section 3 we explored the ECM model which discusses case memories as

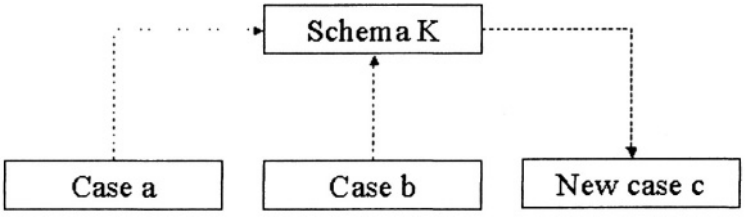

Figure 10.6. Experiences lead to schema which in turn index new experiences

evolutionary systems. We saw how various processes act as selective pressures and lead to incremental evolution in the case-base. There is however no reason why evolution should be restricted to the world of cases. In the discus-

sion on *Restructure* in section 3.3 we discussed the notion that a dense set of cases might require a higher level structure for better differentiability. But how will such a structure come into existence? In the next section we explore the idea that even schema creation can be viewed as an evolutionary exercise over cases. This approach borrows ideas from the very different notion of schemas in genetic algorithms.

5.3.1 C-Schemas as Building Blocks. A schema in genetic algorithms is a very different beast from the schema discussed in the preceding sections. A schema as defined by Goldberg (Goldberg, 2000), is, "*a similarity template describing a subset of strings with similarities at certain string positions.*" To distinguish the two we henceforth call the Cognitive Schema as C- Schema and the Genetic algorithm schema as G-Schema. The notion of a G-Schema was introduced to explain why the seemingly random processes of genetic algorithms yet lead to effectiveness in problem solving. As Goldberg put it, "*Just as a child creates magnificent fortresses through the arrangement of simple blocks of wood, so does a genetic algorithm seek near optimal performance through the juxtaposition of short, low order, high performance schemata or building blocks.*" The assumption in the Building Block Hypotheses (BBH) as it has been labeled is that as the population moves towards higher and higher fitness levels, highly fit gene sequences emerge that can be found in increasing frequencies in the population. Genetic algorithms work, it was argued, because of this process of agglomeration of highly fit blocks. While there have been no unconditional proofs of the BBH, it yet gives a plausible and intuitively appealing reason for the effectiveness of the technique.

We are interested in how C-Schemas emerge from experiences. C-Schemas as we discussed in the previous section are higher level knowledge structures. The direction we explore takes two ideas from Evolutionary Algorithms. The first is to define C-Schemas as abstract patterns operating over cases much as G-Schemas are abstractions over individual chromosomes. The second is to model generation of C-Schemas from cases as an evolutionary process. Let

$$\begin{aligned} C^* &= \{C_1, \ldots, C_i, \ldots, C_n), \\ \text{where, } C_i &= v_{i1}v_{i2}\ldots, v_{im}, \quad \text{and} \quad v_{ij} \in [0, 1]. \end{aligned}$$

In other words, we define each case as a bit string. Define the set of C-Schemas,

$$\begin{aligned} S^* &= \{S_1, \ldots, S_i, \ldots, S_p\}, \\ \text{where, } S_i &= y_{i1} \ldots y_{ij} \ldots y_{im}, \quad \text{and} \quad y_{ij} \in [0, 1, *]. \end{aligned}$$

We assume that experiences are encoded as bit strings. Each C-Schema is a pattern based on the ternary alphabet of [0,1, $*$] that subsumes a set of experiences much like G- Schemas are defined. However, while G-Schemas are im-

plicit in the population, C-Schemas are assumed to be explicit structures. Thus the C-Schema 101 $*$ $*$011 would subsume the cases {10111011, 10101011}.

In this view, as cases accumulate, C-Schema emerge as highly fit patterns or building blocks that subsume and index a set of related cases. Unlike G-Schemas which in a population of N chromosome strings, exist in the order of $N \times 2^m$ where m is the string length, C-Schemas being highly recurrent abstract patterns are assumed to be far fewer than the number of cases. This is because cognitively, C-Schemas may assist in 2 ways, first by improving speed of top-down retrieval as search can begin at the level of C-Schemas which being fewer than experiences would be easier to process and secondly as a means of organization of memory, where abstract structures act as preliminary filters and index experiences which can provide the fine grain processing. An interesting perspective borrowed from G-Schemas is to consider C- Schemas as sampling the solution space. When a C- Schema is explored, it can be said to simultaneously explore a number of potential solutions. From this point of view too, C-Schemas can be seen as preventing exhaustive search among fine grained knowledge by acting as a preliminary filter.

A C-Schema is a highly fit pattern. But what does fitness mean in this scenario? We hypothesize that the fitness of a C-Schema is based on the following 2 considerations:

1 **Genericity:** A fit C-Schema should be able to maximally subsume existing cases. I.e. Fewer the schema the better. This is Genericity, or to use GA terminology, short order C-Schema. More the abstraction, better the C-Schema.

2 **Utility:** A general C-Schema is useless if the C- Schema is not useful in actual problem solving. Thus the second factor that makes a fit schema is Utility, being able to solve the problems it faces, in other words our old friend, *TaskPer formance.* The task performance of a C-Schema may be derived either using the same principles used to derive task performances of an individual case or it can be based on the average task performance of the cases it subsumes.

These are conflicting considerations. The Genericity factor would drive the evolution towards very general C-Schemas while the Utility factor would drive the evolution towards very specific schemas in the regions of maximal task performance. If the fitness function is chosen carefully, this would lead at the end to a set of C-Schemas that provide adequate coverage and are yet effective.

Let $C^* = \{C_1, \ldots, C_i, \ldots, C_n)$ be the set of cases in the case-base. Let $S^* = \{S_1, \ldots, S_i, \ldots, S_p)$ be the set of C-Schemas. Associated with each C-schema is a set M_j defined as the set of cases in the case-base C^* subsumed by S_j. That is,

$$M_j : S_j \times C^* \rightarrow C_j^* \subseteq C^*$$

The utility U_j of a C-Schema C_j is defined as the average over the task performances of the set of cases subsumed by the C-Schema. That is,

$$U_j = \frac{\sum_{k=1}^{|M_k|} PM_k}{|M_j|}$$

The fitness of a C-Schema is a function over its generality and utility. That is,

$$F_j : |M_j| \times U_j \rightarrow [0, 1]$$

Once the fitness is calculated for each C-schema the evolutionary selection operator $E : S_t^* \rightarrow S_{t+1}^*$ is applied on the current S^* to generate the next S^*.

The evolutionary process begins with a set of cases, C^* and uses operators to propose candidate C-Schema. fitness of these C-Schema are then evaluated and new populations generated using the operators which process continues until the set of C-Schemas realized have reached a threshold fitness level. We now give 2 hypotheses:

Hypothesis A: In domains where a few general principles are operational over a wide number of experiences, we might expect that the evolutionary process would lead to a small number of short order C-Schema.

Hypothesis B: In domains where the specific context of the problem solving affects the quality of solution, there are likely to be a number of high order C-Schema.

