# Genetic Algorithms and Genetic Programming for Multiscale Modeling: Applications in Materials Science and Chemistry and Advances in Scalability

**Kumara Sastry**

**IlliGAL Report No. 2007019**
**September 2007**

Illinois Genetic Algorithms Laboratory
University of Illinois at Urbana-Champaign
117 Transportation Building
104 S. Mathews Avenue Urbana, IL 61801
Office: (217) 333-2346
Fax: (217) 244-5705

# GENETIC ALGORITHMS AND GENETIC PROGRAMMING FOR MULTISCALE MODELING: APPLICATIONS IN MATERIALS SCIENCE AND CHEMISTRY AND ADVANCES IN SCALABILITY

BY

KUMARA NARASIMHA SASTRY

MENGR., Birla Institute of Technology & Science, 1998
M.S., University of Illinois at Urbana-Champaign, 2001

DISSERTATION

Submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy in Systems and Entrepreneurial Engineering
in the Graduate College of the
University of Illinois at Urbana-Champaign, 2007

Urbana, Illinois

*CERTIFICATE OF COMMITTEE APPROVAL*

*University of Illinois at Urbana-Champaign*
*Graduate College*

September 7, 2007

*We hereby recommend that the thesis by:*

**KUMARA NARASIMHA SASTRY**

*Entitled:*

**GENETIC ALGORITHMS AND GENETIC PROGRAMMING FOR MULTISCALE MODELING: APPLICATIONS IN MATERIALS SCIENCE AND CHEMISTRY AND ADVANCES IN SCALABILITY**

*Be accepted in partial fulfillment of the requirements for the degree of:*

**Doctor of Philosophy**

*Signatures:*

David E. Goldberg [illegible]

*Director of Research* - David E. Goldberg, Duane D. Johnson

[illegible]

*Head of Department* - Jong-Shi Pang

Committee on Final Examination*

David E. Goldberg

*Chairperson* - David E. Goldberg

[illegible]

*Committee Member* - Duane D. Johnson

[illegible]

*Committee Member* - Ramavarapu S. Sreenivas

[illegible]

*Committee Member* - Ali A. Yassine

*Committee Member* -

*Committee Member* -

* Required for doctoral degree but not for master's degree

# Abstract

Effective and efficient multiscale modeling is essential to advance both the science and synthesis in a wide array of fields such as physics, chemistry, materials science, biology, biotechnology and pharmacology. This study investigates the efficacy and potential of using genetic algorithms for multiscale materials modeling and addresses some of the challenges involved in designing *competent* algorithms that solve hard problems quickly, reliably and accurately. In particular, this thesis demonstrates the use of genetic algorithms (GAs) and genetic programming (GP) in multiscale modeling with the help of two non-trivial case studies in materials science and chemistry.

The first case study explores the utility of genetic programming (GP) in multi-timescaling alloy kinetics simulations. In essence, GP is used to bridge molecular dynamics and kinetic Monte Carlo methods to span orders-of-magnitude in simulation time. Specifically, GP is used to regress symbolically an *inline* barrier function from a limited set of molecular dynamics simulations to enable kinetic Monte Carlo that simulate seconds of real time. Results on a non-trivial example of vacancy-assisted migration on a surface of a face-centered cubic (fcc) Copper-Cobalt ($\mathrm{Cu}_x\mathrm{Co}_{1-x}$) alloy show that GP predicts all barriers with 0.1% error from calculations for less than 3% of active configurations, independent of type of potentials used to obtain the learning set of barriers via molecular dynamics. The resulting method enables 2–9 orders-of-magnitude increase in real-time dynamics simulations taking 4–7 orders-of-magnitude less CPU time.

The second case study presents the application of multiobjective genetic algorithms (MOGAs) in multiscaling quantum chemistry simulations. Specifically, MOGAs are used to bridge high-level quantum chemistry and semiempirical methods to provide accurate representation of complex molecular excited-state and ground-state behavior. Results on ethylene and benzene—two common building-blocks in organic chemistry—indicate that MOGAs produce high-quality semiempirical methods that (1) are stable to small perturbations, (2) yield accurate configuration energies on untested and critical excited states, and (3) yield *ab initio* quality excited-state dynamics. The proposed method enables simulations of more complex systems to realistic multi-picosecond timescales, well beyond previous attempts or expectation of human experts, and 2–3 orders-of-magnitude reduction in computational cost.

While the two applications use simple evolutionary operators, in order to tackle more complex systems, their scalability and limitations have to be investigated. The second part of the thesis addresses some of the challenges involved with a successful design of genetic algorithms and genetic programming for multiscale modeling. The first issue addressed is the scalability of genetic programming, where facetwise models are built to assess the population size required by GP to ensure adequate supply of raw building blocks and also to ensure accurate decision-making between competing building blocks.

This study also presents a design of *competent* genetic programming, where traditional fixed recombination operators are replaced by building and sampling probabilistic models of promising candidate programs. The proposed scalable GP, called *extended compact GP (eCGP)*, combines the ideas from extended compact genetic algorithm (eCGA) and probabilistic incremental program evolution (PIPE) and adaptively identifies, propagates and exchanges important subsolutions of a search problem. Results show that eCGP scales cubically with problem size on both GP-easy and GP-hard problems.

Finally, facetwise models are developed to explore limitations of scalability of MOGAs, where the scalability of multiobjective algorithms in reliably maintaining Pareto-optimal solutions is addressed. The results show that even when the building blocks are accurately identified, massive multimodality of the search problems can easily overwhelm the nicher (diversity preserving operator) and lead to exponential scale-up. Facetwise models are developed, which incorporate the combined effects of model accuracy, decision making, and sub-structure supply, as well as the effect of niching on the population sizing, to predict a limit on the growth rate of a maximum number of sub-structures that can compete in the two objectives to circumvent the failure of the niching method. The results show that if the number of competing building blocks between multiple objectives is less than the proposed limit, multiobjective GAs scale-up polynomially with the problem size on boundedly-difficult problems.

*To my parents, Shivaswamy and Lalitha*

# Acknowledgments

First and foremost, words cannot express the gratitude I owe my advisors and my committee co-chairmen Professors David E. Goldberg and Duane D. Johnson for their guidance and steadfast support. If graduate students have to be lucky to get a good advisor, then I am the luckiest person, as I have two great advisors. They have been more than research advisors to me, and I greatly benefited from their teaching, advice and guidance on a number of academic, professional and career choices.

I am really glad that I was persistent in sending emails about wanting to join IlliGAL to Dave Goldberg and I was lucky that he finally accepted. I thank him for letting me stay at IlliGAL for this long and work on exciting research projects. He is a great advisor, teacher, mentor, and source of inspiration and I have learned so much from him over these years. His amazing ability to see the big picture always kept me from missing the forest for the trees. I sincerely hope that I can emulate at least a small percentage of what I have learned from him in the coming years. I have throughly enjoyed working with and learning from him; and, if it were possible, I would have stayed at IlliGAL for a long time.

I first met Duane Johnson while searching for a topic for my Ph.D. thesis. I still remember our first meeting where I said that the task was too tough and most likely would not work. Fortunately, he encouraged me to continue on the project for a few months, which was one of the best decisions I made, and I am very grateful to him for that. I greatly benefited from his amazing ability to simplify complex and complicated subject matter. The first paper that we worked on was a great learning experience for me in seeing how he was able to shorten the paper and still say more than what I was able to say in the longer version.

I thank the rest of the committee, Professors R. S. Sreenivas and A. A. Yassine for their time, suggestions and support.

I thank my parents for their support and for taking my answers for their favorite question "when are you going to finish your PhD?" with a good sense of humor. From early on they instilled in me the value of hard work and dedication towards any work including finding missing buffaloes and cows and I thank them for that.

Collaborations with a number of researchers were instrumental for this work. I thank Professor Pascal Bellon for his collaboration on using genetic programming for multi-timescale modeling of alloy kinetics. I would like to thank to Alexis Thompson and Professor Todd Martinez for their collaboration on multiscale modeling of quantum chemistry simulations via multiobjective genetic algorithms. Their help and insights were critical to one of the key pieces of this thesis.

I am grateful to Una-May O'Reilly for her collaboration and guidance on the scalability analysis of genetic programming. Una-May's collaboration and guidance were critical in my understanding of key concepts in genetic programming. It was a pleasure and a great learning process to work with Una-May, and I hope I get a chance to do so in the future.

I would also like to thank Martin Pelikan for his help throughout my stay at IlliGAL and acknowledge his collaboration on analysis of scalability limits of multiobjective GA. When I first joined IlliGAL, Martin took me under his wing and helped me immensely in getting started with research. Over the years, we have worked together very regularly and successfully, and I sincerely hope we continue to do so in the future. Interactions, conversations and collaboration with Martin are some of the things that I cherish most.

I thank all current and former IlliGAL members and visitors, with whom I had the good fortune to work and have numerous helpful discussions. I particularly thank Hussein Abbass, Jaume Bacardit, Jacob Borgerson, Martin Butz, Erick Cantú-Paz, Ying-ping Chen, Kazuhisa Inaba, Nazan Khan, Dimitri Knjazew, Pier Luca Lanzi, Min Qiang Li, Claudio F. Lima, Xavier Llorà, Fernando Lobo, Kei Ohnishi, Albert Orriols-Puig, Luis de la Ossa, Prasanna Parthasarathy, Gerulf Pedersen, Martin Pelikan, Franz Rothlauf, Shunsuke Saruwatari, Abhishek Sinha, Ravi Srivastava, Dirk Thierens, Shigeyoshi Tsutsui, Takaoki Ueda, Clarissa Van Hoyweghen, Noriko Imafuji Yasui and Tian-Li Yu for making the lab the best place to work and also for putting up with my constant chattering and jokes and for having a great sense of humor. I will definitely miss the *lab culture* and a great opportunity to meet, collaborate and learn from so many outstanding researchers and friends. I will also miss our daily lunch and coffee, occasional foosball, and Saturday brunch sessions. I could not have acquired my *multilingual* talent, if it weren't for the help of labbies.

I would like to thank Tian-Li Yu and Xavier Llorà, who were always ready and willing to help out on any number of things, and I learned a lot from them. We were like a team, and, if it wasn't for their help, I don't think I could have been productive in research. I have always looked forward to and throughly enjoyed Pier Luca Lanzi's regular visits to IlliGAL and thank him for putting up with my *not-so-nice* explanation skills. I thank Jacob Borgerson for his help with the quals and prelim and also for many useful discussions.

I am grateful to Martin Pelikan, Xavier Llorà, Albert Orriols-Puig, Claudio F. Lima, Donna Eiskamp, Pier Luca Lanzi, Shunsuke Saruwatari, Tian-Li Yu, and Ying-ping Chen for taking the time to read and give many helpful comments on early drafts of this thesis.

Thanks are also due to IESE (formerly GE) staff members Randy Elkins, Debra Hilligoss, and Donna Eiskamp for their support and help.

This work was sponsored by the Air Force Office of Scientific Research, Air Force Materiel Command, USAF, under grants F49620-00-0163, F49620-03-1-0129, and FA9550-06-1-0096, the National Science Foundation under ITR grants DMI-99-76550, DMR-0121695, and DMR-03-25939 at Materials Computation Center, UIUC, and Computational Science and Engineering fellowship, UIUC, and the Dept. of Energy through the Fredrick Seitz Materials Research Laboratory under grant DEFG02-91ER45439. The U.S. Government is authorized to reproduce and distribute reprints for government purposes notwithstanding any copyright notation thereon.

The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of the Air Force Office of Scientific Research, the National Science Foundation, Department of Energy or the U.S. Government.

# Table of Contents

# List of Tables

# List of Figures

# Chapter 1

# Introduction

In recent years there has been growing interest in developing effective modeling and simulation methods to advance both the science and synthesis in a wide array of fields such as physics, chemistry, materials science, biology, biotechnology, and pharmacology. Many of the underlying phenomena involved are inherently multiscale acting over multiple time- and length-scales. For example, to simulate realistic nano-layered film growth, we often require systems with sizes in the range of 10–100 nm (simulation cells over one million atoms), and simulation times of about 10–1000 seconds (Jacobsen, Cooper & Sethna, 1998). To simulate realistic systems, we should be able to accommodate different system sizes, but also different time scales. Thus, there is a significant premium on cost-effective modeling techniques that can simulate physical, chemical or biological phenomena across multiple scales in both time and space, even at the price of losing information at intermediate scales.

Many powerful methods exist that address modeling at single scales. For example, molecular dynamics (MD) can accurately simulate pico-nano seconds of materials kinetics, but requires hours-days of CPU time. On the other hand, kinetic Monte Carlo (KMC) can simulate kinetics from seconds to hours, but requires the knowledge of *all* activation barriers. If we can effectively bridge these methods, then we can simulate complex materials, chemical, physical, and biological systems up to realistic time and space scales with high accuracy and significant reduction in CPU time. However, bridging these powerful methods across scales for effective multiscaling is non-trivial.

One approach is to apply modeling methods of a single scale and couple them by transferring key information from the finer scale to a coarser scale (Picu, 2003). For example, microstructural information from atomistics can be used to regress a constitutive relation between stress and strain that in turn can be used in a finite-element analysis. An important and often daunting task in this multiscaling approach is the development of proper coupling methods. This work proposes the use of genetic algorithms (GAs) (Goldberg, 1989b; Goldberg, 2002; Holland, 1975)—search, optimization, and machine-learning methods based on natural selection and genetics—and genetic programming (GP) (Koza, 1989; Koza, 1992; Koza, 1994; Koza et al., 1999; Koza et al., 2003)—GAs that evolve computer programs—as effective methods for coupling modeling methods from different scales and is applicable in various multiscaling areas, for example, modeling multi-timescale kinetics (Sastry et al., 2005), obtaining constitutive rules (Sastry et al., 2004) and finding chemical reaction pathways (Sastry et al., 2006; Sastry et al., 2007).

The purpose of this study is twofold.

1. Demonstrate the effectiveness and efficacy of using GAs and GP for multiscale modeling with the help of two important and non-trivial case studies in materials science and chemistry.

2. Advance our understanding of the scalability of multiobjective GAs and GP with the help of *facetwise* models and propose scalable designs of GP and multiobjective GAs.

The two applications used to demonstrate the potential of GAs and GP in multiscale materials modeling are as follows:

**Multi-timescaling alloy kinetics,** which is critical for designing functional nanomaterials. Specifically, GP is used to bridge molecular dynamics (MD) and kinetic Monte Carlo (KMC) (Sastry et al., 2004; Sastry et al., 2005) to span simulations by orders-of-magnitude in time.

**Multiscaling quantum chemistry simulation,** with particular focus on photochemical reactions, which are fundamental in many settings such as biological (for example, photosynthesis and vision) and technological (for example, solar cells and LEDs). Specifically, a multiobjective GA is used to *tune* semiempirical methods to enable orders-of-magnitude faster reaction dynamics simulations with *ab initio* accuracy.

The applications part of this thesis uses fairly straightforward flavor of GP and multiobjective GA. However, how these algorithms scale with increasing problem sizes and complexity is not fully understood. The second part of the thesis addresses these issues and also proposes scalable designs of both GP and multiobjective GAs that can *tractably* solve boundedly-difficult problems which are intractable with the first generation GAs and GPs. Specifically, the thesis advances our understanding of scalability of GP and GAs in the following aspects:

**Scalability of GP.** Facetwise models of (i) population sizing required for adequate supply of raw subsolutions required to obtain optimal solutions (Sastry, O'Reilly & Goldberg, 2003), and (ii) population sizing required to ensure accurate decision making between competing subsolutions (Sastry, O'Reilly & Goldberg, 2004) are developed.

**Competent GP design.** A scalable design of GP, called extended compact genetic program (eCGP) (Sastry & Goldberg, 2003a), which is based on the extended compact genetic algorithm (eCGA) (Harik, 1999; Harik, Lobo & Sastry, 2006) is developed. The proposed competent GP solves a broad class of *adversarially-designed* boundedly-difficult problems using only polynomial (cubic) number of function evaluations. In contrast, a standard GP with fixed recombination operators scales exponentially on GP-hard problems.

**Scalability of multiobjective GAs.** Limits on the scalability of multiobjective GAs in reliably maintaining a diverse set of optimal solutions—also known as Pareto-optimal solutions—is modeled and verified with empirical results (Sastry, Pelikan & Goldberg, 2005). Along the way, based on the multiobjective Bayesian optimization algorithm (Khan, Goldberg & Pelikan, 2002; Pelikan, Sastry & Goldberg, 2005), a competent MOGA design—the multiobjective extended compact genetic algorithm (meCGA)—is proposed that can solve boundedly-difficult problems using only a polynomial (quadratic) number of function evaluations (Pelikan, Sastry & Goldberg, 2006; Sastry, Pelikan & Goldberg, 2005).

While the scalability studies were motivated with the specific application of GAs and GP to multiscale modeling problems, the models developed and the lessons learned are not just limited to the multiscale modeling domain, but are broadly applicable to a wide range of problems of interest to both theoreticians and practitioners of genetic and evolutionary computation.

## 1.1 Roadmap

This thesis is organized as follows.

Chapter 2 gives a brief introduction to genetic algorithms, genetic programming, and multiobjective GAs. The chapter also presents a brief overview of GA design decomposition theory, scalable or *competent* GAs, and principled efficiency-enhancement techniques. While competent

GAs take problems that were intractable with first generation GAs and render them tractable, efficiency enhancement takes us from *tractability* to *practicality*. More recent studies on synergistic integration of competent GAs with efficiency enhancement techniques that lead to *supermultiplicative* speedups are also briefly mentioned. These efforts are now leading to GA designs for routine giga-variable optimization (Goldberg, Sastry & Llorà, 2007; Sastry, Goldberg & Llorà, 2007).

The rest of the thesis can be decomposed into two parts. The first part of the thesis deals with applications of GAs and GP to multiscale modeling in materials science and chemistry. The second part discusses advances in the scalability analysis of these methods, particularly GP and multiobjective GAs.

Chapter 3 discusses the potential of using GAs and GP for multiscale materials modeling. In essence, GAs and GP are used to bridge critical information between existing modeling methods that are powerful at single scales. In other words, GAs and GP could be used to derive accurate custom-made constitutive relationships for a higher level model using sparsely-sampled data from lower-level models.

Chapter 4 provides details on using genetic programming for bridging molecular dynamics and kinetic Monte Carlo methods for fast and accurate alloy kinetics simulations. GP is used to *regress symbolically* an *inline* barrier function using a limited set of molecular dynamics calculations. The GP regressed barrier function is then used with kinetic Monte Carlo to span about 15 orders-of-magnitude in time.

Chapter 5 discusses application of multiobjective GAs in bridging *ab initio* quantum chemistry methods and semiempirical methods for fast and accurate reaction dynamics simulation. Specifically, based on limited *ab initio* computations and experimental results, multiobjective genetic algorithms are used to *tune* semiempirical parameters to yield globally accurate potential energy surfaces. The multiobjective GA tuned semiempirical methods yield reaction dynamics with *ab initio* accuracy enabling us to predict reaction dynamics of more complex systems such as nanotubes and proteins for orders-of-magnitude longer and realistic timescales.

Chapter 6 develops facetwise models of population sizing in GP for raw subsolution supply and for accurately deciding between competing subsolutions. Models show that like in GAs, the population sizing for decision making usually governs the population sizing in GP over population sizing for adequate subsolution supply. The population sizing for good decision making suggests that the population size grows quadratically with the optimal program (tree) size. Moreover, facetwise models show that in GP the number of function evaluations scales cubically with optimal program size.

Chapter 7 proposes a design for scalable GP developed based on the extended compact genetic algorithm (eCGA). Unlike traditional genetic programming, which use fixed recombination operators, the proposed probabilistic model-building GP automatically discovers important substructures of the underlying problems and effectively samples them to evolve high-quality solutions. The proposed algorithm, called the extended compact genetic programming (eCGP), adaptively identifies and exchanges non-overlapping subsolutions by constructing and sampling probabilistic models of promising solutions. The results show that in contrast to standard GP, eCGP scales polynomially with the problem size (the number of functionals and terminals) on both GP-easy and boundedly difficult GP-hard problems.

Chapter 8 studies the limits of scalability of multiobjective GAs in reliably evolving and maintaining a diverse set of optimal solutions. Specifically, facetwise models and empirical results suggest that the number of optimal subsolutions that differ between different objectives has to be limited for multiobjective GAs to be competent. In other words, if the optimal subsolutions are different for different objectives in all partitions, then, for a class of additively separable problems, multiobjective GAs scale exponentially due to the presence of exponential number of optimal solutions. On the other hand, if the optimal subsolutions that are different for different objectives scale as $O(\log m)$, multiobjective GAs scale polynomially (usually subquadratically) with problem size.

Finally, summary and key conclusions are given in chapter 9.

# Chapter 2

# Genetic Algorithms (GAs) and Genetic Programming (GP)

This chapter provides a brief introduction to genetic algorithms (GAs)—search, optimization, and machine learning methods based on principles of genetics and natural selection. While the first generation GAs were designed mostly on an ad hoc basis, over the last few decades, a decomposition-based design theory has been proposed, an overview of which is presented in this chapter. The design theory has led to the development of *competent* GAs—GAs that solve boundedly difficult problems quickly, reliably, and accurately—and principled *efficiency-enhancement* techniques to speedup GAs—which are surveyed in this chapter. Key concepts of genetic programming (GP)—GAs that evolve computer programs—and multiobjective GAs—GAs that optimize multiple objectives and discover optimal tradeoffs between given objectives—are also briefly introduced.

## 2.1 Genetic Algorithms

Genetic algorithms (GAs) are search methods based on principles of natural selection and genetics (Goldberg, 1989b; Goldberg, 2002; Holland, 1975). Over the past few decades, GAs have been successfully applied to solve hard search problems in science and engineering ranging from gas-pipeline design (Goldberg, 1983), analog circuit design (Koza et al., 2003), finger-print recognition (Grasemann & Miikkulainen, 2005), and design optimization (Deb, 2001). Significant progress has also been made in understanding and designing *competent* genetic algorithms, that solve hard problems quickly, reliably, and accurately (Goldberg, 2002). The population sizing, convergence time and scalability of competent GAs are well understood on different classes of boundedly difficult problems (Goldberg, 2002; Harik et al., 1999; Miller & Goldberg, 1996a; Pelikan, 2005; Pelikan, Sastry & Cantú-Paz, 2006).

Analogous to genetics, GAs encode the decision variables of a search problem into finite-length strings of alphabets of certain cardinality. The strings, which are candidate solutions to the search problem are referred to as *chromosomes*, the alphabets are referred to as *genes* and the values of genes are referred to as *alleles*. For example, in a problem such as the traveling salesman problem, a chromosome represents a route, and a gene may represent a city. In contrast to traditional optimization techniques, GAs work with coding of parameters, rather than the parameters themselves. Several encoding methods such as binary, gray, real, permutation, and program codes can be and have been used.

In order to evolve good solutions and to implement natural selection, the notion of *fitness*, or a relative goodness measure, of a candidate solution is used. The measure could be an *objective* function that is a mathematical model, computation or a computer simulation, or it can be a *subjective* function where humans choose better solutions over worse ones, or it can be *co-evolved* arising out of co-operative or competitive environments. In essence, the fitness measure must determine a candidate solution's relative fitness, which will subsequently be used by the GA to guide the evolution of good solutions. Furthermore, the fitness of a solution can be quantified by a single measure (as in single-objective optimization) or multiple measures (as in multi-objective optimization).