We point to an interesting work by Perez, Coello Coello and Aguirre (Perez et al., 2001) on extracting design patterns from cases in combinatorial logic design. Part of this work has been discussed in section 4.1. However, besides using cases to inject solutions into the population, the cases are also used to deduce highly fit building blocks. Information was stored on each individual in a GA population and data collected over several generations. This set of individuals was treated as a case-base and a tree clustering algorithm generated a tree where each leaf node is a case. Higher level nodes obviously are abstractions over the leaf nodes under them. In terms of our model, each higher level node is a C-Schema that subsumes the set of cases beneath. Nodes close to the leaves will obviously be high order C-Schema and those higher up lower order. We had stated in Hypotheses A that domains where a few general principles can subsume all experiences should lead at the end of the evolutionary process to a small number of short order C-Schema. Perez's work confirms this intuition. The highly fit building blocks when analyzed were found to implicitly represent De Morgan's laws, the commutative law and the distributive law.

As far as we are aware, little is known about how schema or higher structures actually evolve from experiences. We feel the main use of using an evolution-

ary approach to C- Schema generation is to cast the problem in a different light, within a different framework and moreover a robust formal framework. From the CBR community perspective, viewing the dynamism of human memory as an evolutionary operation may enable questions to be framed in a problem solving universe with the potential to provide interesting insights as has hopefully been demonstrated here. While Perez's work was not aimed at memory modeling, it showed that Hypotheses A and B might be testable and moreover that interesting and even non- intuitive higher level structures can be realized.

The approach throws up several interesting problems, such as:

[Q9] How should case knowledge and schema knowledge be encoded? While we have used bit strings to illustrate the approach, there is every likelihood that different encodings will be required.

[Q10] The ECM framework views evolution in case memories as incremental. By the same token, the evolution of C-Schemas from cases should also be an incremental operation rather than occurring in a flurry. What evolutionary operators ? should be used?

[Q11] What form should the fitness function F take?

[Q12] Can Hypotheses A and B be confirmed?

[Q13] If the operation of evolving schemas from cases is to be added to the ECM model, how will operations like *Reorganize* and *Retrieve* change, one to create such higher level structures and the other to reason over them?

[Q14] What case representations will enable similarity based reasoning as well as play their part in an evolution mechanism?

[Q15] How can this approach be enriched to enable the generation of the levels of expertise discussed in section 5.2? The current approach merely discusses how a C-Schema can be an abstraction over cases. The preceding sections however pointed to a growth both in number and complexity of C-Schema as expertise increases.

[Q16] How will C-Schema index cases? What should be the organization of a case-base that includes both C-Schema and cases? Since cases also evolve, how will the indexing change over time?

6. CONCLUSIONS

Computational models of human memory while mere approximations of the original are also complex in their own right. Just as we become increasingly competent at tasks over time, progressing from the level of beginner to increased levels of expertise, so do computational models seek to progress towards increased expertise with time. But how do case memories progress towards expertise and how do the various case memory processes guide it in it's quest? We felt that this question needed to be better understood. To answer the question, we cast case memories as evolutionary systems to focus attention on the nature of change in case memories. The essay can be seen as a beginning attempt to get to grips with this difficult problem. It has led us to some interesting observations and a number of questions which, if addressed may lead to better clarity. We further extended the field of enquiry to look not just at the case world but into the even mistier higher world of schema. Here again, an evolutionary approach to understand how schema can be created from cases lead us to some interesting insights and a number of open questions.

Are case memories evolutionary systems? In his critique of the Sociobiology program, Richard Lewontin warned against the urge to "Darwinize" phenomena.

> "Darwin's theory of evolution by means of natural selection is an extremely powerful explanatory device ... I shall call this practice of providing an *ad hoc* Darwinian explanation for any phenomenon, Darwinizing, by analogy with 'harmonizing' in which facile harmonies are built spontaneously around a theme for the sake of a few moment's enjoyment." (Lewontin, 1976).

Is this essay therefore an attempt to Darwinize case-based reasoning? We have argued why the analogy not only makes sense but is also useful. It may even be necessary. We believe that models such as the Evolutionary Case Memory are required not only to understand the process of incremental movement towards expertise but to provide a sound basis to progress to the next generation of case based problem solvers.

ACKNOWLEDGEMENTS

Anil Menon for convincing me to write the essay, the patience to wait for it and many useful comments. Swaroop Vattam, Sutanu Chakraborti, Deepak Khemani for useful comments. Lakshmi Ramamurthy for reviews and much else besides.

REFERENCES

Balaraman, V., Chakraborti, S., and Khemani, D. (2003). Knowledge light data heavy systems: Trawlers and locators. (Under Review).

Balaraman, V. and Vattam, S. (1998). Finding common ground in case based systems. In Sasikumar, M., Durgesh, R., Praksah, P. R., and Ramani, S., editors, *Proceedings of the International Conference on Knowledge Based Systems, India*, pages 25–37.

Bartlett, F. (1932). *Remembering.* Cambridge University Press.

Bergmann, R., Breen, S., Goker, M., Manago, M., and Wess, S. (1999a). *Developing Industrial Case-Based Reasoning Applications: The INRECA Methodology,* volume 1612 of *Lecture Notes in Artificial Intelligence.* Springer Verlag, Berlin.

Bergmann, R., Breen, S., Goker, M., Manago, M., and Wess, S. (1999b). *Developing Industrial Case-Based Reasoning Applications: The INRECA Methodology,* volume 1612 of *Lecture Notes in Artificial Intelligence.* Springer Verlag, Berlin.

Bisio, R. and Malabocchia, A. (1995). Cost estimation of software projects through case-based reasoning. In Veleso, M. and Aamodt, A., editors, *Proceedings of First International Conference on CBR, ICCBR-95*, pages 11–22. Springer.

Bohm, W. and Geyer-Schulz, A. (1996). Exact uniform initialization for genetic programming. In Belew, R. K. and Vose, M. D., editors, *Foundations of Genetic Algorithms IV,* pages 379–407, San Francisco, California, USA. Morgan Kaufmann.

Borner, K. (1998). Cbr for design. In Lenz, M., Bartsch-Sporl, B., Burkhard, H.-D., and Wess, S., editors, *Case Based Reasoning Technology*, volume 1400 of *Lecture Notes in Artificial Intelligence,* pages 201–233. Springer-Verlag, Berlin.

Bradley, K. and Smyth, B. (2001). Improving recommendation diversity. In *Proceedings of the Twelfth Irish Conference on Artificial Intelligence and Cognitive Science*, *Maynooth*, *Ireland,* pages 85–94.

Brewer, W. (2000). Bartlett's concept of the schema and its impact on theories of knowledge representation in contemporary cognitive psychology. In Saito, A., editor, *Bartlett, Culture and Cognition*, pages 69–89. Psychology Press.

Bunge, M. (1968). Analogy in quantum theory: From insight to nonsense. *The British Journal for the Philosophy of Science,* 18(4):265–286.

Burkhard, H.-D. and Lenz, M. (1996). Case retrieval nets: Basic ideas and extensions. In Burkhard, H.-D. and Lenz, M., editors, *Fourth German Workshop on Case-Based Reasoning: System Development and Evaluation*, volume 55 of *Informatik-Berichte,* pages 103–110. Berlin.

Carlos, R. (1997). Schemata, frames and dynamic memory structures. Technical Report 7-97, University of Kent at Canterbury, United Kingdom.

Chakraborti, S. and Balaraman, V. (2003). A fast algorithm for pruning of search spaces in retrieval over very large case bases. (Under Review).

Charness, N. (1981). Search in chess: Age and skill differences. *Journal of experimental psychology,* 7:467–476.

Dreyfus, H. and Dreyfus, S. (1986). *Mind over Machine.* The Free Press.

Forbus, K., Gentner, D., and Law, K. (1991). MAC/FAC: a model of similarity based retrieval. *Cognitive Science,* 19(2): 141–205.

Goldberg, D. E. (2000). *Genetic Algorithm in Search, Optimization & Machine Learning.* Addison Wesley.

Gomez, d., Garza, A., and Maher, M. L. (1999). An evolutionary approach to case adaptation. In *Case based reasoning research and applications, Third international conference on case-based reasoning,* pages 167–172, Berlin. Springer-Verlag.

Gonzalez, A. J. and Laureano-Ortiz, R. (1992). A case-based reasoning approach to real estate property appraisal. *Expert systems with Applications,* 4(2):229–246.

Holyoak, K. J. and Koh, K. (1987). Surface and structural similarity in analogical transfer. *Memory and Cognition*, 15:332–340.

ICSR (1995-2002). International conferences on case-based reasoning ICCBR -1995 to 2002.

Kelly, J. D. and Davis, L. (1991). Hybridizing the genetic algorithm and the k-nearest neighbors classification algorithm. In *Proceedings of the fourth international conference on genetic algorithms,* pages 377–383, San Diego, CA. Morgan Kauffmann.

Khemani, D., Selvamani, R. B., Dhar, A. R., and Michael, S. (2002). Infofrax: Cbr in fused cased refractory manufacture. In Craw, S. and Preece, A. D., editors, *Proceedings of the 6th European Conference on Case-Based Reasoning (ECCBR) 2002,* volume 2416 of *Lecture Notes in Artificial Intelligence,* pages 560–574. Springer-Verlag.

Kolodner, J. (1993). *Case-based reasoning.* Morgan Kauffmann, San Mateo, CA.

Laszlo, M. (1990). *Ways of thinking.* World Scientific Publishing Co., Singapore.

Leake, D. (1996). *Case Based Reasoning: Experiences, Lessons & Future Directions.* AAAI Press/MIT Press, Boston.

Leake, D. and Wilson, D. (1999). When experience is wrong: Examining cbr for changing tasks and environments. In Althoff, D., Bergmann, R., and Branting, K., editors, *Proceedings of the Third International Conference on Case-Based Reasoning,* volume 1650 of *Lecture Notes in Artificial Intelligence,* pages 218–232. Springer-Verlag.

Lenz, M., Bartsch-Spoerl, B., D., B. H., and Wess, S. (1998). *Case Based Reasoning Technology: From Foundations to Applications.* Springer-Verlag, Berlin.

Lewontin, R. C. (1976). Sociobiology - a caricature of darwinism. *Proceedings of the Biennial Meeting of the Philosophy of Science Association*, 2:22–31.

Louis, S. (2002). Genetic learning for combinational logic design. In et. al., W. B. L., editor, *Proceedings of the Genetic and Evolutionary Computation Conference, (GECCO'2002) Workshop on Approximation and Learning in Evolutionary Computing,* pages 21–26, San Francisco, CA. Morgan Kaufmann.

McSherry, D. (2002). Diversity-conscious retrieval. In Craw, S. and Preece, A. D., editors, *Proceedings of the 6th European Conference on Case-Based Reasoning (ECCBR) 2002*, volume 2416 of *Lecture Notes in Artificial Intelligence,* pages 218–233. Springer-Verlag.

Medin, D. L. and Ross, B. H. (1989). The specific character of abstract thought: Categorization, problem solving and induction. In Sternberg, R. J., editor, *Advances in the psychology of human intelligence*, volume 5. Lawrence Erlbaum, Hillsdale, N. J.

Minsky, M. (1981). A framework for representing knowledge. In Haugeland, J., editor, *Mind design - Philosophy, Psychology, Artificial Intelligence,* pages 95–128. MIT Press, Cambridge, MA.

Neisser, U. (1976). *Cognition and reality.* Freeman Publishers.

Nguyen, T., Czerwinski, M., and Lee, D. (1993). Compaq Quicksource - providing the consumer with the power of AI. *AI Magazine,* 14(3):50–60.

Perez, E. I., Coello, Coello, C. A., and Aguirre, A. H. (2001). Extracting and reusing design patterns from genetic algorithms using case-based reasoning. In Y. Liu et. al., editor, *Evolvable Systems: From Biology to Hardware (ICES 2001)*, volume 2210 of *Lecture Notes in Computer Science,* pages 244–255. Springer-Verlag.

Portinale, L., Torasso, P., and Tavano, P. (1999). Speed-up, quality and competence in multi-modal case-based reasoning. In Althoff, D., Bergmann, R., and Branting, K., editors, *Proceedings of the Third International Conference on Case-Based Reasoning,* volume 1650 of *Lecture Notes in Artificial Intelligence,* pages 303–317. Springer-Verlag.

Quinlan, J. R. (1986). Induction from decision trees. *Machine Learning,* 1(1): 81–106.

Quinlan, J. R. (1993). *C4.5: Programs that Learn.* Morgan Kaufmann, San Mateo.

Reynolds, R. I. (1982). Search heuristics of chess players of different calibers. *American journal of psychology,* 95:383–392.

Rich, C. and Waters, R. C. (1990). *The Programmer's Apprentice.* ACM Press.

Schank, R. (1972). Conceptual dependency: A theory of natural language understanding. *Cognitive Psychology,* 3:552–631.

Schank, R. (1982). *Dynamic memory: A theory of reminding and learning in computers and people.* Cambridge University Press, Cambridge, MA.

Schank, R. and Abelson, K. (1977). *Scripts, plans, goals and understanding.* Lawrence Erlbaum, Hillsdale, N. J.

Schank, R. and Riesbeck, C. (1987). *Inside case-based reasoning.* Lawrence Erlbaum, Hillsdale, N. J.

Shin, K. and Han, I. (1999). Case-based reasoning supported by genetic algorithms for corporate bond rating. *Expert Systems With Applications,* 16(2): 85–95.

Smith, E., Nisbett, R., and Langston, C. (1992). The case for rules in reasoning. *Cognitive Science,* 16(1): 1–43.

Smyth, B. and McKenna, E. (1999). Building compact competent case bases. In Althoff, D., Bergmann, R., and Branting, K., editors, *Proceedings of the Third International Conference on Case-Based Reasoning,* volume 1650 of *Lecture Notes in Artificial Intelligence,* pages 329–342. Springer-Verlag.

Soh, L.-K. and Tsatsoulis, C. (2001). Combining genetic algorithms and case based reasoning for genetic learning of a casebase: A conceptual framework. In L. Spector et. al., editor, *Proceedings of the Genetic and Evolutionary Computation Conference, (GECCO'2001),* pages 376–383, San Francisco, CA. Morgan Kaufmann.

TRDDC (2000). CBDM: A case-base development methodology. Technical Report 2001-AI-CBR-02, Tata Research Development and Design Centre, Pune, India.

This page intentionally left blank

Chapter 11

THE CHALLENGE OF COMPLEXITY

Wolfgang Banzhaf
Department of Computer Science
University of Dortmund, Germany
banzhaf@cs.uni-dortmund.de

Julian Miller
School of Computer Science
The University of Birmingham, UK
j.miller@cs.bham.ac.uk

Abstract In this chapter we discuss the challenge provided by the problem of evolving large amounts of computer code via Genetic Programming. We argue that the problem is analogous to what Nature had to face when moving to multi-cellular life. We propose to look at developmental processes and there mechanisms to come up with solutions for this "challenge of complexity" in Genetic Programming.