Another important concept of GAs is the notion of a population. Unlike traditional search methods, genetic algorithms rely on a population of candidate solutions. Population-based search is not only robust, but also efficient in handling noise, scaling, and hard fitness landscapes. The population size, which is usually a user-specified parameter, is one of the important factors affecting the scalability and performance of genetic algorithms (Goldberg, 2002). Small population sizes might lead to premature convergence and yield substandard solutions. On the other hand, large population sizes lead to unnecessary expenditure of valuable computational time. Facetwise population-sizing models have been developed to understand the scalability on a broad class of boundedly-difficult search problems (Goldberg, 1989c; Goldberg, 2002; Goldberg, Deb & Clark, 1992; Goldberg, Sastry & Latoza, 2001; Harik et al., 1999; Mahfoud, 1994; Sastry & Goldberg, 2003b).

Once the problem is encoded in a chromosomal manner and a fitness measure for discriminating good solutions from bad ones has been chosen, we can start to *evolve* solutions to the search problem using the following steps (Goldberg, 1989b; Goldberg, 2002; Sastry, Goldberg & Kendall, 2005):

1. **Initialization:** The initial population of candidate solutions is usually generated randomly across the search space. However, domain-specific knowledge or other information can be easily incorporated in the generation of the initial population.

2. **Evaluation:** Once the population is initialized or an offspring population is created, the fitness values of the candidate solutions are evaluated.

3. **Selection:** The selection operator allocates more copies to solutions with better fitness values and thus imposes the survival-of-the-fittest mechanism on the candidate solutions. The main idea of selection is to prefer better solutions to worse ones, and many selection procedures have been proposed to accomplish this idea, including fitness-proportionate methods such as roulette-wheel selection (Goldberg, 1989b; Holland, 1975) and stochastic universal selection (Baker, 1985; Grefenstette & Baker, 1989), and ordinal methods such as tournament selection (Goldberg, Korb & Deb, 1989; Sastry & Goldberg, 2001) and truncation selection (Mühlenbein & Schlierkamp-Voosen, 1993). Usually, ordinal selection schemes are preferred over fitness-proportionate methods as they are less noisy and are not affected by scaling of fitness measures (Goldberg, 2002).

4. **Recombination:** Crossover or recombination operator combines bits and pieces of two or more parental solutions to create new, possibly better solutions (that is, offspring). There are many ways of accomplishing this (Booker et al., 1997; Goldberg, 1989b; Spears, 1997), and achieving competent performance does depend on getting the recombination mechanism designed properly; but the primary idea to keep in mind is that the offspring under recombination will not be identical to any particular parent and will instead combine parental traits in a novel manner (Goldberg, 2002). Competent recombination operators that automatically identify the important traits of parental solutions and effectively exchange them have been developed for genetic algorithms (Goldberg, 1989b; Goldberg, 2002; Pelikan, 2005; Pelikan, Goldberg & Cantú-Paz, 2000; Pelikan, Sastry & Cantú-Paz, 2006; Sastry & Goldberg, 2003a).

5. **Mutation:** While recombination operates on two or more parental chromosomes, mutation operator makes random modifications locally around a solution. Again, there are many

variations of mutation (Bäck, 1996; Beyer, 1996; Goldberg, 1989b; Hansen & Ostermeier, 2001; Rechenberg, 1973; Schwefel, 1977), but it usually involves one or more changes that are made to an individual's trait or traits. In other words, mutation performs a random walk in the vicinity of a candidate solution. Competent mutation operators that automatically identify the important traits of parental solutions and effectively search in the substructural neighborhoods have been developed (Lima et al., 2006; Sastry & Goldberg, 2004a).

6. **Replacement:** The offspring population created by selection, recombination, and mutation replaces the original parental population. Many replacement techniques such as elitist replacement, generation-wise replacement and steady-state replacement methods are used in GAs.

7. Repeat steps 2–6 till one or more stopping criteria are met. Examples of stopping criteria include reaching maximum number of function evaluations, maximum number of generations, etc.

Elsewhere, Goldberg (1983, 1999a, 2002) has likened GAs to mechanistic versions of certain modes of human innovation and has shown that, though selection, crossover, and mutation operators when analyzed individually are ineffective, when combined together they can work well. This aspect has been explained with the concepts of the *fundamental intuition* and *innovation intuition*. The same study compares a combination of selection and mutation to *continual improvement* (a form of hill climbing), and the combination of selection and recombination to *innovation* (*cross-fertilizing*). These analogies have been used to develop a design-decomposition methodology (Goldberg, 2002; Goldberg, Deb & Clark, 1992; Goldberg & Liepens, 1991), *competent* genetic algorithms, or GAs that solve hard problems quickly, reliably, and accurately (Goldberg et al., 1993; Pelikan, 2005; Pelikan, Sastry & Cantú-Paz, 2006; Yu, 2006), and the design of *principled efficiency-enhancement* techniques (Cantú-Paz, 2000a; Goldberg & Voessner, 1999; Sastry, 2001; Sastry & Goldberg, 2004b; Sastry, Pelikan & Goldberg, 2006).

## 2.2 Scalable Genetic Algorithms

While using innovation for explaining working mechanisms of genetic algorithms is very useful, as a design metaphor it poses difficulty as the processes of innovation are themselves not well understood. However, if we want GAs to solve successfully increasingly difficult problems across a wide spectrum of areas, we need a principled, but mechanistic way of designing genetic algorithms. The last few decades have witnessed great strides toward the development of *competent* genetic algorithms—GAs that solve hard problems, quickly, reliably, and accurately (Goldberg, 1999a). From a computational standpoint, the existence of competent GAs suggests that many difficult problems can be solved in a scalable fashion. Furthermore, it significantly reduces the burden on a user to decide on a good coding or a good genetic operator that accompanies many GA applications; if the GA can adapt to the problem, there is less reason for the user to have to adapt the problem, coding, or operators to the GA.

### 2.2.1 GA Design Decomposition

Some of the key lessons of competent GA design are briefly reviewed in this section. We restrict the discussion to selectorecombinative GAs, and focus on the cross-fertilization type of innovation and briefly discuss key facets of competent GA design. Using Holland's notion of a building block (BB) (Holland, 1975), Goldberg proposed decomposing the problem of designing a competent selectorecombinative GA (Goldberg, 1991; Goldberg, Deb & Clark, 1992; Goldberg & Liepens, 1991). This design decomposition has been explained in detail elsewhere (Goldberg, 2002), but is briefly reviewed in what follows.

**Know that GAs process building blocks (BBs).** The primary idea of selectorecombinative GA theory is that genetic algorithms work through a mechanism of *decomposition* and *reassembly*. Holland (Holland, 1975) called well-adapted sets of features that were components of effective solutions *building blocks* (BBs). The basic idea is that GAs (1) implicitly identify building blocks or sub-assemblies of good solutions, and (2) recombine different sub-assemblies to form very high performance solutions.

**Understand BB hard problems.** From the standpoint of cross-fertilizing innovation, problems that are hard have BBs that are hard to acquire. This may be because the BBs are complex, hard to find, or different BBs are hard to separate, or low-order BBs may be *misleading* or *deceptive* (Deb & Goldberg, 1994; Goldberg, 1987; Goldberg, 1989a; Goldberg, Deb & Horn, 1992).

**Understand BB growth and timing.** Another key idea is that BBs or notions exist in a kind of competitive *market economy of ideas*, and steps must be taken to ensure that (1) the best ones grow and take over a dominant market share of the population, and (2) the growth rate is neither too fast, nor too slow.

The growth in market share can be easily satisfied appropriately by setting the crossover probability, $p_c$, and the selection pressure, $s$, which quantifies the strength of selection (Goldberg & Sastry, 2001)

$$p_c \leq \frac{1 - s^{-1}}{\epsilon}, \tag{2.1}$$

where $\epsilon$ is the probability of BB disruption.

Three main approaches have been used in understanding time:

- *Takeover time models,* where the dynamics of the best individual is modeled (Bäck, 1994; Cantú-Paz, 1999; Goldberg & Deb, 1991; Sakamoto & Goldberg, 1997; Smith & Vavak, 1999; Rudolph, 2000).
- *Selection-intensity models,* where approaches similar to those in quantitative genetics (Bulmer, 1985) are used and the dynamics of the average fitness of the population is modeled (Bäck, 1995; Miller & Goldberg, 1995; Miller & Goldberg, 1996a; Mühlenbein & Schlierkamp-Voosen, 1993; Thierens & Goldberg, 1994a; Thierens & Goldberg, 1994b; Voigt, Mühlenbein & Schlierkamp-Voosen, 1996).
- *Higher-order cumulant models,* where the dynamics of average and higher-order cumulants are modeled (Blickle & Thiele, 1995; Blickle & Thiele, 1996; Cantú-Paz, 2000b; Prügel-Bennet & Shapiro, 1994; Rattray & Shapiro, 1997; Rogers & Prügel-Bennet, 1999; Shapiro, Prügel-Bennet & Rattray, 1994).

The time models suggest that for a problem of size (number of binary variables) $\ell$ , with all BBs of equal importance or salience, the convergence time of GAs is given by (Miller & Goldberg, 1995),

$$t_c = \frac{\pi}{2I}\sqrt{\ell}, \tag{2.2}$$

where $I$ is the selection intensity (Bulmer, 1985), which is a parameter dependent on the selection method and selection pressure. For tournament selection, $I$ can be approximated in terms of $s$ by the relation (Blickle & Thiele, 1995):

$$I = \sqrt{2\left(\log(s) - \log\left(\sqrt{4.14\log(s)}\right)\right)}. \tag{2.3}$$

On the other hand, if the building blocks of a problem have different salience, then the convergence time scales differently. For example, when BBs of a problem are exponentially scaled, with a particular BB being exponentially better than the others, then the convergence time of a GA is linear with the problems size (Thierens, Goldberg & Pereira, 1998):

$$t_c = \frac{-\log 2}{\log\left(1 - I/\sqrt{3}\right)}\ell. \tag{2.4}$$

To summarize, the convergence time of GAs scales as $\mathcal{O}\left(\sqrt{\ell}\right)$–$\mathcal{O}\left(\ell\right)$.

**Understand BB supply and decision making.** One role of the population is to ensure adequate *supply* of the raw building blocks. Randomly generated populations of increasing size will, with higher probability, contain larger numbers of more complex BBs (Goldberg, 1989c; Goldberg, Sastry & Latoza, 2001; Holland, 1973; Holland, 1975; Reeves, 1993). For a problem with $m$ building blocks, each consisting of $k$ alphabets of cardinality $\chi$, the population size required to ensure the presence of at least one copy of all the raw building blocks is given by (Goldberg, Sastry & Latoza, 2001),

$$n = \chi^k \log m + k\chi^k \log\chi. \tag{2.5}$$

Just ensuring the raw supply is not enough, decision making among different, competing notions (BBs) is *statistical* in nature, and, as we increase the population size, we increase the likelihood of making the best possible decisions (De Jong, 1975; Goldberg, Deb & Clark, 1992; Goldberg & Rudnick, 1991; Harik et al., 1999). For an additively decomposable problem with $m$ building blocks of size $k$ each, the population size required to not only ensure supply, but also ensure correct decision making is approximately given by (Harik et al., 1999),

$$n = -\frac{\sqrt{\pi}}{2}\frac{\sigma_{BB}}{d}2^k\sqrt{m}\log\alpha, \tag{2.6}$$

where $d/\sigma_{BB}$ is the signal-to-noise ratio (Goldberg, Deb & Clark, 1992), and $\alpha$ is the probability of deciding incorrectly among competing building blocks. In essence, the population-sizing model consists of the following components:

- **Competition complexity**, quantified by the total number of competing building blocks, $2^k$.
- **Subcomponent complexity**, quantified by the number of building blocks, $m$.
- **Ease of decision making**, quantified by the signal-to-noise ratio, $d/\sigma_{BB}$.
- **Probabilistic safety factor**, quantified by the coefficient $-\log\alpha$.

On the other hand, if the building blocks are exponentially scaled, the population size scales as (Goldberg, 2002; Rothlauf, 2002; Thierens, Goldberg & Pereira, 1998):

$$n = -c_o\frac{\sigma_{BB}}{d}2^k m\log\alpha, \tag{2.7}$$

where, $c_o$ is a constant dependent on the drift affects (Asoh & Mühlenbein, 1994; Crow & Kimura, 1970; Goldberg & Segrest, 1987).

To summarize, the population size required by GAs scales as $\mathcal{O}\left(2^k\sqrt{m}\right)$–$\mathcal{O}\left(2^k m\right)$.

**Identify BBs and exchange them.** Perhaps the most important lesson of current research in GAs is that the *identification and exchange of BBs* is the critical path to innovative success. First-generation GAs usually fail in their ability to promote this exchange reliably. The primary design challenge to achieving competence is the need to identify and promote effective BB exchange. Theoretical studies using *facetwise* modeling approach (Goldberg, Thierens &

Deb, 1993; Sastry & Goldberg, 2002; Sastry & Goldberg, 2003b; Thierens & Goldberg, 1993; Thierens, 1999) have shown that while fixed recombination operators such as uniform crossover, due to inadequacies of effective identification and exchange of BBs, demonstrate polynomial scalability on simple problems, they scale up exponentially with problem size on boundedly-difficult problems. The mixing models also yield a *control map* as a function of control parameters for selection and crossover operators, delineating the region of good performance for a GA. Such a control map can be a useful tool in visualizing GA sweet-spots and provide insights in parameter settings (Goldberg, 1999a). This is in contrast to recombination operators that can automatically and adaptively identify and exchange BBs, which scale up polynomially (subquadratically-quadratically) with problem size (Goldberg, 2002; Goldberg et al., 1993; Pelikan, 2005; Pelikan, Sastry & Cantú-Paz, 2006).

### 2.2.2 Competent GA Designs

Efforts in principled design of effective BB identification and exchange mechanisms have led to the development of competent genetic algorithms. Competent GAs are a class of GAs that solve hard problems quickly, reliably, and accurately. Hard problems are loosely defined as those problems that have large sub-solutions that cannot be decomposed into simpler sub-solutions, have badly scaled sub-solutions, have numerous local optima, or are subject to a high stochastic noise. While designing a competent GA, the objective is to develop a GA that can solve problems with bounded difficulty and exhibit polynomial (usually subquadratic) scalability with the problem size.

Interestingly, the mechanics of competent GAs vary widely, but the principles of innovative success are invariant. Competent GA design began with the development of the *messy genetic algorithm* (Goldberg, Korb & Deb, 1989), culminating in 1993 with the *fast messy GA* (Goldberg et al., 1993). Since those early scalable results, a number of competent GAs have been constructed using different mechanism styles that can be classified based on the following facets (Chen, 2005; Chen et al., 2007):

1. means to distinguish between good and bad linkage;

2. methods to express or represent linkage;

3. ways to store linkage information.

Based on the mechanisms used to distinguish good linkages from bad linkages, competent GAs can be classified into the following categories:

**Unimetric methods,** which act solely on the fitness value given by the fitness function. No extra criteria or measurements are involved for deciding whether an individual or a model is better. Unimetric approaches, loosely modeled after natural environments, are believed to be more biologically plausible. Methods such as linkage learning GAs (Chen, 2005; Harik, 1997; Harik & Goldberg, 1997) are examples of unimetric methods.

**Multimetric methods,** which in contrast to unimetric approaches, employ extra criteria or measurements other than the fitness function given by the problem for judging the quality of individuals or models. Multimetric approaches are of artificial design and employ certain bias which does not come from the problem at hand to guide the search. Methods such as gene expression messy GA (gemGA) (Kargupta, 1996), estimation of distribution algorithms (EDAs) (Larrañaga & Lozano, 2002; Pelikan, 2005; Pelikan, Goldberg & Lobo, 2002; Pelikan, Sastry & Cantú-Paz, 2006), and design structure matrix GA (Yu, 2006) are examples of multimetric methods.

Additionally, approaches such as the messy GA (mGA) (Goldberg, Korb & Deb, 1989), the fast messy GA (fmGA) (Goldberg et al., 1993), the ordering messy GA (OmeGA) (Knjazew, 2002), the structured messy GA (Halhal et al., 1999), and the incremental commitment GA (Watson & Pollack, 1999) use implicit multimetric approaches and therefore can be considered to be in between uni- and multi-metric approaches.

Based on methods used to represent linkage, we can broadly classify scalable GA designs into two categories:

**Physical linkage,** where linkage emerges from physical locations of two or more genes on the chromosome. Physical linkage is closer to biological plausibility and inspired directly by it. Examples of competent GAs that use physical linkage are the messy GA (Goldberg, Korb & Deb, 1989), the fast messy GA (Goldberg et al., 1993), and the linkage learning GA (Chen, 2005; Harik, 1997; Harik & Goldberg, 1997).

**Virtual linkage,** where linkage information is explicitly represented using graphs, groupings, matrices, pointers, or other data structures that control the subsequent pairing or clustering organization of decision variables. Virtual linkage is an engineering or computer science approach towards effectively representing key linkages of the underlying problem. Estimation of distribution algorithms (Larrañaga & Lozano, 2002; Pelikan, 2005; Pelikan, Goldberg & Lobo, 2002; Pelikan, Sastry & Cantú-Paz, 2006), general linkage crossover (Salman, Mehrotra & Mohan, 2000), and the dependency structure matrix GA (Yu, 2006) are examples of competent GAs that employ virtual linkage.

Based on the ways to store linkage information, competent GAs can be classified into two categories:

**Distributed Model,** where there is no centralized storage of linkage information, but it is maintained in a distributed manner. Similar to the unimetric approach, distributed-model approaches are also loosely modeled after evolutionary conditions in nature and more biologically plausible. Examples of competent GAs with distributed model include linkage learning GA (Chen, 2005; Harik, 1997; Harik & Goldberg, 1997), the messy GA (Goldberg, Korb & Deb, 1989), the fast messy GA (Goldberg et al., 1993), the gene expression messy GA (Kargupta, 1996), and perturbation based linkage identification procedures (Coffin & Clack, 2006; Heckendorn & Wright, 2004; Munetomo & Goldberg, 1999; Tsuji, Munetomo & Akama, 2006).

**Centralized Model,** where linkage information is stored in centralized manner, such as a global probabilistic vector or dependency table, to handle and process linkage. Centralized-model approaches are developed to achieve the maximum information exchange and to obtain the desired results. Examples of competent GAs with centralized linkage models include estimation of distribution algorithms (Larrañaga & Lozano, 2002; Pelikan, 2005; Pelikan, Goldberg & Lobo, 2002; Pelikan, Sastry & Cantú-Paz, 2006), general linkage crossover (Salman, Mehrotra & Mohan, 2000) and the design structure matrix GA (Yu, 2006).

## 2.3 Efficiency Enhancement of Genetic Algorithms

Competent GAs have successfully solved hard problems oftentimes requiring only a subquadratic number of function evaluations. That is, competent GAs usually solve an $\ell$-variable search problem, requiring only $\mathcal{O}(\ell^2)$ number of function evaluations. While competent GAs take problems that were intractable with first generation GAs and render them tractable, for large-scale problems, the task of computing even a subquadratic number of function evaluations can be daunting. If the fitness function is a complex simulation, model, or computation, then a single evaluation might take hours, even days. For such problems, even a subquadratic number of function evaluations is very high. For example, consider a 20-bit search problem and assume that a fitness evaluation takes one hour. We will require about an order of a month to solve the problem. This places a premium on a variety of *efficiency enhancement techniques*. While competence leads us from *intractability* to *tractability*, efficiency enhancement takes us from *tractability* to *practicality*. Efficiency-enhancement techniques often used in GAs can be broadly classified into four principal categories:

**Parallelization,** where GAs are run on multiple processors and the computational resource is distributed among these processors (Cantú-Paz, 2000a; Cantú-Paz, 1997). Evolutionary algorithms are by *nature* parallel, and many different parallelization approaches such as a simple master-slave (Bethke, 1976; Grefenstette, 1981), coarse-grained (Grosso, 1985; Pettey, Leuze & Grefenstette, 1987; Tanese, 1989), fine-grained (Gorges-Schleuter, 1989a; Gorges-Schleuter, 1989b; Manderick & Spiessens, 1989; Robertson, 1987), or hierarchical (Goldberg, 1989b; Gorges-Schleuter, 1997; Gruau, 1994; Lin, Goodman & Punch, 1997) architectures can be readily used. Regardless of how parallelization is done, the key idea is to distribute the computational load on several processors thereby speeding-up the overall GA run. Moreover, there exists a principled design theory for developing an efficient parallel GA and optimizing the key facts of parallel architecture, connectivity, and deme size (Cantú-Paz, 2000a; Cantú-Paz & Goldberg, 1999; Cantú-Paz & Goldberg, 2000).

For example, when the function evaluation time, $T_f$, is much greater than the communication time, $T_c$, which is very often the case, then a simple master-slave parallel GA—where the fitness evaluations are distributed over several processors and the rest of the GA operations are performed on a single processor—can yield linear speed-up when the number of processors is less than or equal to $\sqrt[3]{\frac{T_f}{T_c}n}$, and a maximum speed-up when the number of processors equals $\sqrt{\frac{T_f}{T_c}n}$, where $n$ is the population size (Cantú-Paz, 1998).

**Hybridization,** where domain-specific knowledge and other local-search techniques are coupled with GAs to obtain high-quality solutions in reasonable time (Davis, 1991; Goldberg & Voessner, 1999; Hart, 1994; Krasnogor, 2002; Moscato, 1989; Sinha, 2003b). Most industrial strength GAs employ some sort of local search for a number of reasons such as achieving faster convergence (Bosworth, Foo & Zeigler, 1972; Hart, 1994; Sinha, 2003b), repairing infeasible solutions into legal ones (Ibaraki, 1997; Orvosh & Davis, 1993), initializing GA population (Fleurent & Ferland, 1994; Ramsey & Grefenstette, 1993), and refinement of solutions obtained by a GA (Hartmann & Rieger, 2001).

There are two methods to utilize the information gained through local search called *Baldwinian* and *Lamarckian* approach. In Lamarckian evolution, both the fitness and alleles (genotypes) of an individual are changed to obtained through local search (in the phenotypic space). On the other hand, in Baldwinian learning, the individual retains its original alleles, but has its fitness changed to that obtained through local search. Orvosh and Davis (1993) have suggested an empirical rule of 20, according to which a Lamarckian step should be used about once in twenty trials.

While GA practitioners have often understood that real-world or commercial applications often require hybridization, there have been limited efforts in developing a principled design framework on answering critical issues such as the optimal division of labor between global and local searches (or the right mix of exploration and exploitation) (Goldberg & Voessner, 1999; Sinha, 2003a), the effect of local search on sampling (Hart, 1994; Hart & Belew, 1996), and the optimal duration of local search (Hart, 1994; Land, 1998)

**Time Continuation,** where capabilities of both mutation and recombination are optimally utilized to obtain a solution of as high quality as possible with a given limited computational resource (Goldberg, 1999b; Sastry & Goldberg, 2004a; Sastry & Goldberg, 2004b; Sastry & Goldberg, 2007; Srivastava, 2002; Srivastava & Goldberg, 2001). Time utilization (or continuation) exploits the tradeoff between search for solutions with large population and a single convergence epoch or using a small population with multiple convergence epochs.