Keywords: Genetic Programming, Evolutionary Algorithm, Complexity, Scaling Problem, Development, Heterochrony

INTRODUCTION

The purpose of this chapter is to pose a challenge to the sub-area of Evolutionary Computation (EC) dealing with algorithm evolution, Genetic Programming (Koza, 1992). Genetic Programming (GP) has a fundamental mechanism which distinguishes it from other branches of EC, namely a means to adapt the complexity of its solutions (Banzhaf et al., 1998). Such a mechanism needs to be in place in GP since the resulting solutions are programs and algorithms, or, in other words, active entities which usually require input from somewhere that is subsequently transformed into output through the target program or algorithm.

It has been shown in recent years, that there are lower bounds on the complexity of solutions to algorithmic problems in GP (Langdon, 1999b). Below

a certain threshold, no algorithm would be able to perform a predefined task. Above that threshold, however, numerous programs would be able to perform the task. Evolution in GP is thus expected to lead the programs it breeds past this threshold, to be able (only after passing) to home in on one or the other of the many solutions that exist there. One might expect that GP would be well equipped to handle tasks of varying complexity because of its basic ability to adapt complexity.

As it turns out, however, GP is not able to handle complexity gracefully, it has a scaling problem. As is well known from other search algorithms, more complexity means larger search spaces. Larger search spaces in turn mean a combinatorial explosion in the number of possible solutions which need to be visited. Even a path-oriented algorithm like an evolutionary algorithm suffers from the problem of scaling under such circumstances. Although GP is regularly able to evolve programs of length 50 to 100 lines of code, this is a far cry from what would be needed to provide a useful method for day-to-day assistance for programmers.

Various remedies have been looked at over the years. Modularization of programs is one important method to improve scalability. The problem at hand is divided into sub-problems which are supposed to be less difficult (and thus would require less complex solutions). These sub-problems could be solved in a divide-and-conquer method, whereby the overall solution is put together from the various sub-solutions evolved independently. Koza (Koza, 1994) has done an entire series of well thought-out experiments in order to show, that GP is indeed able to proceed along those lines, provided it is equipped with appropriate means (ADFs in his approach). ADFs are good at structuring a global solution into parts, and by repeated use through calls from the main program with different arguments they provide reusability features for code in multiple sub-tasks. There have been other approaches toward modularization in the last decade (Angeline and Pollack, 1994; Banzhaf et al., 1999; Rosca and Ballard, 1994), all trying to develop methods for better scalability.

However, all of these methods have failed to deliver on the fundamental challenge to GP which can be summarized in the following task:

Using GP, evolve a program whose purpose is so complex that it requires 100,000 or a million lines of hand-written code or 10,000 modules of average size 100 lines of code.

Application examples coming to mind are the following tasks

- Direction and control of the processes in a production plant
- Safe operation of an aircraft under a variety of weather conditions
- Design of a convenient multi-functional desktop computer tool, such as an editor or a mailer

- Maintaining a large network of computers as a self-repairing system
- Translation of one human language into another
- Recognition of pieces of art and music from visual or audio clues
- Evolution of a program playing Go with human-competitive performance
- A computer operating system based on self-regulation
- etc.

In other words, the challenge is to radically dispose of the complexity limits for the evolution of computer code, and aim at complexities heretofore only achieved by large teams of human programmers.

This chapter is therefore devoted to offering a possible solution to this challenge. This solution, however, can only be framed in very abstract, sometimes speculative words. Taken literally, it will not suffice to arrive at a workable mechanism. But the goal here is to set the mind of the reader into such a framework that she or he might come up with appropriate ideas to approach this challenge.

The rest of the chapter is organized as follows. Sec 1 summarizes very shortly the fundamental idea behind GP, Sec 2 looks at an ostensibly similar scaling problem situation in the area of Biology. Sec. 3 discusses Nature's way to deal with this problem, the introduction of a developmental process between the information storage in the genotype and the active entity, the phenotypic organism that results from its expression. Sec. 4 then tries to formulate a few principles of this solution to the problem that might be transferable into Genetic Programming. Sec. 5, finally goes one step further and proposes a possible scenario for the introduction of development into GP. Sec. 6 briefly discusses earlier experiences with the introduction of development, mostly treated under the heading genotype-phenotype-mapping.

1. GP BASICS AND STATE OF THE ART

Genetic Programming is part of the area of Evolutionary Algorithms which apply search principles analogous to those of natural evolution in a variety of different problem domains, notably parameter optimization. The major distinction between GP and these other areas of Evolutionary Algorithms is that GP controls active components like symbolic expressions or instructions as opposed to simple parameters, and that GP is able to develop its own representation of a problem by allowing variable complexity of its individuals.

As other evolutionary algorithms GP follows Darwin's principle of differential natural selection. This principle states the following preconditions for evolution to occur via (natural) selection:

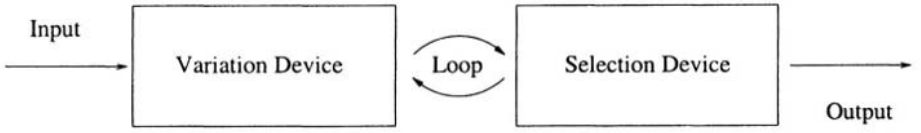

Figure 11.1. The variation selection loop of GP and other artificial evolutionary systems.

- A population of entities called individuals is formed which can reproduce or can be reproduced.
- There is heredity in reproduction, i.e. individuals produce similar offspring.
- In the course of reproduction variation occurs that affects the likelihood of survival and therefore of reproducibility of individuals.
- Due to excessive reproduction individuals are caused to compete for finite resources. Not all can survive the struggle for existence. Differential natural selection exerts pressure towards improved individuals.

Thus a variation and selection loop is iterated which constantly tries to improve solutions (see Figure 11.1).

The representation of programs, or generally structures, in GP has a strong influence on the behavior and efficiency of the resulting algorithm. As a consequence, many different approaches toward choosing representations have been adopted in GP. The resulting principles have been applied even to other problem domains such as design of electronic circuits or art and musical composition.

The mechanism behind GP works with a population of programs which are executed or interpreted in order to judge their behavior. Usually, a scoring operation called fitness measurement is applied to the outcome of the behavior. For instance, the deviation between the quantitative output of a program and its target value (defined through an error function) could be used to judge the behavior of the program. This is straight-forward if the function of the target program can be clearly defined. Results may also be defined as side-effects of a program, such as consequences of the physical behavior of a robot controlled by a genetically developed program. Sometimes, an explicit fitness measure is missing, for instance in a game situation, and the results of the game (winning or loosing) are taken to be sufficient scoring for the program's strategy. The general approach is to test a variety of programs at the same problem and to compare their performance relative to each other.

The outcome of fitness measurement are used to select programs. There are a number of different methods for selection, both deterministic and stochastic.

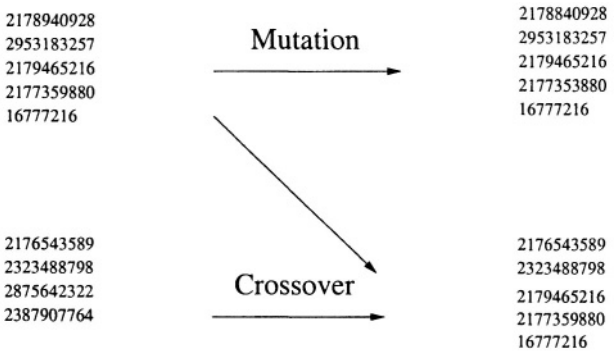

Figure 11.2. *The primary operations of GP, mutation and crossover, as applied to programs represented by sequences of instructions. The instructions are coded as integer numbers.*

These selection schemes determines (i) which programs are allowed to survive (overproduction selection), and (ii) which programs are allowed to reproduce (mating selection). Once a set of programs has been selected for further reproduction, the following operators are applied:

- reproduction
- mutation
- crossover

Reproduction simply copies an individual, *mutation* varies the structure of an individual under control of a random number generator, and *crossover* mixes the structure of two (or more) programs to generate one or more new programs (see Figure 11.2). Additional variation operators are applied in different applications. Most of these contain problem-specific knowledge in the form of heuristic search recipes adapted to the problem domain.

In this way, fitness advantages of individual programs are exploited in a population to lead to better solutions. A key effort in Genetic Programming is the definition of the fitness measure. Sometimes the fitness measure has to be iteratively improved in order for the evolved solutions to actually perform the function they were intended for. The entire process can be seen in close analogy to breeding animals. The breeder has to select those individuals from the population which carry the targeted traits to a higher degree than others.

In the meantime, many different representations for GP were studied, among them generic data structures such as sequences of instructions or directed graphs, as well as more exotic data structures such as stacks or neural networks. Today, many different approaches are considered as GP, from the evolution of parse trees to the evolution of arbitrary structures. The overarching principle is to

subject structures with variable complexity to forces of evolution by applying mutation, crossover and fitness-based selection. The results are not necessarily programs.

When analyzing search spaces of programs it was realized that their size is many orders of magnitude larger than search spaces of combinatorial optimization problems. A typical size for a program search space might be $10^{100,000}$, as opposed to a typical search space for a combinatorial optimization problem of the order of 10^{100}. Although this might be interpreted as discouraging for search mechanisms, it was also realized that the solution density in program spaces is, above a certain threshold, constant with changing complexity (Langdon, 1999a). In other words, there are proportionally many more valid solutions in program spaces than in the spaces of combinatorial optimization problems.

2. THE SITUATION IN BIOLOGY

The situation in biology is also complicated. Life needs many supporting structures. Even single-cell organisms are already very sophisticated. Let's take the bacterium E.Coli as an example (Harold, 2001b).

A bacterium is an autonomous living system and organizes molecules into a particular dynamic pattern that keeps it alive. Following Neidthardt et al, (Neidhardt, 1996) there are a total of $300 \cdot 10^6$ molecules (excluding water with $40 \cdot 10^9$ molecules) in appr. 3250 different varieties (proteins, m-,t-RNA, DNA, lipids, small metabolites and ions, peptidoglycan, etc.). The genome of E.Coli is a single, circular molecule of $4.6 \cdot 10^6$ base pairs, which is to say it contains 6 Mbits of information (again accounting for redundancy in the code) Notably, E. Coli has approximately 4300 protein coding genes (88 % of the genome) 0.8 % stable RNAs, 0.7 % repeats. 11 % of the genome might contain regulatory information (for a recent classification, see (Thomas, 1999)). Mushegian and Koonin (Mushegian and Koonin, 1996) identify a subset of 256 shared genes between two very simple bacterial organisms (H. influenzae and M. genitalium) which seem to provide the essential functions of life for those creatures. So how can all this multitude be organized by such a little genome?

Even more difficult is the situation in multicellular life. Take a human genome with its $3 \cdot 10^9$ nucleotides. Each nucleotide carries 2 bits, hence for a rough estimate we arrive at 4 Gbit maximum information content of the genome (the number was reduced from a simple multiplication, since due to code redundancy the information content is about 1/3 smaller).

On the other hand, take the number of cells of a human body as a rough estimate of the phenotype's information content: According to various estimates, the body amounts to approximately $50 \cdot 10^{12}$ cells. The estimate is raw

and difficult to quantify more accurately because the number of cells changes dynamically. Cells are produced and die during the life of an individual. Now assuming that each cell has an information content of at least 1 Mbit, this results in the requirement for $5 \cdot 10^{17}$ bits $> 10^8 \cdot 4$ Gbit, or 10^8 times the human genome! Note that the estimate of 1 Mbit per cell is unrealistically low, as we shall see when we consider free-living single-cell creatures. According to Calow (1976) (Calow, 1976), the cells of the human body have to be weighed in with a much larger information content, resulting in a total of $5 \cdot 10^{28}$ bits for the body!

We can see easily, that these numbers are completely out of proportion, which means that the information in the genome must be used in a sophisticated way so as to produce a viable organism. We might call this the information dilemma of the genotype-phenotype relation.

3. NATURE'S WAY TO DEAL WITH COMPLEXITY

In his now famous book '*The way of the cell*' biologist Frank M. Harold explains: "Genes specify the cell's building blocks; they supply raw materials, help regulate their availability and grant the cell independence of its environment. But the higher levels of order, form and function are not spelled out in the genome. They arise by the collective self-organization of genetically determined elements, effected by cellular mechanisms that remain poorly understood." (Harold, 2001b), p.69.

Thus, there are other aspects of natural biochemical systems, so far not fully understood, that structure interactions and determine the fate of molecules. These aspects constrain the possible directions that genes could affect their products. Self-organization and self-assembly are among them as are physical (and other) laws. In addition, the natural abundance of certain materials, energy, or even information plays an important role. These aspects are providing the environment in which a living system is supposed to survive.

Nature's self-organizing properties are beginning to be seen in all scientific and technical disciplines (Banzhaf, 2002). But is self-assembly without a genome sufficient to explain the intricate organization of a cell? For example, if all the necessary substructures and molecules were present in a medium, would they be able to form an E.Coli bacterium? Here we follow again the argument of Harold (Harold, 2001b) and Rosen (Rosen, 1994). The answer is "No", because self-assembly can never be a fully autonomous process. In addition, some cell components cannot be formed by self-assembly since they need to be formed by, e.g., cutting and splicing. Further, membrane proteins catalyze directional reactions (uni-directional through the membrane) (Harold, 2001a). The direction itself is, however, provided by the cell, not by the amino