Early theoretical investigations indicate that when the building blocks are of equal (or nearly equal) salience and both recombination and mutation operators have the linkage information, then a small population with multiple convergence epochs is more efficient. However, if the fitness function is noisy or has overlapping building blocks, then a large population with single convergence epoch is more efficient (Sastry & Goldberg, 2004a; Sastry & Goldberg, 2004b). On the other hand, if the building blocks of the problem are of non-uniform salience, which essentially require serial processing, then a small population with multiple convergence epochs is more efficient (Goldberg, 1999b; Sastry & Goldberg, 2007). Interestingly, recent efforts at using linkage models—for example, probabilistic models built by estimation of distribution of algorithms—to decide between single-epoch GA with large population vs. multiple-epoch GA with small population have yielded what promises to be *supermultiplicative* speedups (Lima et al., 2006; Lima et al., 2005; Sastry & Goldberg, 2004a). These studies are leading towards *adaptive* time-continuation techniques which yield speed-ups far in excess of those obtainable through traditional means.

**Evaluation relaxation,** where an accurate, but computationally-expensive fitness evaluation is replaced with a less accurate, but computationally inexpensive fitness estimator. The low-cost, less-accurate fitness estimator can either be (1) *exogenous*, as in the case of surrogate (or approximate) fitness functions (Jin, 2005), where external means can be used to develop the fitness estimate, or (2) *endogenous*, as in the case of *fitness inheritance* (Smith, Dike & Stegmann, 1995) where the fitness estimate is computed internally based on parental fitnesses.

Evaluation relaxation in GAs dates back to early, largely empirical work of Grefenstette and Fitzpatrick (1985) in image registration (Fitzpatrick, Grefenstette & Van Gucht, 1984) where significant speed-ups were obtained by reduced random sampling of the pixels of an image. Approximate models have since been used extensively to solve complex optimization problems in many engineering applications such as aerospace and structural engineering (Barthelemy & Haftka, 1993; Booker et al., 1998; Dennis & Torczon, 1997).

While early evaluation-relaxation studies were largely empirical in nature, design theories have since been developed to understand the effect of approximate surrogate functions on population sizing and convergence time and to optimize speed-ups in approximate fitness functions with known variance (Miller & Goldberg, 1996b; Miller, 1997), in integrated fitness functions (Albert & Goldberg, 2001), in simple functions of known variance or known bias (Sastry, 2001), and in fitness inheritance (Pelikan & Sastry, 2004; Sastry, Goldberg & Pelikan, 2001; Sastry, Lima & Goldberg, 2006; Sastry, Pelikan & Goldberg, 2004). Recent work on using linkage models to induce the functional form of surrogates combined with standard techniques for estimating the coefficients of the induced surrogates have yielded super-multiplicative speedups orders-of-magnitude above those achievable by conventional means (Pelikan & Sastry, 2004; Sastry, Lima & Goldberg, 2006; Sastry, Pelikan & Goldberg, 2004; Sastry, Pelikan & Goldberg, 2006).

### 2.3.1 Integration of Efficiency Enhancement Techniques

Speed-up obtained by employing an efficiency-enhancement technique (EET) is measured in terms of a ratio of the computation effort required by a GA when the EET is used to that required by GA in the absence of the EET. That is, $\eta = T_{\text{base}}/T_{\text{efficiency}-\text{enhanced}}$. Speed-up obtained by employing even a single EET can potentially be significant. Furthermore, assuming that the performance of one of the above methods does not affect the performance of others, if we employ more that one EET, the overall speed-up is the product of individual speed-ups. That is, let the speed-ups obtained by employing parallelization, hybridization, time continuation and evaluation relaxation be $\Psi_{parallel}$, $\Psi_{hybrid}$, $\Psi_{time}$, and $\Psi_{evaluation}$ respectively. Given a speedup of $\Psi_{competentGA}$ can be obtained by using a competent genetic algorithm, if one uses all these EETs, then the overall speed-up obtained is

$$\Psi_{\text{total}} = \Psi_{competentGA}\Psi_{\text{parallelel}}\Psi_{\text{hybird}}\Psi_{\text{time}}\Psi_{evaluation}.$$

Even if the speed-up obtained by a single EET is modest, a combination of two or more EETs can yield a significant speed-up. For example, if we use a parallel GA that yields linear speed-up with 128 processors, and each of the other three EETs makes GAs 25% more efficient, then together they yield a speed-up of $128 * 1.25^3 = 250$. That is, evaluation relaxation, time continuation, and hybridization would give over 100 processors' worth of additional computation power.

While the prospect of multiplicative speedups is exciting enough, recent studies on tightly integrating competent GAs with efficiency-enhancement techniques have revealed the possibility of obtaining *supermultiplicative speedups* far in excess of those predicted by conventional means (Lima et al., 2005; Pelikan & Sastry, 2004; Sastry, Lima & Goldberg, 2006; Sastry, Pelikan & Goldberg, 2004). For example, following multiplicative speedups, the use of inheritance with a competent GA would have suggested a 20–25% efficiency gain by combining these two sources of improvement (Sastry, Goldberg & Pelikan, 2001), but instead a tight integration of competence and inheritance yields speedups of between 30–50 (Pelikan & Sastry, 2004; Sastry, Pelikan & Goldberg, 2004).

This initially unexpected *supermultiplicative* speedups can be explained using extensions of design and efficiency enhancement theory (Sastry, Pelikan & Goldberg, 2006). This is an exciting possibility that promises orders-of-magnitude improvements above those obtained independently by competent GAs and efficiency enhancement techniques. A systematic study of the integration of probabilistic model building and the four main sources of efficiency enhancement will enable routine solutions to problems with millions and billions of variables (Goldberg, Sastry & Llorà, 2007; Sastry, Goldberg & Llorà, 2007)

## 2.4 Genetic Programming

*Genetic programming* (Koza, 1989; Koza, 1992; Koza, 1994; Koza et al., 1999; Koza et al., 2003) is a genetic algorithm that evolves computer programs. Over the last two decades, GP has been successfully used to solve problems in a wide range of areas from novel analog circuit design to quantum computing that have resulted in several reinvention of patented inventions and some patentable new inventions (Koza et al., 2003). A typical GP consists of the following components:

**Representation:** In GP a chromosome is a candidate computer program, which is usually represented by a tree (Koza, 1992). For example, see Figure 2.1. However, it should be noted that other representations such as linear codes (Banzhaf et al., 1998; Pelikan, Kvasnicka & Pospichal, 1997), grammar-based codes (Ratle & Sebag, 2001; Ryan, Collins & O'Neill, 1998),

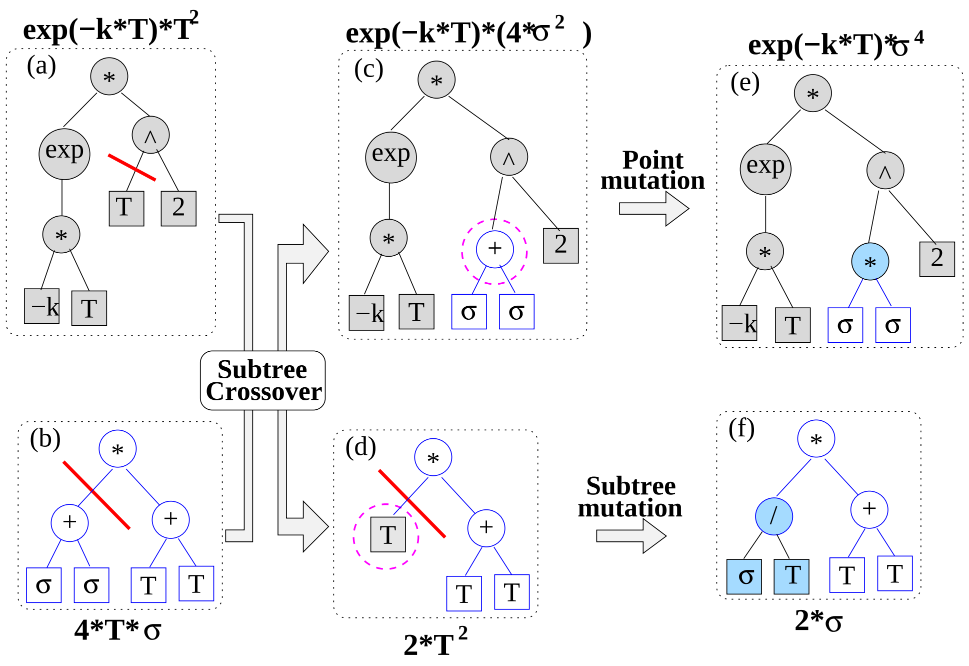


Figure 2.1: Illustration of tree representation, subtree crossover, subtree mutation, and point mutation used in GP.

and domain-specific representations (Babovic & Keijzer, 2000; Koza et al., 2003) have also been used.

In tree representations, the internal nodes of a tree are composed of elements from a set of *primitive functions* and the leaf nodes consist of elements from a set of *terminals* as shown in the example of Figure 2.1. Both the primitive functions and the terminals are user specified. The primitive functions can be arithmetic functions (for example, "+", "−", "*", "/"), logical expressions (for example, "`if-then-else`", "`and`", "`or`", "`not`"), boolean functions (for example, "`AND`", "`OR`", "`XOR`", "`NOT`"), loops (for example, "`while`", "`for`"), and other program constructs, including user-specified subroutines, or domain-specific functions. The terminals usually consist of independent variables of a problem, constants, and *ephemeral random constants* (Koza, 1992).

Choosing appropriate primitive functions and terminals is one of the key factors influencing GP performance. In some cases, the choice of the primitive functions can be straightforward and might consist of arithmetic functions, and a branching operator. In other cases, the primitive functions might include specialized functions from the problem domain. For example, for thermal processes such as various creep mechanisms, a specialized function such as $Q/(k_B T)$ can be beneficial. Similarly, for thermal activation process, such as vacancy-assisted climb (which is one of the creep-recovery mechanisms), using $\exp[-Q/(k_B T)]$ as a primitive function might be beneficial in obtaining an optimal solution. Koza (1992) suggests selecting primitive functions that are capable of expressing the solution to the problem, and calls this the *sufficiency* property. Kinnear (1996b) suggests using "the most powerful useful seeming functions from the problem domain that you can think of". It should be noted, however, that if one uses advanced GP features like *automatically defined functions* and *architecture altering operations*, then missing, but necessary primitive functions, can be automatically co-evolved (Koza et al., 1999).

**Initialization:** A common initialization scheme used in GP is the *ramped half-and-half* method (Koza, 1992), where trees of different sizes (between user-specified minimum and maximum tree sizes) and shapes are initialized using the *grow* and *full* methods in equal proportion. In a grow method, a tree of arbitrary size is generated by selecting terminals or primitive functions with equal probability. In a full method a tree of specified size is generated by selecting only primitive functions for the nodes till the tree size approaches a specified size after which only terminals are selected.

**Recombination:** Typically, in a GP a *subtree-crossover* method is used (Koza, 1992). In subtree crossover, a crossover point for each solution is randomly chosen and subtrees below the crossover points are swapped to create two new solutions (see Figure 2.1).

**Mutation:** In GP, usually two mutation techniques are used, see Figure 2.1: *Subtree mutation*, where a subtree is randomly replaced with another randomly created subtree, and *point mutation* where a node is randomly modified.

### 2.4.1 Advanced Genetic Programming Features

The scalability and applicability of GP can be significantly enhanced with the help of one or more of the advanced features (Koza et al., 1999; Koza et al., 2003), two of which are outlined in the following:

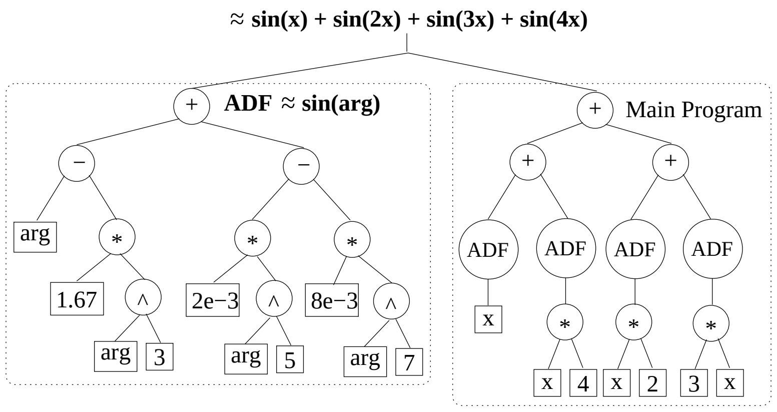


Figure 2.2: Illustration of an ADF. The individual approximates $\sin(x)+\sin(2x)+\sin(3x)+\sin(4x)$ by repeatedly using an ADF which in turn approximates the sin function.

**Constrained Syntactic Structures:** In simple GP, the search space—or, the possible set of valid combinations of functions and terminals—is usually unconstrained. However, when solving complex problems, it might be advantageous, or even necessary, to search among a restricted program space. Usually, grammar-based representations such as *strong typing* are used to restrict the space of valid syntactic structures (Haynes, Schoenfeld & Wainwright, 1996; Koza et al., 2003; Montana, 1995). One such restriction (or constraint) that is useful for symbolic regression is imposing the requirement that the GP operators create (or promote) only *dimensionally-correct* functions (or programs) (Babovic & Keijzer, 2000; Ratle & Sebag, 2001). For example, when evolving a model for predicting activation energies, we need to search among only those relations whose dimensions are consistent with energy units. By restricting the GP to evolve only dimensionally-correct programs, the search space can be significantly reduced and the evolutionary process can be accelerated.

**Automatically Defined Functions (ADFs):** Sometimes it is difficult or even impossible to have prior knowledge of all the important primitive functions that are useful for solving the problem at hand. Even when appropriate primitive functions are used, the solution might contain repeated use of certain highly favorable combinations of functions and terminals. For example, when modeling activation energies from atomistic simulations, a relation representing the attractive and repulsive forces between atom pairs, such as $(r/\sigma)^n - (r/\sigma)^m$, might be repeatedly used. In such cases, GP can automatically evolve highly favorable combinations of functions and terminals and *encapsulate* them into reusable subroutines (Koza, 1994). These automatically defined functions eliminate the need to "reinvent the wheel" and efficiently utilizes the modularities, symmetries, regularities, and hierarchy of the problem (Koza et al., 2003). Moreover, the use of ADFs decreases the possibility of disrupting the important sub-programs via the genetic operators such as recombination and mutation.

In GP, an ADF is evolved from a population of candidate ADFs in parallel to the main program. That is, every individual comprises multiple trees, one representing the main program and the others representing the ADFs as shown in Figure 2.2. The main program invokes an ADF as a primitive function with predefined number of arguments. For example, in Figure 2.2, the ADF takes a single argument. Other program features, such as iterations, loops, recursions, and stores, can also be defined automatically along the lines of an ADF and further details are available elsewhere (Angeline, 1996; Kinnear, 1996a; Koza, 1994; Rosca & Ballard, 1996). Furthermore, the structure, content, and subroutine topologies can also be evolved with the help of *architecture altering operations* such as addition, deletion, and duplication of subroutines and arguments (Koza et al., 1999).

## 2.5 Multiobjective Genetic Algorithms

Another area where GAs are particularly effective is multiobjective optimization. Many real-world optimization problems contain multiple competing objectives and there is a premium on methods that can handle multiple objectives and discover optimal tradeoff (Pareto-optimal front) between these objectives. Traditional approaches for handling multi-objective problems usually convert multiple objectives into single-objective problems by using a priori weights denoting the relative importance of the different objectives. They rely on multiple runs of single-objective optimization with different weights to obtain different Pareto-optimal solutions. However, the choice of weights is a non-trivial task and furthermore uniform coverage of the Pareto-optimal front is usually improbable—sometimes, impossible—and the methods are usually inefficient and less robust (Deb, 2001).

On the other hand, population-based approaches such as genetic algorithms are particularly suited to handle multiple objectives as they can process a number of solutions in parallel and

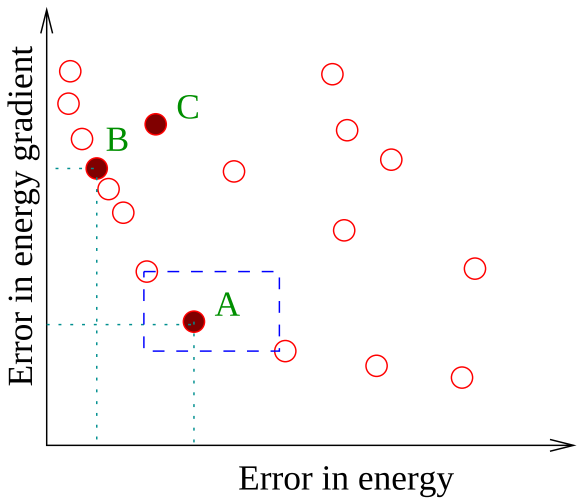
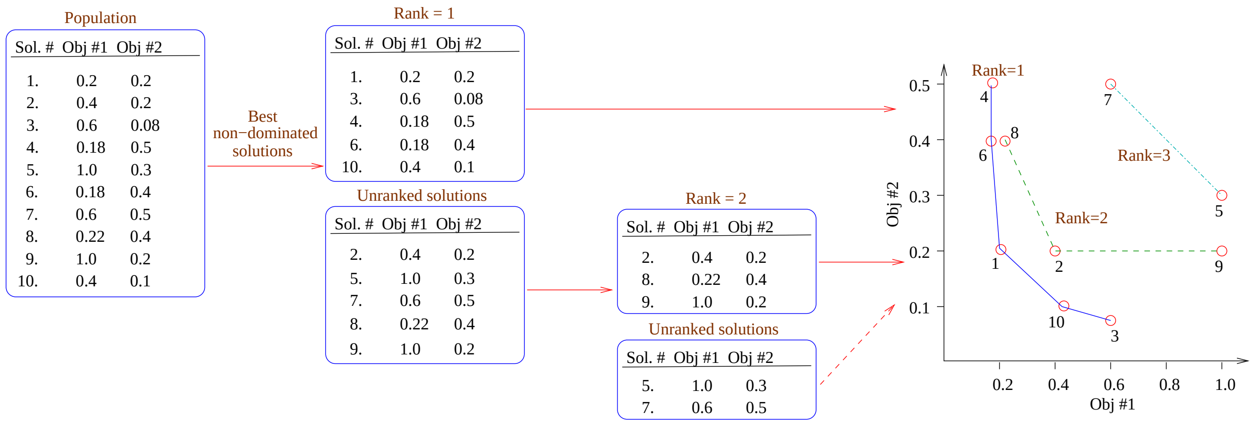


Figure 2.3: Illustration of non-domination and crowding for a two-objective minimization problem. Solution A and B are non-dominant and solution B dominates C. Furthermore, solution A is less crowded (less dense) than solution B. Therefore, when solutions B and C compete, solution B is preferred as it is more dominant, and when solutions A and B compete, solution A is preferred as it is more diverse, or less crowded (as it lies in a less dense area of the non-dominated set).

find all or majority of the solutions in the Pareto-optimal front. Based on Goldberg's suggestion (Goldberg, 1989b) of implementing a selection procedure that uses a *non-domination* principle, many multiobjective evolutionary algorithms have been proposed (Coello Coello, Van Veldhuizen & Lamont, 2002; Corne et al., 2001; Deb, 2001; Deb et al., 2002; Erickson, Mayer & Horn, 2001; Fonseca & Fleming, 1993; Horn, Nafpliotis & Goldberg, 1994; Pelikan, Sastry & Goldberg, 2006; Srinivas & Deb, 1995; Van Veldhuizen & Lamont, 2000; Zitzler, Laumanns & Thiele, 2001; Zydallis, Van Veldhuizen & Lamont, 2001).

Two key components enable GAs to handle multiple objectives: (1) selection based on a *non-domination* principle, and (2) niching in objective and/or decision variable space. Both these mechanisms are illustrated in the following paragraphs with the non-dominated sorting genetic algorithm II (NSGA-II) (Deb et al., 2002) as the exemplar of multiobjective GAs.

**Non-dominated sorting:** The non-dominated sorting procedure assigns domination ranks to individuals in the population based on their multiple objective values. A candidate solution X dominates Y, if X is no worse than Y in all objectives and if X is better than Y in at least one objective. For example, in Figure 2.3, solution B dominates solution C, whereas solutions A and B are non-dominant.



Population

| Sol. # | Obj #1 | Obj #2 |
|---|---|---|
| 1. | 0.2 | 0.2 |
| 2. | 0.4 | 0.2 |
| 3. | 0.6 | 0.08 |
| 4. | 0.18 | 0.5 |
| 5. | 1.0 | 0.3 |
| 6. | 0.18 | 0.4 |
| 7. | 0.6 | 0.5 |
| 8. | 0.22 | 0.4 |
| 9. | 1.0 | 0.2 |
| 10. | 0.4 | 0.1 |

Rank = 1

| Sol. # | Obj #1 | Obj #2 |
|---|---|---|
| 1. | 0.2 | 0.2 |
| 3. | 0.6 | 0.08 |
| 4. | 0.18 | 0.5 |
| 6. | 0.18 | 0.4 |
| 10. | 0.4 | 0.1 |

Unranked solutions

| Sol. # | Obj #1 | Obj #2 |
|---|---|---|
| 2. | 0.4 | 0.2 |
| 5. | 1.0 | 0.3 |
| 7. | 0.6 | 0.5 |
| 8. | 0.22 | 0.4 |
| 9. | 1.0 | 0.2 |

Rank = 2

| Sol. # | Obj #1 | Obj #2 |
|---|---|---|
| 2. | 0.4 | 0.2 |
| 8. | 0.22 | 0.4 |
| 9. | 1.0 | 0.2 |

Unranked solutions

| Sol. # | Obj #1 | Obj #2 |
|---|---|---|
| 5. | 1.0 | 0.3 |
| 7. | 0.6 | 0.5 |

Figure 2.4: Illustration of non-dominated sorting procedure for a two-objective minimization problem from a set of randomly generated population of 10 individuals. The corresponding positions of the individuals in the non-dominated fronts are also shown.

In non-dominated sorting, we start with the set of solutions that are not dominated by any solution in the population and assign them rank 1. Next, solutions that are not dominated by any of the remaining solutions are assigned rank 2. That is, all solutions with rank 2 are dominated by at least one solution with rank 1, but are not dominated by others in the population. Thus the sorting and ranking process continues by assigning increasing ranks to those solutions that are not dominated by any of the remaining unranked solutions. After non-dominated sorting, we are left with subsets of the population with different ranks. Solutions with a given rank are not dominated by solutions that have the same rank or higher and are dominated by at least one solution with a lower rank. Therefore, with respect to Pareto optimality, solutions with lower ranks should be given priority. The non-dominated sorting procedure is illustrated for a two objective minimization problem in Figure 2.4.

**Crowding distance computation:** Apart from finding solutions in the Pareto front, it is also essential to achieve good coverage or spread of solutions in the front. The diversity of solutions in the objective space is usually maintained with a niching mechanism and NSGA-II uses crowding for doing so. Each solution in the population is assigned a crowding distance, which estimates how dense the non-dominated front is in the neighborhood of the solution. Therefore, the higher the crowding distance of the solution, the more diverse the solution is in the non-dominated front. For example, in Figure 2.3, solution A is less crowded, and hence more diverse, than solution B.

The pseudocode for computing the crowding distance is outlined below:

```
crowding_distance_computation(P)
   for rank r = 1 to R
       $P_r$ = subset of solutions in P with rank r
       $n_r$ = size($P_r$)
       for i = 1 to $n_r$
           $d_c(P_r(i))$ = 0
       for j = 1 to M
           $Q_r$ = sort $P_r$ using $j^{th}$ objective, $f_j$.
           $d_c(Q_r(1)) = \cdots = d_c(Q_r(n_r)) = \infty$
           for i = 2 to $n_r$-1
               dist = $Q_r(i+1).f_j$ - $Q_r(i-1).f_j$
               $d_c(Q_r(i))$ = $d_c(Q_r(i))$ + dist
   return $d_c$
```

where, P is the population, R is the maximum rank assigned in the population, M is the number of objectives, and $Q_r(i).f_j$ is the value of the $j^{th}$ objective of the $i^{th}$ individual.

The selection operator in multiobjective GAs uses the non-domination principle and also typically acts as a diversity preserving operator. For example, NSGA-II uses an *individual comparison operator* to compare the quality of two solutions and to select the better individual. Both the rank and the crowding distance of the two solutions are used in the comparison operator, a pseudo-code of which is given below. First, the rank of the two individuals are considered and the solution with a lower rank is selected. If the two individuals have the same rank, then the solution with the highest crowding distance is selected.

```
compare(X,Y)
   if rank(X) < rank(Y) then return X
   if rank(X) > rank(Y) then return Y
   if rank(X) = rank(Y)
      if $d_c$(X) > $d_c$(Y) then return X
if $d_C$(X) < $d_C$(Y) then return Y
if $d_C$(X) = $d_C$(Y)
   then randomly choose either X or Y
```

## 2.6 Summary

The chapter provided a brief introduction to genetic algorithms and design decomposition theory of scalable GAs. Following the design-decomposition theory several designs of scalable or competent GAs have been proposed and were briefly reviewed. While competence takes us from intractability to tractability, efficiency-enhancement techniques take us from tractability to practicality. The chapter briefly reviewed four classes of efficiency-enhancement techniques: (1) parallelization, (2) hybridization, (3) time continuation, and (4) evaluation relaxation. While conventional view of integrating competent GAs and efficiency-enhancement techniques predicts the combined speedup to be multiplicative of speedups from individual sources, recent work on synergistically integrating competent GAs with efficiency-enhancement techniques reveal the possibility of obtaining super-multiplicative speedups. Finally, an overview of genetic programming and multiobjective GAs, which are used in the applications part of this thesis for demonstrating the potential of GAs and GP for multiscale materials modeling, was also provided in this chapter.

# Chapter 3

# Genetic Algorithms and Genetic Programming for Multiscale Materials Modeling

Although there has been a rise in interest in multiscaling as of late (Barkema & Mousseau, 1996; Barkema & Mousseau, 2001; Cai et al., 2002; Chen, 1999; Diaz De La Rubia et al., 1998; Fish & Schwab, 2003; Henkelman & Jónsson, 1999; Jacobsen, Cooper & Sethna, 1998; Mukherjee et al., 2000; Sørensen & Voter, 2000; Steiner & Genilloud, 1998; Voter, 1997; Voter, 1998; Voter, Montalenti & Germann, 2002), the very notion of multiscaling has been an important part of mathematical physics for some time. Perhaps the most commonplace examples are drawn from the physics of solids, liquids, and gases, where assumptions are made about the molecular behavior of matter and statistical mechanics enables us to derive constitutive relationships that enable us to treat the mechanics of solids, liquids, and gases in a continuum. These cases are, of course, somewhat special, depending on the microscopic homogeneity of the molecular level. As engineers and scientists want to understand less regular low-level phenomena, multiscale speed-up depends on our ability to derive custom-made constitutive relationships for the special case at hand. No longer can we reliably rely on specialized mathematical tools to bridge the gap, and the two modeling levels being bridged may not be assumed to exist in mathematical closed from. Instead, we must find automatic ways to (1) sample sparsely low-level models, and (2) derive accurate custom-made constitutive relationships for a higher level using a uniform, competent computational procedure.

Genetic algorithms (GAs) and genetic programming (GP) are a class of such effective tools for bridging modeling methods working on different scales (see Figure 3.1)[1]. For example, GP (Koza, 1989; Koza, 1992; Koza, 1994; Koza et al., 1999; Koza et al., 2003) could be used to evolve automatically (reduced-order) models between macroscopic variables based on microscopic data. Unlike traditional regression methods, GP does not require the knowledge of the functional

[1]It should be noted that like genetic programming, other machine-learning and data-mining techniques, such as artificial neural networks and Bayesian-learning techniques, can also be used for multiscale modeling. For example, see Tiley et al. (2004).

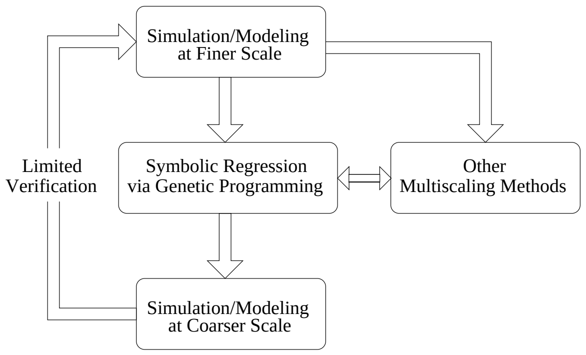

Simulation/Modeling at Finer Scale
Limited Verification
Symbolic Regression via Genetic Programming
Other Multiscaling Methods
Simulation/Modeling at Coarser Scale

Figure 3.1: Schematic of GA-based approach for multiscale simulation.

form of the coupling relation. Indeed, GP searches for both the functional form (via appropriate choice of primitive functions, terminals, and program-tree structure) and regresses the coefficients in parallel to best fit the data. That is, GP *evolves* important basis functions for regression via appropriate hierarchical combination of primitive functions, and also optimizes the coefficient values simultaneously. Similarly, for cases where the functional form of the constitutive relations are known, GAs could be used to optimize the coefficients of the constitutive relations. Apart from being robust general-purpose solvers, GAs and GP can also be readily and efficiently *hybridized* with other multiscale methods (Sinha, 2003b). Additionally, GAs and GP are inherently parallel making them readily amenable to a variety of parallelization techniques (Cantú-Paz, 2000a), which is a significant advantage over other competitive methods.

The potential of GP in evolving custom constitutive relations is illustrated with the help of a non-trivial example in the remainder of this chapter. The objective of the case study is to create custom constitutive relations based on macroscale data arising from microscale effects. Specifically GP is used evolve a relationship between flow stress and temperature-compensated strain rate for an aluminum alloy based on limited experimental data.

## 3.1 Evolving Constitutive Relations via GP

One of the ways to transfer information from microscopic analysis onto macroscopic analysis is via constitutive relations. A methodology for discovering automatically not only the important variables, but also the functional form of constitutive relations between them can be very effective. Such reduced-order constitutive relations can then be used in macroscopic analysis via finite-element or finite-difference methods.

Here, the emphasis is on the effectiveness of GP in evolving custom constitutive relations and measured data (Padilla et al., 2003) is used to evolve a constitutive relation between flow stress and temperature-compensated strain rate for an aluminum alloy AA7055. Such a constitutive relation is required for the simulation of the hot rolling process, which is an important thermomechanical process involved in the treatment of aluminum alloys (Beaudoin & Cassada, 1998). The effect of temperature, stress, and strain rate on the final product properties and their anisotropy are required to understand the hot rolling process. Furthermore, the material properties are also affected by changes in precipitate distributions, grain structure and texture (Deshpande, 1998). Microstructural characterization can be used to describe the microstructural development with temperature and the evolution of deformation structures. Moreover, high-temperature compression tests can be performed to high strains to collect data that can correlate material properties with microstructural damage characterization (Padilla et al., 2003).

The objective is to find a constitutive relation between the stress and strain rate to model their correlation based on microstructural characterization. Padilla et al. (2003) conducted such high temperature compression tests on AA7055 at temperatures ranging from 340°C–520°C at two different strain rates of 1 s$^{-1}$ and 10$^{-3}$ s$^{-1}$. A detailed experimental procedure used in obtaining the stress-strain rate data is given elsewhere (Padilla et al., 2003). Assuming the power-law relation between stress and strain rate, they obtained the coefficients that best fit the data to give the following constitutive relation (Sastry et al., 2004):

$$\dot{\epsilon} = A_o \exp\left(\frac{Q_d}{RT}\right)\left(\frac{\sigma}{\mu}\right)^{4.56}, \tag{3.1}$$

where $\dot{\epsilon}$ is the strain rate, $\sigma$ is the stress, $\mu$ is viscosity, $Q_d( = 125$ kJ/mol) is the activation energy for diffusion of zinc in aluminum, $R$ is universal gas constant, $T$ is the temperature, and $A_o$ is an Arrhenius exponent.

For evolving a custom constitutive relation between stress and strain rate, the following function set $\mathcal{F} = \{+, -, *, /, \hat{}, \exp, \sin\}$ and the terminal set $\mathcal{T} = \{\dot{\epsilon}, T, \exp[1/(RT)], \text{and } \mathcal{R}\}$ are used. Here $\mathcal{R}$ is an *ephemeral random constant* (Koza, 1992). Both $T$ as well as $\exp[1/(RT)]$ are used as possible temperature-related variables to see if GP can automatically decide between the more likely useful primitive function $\exp[1/(RT)]$ over the less likely $T$. The output of the candidate program is the ratio of stress and viscosity, that is, $\sigma/\mu$.

The fitness of a solution is computed as the absolute error between the predicted and experimental data for $\sigma/\mu$:

$$f = \frac{1}{M}\sum_{i=1}^{M}\left|\left(\frac{\sigma}{\mu}\right)_{\text{pred}} - \left(\frac{\sigma}{\mu}\right)_{\text{expt}}\right| \tag{3.2}$$

where $M$ is the number of experimental data points. This fitness function is one of the obvious choices for data-fitting problems. Here we note that experimental uncertainties can be incorporated into the fitness function in a straightforward manner. For example, a Gaussian noise (or other models of the uncertainty) can be added to the fitness function, which would prevent GP from over-fitting the data. Alternatively, if any of GP-regressed data is within the error-bar of its corresponding measured data, we can set the error to zero.

First, we start by obtaining a constitutive relation between stress-strain rate using only low-strain-rate data. That is, we only use experimental data obtained for strain rate of $10^{-3}$ s$^{-1}$. This is because the power-law relation described by Padilla et al. (2003) was fit only to the low-strain-rate data. Therefore, we want to verify if GP could find a similar relation as that of Equation 3.1, and expect that this linear regression should be trivially reproduced—albeit by a highly non-trivial approach. We find that GP does indeed discover a stress-strain rate relation that is in agreement with the constitutive relation of Padilla et al. (2003) and other power-law relations used to describe creep in metals (Kassner & Pérez-Prado, 2000; Mukherjee, 1975):

$$\dot{\epsilon} = c_o \exp\left(\frac{1}{RT}\right)\left(\frac{\sigma}{\mu}\right)^{4.55}, \tag{3.3}$$

where $c_o \approx A_o \exp(Q_d)$ is a constant.

The above results are based on 10 independent runs of GP (which took wall time of 13–41 seconds per GP run on a 1.67 GHz AMD Athlon XP workstation with 2 parallel jobs) and in all 10 runs similar relation as above were obtained, suggesting that the above expression might be an optimum. Furthermore, GP was able to select the appropriate form for incorporating the effect of temperature in the constitutive relation. The comparison of the constitutive relation developed by GP and that developed by Padilla et al. (2003) (Equation 3.1) to experimental data is shown in

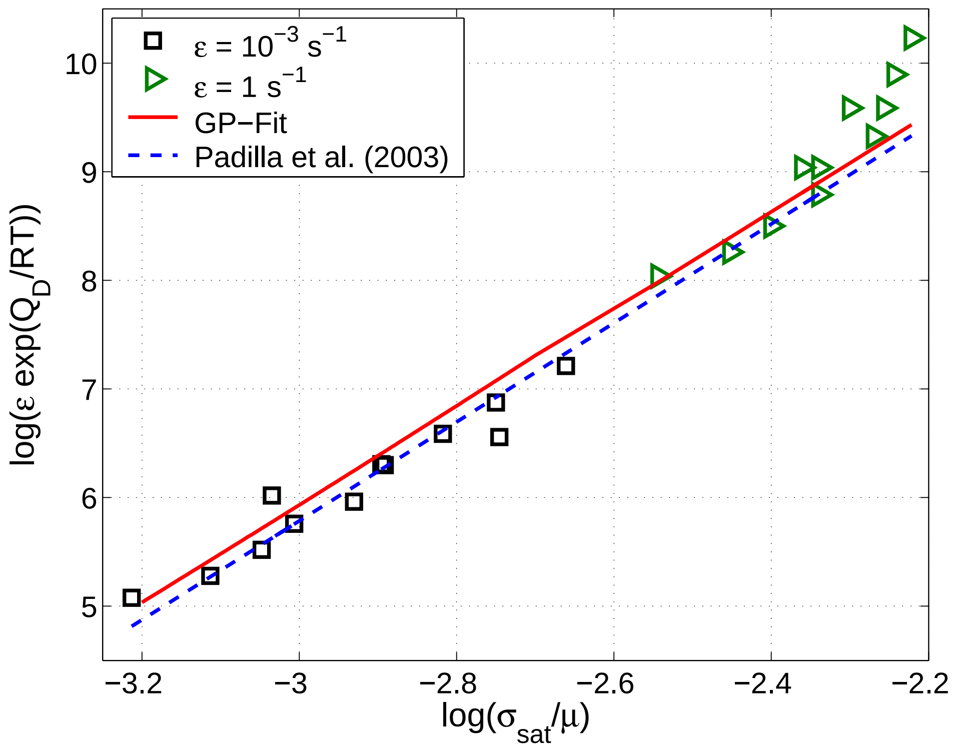


Figure 3.2: Constitutive relation developed by GP (Equation 3.3) and used by Padilla et al. (2003) (Equation 3.1) between stress and strain rate for accounting variations in microstructural deformation. Only low-strain-rate data is used for the symbolic regression via GP.

.

Figure 3.2.

Now, more importantly, GP regression is used to fit both the low-strain-rate and high-strain-rate data to evolve a stress–strain-rate constitutive relation. We are interested in finding out if GP can identify a single relation to fit both the data sets. Note that the GP does not have any knowledge of the two different sets of data, that is, high- or low-strain rates. As far as GP is concerned, there is no qualitative difference between high-strain-rate and low-strain-rate data. We used the same set of functions and terminals as in the previous case and obtained the following constitutive relation:

$$\dot{\epsilon} = \frac{c_o}{g\left[\dot{\epsilon}, \exp\left(\frac{1}{RT}\right)\right]} \exp\left(\frac{1}{RT}\right)\left(\frac{\sigma}{\mu}\right)^4\left[1 - \frac{\sigma}{\mu}\right], \tag{3.4}$$

where the denominator $g$ is a complex mathematical expression:

$$g = 10.3 \left[ 1 - 3\left(\frac{\sigma}{\mu}\right) \sqrt{6 + \sqrt{\left| A_o e^{-\frac{Q_D}{RT}} - 2\frac{\sigma}{\mu} \right|}} \cdot \left\{ 1 - 3\frac{\sigma}{\mu} \cdot A_o e^{-\frac{Q_D}{RT}} \cdot \left( 1 - A_o^2 e^{-\frac{2Q_D}{RT}} + 2\frac{\sigma}{\mu} \right) \right\} \right]. \tag{3.5}$$

The behavior and functionality of $g$ is explained in the later paragraphs. Equation 3.4 indicates a competition between the 4th-order power-law ($(\sigma/\mu)^4$) and 5th-order power-law ($(\sigma/\mu)^5$) in stress. Such power-law behavior is suggestive of possible competing mechanisms taking place during the microstructural deformation process. Furthermore, three out of ten independent GP runs (which took wall time of 34.4–58.2 seconds per GP run on a 1.67 GHz AMD Athlon XP workstation with 2 parallel jobs) yield similar constitutive relation as Equation 3.4. The rest of the seven GP runs gave either a 4th-order power-law or a 5th-order power-law equation still highlighting the competition between the two.

Notably, a 5th-order power-law represents creep in metals (Kassner & Pérez-Prado, 2000). However, it is unclear as to the physical mechanism represented by the 4th-order power-law. Nonetheless, GP regression appears to have consistently identified a competing set of power-law-mechanisms; one of which has a clear physical meaning (creep), while the other remains unknown, but, as discussed below, yield crossover from low-strain-rate to high-strain-rate.

Comparison of the constitutive relation (Equation 3.4) with experimental data is shown in Figure 3.3. The results show that the constitutive relation developed by GP does indeed agree with experimental data. Furthermore, there is a noticeable kink incorporated in Equation 3.4 at the transition between low- and high-strain-rate data points. This indicates that GP implicitly identified the transition point between high-strain-rate and low-strain-rate data. After some simplification and analysis of the denominator $g$ in Equation 3.4, we found that the kink was represented in the denominator. This indicates that GP was able to identify a missing variable and compensate for the missing variable by a step function as shown in Figure 3.4. It is important to note that this transition is well established. For example, a hyperbolic function is often used to specify the transition between low-strain-rate and high-strain-rate data (Kassner & Pérez-Prado, 2000), which approximates a step-function-like behavior in analytic models.

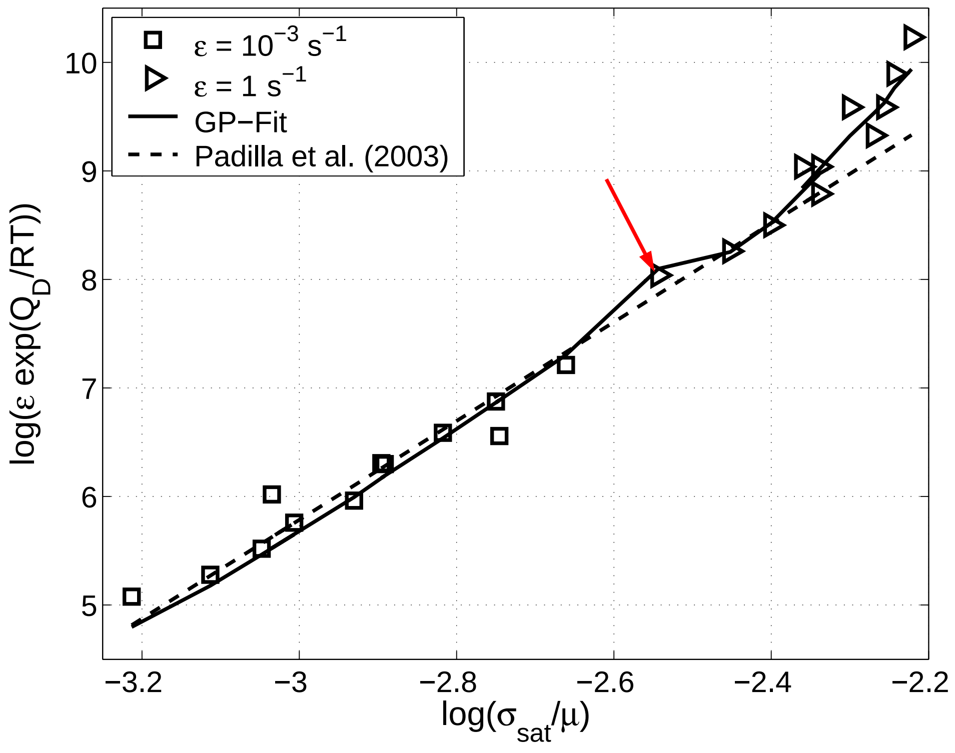
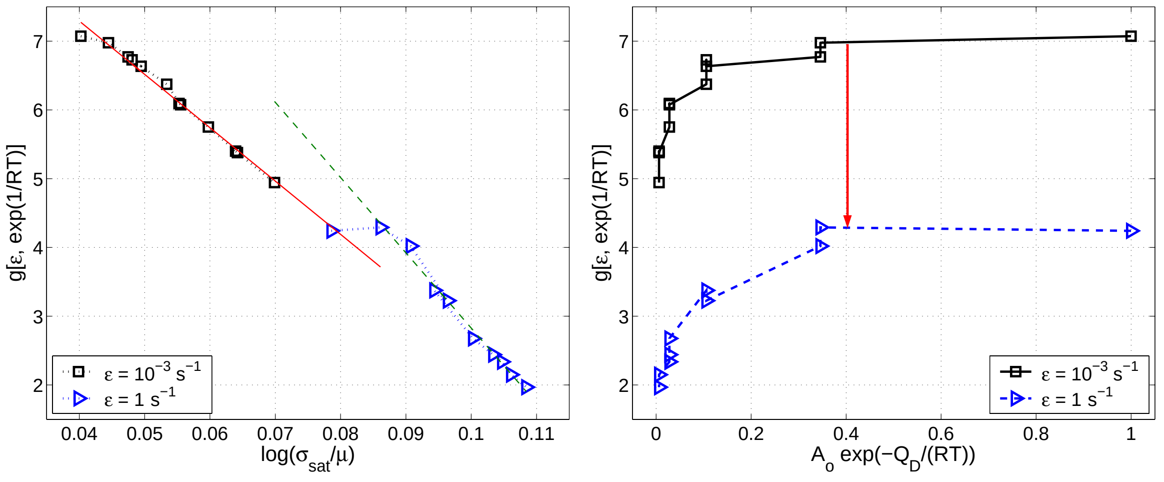


Figure 3.3: Constitutive relation developed by GP (Equation 3.4) and used by Padilla et al. (2003) (Equation 3.1) between stress and strain rate for accounting variations in microstructural deformation. Both low-strain-rate and high-strain-rate data are used for the symbolic regression via GP. The arrow indicates a crossover between low-strain-rate and high-strain-rate data.



(a) $g$ vs. $\sigma/\mu$ (b) $g$ vs. $A_o \exp\left(-Q_D/(RT)\right)$

Figure 3.4: The effect of the denominator $g(\dot{\epsilon}, \exp(1/(RT)))$ for different strain rates and temperature. The results show that the denominator is a complex way of expressing a simple step function. The step function which was automatically discovered by GP represents the transition between low-strain-rate and high-strain-rate data points.

## 3.2 Summary

The multiscaling example discussed in this section demonstrates the effectiveness of genetic programming in regressing constitutive relation between key macroscopic variables using microscopic information. Such a reduced-order model can be highly effective in multiscaling not only in space coordinates, but also in time coordinate. The constitutive relations developed here can be readily used with finite-element or finite-difference methods along the lines of Padilla et al. (2003) to simulate the hot rolling process used in the treatment of aluminum alloys. However, we stress that while the GP provided a better constitutive law that incorporates strain-rate effects, it does not reveal the underlying physics—in this case a competing creep mechanism and as yet unknown mechanism.

# Chapter 4

# Multi-Timescale Alloy Kinetics Modeling via Genetic Programming

The previous two chapters introduced basic concepts of GAs and GP and how they potentially can be used for multiscale modeling. With this background information, the next two chapters of this thesis will concentrate on multiscaling in the broad class of materials and chemistry phenomena where the accuracy of the simulations depends on the accuracy of the potential energy surfaces (PES). *Ab initio* methods compute the potential energy surfaces from scratch and are highly accurate, but are prohibitively expensive even for small systems. These methods simulate only a fraction of the real-time dynamics and take hours to days of CPU time. On the other hand, faster methods are available which can simulate real-time dynamics orders-of-magnitude longer than *ab initio* methods and do so using orders of magnitude less CPU time than *ab initio* methods. However, in order to simulate systems with *ab initio* accuracy, these faster methods need the knowledge of the *entire* potential energy surface. Here, two specific cases are considered, one where the form of the potential energy function is known (Sastry et al., 2006; Sastry et al., 2007) and the other where it is unknown (Sastry et al., 2004; Sastry et al., 2005). However, it should be noted that in both cases the *entire* potential energy surface is unknown.