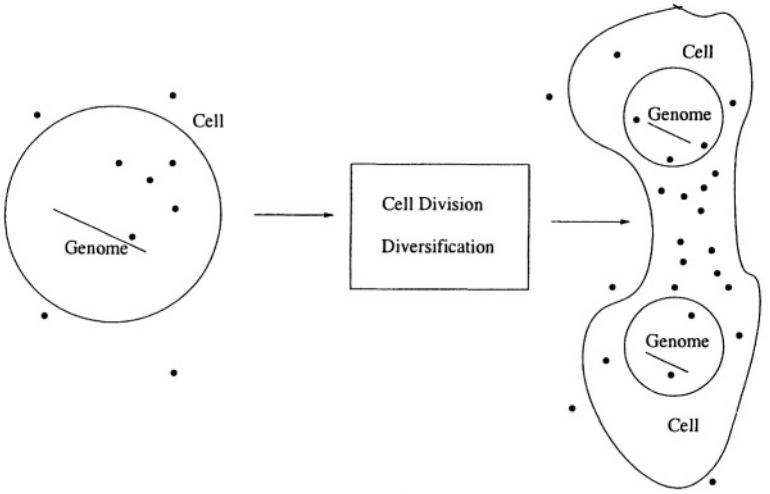

Figure 11.3. Single cell and multi-cellular system. The environment of a genome is primarily the cell in which it is residing. Control is exerted both by the cell and its environment via substances (black dots) diffusing around in intra- and extracellular space. The genome in turn tries to influence its environment by providing orders to produce certain substances. If a multicellular being is constructed a division and differentiation process is set into motion which leads to a number of cells with a boundary to the outside environment. The organism is the primary environment of a cell, with intra- and extra- organismal message transfer via molecules (black dots).

acid sequence of the protein or its gene. More generally, a great deal of localized behavior takes place within a cell. Localization, however, cannot be provided by the genes, it is a feature of their environment, i.e. of the cell (see Figure 11.3).

The conclusion is inevitable: Cells do not self-assemble. But how do they succeed instead? They grow! Rudolf Virchow (1858) was the first to formulate this realization[1] in a now famous biological law: Omnis cellula e cellula (every cell originates from a cell). No cell has not come from another cell.

In other words, there is a tight coupling between what the genome instructs and what natural laws and resources in the environment allow the cell to do. In a way the genome exploits all physical laws available (together with all sorts of material, energy and information fluxes) in order to organize a living being.

The real trick of Nature was to hit upon a system of organizing characters (RNA, then DNA and protein) that allows open-ended evolution to proceed. That is to say that the system does not close down upon encountering enormous

[1]One should be careful to include both (i) scaling up and (ii) diversification / specialization in one's notion of growth.

complexity, both in the environment and in handling its inner mechanisms. Clearly, only a combinatorial system has enough power to grow to each level of complexity demanded (and also to shrink to a lower level if necessity dictates).

"Biological forms are not fragile or contrived, quite the contrary, they are the 'generic forms' most likely to be found by self-organizing dynamic systems, and therefore both probable and robust. We may imagine systems 'exploring the space' available to the particular dynamics of each kind, and see evolution as the process by which their morphologies are transformed one into the other." (Harold, 2001b), p. 198. It may be added that natural evolution is an opportunistic process in the sense that whatever works is exploited as much as possible. Thus, the notion of a very limited exploration of the design space, as put forward by Gould (Gould, 1980; Gould, 2002) can be brought into agreement with the above opinion. Going back to the question of how development could organize the massive amount of molecules into orchestrated multicellular organisms, it seems to us that the exploitation of the natural (physical) tendencies to self-organize, i.e. to form self-maintaining networks of structures on which matter, energy and information flows, is the key recipe that genomes use. In other words, genomes are specifying or, better, influencing the interactions that lead to these networks and take place in them. What was built on top of single-cell life, then, were elaborate mechanisms for cell communication and differentiation, based on the same principles as single-celled life was. The enormous number of genes added to single cell organisms can be put to use for the purpose of (a) adaptation of the cells to multicellular environments and (b) coordination between cells, a task that is obviously very complicated.

A proper definition of biological development is in order here. At present, biological understanding might be summarized in the following statement: Development is a differential transcription (and translation) of genes in different cells and tissues at different times and rates, with each step ultimately initiated by the transcription and translation of the previous step.

The operations of transcription and translation probably warrant some explanation. Figure 11.4 shows the typical sequence of events from DNA to protein activity. After the mRNA copy is transcribed from DNA, it is processed and transported out of the nucleus of the cell. It then is translated at a ribosome into a sequence of amino acids which fold into a native structure able to perform biochemical activity.

The control and timing of transcription and translation in cells is called regulation and can be imagined as follows: The products of certain genes are not used in building the organism directly but rather are used to interact with other genes' products, with environmental cues, or with the DNA of other genes (both expressed and non-expressed parts thereof). By interaction they change the course of events in a cell, depending on the presence of interaction partners and the strength of their mutual effects. In this way, networks of

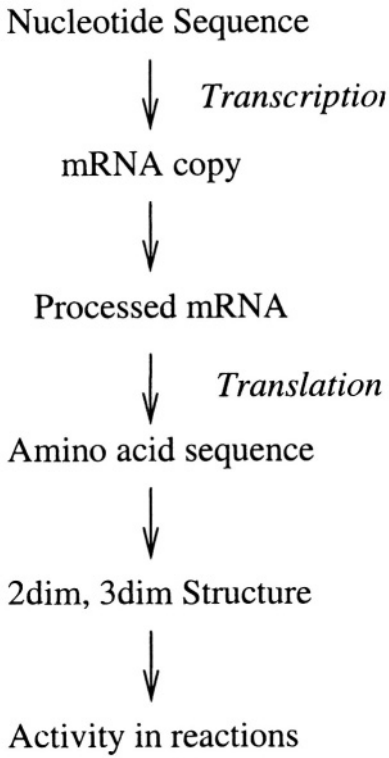

Figure 11.4. Transcription and translation as two important steps in the process of mapping information from genotype to phenotype.

interaction are formed among genes, called regulatory networks. As already mentioned, however, genes do not restrict their interactions to other genes, but may also interact with environmental material. In this way, they can interfere with another network of biochemical reactions that is formed within a cell, the metabolic network. However: "Genes seem to be distant from the biochemical network, maintaining control only by carefully timed 'injections' of their products into crucial 'branching points' where small inputs have big effects." (Harold, 2001b). Notably, most of the order of a cell is created by the underlying network, with only occasional but decisive intervention by genes.

In summary: The biochemical network of interactions between substances is the underlying substrate of a system of control built upon the effect of additional substances (signalling substances), that are itself produced by genes. The system is highly combinatorial in that many of the biochemical (maintenance) substances can interact with each other and with the signal substances. It is through a control of the expression of the where and when of the signals that genes exert their control on the underlying networks. One other key insight of developmental biology is the notion of heterochrony (Haeckel, 1866; Gould, 1977; McKinney and McNamara, 1991) which seems to be able to explain a whole plethora of phenomena found in the developmental process (and in evolution, for that matter) (McKinney, 1999). Heterochrony describes the fact that during differentiation, a large amount of control can be exerted on development by controlling three variables only: (a) the onset, (b) the rate and (c) the offset of the expression of certain genes. As such, the phenomenon is not very much different from what must happen in single-celled organisms where, in response to changing growth and environmental conditions, certain genes alter their rate of expression.

Though we don't have much space to delve into this very interesting phenomenon, one angle on heterochrony is worth looking at more closely: Its relation to the discovery of novelty in evolution. Citing McKinney and McNamara, (McKinney and McNamara, 1991): "Heterochrony is the cause of most developmental variation and heterochrony can cause major novelties. The main reason for heterochrony to be able to cause major novelties, even new tissues is the fact that it can alter the regulative development of cells already early on in development which will give rise to major 'jumps' in morphospace." And later, the authors write: "Heterochrony can be applied at different levels (molecular, cellular, tissue, organism). It is interesting to note that 'small' rate or timing changes at the lower levels will often translate into complex result at the higher levels. The nonlinearity of the system will amplify some changes (pos. feedback) and dampen others (neg. feedback) as they cascade upwards." (McKinney and McNamara, 1991), p.48.