This chapter considers the case where the form of the potential energy function is unknown. Specifically, we consider the utility of genetic programming in bridging molecular dynamics and kinetic Monte Carlo methods for fast and accurate alloy kinetics simulations. On one hand, molecular dynamics while accurate, is prohibitively expensive and therefore falls orders-of-magnitude short of real processing time. On the other hand, kinetic Monte Carlo methods are fast, but need the information on the *entire* potential energy surface *a priori*. Therefore, this study proposes the use of GP to regress symbolically the potential energy surface using limited molecular dynamics simulations and thereby enabling the use of kinetic Monte Carlo for complex alloy systems with orders of magnitude scale-up in simulation time.

This chapter is organized as follows. The following section gives a brief description of the proposed approach. Section 4.2 provides details of GP used to bridge molecular dynamics and kinetic Monte Carlo methods. The proposed GP-based multi-timescale modeling approach is illustrated with the help of non-trivial example of surface vacancy migration in a copper-cobalt alloy, details of which is provided in section 4.3 followed by discussion of the results. Section 4.5 provides a brief note on the CPU and time savings obtained by the proposed multi-timescale modeling approach followed by summary and conclusions.

## 4.1 Symbolically-Regressed Table KMC (sr-KMC)

Molecular dynamics (MD) is extensively used for kinetic modeling of materials. Yet MD methods are limited to nanoseconds of real time, and hence fail to model directly many processes. Recently several approaches were proposed for multiscaling (Barkema & Mousseau, 1996; Barkema & Mousseau, 2001; Cai et al., 2002; Diaz De La Rubia et al., 1998; Henkelman & Jónsson, 1999; Jacobsen, Cooper & Sethna, 1998; Sørensen & Voter, 2000; Steiner & Genilloud, 1998; Voter, 1997; Voter, 1998; Voter, Montalenti & Germann, 2002). Methods such as temperature-accelerated dynamics (TAD) (Voter, 1998) provide significant acceleration of MD but they still fall 3–6 orders of magnitude short of real processing times. These methods assume that transition-state theory applies, and concentrate only on infrequent events. An alternative approach to bridge timescales (Jacobsen, Cooper & Sethna, 1998) uses kinetic Monte Carlo (Binder, 1986) (KMC) combined with MD by constructing an *a priori* list of events (that is, “look-up table”). The table look-up KMC yields several orders of magnitude increase in *simulated* time over MD depending on temperature as discussed in section 4.5. The table of events is commonly comprised of atomic jumps, but collective motions (or off-lattice jumps), for example, see Sørensen and Voter (2000), can be added if they have been identified, for instance, by MD. Additionally, tabulating barrier energies from a list of events is a serious limitation. For example, multicomponent alloys have an impossibly large set of barriers, due to configurational dependence, making their tabulation impractical, especially from first principles. An alternative approach is calculating energies “on-the-fly” (Bocquet, 2002; Henkelman & Jónsson, 1999), but it too has serious time limitations (see Figure 4.1). Recent developments and limitations of KMC methods are given, for example, in (Bocquet, 2002).

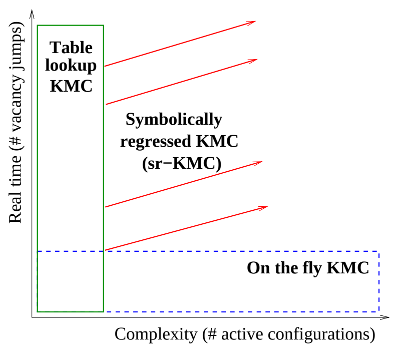

Real time (# vacancy jumps)
Table lookup KMC
Symbolically regressed KMC (sr−KMC)
On the fly KMC
Complexity (# active configurations)

Figure 4.1: Schematic illustration of simulation capabilities and bottlenecks of on-the-fly KMC, table look-up KMC, and symbolically-regressed KMC (sr-KMC).

To avoid the need or expense of explicit calculation of all activation barriers—frequent or infrequent—and thereby facilitate an effective hybridization of MD and KMC for multiscale dynamics modeling, genetic programming (GP) could be used to regress symbolically the PES (in the present, non-trivial case, saddle-point barriers only) from a limited set of directly calculated points on the PES using MD via semi-empirical, tight-binding, or *ab initio* potentials. Importantly, from only a few calculated barriers relative to the total, GP-regression provides an *in-line* barrier function for increasing number of active configurations (or complexity) as a machine-learned replacement to the "look-up table" approach. A key point is that multiscale modeling requires only relevant (often referred to as coarse-grained) information at the appropriate length or time scales. Hence, only the diffusion barriers are needed for kinetics, not the underlying atomic-scale details; how that information is obtained, direct calculation or machine-learning, is not relevant to the scaling, only that the barriers are accurate. Therefore, an accurate GP-regressed PES extends the KMC paradigm, as suggested in Figure 4.1, permitting simulation over experimentally relevant time frames, which may not be possible from standard *table look-up* or *on-the-fly* KMC. Interfacing GP with TAD-MD and/or pattern-recognition methods will further extend its applicability, for example, by finding system-specific mechanisms. This new approach is referred to as as *Symbolically-Regressed Table KMC (sr-KMC)*. Of course, sr-KMC benefits from any advances in KMC methods. In addition, GP-based symbolic regression holds promise in other multiscaling areas, for example, regressing constitutive rules (Sastry et al., 2004) and chemical reaction pathways (Sastry et al., 2006; Sastry et al., 2007). Also, as exemplified, standard basis-set regression are generally not competitive to GP for fixed accuracy due to the difficulty in choosing (that is, guessing) appropriate basis functions to represent that space, which are here configurationally-dependent diffusion barriers.

To demonstrate the potential of sr-KMC, the application of GP to a non-trivial case of vacancy-assisted migration on (100) surface of phase-separating $\mathrm{Cu}_x\mathrm{Co}_{1-x}$ at a concentrated alloy composition, that is, $x = 0.50$ is discussed. Although there are millions of configurations, only the atoms in the environment locally around vacancy and migrating atom significantly influence the barrier energies. We refer to these as the *active* configurations. The results show that GP predicts barriers within 0.1–1% error using calculated barriers of less than 3% of the total *active* configurations, depending on the type of potential (error is less for the more accurate potentials, and greater for semi-empirical). The results also show the efficacy of GP approach relative to polynomial regression, where the more complex the space the less percentage of total barriers is required to regress the potential energy surface, in contrast to standard regression. (Basically it is too difficult to guess a good basis to fit a complicated PES, but a computer can machine-learn it efficiently). These initial results hold promise to enable the use of KMC (even with realistic potentials) for increased problem complexity with a scale-up of simulation time.

## 4.2 Genetic Programming for Bridging Molecular Dynamics and Kinetic Monte Carlo

The inline barrier function is represented by a GP tree generated from the function set $\mathcal{F} = \{+, -, *, /, \hat{}, \exp, \sin\}$ and the terminal set $\mathcal{T} = \{\vec{x}, \mathcal{R}\}$. Here $\vec{x}$ is a vector representing the *active* alloy configuration, and $\mathcal{R}$ is an ephemeral random constant. Since GP is used for predicting the barriers, a tree represents a PES-prediction function that takes a configuration and ephemeral constants as inputs and returns the barrier for that configuration as output.

A tree's quality is given by its fitness $f$. For this, we calculate the barriers $\{\Delta E_{\text{calc}}(\vec{x}_1), \cdots, \Delta E_{\text{calc}}(\vec{x}_M)\}$ for $M$ random configurations $\{\vec{x}_1, \vec{x}_2, \cdots, \vec{x}_M\}$. These configurations are used as inputs to the tree and the barriers $\{\Delta E_{\text{pred}}(\vec{x}_1), \cdots, \Delta E_{\text{pred}}(\vec{x}_M)\}$ are predicted. The fitness is then computed as a weighted average of the absolute error between the predicted and calculated barriers:

$$f = \frac{1}{M} \sum_{i=1}^{M} w_i \left| \Delta E_{\text{pred}}(\vec{x}_i) - \Delta E_{\text{calc}}(\vec{x}_i) \right| \tag{4.1}$$

Table 4.1: Number of active configurations for 1st and 2nd n.n. jumps, and for 1st and 2nd n.n. active atoms.

| | **1st n.n. jumps** | **2nd n.n. jumps** |
|---|---|---|
| **1st n.n. active configurations** | 128 | 128 |
| **2nd n.n. active configurations** | 2048 | 8192 |
| **Total configurations** | $\gg 2^{100}$ | $\gg 2^{100}$ |

with $w_i = |\Delta E_{\text{calc}}|^{-1}$, which gives preference to predicting accurately lower-energy (most significant) events over higher-energy events.

The GP population which represents candidate PES prediction functions is initially created using the *ramped half-and-half* method. We use an *s-wise tournament selection*, *subtree crossover*, *subtree mutation*, and *point mutation*.

## 4.3 Vacancy-Assisted Surface Diffusion in Binary Alloy

We consider the prediction of diffusion barriers for vacancy-assisted migration on (100) surface of phase-separating fcc $\text{Cu}_x\text{Co}_{1-x}$[1] for a concentrated alloy with $x = 0.5$, a non-trivial case with many configurations that affect the diffusional barrier height. The system consists of five layers with 100 to 625 atoms in each layer, see Figure 4.2. The bottom three layers are held fixed to their bulk bond distances, while the top layers are either held fixed (as a test) or fully relaxed via MD. The input to the barrier regression/prediction function, $\vec{x} = \{x_j\}$ is a binary-encoded vector sequence, where $x_j = 0$ (1) represents a Cu (Co) atom. We consider only first and second nearest-neighbor (n.n.) jumps, along with 1st (as a test) and 2nd n.n. environmental atoms in the active configuration, as shown in Figure 4.2. This system already exhibits large complexity and is still small enough so that table look-up and GP-regressed KMC can be implemented and directly compared. Table 4.1 gives the number of active configurations when 1st and 2nd n.n. environments are considered for a binary alloy.

Note that, for the sake of simplicity, we have restricted the dynamics of the atoms to vacancy-assisted jumps. This simplification, however, does not limit the generality of our demonstration. Indeed, as long as the list of possible jumps is known *a priori*, which is a standard requirement for

[1] We note that Cu-Co alloys are characterized by a small size misfit (King, 1966), and the effectiveness of the GP-based symbolic regression for Au-Ni or Cu-Ag alloys, which have a much larger size misfit is left as a future study

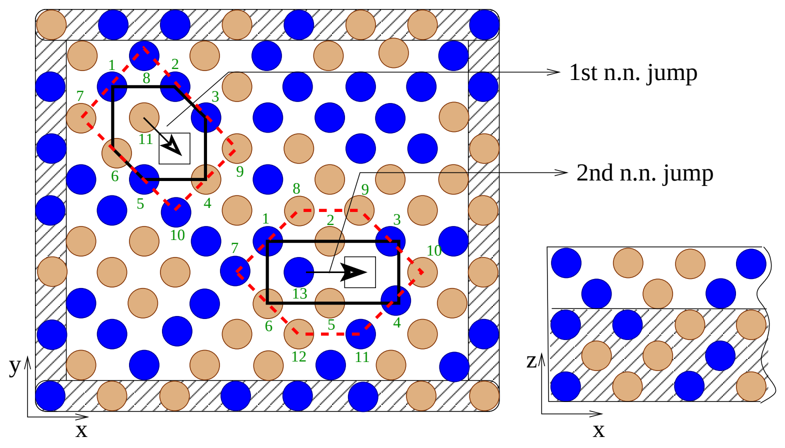


Figure 4.2: Sketch of simulation cell for vacancy-assisted migration on (100)-surface of an fcc binary alloy. Atoms in all but the bottom layers and the boundary can fully relax. The solid (dashed) lines around the migrating atom and vacancy represent 1st (2nd) n.n environmental atoms. Atoms for 1st (2nd) n.n. jumps are labeled from $1-7$ $(1-13)$ as they occur in the encoded vector $\vec{x}$ along with the barrier energy $\Delta E(\vec{x})$.

lattice KMC simulations (Sørensen & Voter, 2000), the present approach can easily be extended. One would simply use one symbolically-regressed function for activation energies corresponding to each migration mechanism, for instance, di-vacancy migration, ad-atom migration and atom exchanges at step surfaces.[2]

We model atomic interactions with a Morse potential (Girifalco & Weizer, 1959), which includes at least 2nd n.n. interactions, and a tight-binding potential within a second-moment approximation (TB-SMA) (Cleri & Rosato, 1993; Levanov et al., 2000; Mazzone & Rosato, 1997; Stepanyuk et al., 2000; Stepanyuk et al., 2001; Stepanyuk et al., 2001). The atomic interactions of TB-SMA range over 5th nearest neighbors, which are longer range and, hence, more computationally intensive (as in timings given later) but more accurate (Roussel, 2005). For the TB-SMA with interactions up to 5th n.n. atoms, we only consider up to 2nd n.n. environmental atoms as variables for GP-regression of the barrier function (which is just to minimize the large variable space). If the TB-SMA potential is truncated to 2nd n.n. interactions, its timings would be approximately equal to those of the Morse potential, but it requires additional terms such as those to ensure continuity of potential and no truncation forces (Roussel, 2005). To validate interactions, we model vacancy-assisted migration

[2]Although di-vacancy diffusion is faster than single-vacancy diffusion in metals, a di-vacancy will have a shorter lifetime due by reaching vacancy sinks more quickly; so single-vacancy diffusion is relevant to long-time diffusion, even in our non-trivial case study.

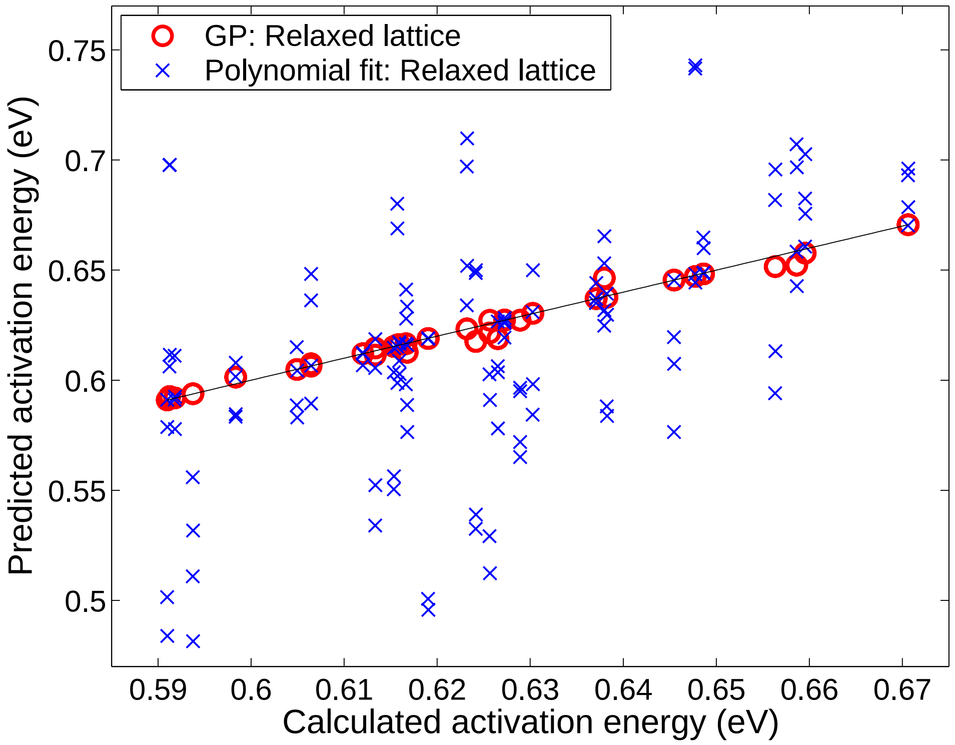


Figure 4.3: Activation energies (in eV) predicted by regression. GP (circles) and a quadratic polynomial (crosses) are compared to the calculated (Morse) barriers for 1st n.n. jumps on (100)-surface of $Cu_{0.5}Co_{0.5}$ for relaxed lattices. As a simple test, only first n.n. environments are considered in the active configuration. The line is a guide for the eye.

on (100)-surface of Cu and consider only first n.n. jumps. The predicted barrier for n.n. vacancy jumps with fully relaxed lattice in Cu is 0.39 eV for Morse and 0.45 eV for TB-SMA, agreeing with $0.42\pm0.08$ ($0.47\pm0.05$) from *ab initio* (EAM) (Boisvert & Lewis, 1997) calculations.

## 4.4 Results: Efficacy of GP Regression

For simplicity, we begin by considering only seven surface 1st n.n. environmental atoms, that is, six neighboring atoms of both the diffusing atom and vacancy—for a total of seven atoms that can be either Cu or Co, yielding 128 *active* configurations. The environment outside this configuration is fixed. About 20, that is, 16%, different active configurations are randomly chosen and their barriers are computed using the conjugate-gradient method and are used in the GP fitness function, see Equation 4.1. The barriers predicted by GP for the relaxed configurations are compared to the exact values in Figure 4.3. We note that the prediction error for rigid lattice case ($0.4\pm0.04\%$) is significantly less than that for relaxed lattice case ($2.8 \pm 0.08\%$). Due to the weighting used in the fitness function, GP predicts barriers for most significant, low-energy events more accurately than for less significant, higher-energy events.

Figure 4.3 also compares the barriers predicted by GP to those predicted by a least-squares fit quadratic polynomial, showing clearly the inadequacy of standard basis-set regression for alloys. Furthermore, while GP requires only 16%, the quadratic (cubic) polynomial fit needs 28% (78%) of the barriers. (For clarity, the large percentage needed for the polynomial regression arises because of the number of variables. In the quadratic case for this simple test case, the variable $\vec{x}$ has 7 occupation components of 0 or 1; so we need to fit coefficient terms from *1 constant*, *7 linear*, and *7 (21) (off-)diagonal quadratic*, for 36/128 or 28%.) In limited cases, such as dilute $\mathrm{Fe}_{1-x}\mathrm{Cu}_x$, the barriers can be predicted via a simple polynomial fit (Bouar & Soisson, 2002).

To test the scalability of GP with *active* configuration size, we consider the 2nd n.n. jumps and 1st and 2nd n.n. environmental atoms in the *active* configuration. As shown in Table 4.1, there are a total of 8192 configurations. The energies predicted by GP are compared with direct calculations in Figure 4.4, along with the error in the Morse (worst case) example. The GP predicts the barriers for most significant events with less than 0.1% error by fitting to energies from only 3% (that is, 256/8192) of the active configurations (see error definition[3]). From Figure 4.4 clearly the non-additive and non-linear tight-binding potential has less error than Morse case for even fewer barriers in the learning set. In comparison, a cubic polynomial fit requires energies for $\sim$6% of the configurations, predicting the barriers with 2.5% error for the most significant events.

The results shown in Figures 4.3 and 4.4 clearly demonstrate the effectiveness of GP in predicting the potential energy surface, with high accuracy and little information. As expected, since the regression and barrier calculation are nearly independent, the GP performance does not depend on the potentials used, for example, Figure 4.4 shows results for both Morse, and non-additive and non-linear tight-binding potentials. The regression only requires a database of barriers and has no knowledge (nor the need) of the underlying potential used. We also find that the GP performance

[3] The average relative error for $N'_{\mathrm{cfgs}}$ configurations within the desired energy range is given by

$$\bar{\varepsilon}_{\mathrm{rel}} = \frac{100}{N'_{\mathrm{cfgs}}} \sum_{i=1}^{N'_{\mathrm{cfgs}}} \left| \frac{\Delta E_{\mathrm{pred}}(\vec{x}_i) - \Delta E_{\mathrm{calc}}(\vec{x}_i)}{\Delta E_{\mathrm{calc}}(\vec{x}_i)} \right|,$$

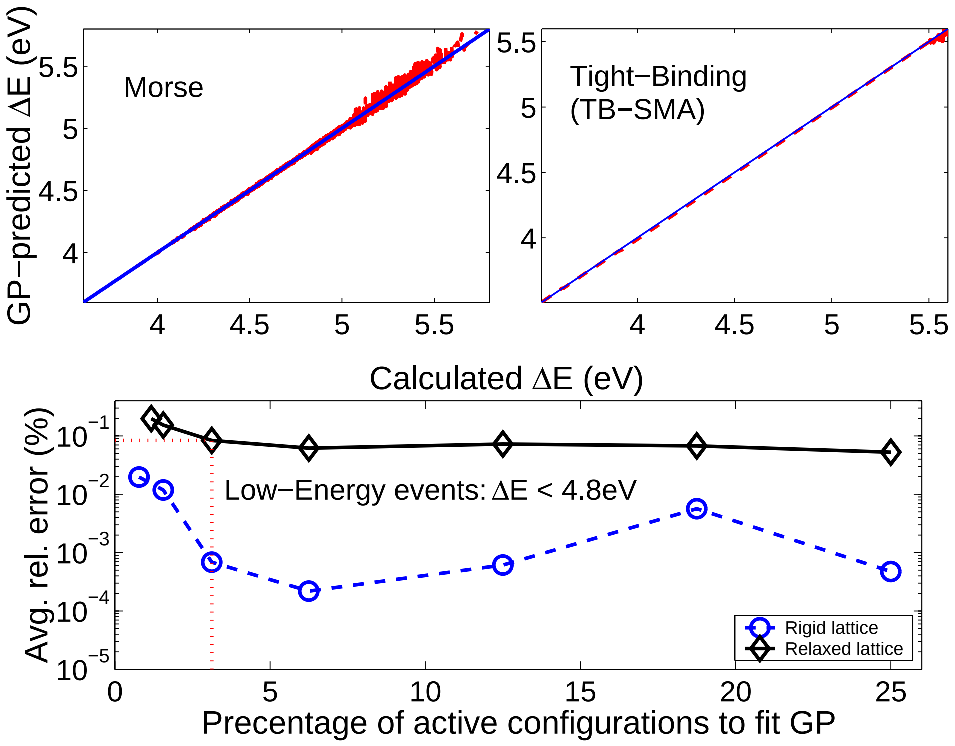


Figure 4.4: (Upper) Calculated vs. GP-predicted (Morse and TB-SMA) barriers (in eV) for 2nd n.n. jumps on relaxed $Cu_{0.5}Co_{0.5}(100)$ from configurations out to 2nd n.n. (Lower) Morse GP-barrier error vs. number of configurations used in learning set: 0.1% (1%) error for most- ($\Delta E < 4.8$ eV) and least- ($\Delta E > 4.8$ eV) significant events from only 3% of active configurations; TB-SMA GP error is 0.1% for less than 3%.

is independent of the configuration set used in calculating the fitness function, the order in which they are used, and the labeling scheme used to convert the configuration into a vector of inputs. Differences in the activation-energy scale on the PES prediction via GP are also negligible. That is, even though the barriers for the 1st and 2nd n.n. jumps differ by an order of magnitude, GP predicts the barriers with similar accuracy. Moreover, for more complex, cooperative effects, such as island diffusion via surface dislocations (Hamilton, Daw & Foiles, 1995), sr-KMC could be interfaced with pattern-recognition methods (Trushin et al., 2005), as well as long-range fields.[4]

[4]For long-range fields (for example, elastic fields from coherent interfaces, such as multilayers or precipitates), a description based solely on local configurations may have to be extended, say, with phase field methods.

## 4.5 Relative Time Comparisons: The Critical Success

The CPU time savings by coupling GP-regressed *inline barrier function* with KMC (that is, sr-KMC) are simple to estimate. For our example, with $\sim$33 times fewer calculated barriers GP symbolically regresses an inline barrier function—rather than the complete look-up table—and thus, sr-KMC provides direct CPU savings of $\sim$100 over table look-up methods. Additionally, each sr-KMC time step requires only $10^{-3}$ CPU-seconds for an *inline function* evaluation, as opposed to *on-the-fly* KMC that requires seconds (empirical potentials) to hours (quantum methods), providing a gain of $10^4$–$10^7$ CPU-seconds. For our example, one relaxed barrier calculation takes $\sim$10 secs ($\sim$1800 secs) for Morse (tight-binding).

An important question, especially for bulk diffusion, is how the gain from sr-KMC scales with system complexity (for example, the range of environment considered, the active environment, or additional alloying components). While we cannot fully answer this question yet, in the present study it is remarkable and promising that the fraction of explicit barrier calculations required by sr-KMC decreases as the number of active configurations increases.

For completeness sake, we note the simulation time enhancements over MD (from nano-seconds to seconds) by sr-KMC. (Of course, if a complete *look-up* table is also calculated, the estimate is the same.) With event occurrence following a Poisson distribution, the real time in KMC is given by (Binder, 1986; Fichthorn & Weinberg, 1991)

$$\tau_r = \sum_{j=1}^{N_{KMC}} \frac{-\ln\, U}{\sum_{i=1}^{N_{cfgs}} \nu_0 e^{-\beta \Delta E(\vec{x}_i)}} \tag{4.2}$$

where $N_{KMC}$ is the number of Monte Carlo steps, $U \in (0, 1]$ is a uniform random number, $N_{cfgs}$ is the number of active configurations. Using $\nu_0 \approx 27 \times 10^{12}$ Hz for Cu-Co (Boisvert & Lewis, 1997), Equation 4.2 gives—per time step of KMC relative to MD (assuming an MD time-step of $10^{-15}$ s)—an increase in *simulated* time of $10^9$ at 300 K, $10^4$ at 650 K, and $10^{2.3}$ at 1000 K. Direct timing runs from KMC agree with these estimates. So the key increase in timings comes from learning the table from very few barriers, saving all the calculating time, and allowing more complex problems to be addressed potentially.

## 4.6 Summary

Here we have presented a new approach using a machine-learning method based upon symbolic regression via genetic programming to determine, accurately, and with little information, complete details of the potential energy surface and output as an *inline* function. We have shown on a non-trivial example of vacancy-assisted migration on a surface of fcc $\mathrm{Cu}_x\mathrm{Co}_{1-x}$ that GP predicts all barriers with 0.1% error from calculations for less than 3% of active configurations, independent of type of potentials used to obtain the learning set of barriers via molecular dynamics. The genetic programming-based KMC approach avoids the need or expense of calculating the entire potential-energy surface, is highly accurate, and leads to significant scale-up in real simulation time for complex cases as it enables the use of KMC and, more importantly, leads to a significant reduction in CPU time needed for KMC ($>$ 7-orders of magnitude for quantum-based calculations), not possible from any other current means. For alloys, we believe the number of explicit barrier calculations for the learning set can be reduced further by over an order of magnitude ($\sim$0.3% of the *active* configurations) using local cluster expansion methods (Van der Ven & Ceder, 2001).

The genetic programming regression allows atomic-scale information (in our example, diffusion barriers on the potential energy surface) to be included in a long-time kinetic simulation without maintaining a detailed description of the all atomistic physics, as done within molecular dynamics. Our multiscale approach does not require finding pertinent “hidden variables”, but just uses necessary information at the appropriate time scale (or length scale)—a coarse-graining of sorts. We emphasize that the genetic programming is non-trivially regressing an *inline* function and its coefficients that approximates the potential-energy surface, and its efficacy over standard basis-set regression was made clear.

# Chapter 5

# Fast and Accurate Chemical Reaction Simulations via Multiobjective Genetic Algorithms

Photochemical reactions are fundamental in many settings such as biological (for example, photosynthesis and vision) and technological (for example, solar cells and LEDs), and the associated dynamics are energetically subtle, requiring highly accurate descriptions of the molecular forces. Reliable quantum chemistry predictions are costly even for small molecular reactions, but rapidly approach the impossible in complex environments, such as in solvents, in solid cages of zeolites, or with protein ion channels. Hence, having substantially faster semiempirical potentials that accurately reproduce high-level quantum chemistry results would make it possible to address critical biological processes and technologically chemical reactions, or dramatically reduce searches for potentially technological useful light-activated reactions.

The *ab initio* multiple spawning (AIMS) methods, which simultaneously solve both the electronic and nuclear Schrödinger equations (Ben-Nun, Quenneville & Martinez, 2000), while very flexible and accurate, can be computationally expensive, especially for large molecules. On the other hand, the semiempirical methods (Dewar & Thiel, 1977; Dewar et al., 1985; Stewart, 1989), which neglect many two-electron integrals of *ab initio* methods and replace others with parameters, while significantly less expensive than AIMS, have an accuracy that depends on the accuracy of the semiempirical parameters. Notably, semiempirical potentials have traditionally had the critical parameters hand-designed and optimized so as to predict ground-state energies—not excited-state energies. Well established parameter sets in quantum chemistry databases (Dewar & Thiel, 1977; Dewar et al., 1985; Stewart, 1989) (known by acronyms of MNDO, AM1, and PM3) and software (Andersson et al., 2001; Stewart, 1999; Werner et al., 2002) give useful information on ground-state energies. However, they fall short of yielding globally accurate potential energy surfaces critical for accurate photochemical reaction simulation. For example, in ethylene, AM1 or PM3 parameter sets incorrectly obtain the so-called pyramidalized structure as the excited-state minimum. Thus, these carefully established parameter sets oftentimes yield inaccurate potential energy surfaces, resulting in unphysical excited-state reaction dynamics.

Therefore, in order to obtain globally accurate energetics, the parameter sets have to be reoptimized for different classes of molecules using a very limited set of *ab initio* and experimental data (Owens, 2004; Toniolo, Thompson & Martinez, 2004). The reparameterization strategy is a promising way to extend direct dynamics simulations of photochemistry to a more realistic multi-picosecond time scales. The reoptimization problem is massively multimodal and involves multiple conflicting and competing objectives, such as minimizing the difference between calculated and predicted energies, gradients of energies, and stationary-point geometries. Previous semiempirical parameter optimization attempts, mostly based on a staged fixed-weight single-objective optimization, have been but partially successful (Brothers & Merz, Jr., 2002; Cundari, Deng & Fu, 2000; Hutter, Reimers & Hush, 2002; Owens, 2004; Rossi & Truhlar, 1995; Toniolo, Thompson & Martinez, 2004). In this chapter, we propose the use of multiobjective genetic algorithms (MOGAs) to reoptimize the parameter sets using a very limited set of ab initio and experimental data to yield globally accurate potential energy surfaces and excited-states, yielding accurate photochemical reaction dynamics (Sastry, Johnson, Thompson, Goldberg, Martinez, Leiding, & Owens, 2006; Sastry, Johnson, Thompson, Goldberg, Martinez, Leiding, & Owens, 2007).

This chapter is organized as follows. In the next section, a brief introduction to reparameterization of semiempirical methods is provided followed by a description of the multiobjective genetic algorithm used for reparameterization in section 5.2. Section 5.3 presents key results obtained via multiobjective genetic algorithms in yielding optimal semiempirical parameters that are stable to random perturbation, yield accurate configurational energies, and yield *ab initio* quality excited-state dynamics.

## 5.1 Reparameterization of Semiempirical Methods

A brief background of current computational methods for performing excited-state dynamics in photochemistry is provided in this section and a more detailed overview is given elsewhere (Owens, 2004; Toniolo et al., 2005; Toniolo, Thompson & Martinez, 2004) and the references therein. As mentioned earlier, a comprehensive understanding of the photochemistry of molecules requires bridging the gap between molecular dynamics and quantum chemistry, and quantum dynamics simulations are required to simultaneously solve both the nuclear and electronic Schrödinger equations (Toniolo et al., 2005). Additionally, the potential energy surfaces (PESs) must be of high quality and very robust because the portions of the PES that are critical to the behavior of the molecule may be far removed from the Franck-Condon region.

The *ab initio* multiple spawning (AIMS) method has been developed in order to address such problems (Ben-Nun & Martinez, 2002; Ben-Nun, Quenneville & Martinez, 2000). While the AIMS method is extremely flexible and can describe quantum mechanical phenomena such as tunneling and non-adiabatic transitions, it is computationally very expensive because of a large number of ab initio electronic structure calculations involved, making long-time dynamics simulations highly improbable, if not impossible.

In order to retain the flexibility of *ab initio* electronic structure methods with less computational cost, semiempirical methods—which ignore some two-electron integrals and use parameters for others—were developed (Dewar & Thiel, 1977; Dewar et al., 1985; Stewart, 1989). Instead of calculating each electron integral, semiempirical methods make certain approximations. First, many of the two electron integrals (those on three or four centers) are assumed to be zero. Also, the remaining one and two electron integrals are replaced with analytic functions that depend on a set of parameters. The semiempirical parameters which are different for each element have been optimized using ground state properties for a small set of molecules without the use of fractional occupation molecular orbitals. Standard parameter sets, such as MNDO (Dewar & Thiel, 1977), AM1 (Dewar et al., 1985), and PM3 (Stewart, 1989), yield useful information concerning the locations of the minimal energy conical intersections (MECIs), which often dominate photochemical reactions. However, they often yield erroneous energetics, resulting in unphysical dynamics. Therefore, the parameter sets must be reoptimized using a very limited set of *ab initio* and experimental data to obtain an acceptable and an accurate description of the photodynamics. The reparameterization strategy is a promising way to extend direct dynamics simulations of photochemistry to multi-picosecond time scales. It is also reasonable to expect *transferability* of the parameter sets optimized on simple molecules such as ethylene and benzene to other complex molecules such as stilbene and phenylacetylene dendrimers. Furthermore, the reparameterization approach opens up the possibility of accurate simulations of photochemistry in complex environments such as proteins and condensed phases.

It should be noted that while the reparameterization procedure only fits energetics of a few important stationary molecular geometries, much larger portions of the PESs will be accessed during dynamics simulations. Therefore, the semiempirical methods have to incorporate enough of the fundamental chemical physics to generate at least qualitatively correct global PESs. While it is possible to include geometries and energetics of the MECIs in the reparameterization, the strategy of using relatively little *ab initio* data is mandatory if reparameterization is to be applicable for larger molecules, where *ab initio* data is extremely expensive to obtain. Therefore, we intentionally use a minimal set of energies and gradients at ground state optimized geometries in our reparameterization.

Here, we will concentrate on reparameterizing two simple molecules, which are fundamental building blocks of organic molecules: ethylene and benzene. The small size of ethylene has many advantages: First, semi-empirical calculations can be run very quickly so a large number of reparameterization runs can be conducted. Second, the small number of atoms, basis functions, and possible geometries imply that the results may be less complex and more easily interpretable. Lastly, the size and simplicity enables the reoptimized parameter sets to be amenable for further analysis of ethylene dynamics and for transferability to stilbene or conjugated polyenes. However, despite its simplicity, ethylene has an associated set of ethylidene geometries that can be used to evaluate performance of the reoptimized parameter sets in calculations for which they were not optimized. Benzene plays an important role in photochemistry and photophysics of aromatic systems and has been extensively studied both experimentally and theoretically (Toniolo, Thompson & Martinez, 2004).

Following Owens (Owens, 2004) for ethylene reparameterization, we use energetics for the ground state planar and ethylidene geometries, twisted geometry (see Figure 5.1) on the excited state as well as the gradients on the excited and ground states. The *ab initio* results used for reparameterization are taken from previously reported calculations (Ben-Nun & Martinez, 2000) and are calculated using CASSCF(2/6)*SDCI wavefunctions with the aug-cc-pVDZ basis set. Following Toniolo, Thompson and Martinez (2004), for reparameterization of benzene, we use four

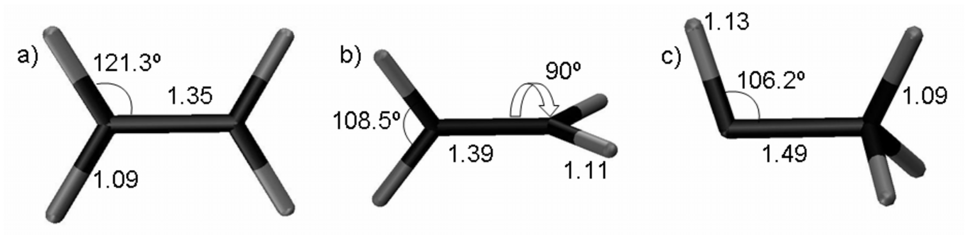


Figure 5.1: Ground state optimized geometries and important minimal energy conical intersections (MECIs) for ethylene. Reproduced with permission from A. Thompson

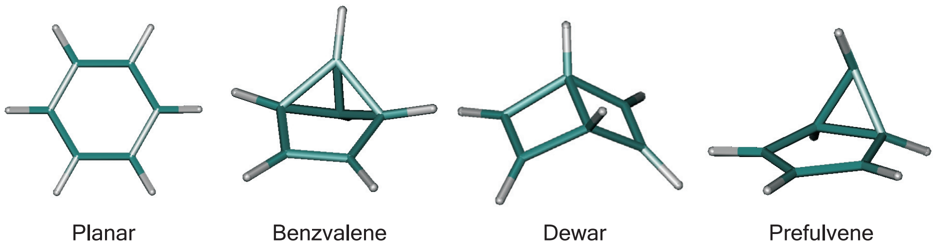


Figure 5.2: Ground state optimized geometries and important minimal energy conical intersections (MECIs) for benzene.

important local minima on $S_0$: planar, Dewar benzene, prefulvene and benzvalene (see Figure 5.2) and use *ab initio* calculations and experimental results reported in and used by Toniolo, Thompson and Martinez (2004).

A floating occupation molecular orbital-configuration interaction (FOMO-CI) calculation is used to describe electronic excited states (Granucci & Toniolo, 2000). The molecular orbitals are optimized using an SCF calculation in which the occupation numbers of some of the orbitals are allowed to fluctuate. These occupation numbers are updated at each SCF iteration according to

$$O_i = \int_{-\infty}^{\varepsilon_F} \sqrt{\frac{2}{\pi\omega^2}}\, e^{-\frac{(\varepsilon-\varepsilon_i)}{2\omega^2}}\, d\varepsilon, \tag{5.1}$$

where $O_i$ is the occupation number of orbital $i$, $\omega$ is the width of the Gaussian function, $\varepsilon_i$ is the energy of orbital $i$, and $\varepsilon_F$ is the Fermi level energy determined such that

$$\sum_i O_i = N_{electrons}, \tag{5.2}$$

where $N_{electrons}$ is the number of electrons in the system. While all orbitals could be allowed to have fractional occupation, only the occupation numbers of orbitals in the active space are allowed to vary. The benefit of this method is that it is a fast, low-cost way of generating better virtual orbitals, which improves the treatment of excited states.

To calculate the excited state properties, multiple configurations need to be included in the wavefunction. Using the floating occupation molecular orbitals, we use a complete active space configuration interaction (CAS*CI) wavefunction. In this technique, all the configurations that involve excitations within the active orbitals defined in the calculation are included. The energies of each of the configurations are calculated using the fractionally occupied orbitals, but the orbitals are not reoptimized at each step.

The semiempirical calculations are performed with a developmental version of MOPAC2000 (Stewart, 1999), while the *ab initio* results are performed with MOLPRO (Werner et al., 2002) and MolCas (Andersson et al., 2001), details of which are beyond the scope of this study. For both ethylene and benzene, 11 parameters for carbon—$U_{ss}$, $U_{pp}$, $\beta_s$, $\beta_p$, $\zeta_s$, $\zeta_p$, $G_{ss}$, $G_{sp}$, $G_{pp}$, $G_{p2}$, and $H_{sp}$—are reoptimized. Following earlier studies (Owens, 2004; Toniolo, Thompson & Martinez, 2004), the core-core repulsion parameters—$\alpha$, $a_i$, $b_i$, and $c_i$—are not reoptimized.

With this general overview of reparameterization of semiempirical methods, the multiobjective genetic algorithm used in reparameterization is described in the next section.

## 5.2 Multiobjective Genetic Algorithms for Optimizing Semiempirical Methods

As mentioned earlier, reparameterization of semiempirical methods involves optimizing the semiempirical parameters based on a very limited set of *ab initio* and/or experimental data. We use a real-valued encoding to represent the 11 parameters—$U_{ss}$, $U_{pp}$, $\beta_s$, $\beta_p$, $\zeta_s$, $\zeta_p$, $G_{ss}$, $G_{sp}$, $G_{pp}$, $G_{p2}$, and $H_{sp}$—of the semi-empirical methods.

The two fitness functions involve minimizing the absolute error in energies and energy-gradients for a very limited set of excited-state and ground-state configurations either calculated by *ab initio*

Table 5.1: The PM3 values and the lower and upper bounds on the percentage deviation from PM3 values used in generating the initial GA population.

| SE parameter | PM3 values | Lower bound (%) | Upper bound (%) |
|---|---|---|---|
| $U_{ss}$ | -47.270320 | -56.7243840 (-20%) | -37.8162560 (20%) |
| $U_{pp}$ | -36.266918 | -43.5203016 (-20%) | -29.0135344 (20%) |
| $\beta_s$ | -11.910015 | -14.2920180 (-20%) | -9.5280120 (20%) |
| $\beta_p$ | -9.802755 | -12.7435815 (-30%) | -6.8619285 (30%) |
| $\zeta_s$ | 1.565085 | 1.2520680 (-20%) | 1.8781020 (20%) |
| $\zeta_p$ | 1.842345 | 1.4738760 (-20%) | 2.2108140 (20%) |
| $G_{ss}$ | 11.200708 | 7.8404956 (-30%) | 14.5609204 (30%) |
| $G_{sp}$ | 10.265027 | 8.2120216 (-20%) | 12.3180324 (20%) |
| $G_{pp}$ | 10.796292 | 8.6370336 (-20%) | 12.9555504 (20%) |
| $G_{p_2}$ | 9.042566 | 7.2340528 (-20%) | 10.8510792 (20%) |
| $H_{sp}$ | 2.290980 | 1.1454900 (-50%) | 3.4364700 (50%) |

methods or obtained by experiments, and those predicted by semiempirical methods. That is,

$$f_1(\mathbf{x}) = \sum_{i=1}^{n_c} |\Delta E_{0,i} - \Delta E_{SE,i}(\mathbf{x})| \tag{5.3}$$

$$f_2(\mathbf{x}) = \sum_{i=1}^{n_g} |\nabla E_{0,i} - \nabla E_{SE,i}(\mathbf{x})| \tag{5.4}$$

where $x$ represents the semiempirical parameters to be optimized, $n_c$ is the number of configurations, and $n_g$ is the number of gradient-energy data used in reparameterization. $\Delta E_{0,i}$ and $\Delta E_{SE,i}$ are the differences in energy between the geometry $i$ and the reference structure (planar ethylene and benzene) calculated by *ab initio* and semiempirical methods, respectively.

For ethylene, we make the restriction that the excited state at the ground state planar geometry must have the correct state symmetry. For benzene, in the first objective we also include geometry difference between the reparameterized semiempirical geometries and the *ab initio* geometries by calculating the sum-squared differences between the corresponding atoms after the molecules have been rotated and translated such that they are in maximum coincidence. $\nabla E_{0,i}$, and $\nabla E_{SE,i}$ represent the excited-state energy gradients using *ab initio* and semiempirical methods, respectively. The semiempirical calculations are done within a development version of MOPAC2000 using a CAS(2/2)*CI wavefunction. All geometries are minimized and then energies and gradients are calculated at this minimum on the potential energy surface.

The initial population of candidate solutions is usually generated randomly across the search space. However, domain-specific knowledge or other information can be easily incorporated in the generation of the initial population. In reparameterization of semi-empirical potentials, the initial population is randomly generated within a certain percentage (20–50%) of the PM3 parameter values (Stewart, 1989) (see Table 5.1). The parameter bounds are restricted around the PM3 set so as to maintain a reasonable representation of the ground-state potential energy surface.

The multiobjective GA used in this study is the non-dominated sorting genetic algorithm II (NSGA II) (Deb et al., 2002), with binary ($s = 2$) tournament selection without replacement (Goldberg, Korb & Deb, 1989; Sastry & Goldberg, 2001), simulated binary crossover (SBX) (Deb & Agarwal, 1995; Deb & Kumar, 1995)—which models the behavior of single-point crossover in binary genetic algorithms—with $\eta_c = 5$, and crossover probability $p_c = 0.9$, and a polynomial mutation (Deb, 2001) with $\eta_n = 10$ and mutation probability $p_m = 0.1$.

In tournament selection without replacement with tournament size $s$, $s$ chromosomes are chosen at random without replacement and entered into a tournament against each other. The best (fittest) individual in the group of $s$ chromosomes wins the tournament and is selected into a *mating pool* for evolving new solutions. The tournaments are continued till all the individuals in the population have competed once, at which point there are exactly $n/2$ chromosomes in the mating pool, where $n$ is the population size.[1] The entire process in repeated again—but this time the competitors in each tournament will be different—so that the mating pool has $n$ chromosomes.

In SBX, individuals in the mating pool are divided into random pairs and each pair undergoes recombination with a probability $p_c$. For each pair participating in the crossover, each gene (or variable) undergoes contracting or expanding crossover operation with a probability 0.5. Therefore, for each pair of chromosomes undergoing recombination on an average half of the genes are modified using either contracting or expanding crossover operations. Assuming that two parents $p_1$ and $p_2$ recombine to yield offspring $c_1$ and $c_2$, let $x_i^{p_1}$ and $x_i^{p_2}$ be the $i^{th}$ gene-value of parents $p_1$ and $p_2$ respectively. Without loss of generality assume $x_i^{p_1} > x_i^{p_2}$, and define a spreading factor $\beta = \left|\frac{x_i^{c_1} - x_i^{c_2}}{x_i^{p_1} - x_i^{p_2}}\right|$, where $x_i^{c_1}$ and $x_i^{c_2}$ are the $i^{th}$ gene-values of offspring $c_1$ and $c_2$ respectively. The polynomial probability distribution for $\beta$, which is used to perform the contracting and expanding

[1] In using tournament selection without replacement, one has to ensure that that the population size is a multiple of tournament size.

operations is defined as:

$$f(\beta) = \begin{cases} 0.5(\eta_c + 1)\beta^n & \beta \leq 1 \\ 0.5(\eta_c + 1)\beta^{-(\eta_c+2)} & \beta > 1 \end{cases} \tag{5.5}$$

Once $\beta$ is chosen based on the probability density function given by Equation 5.5, $x_i^{c_1}$ and $x_i^{c_2}$ are given by

$$x_i^{c_1} = \frac{1}{2}(x_i^{p_1} + x_i^{p_2}) + \frac{\beta}{2}(x_i^{p_1} - x_i^{p_2}) \tag{5.6}$$

$$x_i^{c_2} = \frac{1}{2}(x_i^{p_1} + x_i^{p_2}) - \frac{\beta}{2}(x_i^{p_1} - x_i^{p_2}) \tag{5.7}$$

The polynomial mutation is similar to SBX, and the only difference is in the computation of the polynomial probability. Instead of using genotypic distance between two parents as in SBX, the distance between a gene and its corresponding upper or lower bound, whichever is closer, is considered in computing the contracting and expanding probability distributions. In polynomial mutation, for a chromosome participating in mutation, each gene (or variable) undergoes contracting or expanding operation with a probability $p_m$.