May it suffice to add one more key insight of developmental biology that is just starting to surface in detailed studies of early embryonic development

of multicellular organisms: Interesting recent results suggest that the control of timing of developmental events, i.e. the actual mechanism by which heterochrony can be enacted, is due to an encoding of time and strength of expression of genes into the strength of interaction between (regulatory) genes (Arnone, 2002; Davidson, 2001; Gaudet and Mango, 2002).

4. WHAT WE CAN LEARN FROM NATURE?

In the previous discussion we have seen some similarity to the problems in Genetic Programming. So a natural question would be what we could learn from Nature. Here we list a few of the aspects of the developmental process in Nature which might provide hints to our efforts in artificial evolutionary systems.

1 Nature stands before what we have called the information dilemma: How to instruct a body with so few genes? The size of a genome is very small for to provide the required information for a phenotypic organism. Nature's recipes are:

- The channeling or canalizing of environmental complexity (information, energy, matter, laws, interactions, dynamics, boundaries) into the developing phenotype. The complexity of the organism stems mainly from outside and has not to be provided by the genotype. The genotype mainly directs the assembly.
- The stability of an organism (whether mature or developing) is a steady state, not a static equilibrium. It is in a continual state of growing and dying to maintain itself. Nature is dealing with open systems (due to physical constraints) where energy and entropy considerations are important. Responsiveness to environment is much better this way.
- Development allows for open-ended evolution since it is a constructive process where layers of complexity are built onto each other (with the possibility of ever larger complexity).
- There is a built-in tendency of development to be recursive (see L-systems), which allows hierarchy-building in a very natural way.
- Development happens by way of communication between cells, i.e. it's a social system of cells. More generally, there are many combinatorial subsystems interacting with each other, erecting networks of communication flow.
- Fitness tests for phenotypic organisms are always punctual, i.e. individuals are never tested completely and therefore considered

ready. Instead, multi-functionality is important and punctual fitness tests which would test for, e.g., metabolism efficiency today and for, e.g., adaptive capabilities tomorrow, allow for it to develop.

2 Time is the most important aspect of development. It results in the formation of a 4D space in biological development.

- Time and dynamics is a key to survival in real-time environments. No wonder it plays the major role in development also.
- Different time-scales (usually required by the environment) are easy to achieve, since development is intrinsically hierarchical.
- The time dimension is a way to "mold" results of development, as can be seen by the notion of heterochrony.
- There is labor division (and gradually more so) in the course of development.
- Incremental fitness is an important concept, too, i.e. there is a requirement of primitive functionality from the very beginning which is gradually refined until the organism is "mature."
- In terms of fitness landscapes: The fitness landscape gradually sharpens (becomes more rugged) in the course of development.
- The developmental process has an enormous degree of fault tolerance. Repair mechanisms are abound, as well as adaptability, and the ability of regeneration.
- There is a chain of being - from the first living thing to the last cell in a multi- cellular individual. This would be interrupted without development.
- Sexuality requires a 1-cell stage for each living being (for the uniqueness of information exchange in recombination). Thus Nature needs a mechanism for an organized transition from the one-cell stage to the multi-cell individual.

3 The mechanisms of development are constructive

- Starting from a single cell, whole bodies are constructed, consisting of millions and billions of cells.
- Development erects networks (metabolic, signaling, regulatory) of increasing complexity, within and between cells.
- Development makes use of neutrality, i.e. there are some phases in development where nothing happens if looked at from the behavior of the phenotype.

- Development allows the exploitation of side-effects, perhaps in a very efficient way. Side-effects are an important source of innovation for evolution, since they are unintentional effects which turn out to be useful for other purposes. Producing side-effects is what development can do, discerning their usefulness is left to evolution.

5. A POSSIBLE SCENARIO: TRANSFER INTO GENETIC PROGRAMMING

A linear genetic program is a sequence of instructions that is followed one by one. This might be a good way to organize a genome, as the subsequent execution of steps is a rather natural way of following this information. However, it is not a very natural way to look at program behavior, i.e. the phenotypes. We propose that, instead, complex programs of the type of interest here should be considered as networks of interacting objects which are to behave in complicated ways depending on the flow of input and required output. Thus, if one were to set up a system of interacting objects, designating input and output objects and their communication means, one would have a natural analogue to a biochemical network. Note that this does not necessarily imply that we ought to consider non-sequential programs here. Rather, it is the more general case.

Figure 11.5 shows a dataflow graph of a program phenotype. This is the graphical translation of the following program (line with "!" are not contributing to fitness):

```
void gp(r)
  double r[4];
{
   ...

   r[3] = r[1] - 3;
   r[1] = r[2] * r[1];
!  r[3] = r[1] / r[0];
   r[0] = r[1] - 1;
   r[1] = r[2] * r[0];
   r[1] = r[0] * r[1];
!  r[0] = r[2] + r[2];
   r[2] = pow(r[1], r[0]);
!  r[2] = r[0] + r[3];
!  r[0] = r[3] - 1;
!  r[1] = r[2] - r[0];
!  r[3] = pow(r[0], 2);
!  r[2] = r[2] + r[1];
   r[0] = r[1] + 9;
   r[0] = r[1] / r[3];
!  r[0] = r[2] * r[2];
!  r[2] = r[1] * r[3];
!  r[0] = r[0] + r[2];
```

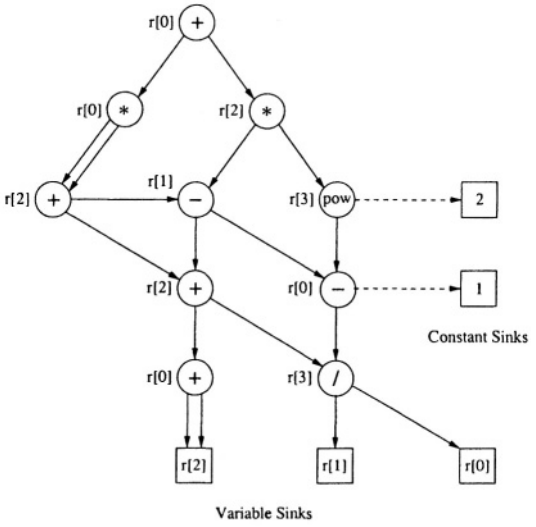

Figure 11.5. The network of data flow on registers as one example of program phenotype. The corresponding program is listed in the text as a linear sequence of instructions. Adopted from (Brameier, 2003)

}

In the language of object-oriented programming, objects possess attributes for receiving messages and methods for sending messages and performing other functions. Input driving the program system would be considered an information flow to be taken advantage of for a given purpose (output), and the right combination of interactions would be searched for by a genetic search method on the level of genes. Taking advantage would mean that those networks are differentially selected that perform, after development, the prescribed task better than others.

As for the "reconfiguration" of the object network, this would not happen through a direct modification of objects but rather through additional layers of message producing objects, and through their corresponding messages. The messages would act like signaling substances and interfere with the object network in a constructive way, e.g. by inhibition or by excitation, and the objects producing these messages would be genes located in sequence on a genome. Thus each gene would specify an object, where such an object would be even allowed to interact with other objects' specification of products. The most difficult part is presumably the latter, since it requires the ability for self-modification.