## 5.3 Results and Discussion

The results presented in this section demonstrate the effectiveness of using multiobjective genetic algorithm in rapid reparameterization of semiempirical methods for ethylene and benzene. We begin with estimating population-sizing and run-duration requirements and then compare the performance of the evolutionary approach in predicting globally accurate PESs—specifically on critical and untested excited states—with previously published results and with single-objective optimization.

### 5.3.1 Population-Sizing and Convergence-Time Analysis

Since the fitness calculations for ethylene are reasonably fast—about 2 seconds per evaluation on a 1.7 GHz AMD Athlon XP workstation—we first verify the population-sizing and run-duration requirements using a limited number of NSGA-II runs. In order to verify population-sizing requirements, five independent runs of NSGA-II with a population size of 2000 for 200 generations

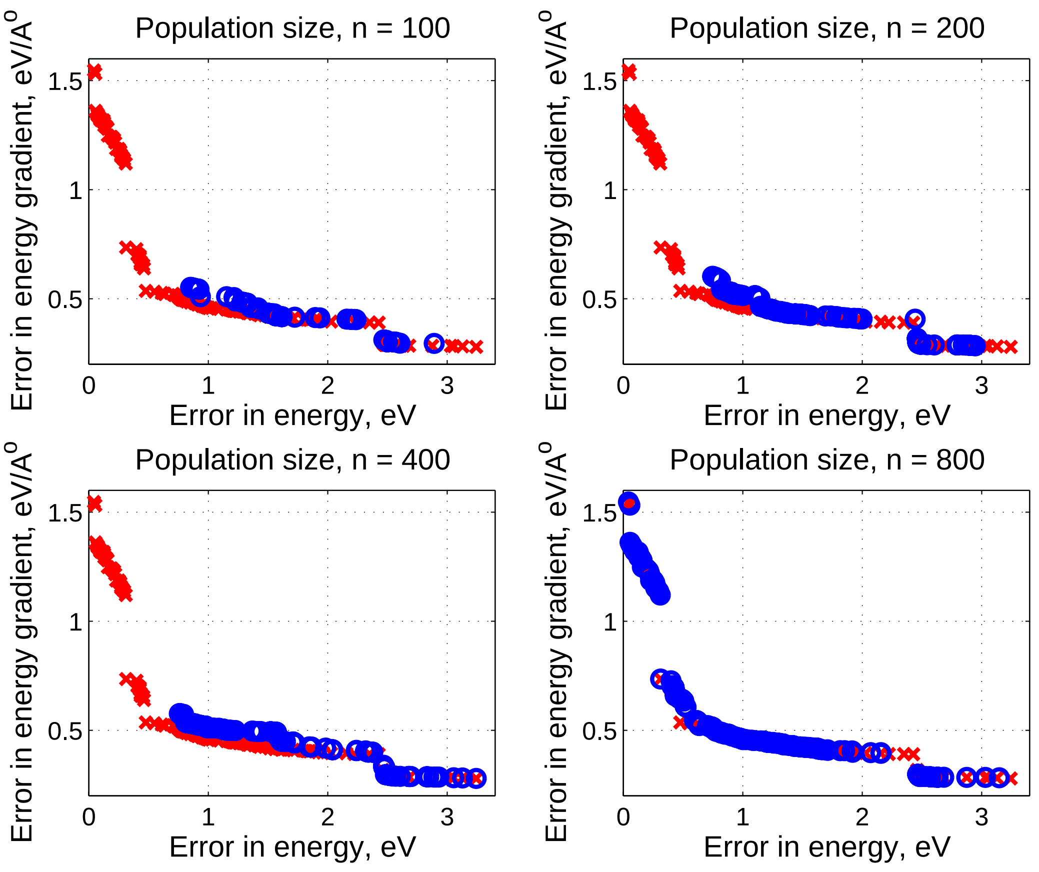


Figure 5.3: Effect of different population sizes on the convergence and coverage of the multi-objective GA. The results are shown for ethylene and are averaged over 10 independent runs. The results show that population sizes below 800 are not capable of converging onto the entire Pareto-front. The empirical results agree with the population size estimate of 750 predicted by Mahfoud's (Mahfoud, 1994) population-sizing model. Points denoted by crosses are obtained with a population of 2000 run for 200 generations and the points represented by circles are the best non-dominated solutions at population sizes of 100, 200, 400, and 800.

were run. The best non-dominated set out of those 5 runs was used as an approximation of the true Pareto-optimal front, which contains 61 distinct solutions. Using the population-size model for niching (Mahfoud, 1994), the population size required to maintain at least 1 copy of each of the Pareto-optimal points with a probability of 0.98 is computed to be 750. To verify this estimate, 10 independent runs of NSGA-II with population sizes between 50–800 were run with a fixed number of function evaluations of 80,000 for each run. The performance of NSGA-II with different population sizes is shown in Figure 5.3. As shown in Figure 5.3, while NSGA-II with population sizes below 800 are unable to converge to the approximate Pareto-optimal front, NSGA-II with a population size of 800 discovers almost all the Pareto-optimal points.

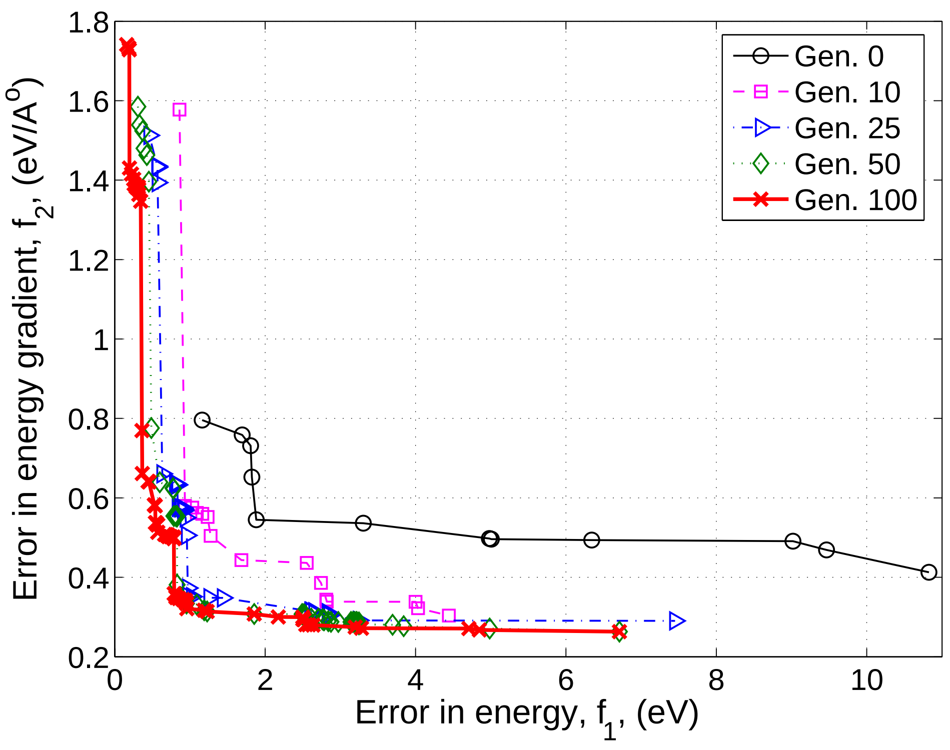


Figure 5.4: Convergence of NSGA-II for reparameterization of semiempirical parameters for ethylene. The best non-dominated front out of 10 independent runs are shown at five different generations. Solutions of reasonable quality start appearing in about 25 generations and high-quality solutions are discovered somewhere between 50–100 generations.

We now look at the convergence rate of NSGA-II and the run-duration requirements for reparameterization. Specifically, ten independent runs of NSGA-II with a population size of 800 were run and the evolution of the best non-dominated front at different generations of the evolutionary process was considered. The results are depicted in Figure 5.4. The results show that reasonably good quality solutions start appearing as early as 10$^{th}$ generation and the solution quality improves at a steady pace till about 25 generations and gradually up to about 100 generations. We found that after about 100 generations the improvement in solution quality was minimal.

Based on population-sizing and run-duration requirements in the remainder of the results a population size of 800 and run duration of 100 generations are used. Since the number of decision variables (semiempirical parameters) remains the same with different molecules involving carbon and hydrogen, the population-sizing and run-duration estimates should hold for the reparameterization of semiempirical parameters for those molecules as well. However, it should be noted that

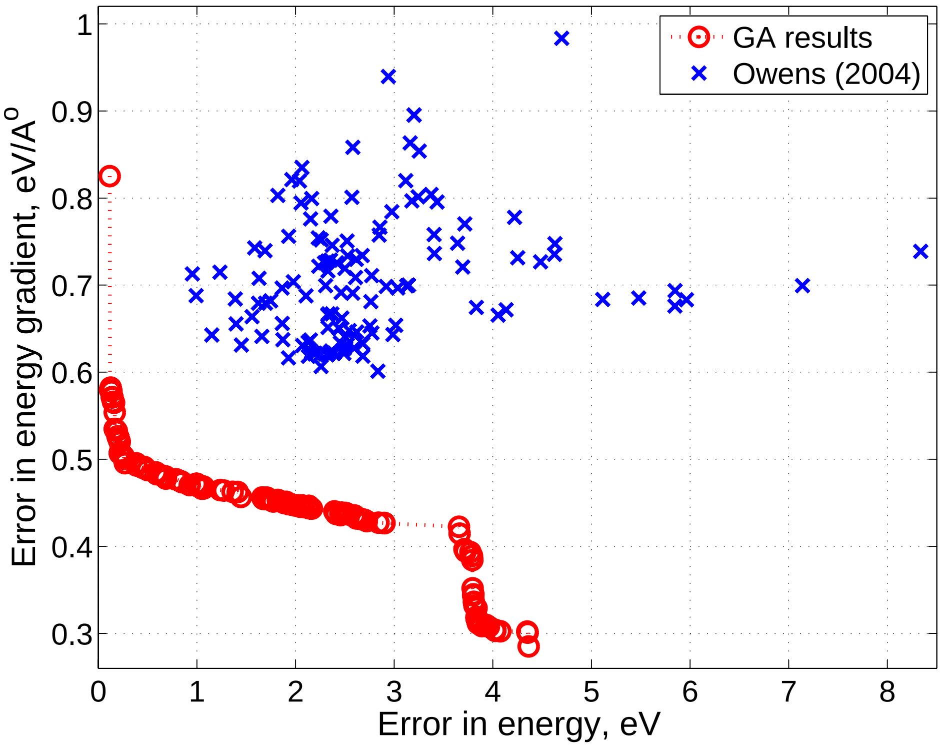


Figure 5.5: The best non-dominated front after 100 generations for ethylene compared to the published results (Owens, 2004). The GA results are for population size $n = 800$, and are averaged over 30 independent runs. The results obtained through GAs are significantly better—226% lower error in the energy, and 32.5% lower error in the energy gradient—than existing reparameterized sets.

the evaluation time increases with the complexity of the molecule under consideration.

### 5.3.2 Multiobjective GA Versus Single-Objective GA

Next, the performance and efficiency of multiobjective optimization to that of single-objective optimization is compared. A weighted sum of the two objectives is taken to convert the multiple objectives into a single objective that has to be minimized:

$$f' = \alpha f_1(\mathbf{x}) + (1 - \alpha) f_2(\mathbf{x}) \tag{5.8}$$

where $f_1$ and $f_2$ are as given by Equations 5.3 and 5.4, respectively, and $\alpha$ is the weighting factor. In order to potentially obtain different Pareto-optimal solutions, 20 different values for $\alpha$ ranging from 0.05 to 1.0 with steps of 0.05 were used. The selection, recombination, and mutation operators and their parameters were kept identical to those of the multiobjective GA.

For the multiobjective GA, a population size of 800 and a run duration of 100 generations are used. Thirty independent GA runs are conducted and the best set of results out of the 30 runs are reported. Therefore, for the multiobjective GA we use a total of 800*100*30 = 2,400,000 function evaluations. For the single-objective GA, for each of the 20 values of $\alpha$, a population size of 100 and a run duration of 50 generations are used. As with multiobjective GA results, the best result out of 30 independent single-objective GA runs are reported. Therefore, for single-objective GA we use a total of 100*50*30*20 = 3,000,000 function evaluations. That is, the single-objective GA runs use 20% more function evaluations than the multiobjective GA runs. Moreover, the population size and run duration settings are consistent with previous reparameterization studies using single-objective GAs. The best non-dominated set obtained by the multiobjective GA is compared to results of single-objective GA in Figure 5.6. The results show that the solutions obtained through multiobjective optimization are consistently superior, both in terms of error in energy and energy-gradient, than the single-objective GA results. It can be easily seen that the single-objective optimization does not yield even one solution comparable to those obtained with the multiobjective GA. We also tried using a population size of 800 and run duration of 100 in the single-objective GA for four different values of $\alpha$ and the results are qualitatively similar. That is, even when single-objective GA used 4 times more function evaluations than the multiobjective GA, the single-objective optimization failed to yield solutions comparable to those of multiobjective GA. These results clearly demonstrate the efficiency of multiobjective approach to reparameterization of semiempirical parameters as opposed to a single-objective optimization approach.

### 5.3.3 Multiobjective GA Versus Published Results

We now compare the solution qualities provided by the best non-dominated front of NSGA-II over the current published results of Owens (2004) for ethylene in Figure 5.5. As shown in the figure, the solutions obtained through the genetic algorithm is significantly superior, both in terms of error in energy and energy-gradient, than those previously reported (Owens, 2004). Specifically, the multiobjective GA yields solutions that are 384% lower error in the energy and 32.5% lower error in the energy gradient than the previously published results. Moreover, the optimality of the

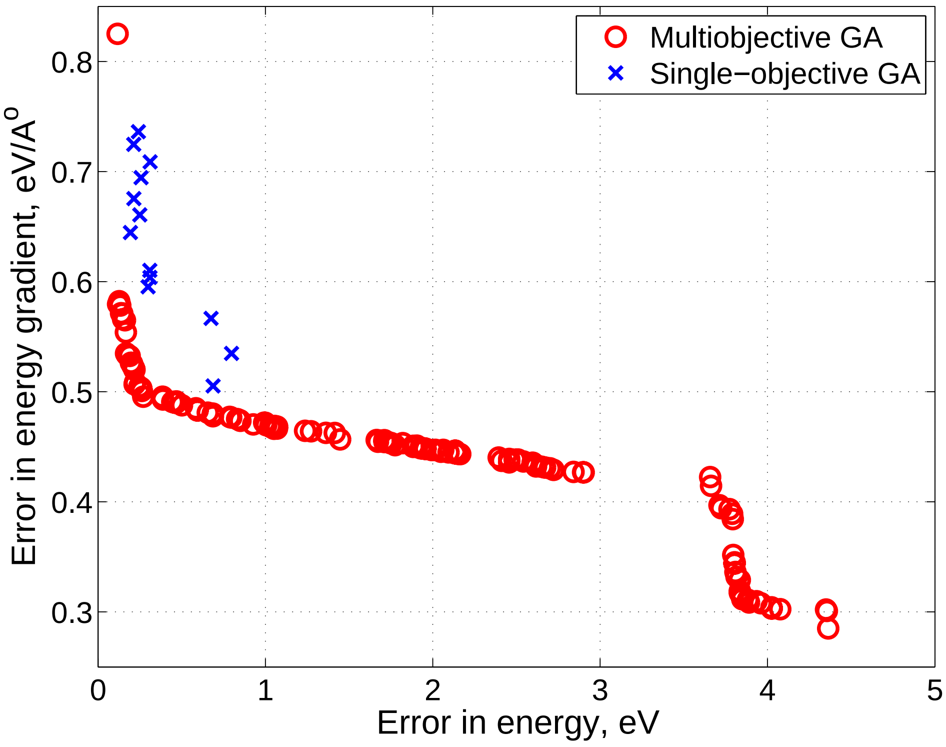


Figure 5.6: Comparison of the best non-dominated solutions for ethylene obtained via the multi-objective genetic algorithm (population size $n = 800$, 30 independent runs) as opposed to multiple runs of single-objective GA (population size $n = 100$ and $n = 800$, 30 independent runs) with different weights for the two objectives.

parameter sets representing the best non-dominated front have been confirmed with local-search and random-perturbation methods.

To verify the effectiveness of the multiobjective GA, we also tested reparameterization on benzene which is more complex than ethylene. The results for benzene reoptimization are shown in Figure 5.7. Similar to the results obtained for ethylene, we observe that the GA provides significant improvement—46% lower error in the energy and 86.5% lower error in the energy gradient—over previously reported results (Toniolo, Thompson & Martinez, 2004).

For ethylene the multiobjective GA found a total of 150 unique semiempirical parameter sets on the best non-dominated front and for benzene the multiobjective GA found a total of 82 unique semiempirical parameter sets on the best non-dominated front. From an optimization point of view, all the parameter sets in the best non-dominated front are equally good. However, from the chemistry perspective this may not be the case, and ultimately we are interested in those

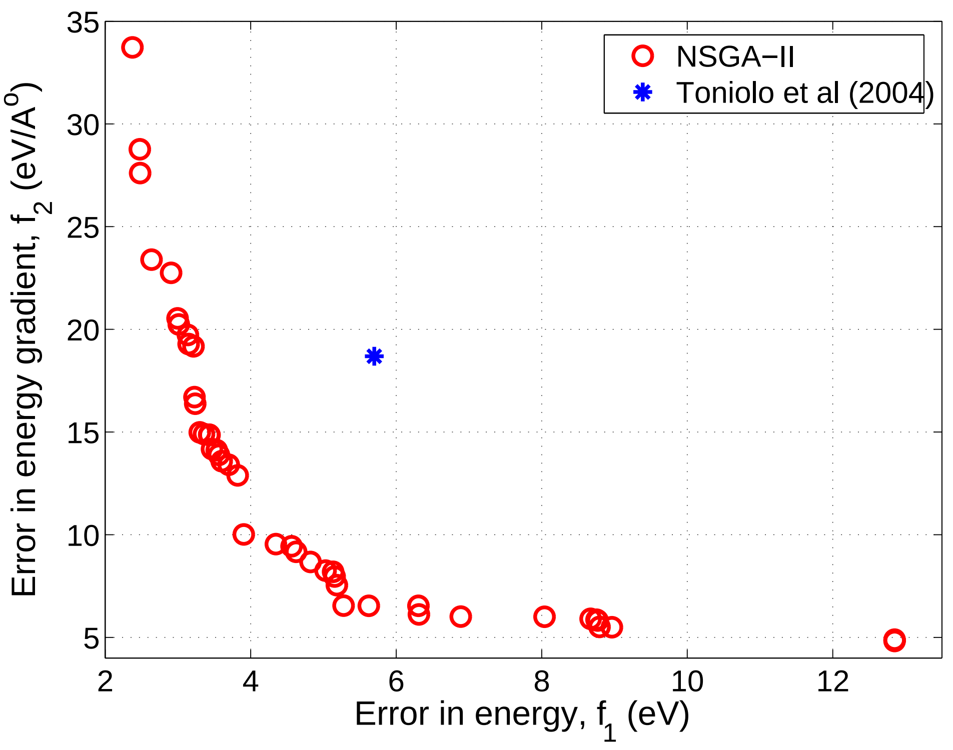


Figure 5.7: The best non-dominated front after 100 generations for benzene compared to the published results (Toniolo, Thompson & Martinez, 2004). NSGA-II results are for population size $n = 800$, and are averaged over 10 independent runs. The results obtained through GAs are significantly better—226% lower in error in energy, and 32.5% lower in error in energy gradient—than existing reparameterized sets.

semiempirical parameters that yield globally-accurate potential energy surface. Therefore, using ethylene results as an example, the remainder of this chapter will consider additional criteria for evaluating the quality of the solution from a chemistry perspective. Specifically, we want those parameter sets that

1. are not sensitive to small perturbations

2. yield accurate excited- and ground-state energies for untested, and critical configurations

3. yield accurate excited-state dynamics

Since we are dealing with two conflicting objectives, we can expect that solutions with low errors in energy yield accurate configurational energies, and the solutions with low errors in energy gradient yield accurate curvature (or shape) of the potential energy surface. Therefore, we are interested in selecting one or more parameter sets that not only yield accurate configurational energies, but also yield accurate shape of the potential energy surface.

The above three criteria are investigated in the following sections.

### 5.3.4 Online Sensitivity Analysis

A preferable property of good-quality semiempirical parameter sets is that they should be less sensitive to small perturbations. That is, if the reoptimized parameter sets are perturbed, we would like the errors in the energy and energy gradient to be similar to those of the Pareto-optimal parameter sets. A sensitivity or stability analysis of the parameter sets can be performed, for example, by considering the errors in energy and energy gradient of randomly perturbed parameter sets around the Pareto-optimal parameter sets. If the error in energy and energy-gradient of the perturbed parameter sets are greater than some threshold, then they are deemed as sensitive. The question remains as to what should the values of these thresholds be?

Standard parameter sets such as PM3 have traditionally been viewed as robust or stable. Therefore, perturbing PM3 parameter set and analyzing the errors in energy and energy-gradient of the parameter sets should give us an idea of what the threshold values for determining the stability of the Pareto-optimal parameter sets. The PM3 parameter sets are randomly perturbed and over 600 perturbed parameter sets are created such that the parameter values of the perturbed parameter are within 2% of the PM3 set. That is, the relative distance between every semiempirical parameter in the perturbed set and the PM3 set is less than 0.01:

$$\left|\frac{x_i' - x_i}{x_i}\right| \leq 0.01, \qquad i = 1, 2, \cdots, 11. \tag{5.9}$$

Here $x_i$ and $x_i'$ are the values of $i^{\text{th}}$ parameter in the PM3 set and perturbed parameter set, respectively.

The errors in energy and energy gradient for each of the perturbed parameter set are computed and are plotted in Figure 5.8. The results in Figure 5.8 indicate that the RMS deviation in error in energy of the perturbed points from that of PM3 is 0.99 eV. Similarly, the RMS deviation in error in energy gradient of the perturbed points from that of PM3 is 0.023 eV/A°. Therefore, threshold values of 0.99 and 0.023 are used for assessing stability of the Pareto-optimal parameter sets.