Perhaps one could even go down to the level of instructions (as equivalent to objects in the above sense). Instructions have an operation (through the op-

code) and operands (input and output) to digest messages. The problem with instructions is that the desired behavior of a program needs to come about by side-effects of instructions only. Since there are flags to be set by instructions, in principle there could be a way. It would have to be decided, what the side effects are (1. one could select for the flags, or 2. one could select for values in registers and define a network of interactions between flags of instructions, although that might be more difficult).

How could heterochrony come into play? The idea would be to influence the underlying information flow in the network of objects by means of variation in "timing" of expression. That would be a very smooth way of variation, even expressible directly as a simple parameter evolution. On the other hand, Nature's example teaches us how to translate timing signals into pattern matching. So there would be another way to control time-dependent development.

Finally, the network of interacting objects could be built up by a developmental process, perhaps starting from one object. In this case the object would act like a cell with its genome directing the expansion into a larger network of interacting objects, possibly using pre-defined objects that have been specified already and only need to be coopted into the network.

Perhaps we have ventured too far now. However, we know that a simple division and diversification process of objects can reach any size of a network in logarithmic time. As such it is perfectly imaginable that the process envisioned here will quickly reach the desired complexity for any prescribed task. Nevertheless we have to leave it to the reader and further considerations how such a scenario could be realized in a computer.

6. CONCLUSION

Some ideas related to the present contribution have been published in the past. Notable is Gruau's (Gruau, 1993) system of cellular encoding which uses a grammar tree to produce programs in a simple developmental process for GP. This work has later been applied by Koza (Koza, 1994) and others to produce electric circuit designs. Cangelosi (Cangelosi, 1999) was the first to try to make use of heterochrony in the context of GAs. A number of people are working on regulation and evolutionary algorithms using regulation, like in the work of Kennedy et al. (Kennedy and Osborn, 2001). The genotype-phenotype mapping has been studied in different papers, see for example (Smith et al., 2001) and just recently has been the subject of a special journal issue (Kargupta, 2002) under the heading "gene expression computing".

In the present contribution we have tried to provide a challenge to Genetic Programming which would be worth to meet in the long run. We have argued that Nature had to solve an analogous problem which it did by inventing the developmental process. We have discussed a number of aspects of development

that seemed to us relevant in the context of artificial evolutionary processes, and sketched one way to achieve a similar mechanism in GP. It remains to be seen whether GP can meet that challenge in the future.

ACKNOWLEDGMENTS

W.B. acknowledges the hospitality of the Institute of Genomics and Bioinformatics at UC Irvine under its director Pierre Baldi, where part of this work has been written.

REFERENCES

Angeline, P. and Pollack, J. (1994). Coevolving high-level representations. In Langton, C., editor, *Proc. Artificial Life III*, pages 55 – 71, Reading, MA. Addison Wesley.

Arnone, M. (2002). Bringing Order to Organogenesis. *Nature Genetics,* 30:348 – 350.

Banzhaf, W. (2002). *Self-Organizing Systems,* volume 14 of *Encyclopedia of Physical Science and Technology,* pages 589–598. Academic Press, New York.

Banzhaf, W., Banscherus, D., and Dittrich, P. (1999). Hierarchical Genetic Programming using local modules. Technical Report CI-56/99 of SFB 531, University of Dortmund.

Banzhaf, W., Nordin, P., Keller, R., and Francone, F. D. (1998). *Genetic Programming - An Introduction.* Morgan Kaufmann, San Francisco, CA.

Brameier, M. (2003). *Linear Genetic Programming.* PhD thesis, Department of Computer Science, University of Dortmund. (to appear).

Calow, P. (1976). *Biological Machines: A cybernetic approach to life*. E. Arnold, London.

Cangelosi, A. (1999). Heterochrony and adaptation in developing neural networks. In W. Banzhaf et al., editor, *Proceedings of GECCO99 Genetic and Evolutionary Computation Conference,* pages 1241–1248, San Francisco, CA. Morgan Kaufmann.

Davidson, E. H. (2001). *Genomic Regulatory Systems.* Academic Press, San Diego.

Gaudet, J. and Mango, S. E. (2002). Regulation of Organogenesis by the Caenorhabditis elegans FoxA Protein PHA-4. *Science,* 295:821 – 825.

Gould, S. J. (1977). *Ontogeny and Phylogeny.* Belknap Press of Harvard University Press, Cambridge, MA.

Gould, S. J. (1980). The Evolutionary Biology of Constraint. *Daedalus,* 109:39 – 52.

Gould, S. J. (2002). *The Structure of Evolutionary Theory.* Belknap Press of Harvard University Press, Cambridge, MA.

Gruau, F. (1993). Genetic Synthesis of Modular Neural Networks. In Forrest, S., editor, *Proceedings of the 5th International Conference on Genetic Algorithms, ICGA-93,* pages 318–325, San Francisco, CA. Morgan Kaufmann.

Haeckel, E. (1866). *Generelle Morphologie der Organismen.* Reimer, Berlin.

Harold, F. (2001a). Gleanings of a chemiosmotic eye. *Bioessays*, 21:848–855.

Harold, F. (2001b). *The Way of the Cell.* Oxford University Press, Oxford.

Kargupta, H. (2002). Editorial: Computation in Gene Expression. *Genetic Programming and Evolvable Machines,* 3:111–112.

Kennedy, P. J. and Osborn, T. R. (2001). A Model of Gene Expression and Regulation in an Artificial Cellular Organism. *Complex Systems*, 13.

Koza, John R. (1992). *Genetic Programming.* MIT Press, Cambridge, MA.

Koza, John R. (1994). *Genetic Programming II.* MIT Press, Cambridge, MA.

Langdon, W. B. (1999a). Boolean function fitness spaces. In Poli, R., Nordin, P., Langdon, W. B., and Fogarty, T., editors, *Proceedings EuroGP'99,* Berlin. Springer.

Langdon, W. B. (1999b). Scaling of Program Tree Fitness Spaces. *Evolutionary Computation,* 7:399 – 428.

McKinney, M. (1999). Heterochrony: Beyond words. *Paleobiology,* 25:149 – 153.

McKinney, M. and McNamara, K. (1991). *Heterochrony: The Evolution of Ontogeny.* Plenum Press, New York.

Mushegian, A. and Koonin, E. (1996). A minimal gene set for cellular life derived by comparison of complete bacterial genomes. *Proc. Natl. Acad. Sci. (USA),* 93:10268 – 73.

Neidhardt, F. C. (1996). *Escherichia Coli and Salmonella typhimurium.* ASM Press, Washington, DC.

Rosca, J. and Ballard, D. (1994). Hierarchical selforganization in genetic programming. In *Proc. of the 11th Int. Conf. on Machine Learning,* pages 252 – 258, San Mateo, CA. Morgan Kaufmann.

Rosen, R. (1994). *Life Itself.* Columbia University Press, New York.

Smith, T., Husbands, P., and O'Shea, M. (2001). Neutral Networks and Evolvability with Complex Genotype-Phenotype mapping. In Kemelen, E. and Socik, S., editors, *Proc. 6th ECAL-01, Prague, 2001,* pages 272 – 281, Berlin. Springer.

Thomas, G. H. (1999). Completing the E. coli proteome: a database of gene products characterised since completion of the genome sequence. *Bioinformatics*, 7:860 – 861.

Author Index

Index