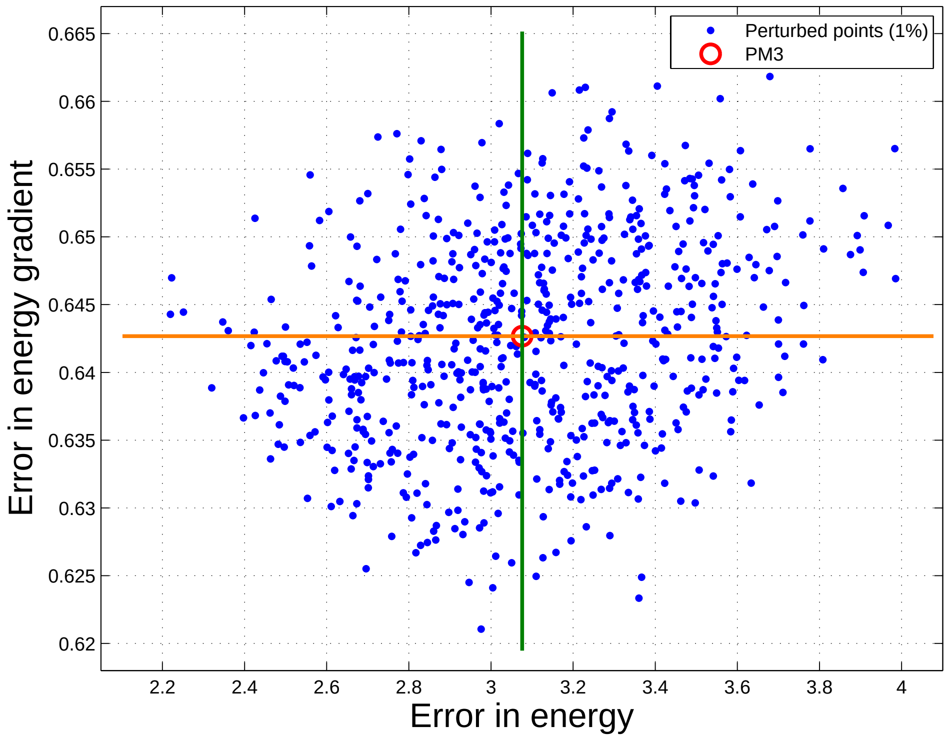


Figure 5.8: Sensitivity of PM3 parameter set to random perturbation. The errors in energy and energy gradient of 606 randomly perturbed parameter sets around PM3 parameter set. The RMS error in energy of the perturbed points is 0.99 eV and the RMS error in energy gradient is 0.023 eV/A°.

One advantage of using genetic algorithms is that we have a population of candidate solutions that can be used to perform an *on-line* sensitivity analysis of the optimal semiempirical parameter sets. Performing such a sensitivity analysis not only reduces the number of acceptable reoptimized parameter sets, but is also more efficient and reliable than a manual stability/sensitivity analysis. That is what we do here, and along with saving the Pareto-optimal semiempirical parameter set, for each of the Pareto-optimal solution we also maintain a list of parameter sets visited during the GA process that are within 2% (as per Equation 5.9) from that Pareto-optimal solution. The RMS deviation in errors in energy and energy gradient can then be computed between the Pareto-optimal solution and the parameter sets around it. If both the RMS deviation in error and energy gradient is less than their respective threshold, then the Pareto-optimal solution is labeled as being *stable*. On the other hand, if either of the deviations is greater than the threshold, then the Pareto-optimal solution is labeled as *sensitive*.

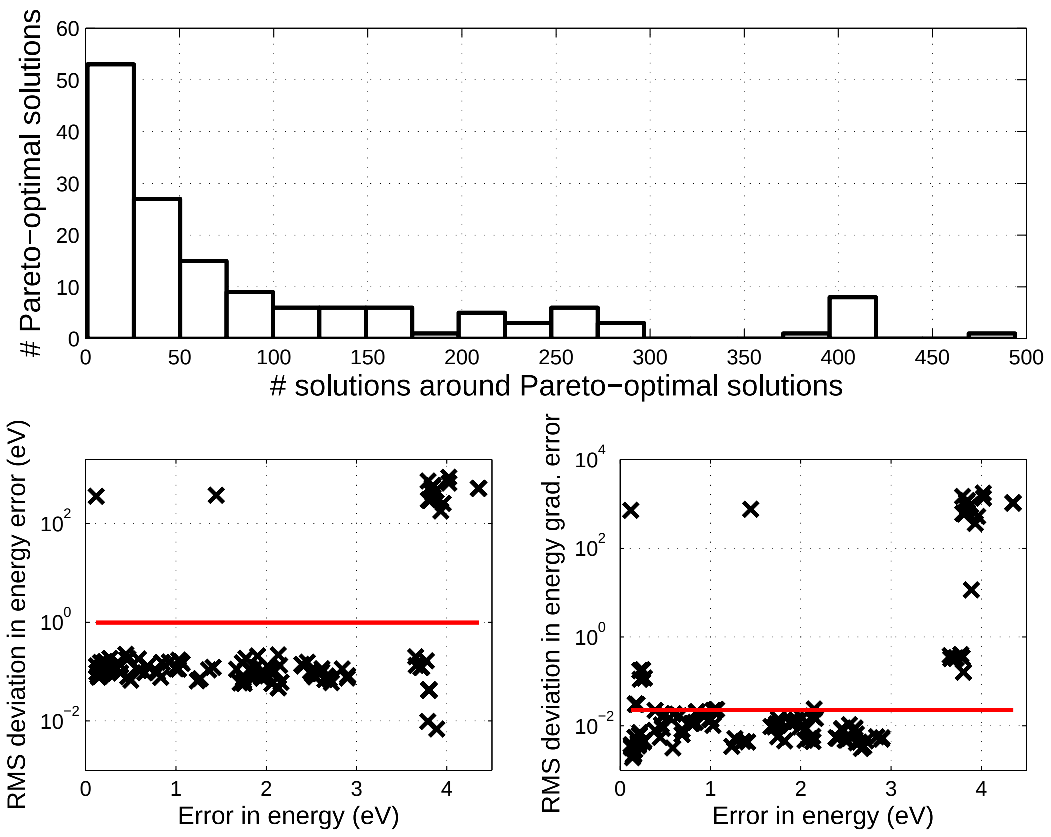
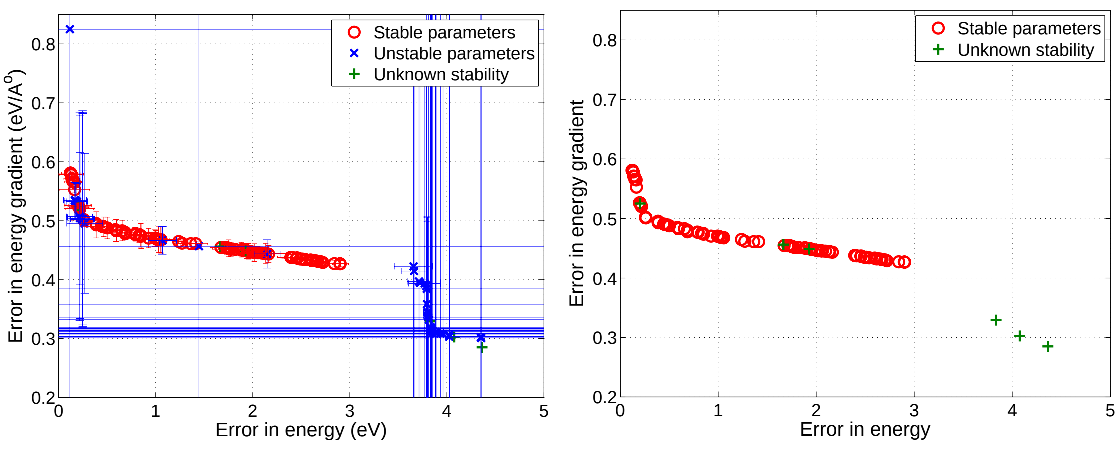


Figure 5.9: (Upper) Histogram of the density of parameter sets around (within 2%) Pareto-optimal solutions. There are a maximum of 495 and a minimum of 1 parameter set within 2% of some Pareto-optimal solution in the GA population. The GA population contains, on an average about 95 parameter sets around the Pareto-optimal solution. (Lower left) RMS deviation in error in energy between the Pareto-optimal solution and corresponding neighboring parameter sets. The threshold determined by analyzing sensitivity of PM3 parameter set is shown as a red line. (Lower right) RMS deviation in error in energy gradient between the Pareto-optimal solution and corresponding neighboring parameter sets. The threshold determined by analyzing sensitivity of PM3 parameter set is shown as a red line.

As shown in Figure 5.9, for majority of the Pareto-optimal solutions, there are sufficient parameter sets within 2% to yield acceptable measure of their sensitivity. Indeed the GA population contains less than 5 parameter sets within 2% of only 6 Pareto-optimal solutions. On an average there are 95 (and a maximum of 495) parameter sets within 2% of the Pareto-optimal solutions. Figure 5.9 also shows the RMS deviations of error in energy and error in energy gradient between the Pareto-optimal solutions and the corresponding neighboring parameter sets. The RMS deviation in errors in energy and energy gradient for each of the Pareto-optimal solution is shown in Figure 5.10. The online sensitivity analysis reveals that 44 out of 150 Pareto-optimal solutions



(a) Sensitivity of Pareto-optimal solutions.
(b) Stable Pareto-optimal solutions.

Figure 5.10: (a) RMS deviations in error in energy and energy gradient for each of the Pareto-optimal solutions. Pareto-optimal solutions with less than 5 neighboring parameter sets are labeled as unknown stability. The stability of these solutions are determined offline by randomly generating perturbed solutions. (b) Stable Pareto-optimal solutions. From a set of 150 Pareto-optimal solutions, 44 solutions with either (or both) the RMS deviation in error in energy greater than 0.99 eV or RMS deviation in error in energy gradient greater than 0.023 eV/A° are labeled as sensitive and eliminated. Six Pareto-optimal solutions which have less than 5 neighboring parameter sets are labeled as those of unknown stability. The remaining 100 Pareto-optimal solutions are found to be stable and not sensitive to perturbation.

have either RMS deviation in error in energy or RMS deviation in error in energy gradient or both are greater than their respective threshold and therefore are sensitive. The results also show that 100 out of 150 Pareto-optimal solutions have both RMS deviations in errors in energy and energy gradient are below the threshold and thus are stable or less sensitive to small perturbations in the parameter values. The parameter sets that are found to be stable via the online sensitive analysis are shown in Figure 5.10(b).

### 5.3.5 Energetics of Untested Excited-State Configurations

We now consider solutions obtained through the GA and evaluate their results on energetic calculations for a set of ethylidene geometries for which they were not reoptimized. Before comparing the results of GA with those of Owens, a brief description of salient properties of *cis-trans* isomerization of ethylene is provided.

Previous *ab initio* work has shown that in addition to the twisting coordinate, a coordinate involving the pyramidalization of one of the carbons is also important. Ultrafast pump-probe

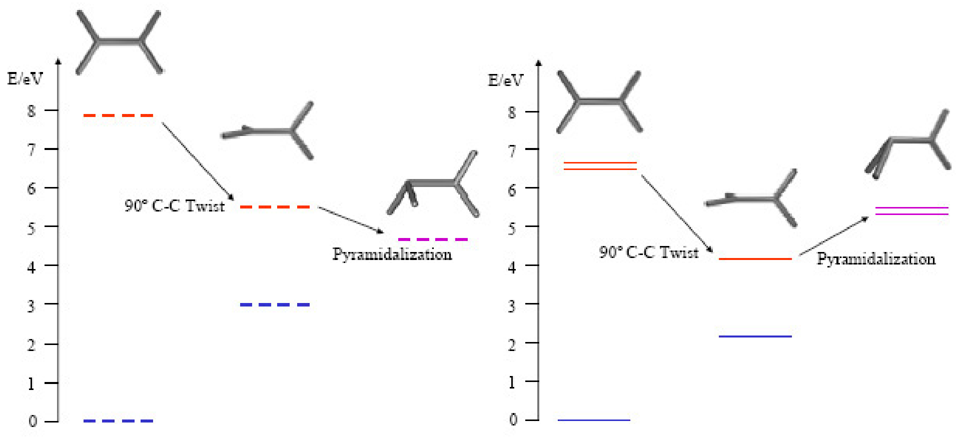


Figure 5.11: Energy levels of ethylene at various geometries: Planar, twisted, and pyramidalized (left) from *ab initio* calculations and (right) from PM3 and AM1 calculations. Reproduced with permission.

experiments have shown a very fast excited state lifetime for ethylene (Famanara, Stert & Radloff, 1998). Recent theoretical works suggest that the twisted-pyramidalized geometry is responsible for the fast non-radiative transfer to the ground state through a conical intersection (Ben-Nun & Martinez, 2000; Ben-Nun, Quenneville & Martinez, 2000). That is, the ground state for ethylene is a planar structure as shown in Figure 5.11 and when it is excited, the carbon-carbon bond twists 90° and decreases in the energy gap from 7.8 eV to 2.5 eV. The twisted geometry, however, is not an excited state minimum but a saddle point with respect to pyramidalization of one of the carbon atoms. As shown in Figure 5.11, using the PM3 and AM1 parameters, however, the pyramidalized geometry is actually higher in energy than the purely twisted geometry on the excited state, which is in direct contrast to the results of experiments and high level calculations.

Specifically, we consider the above two important energetics, the results of which are shown in Figure 5.12:

- Energy differences between planar ethylene (ground state, $S_0$ minimized $D_{2h}$) and twisted geometry ($S_1$ minimized $D_{2d}$), ideal value for which is 2.28 eV as calculated by *ab initio* methods (Ben-Nun & Martinez, 2000). If the energy difference between the planar and twisted geometry is less than zero, then the excited state minimum would be the planar structure, which is erroneous. In other words, for good parameter sets, the energy difference between the planar and twisted geometry should be greater than zero, preferably around 2.28

eV.

- Energy differences between the twisted geometry ($S_1$ minimized $D_{2d}$) and pyramidalized structure, ideal value for which is 0.88 eV as calculated by *ab initio* methods (Ben-Nun & Martinez, 2000). As shown in Figure 5.11 the standard semiempirical parameter sets do not capture this feature, and therefore, this energetics is one of the critical phenomena in determining the quality of the reoptimized parameter sets. If the energy difference between the twisted geometry and the pyramidalized structure is less than zero, then the excited state minimum would be the twisted geometry (as predicted by standard parameter sets) which is inconsistent with *ab initio* and experimental results. Therefore, for good parameter sets, the energy difference between the twisted and pyramidalized geometries must be greater than zero, preferably around 0.88 eV.

Among the Pareto-optimal solutions, we would expect those parameter sets with lower error in energy to be able to yield accurate energetics of untested configuration as opposed to those with higher error in energy. The energy differences between planar and twisted geometry, and twisted geometry and pyramidalized structure, for both the best non-dominated sets are shown in Figure 5.12 along with the corresponding solutions. As expected, from Figure 5.12 we can easily see that the Pareto-optimal solutions with error in energies less than 1.5 eV yield near ideal energies for both excited-state transitions. Moreover, as shown in Figure 5.13, the results obtained via multiobjective GA are clearly superior when compared to previously published results (Owens, 2004). More importantly, the multiobjective GA optimized parameter sets correctly identify the lowest-energy excited state as the pyramidalized structure as opposed to standard semiempirical parameter sets and some of the previously reported reparameterized sets.

### 5.3.6 Energy Dynamics Calculations

The final requirement of good parameter sets from chemistry perspective is that they yield accurate reaction dynamics simulations. It should be noted that the dynamics is controlled by the shape of the potential energy surfaces. Since error in energy gradient gives a measure of the accuracy of the potential energy surface, it is reasonable to expect that Pareto-optimal solutions with lower error in energy gradient yield accurate dynamics over those with higher error in energy gradient.

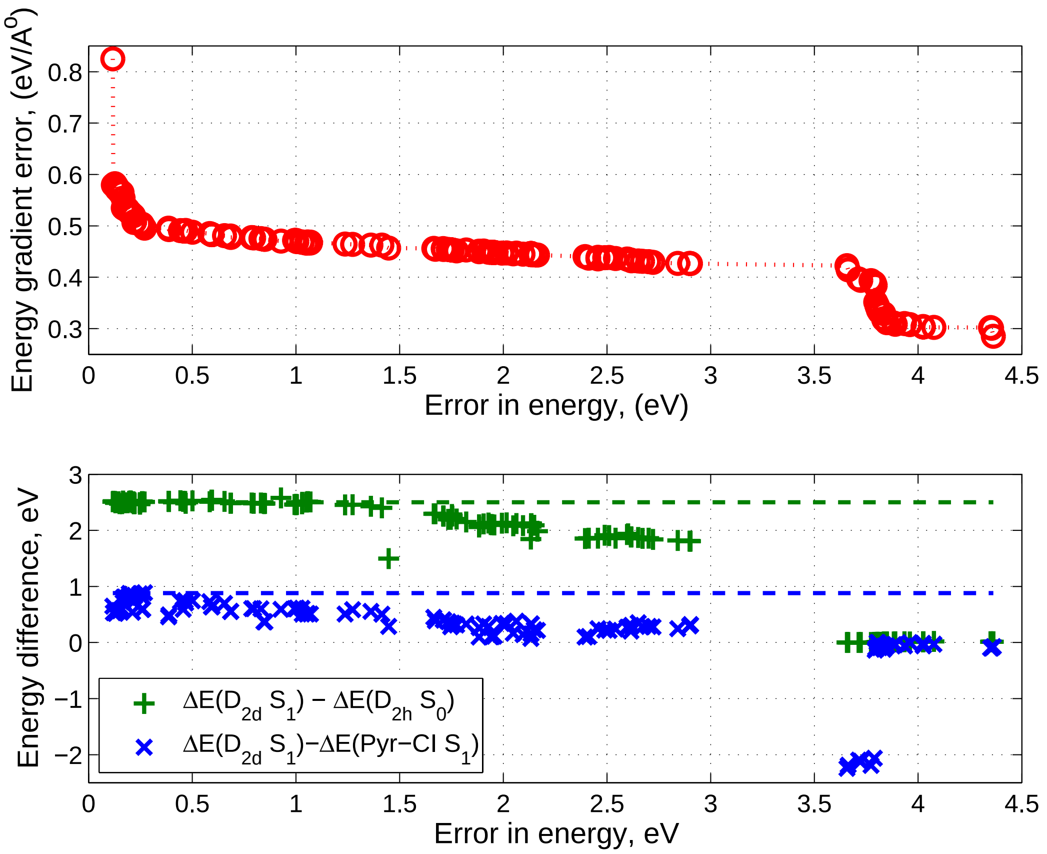


Figure 5.12: (Top) The best non-dominated parameter set obtained via multiobjective GA reproduced from Figure 5.5. (Bottom left) the energy differences between planar and twisted geometries, and between twisted and pyramidalized geometry. The points on the best non-dominated set—especially those with low error in energy—are near ideal values (shown by dashed lines). The ideal value for energy difference between $D_{2d}$ $S_1$ and pyramidalized $S_1$ configuration of 0.9 eV, and that between $D_{2d}$ $S_1$ and planar ($D_{2h}$ $S_0$) is 2.28 eV.

The dynamics calculations conducted by Alexis Thompson, one of our chemistry collaborators, validates this expectation and indeed parameter sets with lower error in energy gradient produce ab-initio-quality reaction dynamics simulations.

Specifically, Alexis considered excited state lifetime and computed the average time for half the population to transfer from $S_1$ (excited state, pyramidalized) to $S_0$ (ground state, planar) following photoexcitation of the gas-phase molecule. This phenomenon has been extensively studied using *ab initio* simulations (Quenneville, Ben-Nun & Martinez, 2001). The AIMS simulations used multi-reference configuration interaction (MRCI) electronic wavefunctions within a double zeta basis set. The simulations did not include Rydberg basis functions and the nuclear dynamics is followed for

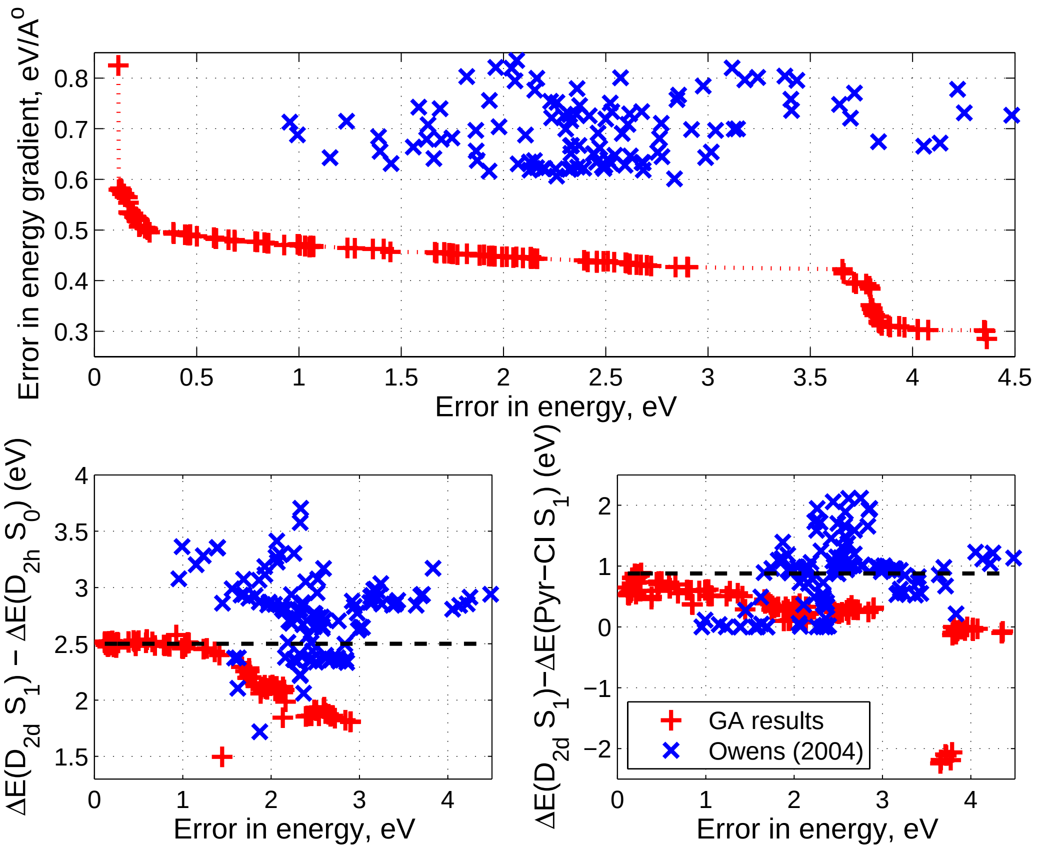


Figure 5.13: (Top) The best non-dominated sets and solution set reported by Owens (Owens, 2004) reproduced from Figure 5.5. (Bottom) The energy difference between planar and twisted geometries and the energy difference between twisted and pyramidalized geometry. The points on the best non-dominated set—especially those with low error in energy and energy-gradient—are near ideal values (shown by dashed lines). The ideal value for energy difference between $D_{2d}$ $S_1$ and pyramidalized $S_1$ configuration of 0.9 eV, and that between $D_{2d}$ $S_1$ and planar ($D_{2h}$ $S_0$) is 2.28 eV.

0.5 picoseconds, and the total dynamics is represented by averaging over results obtained using 10 different representations of the initial wavefunction. Overall, approximately 100 nuclear basis functions are spawned during the simulation time. A more complete description of the technical details is available elsewhere (Ben-Nun, Quenneville & Martinez, 2000; Quenneville, Ben-Nun & Martinez, 2001). The *ab initio* results indicate that the average time for half the population to transfer from $S_1$ to $S_0$ is $180 \pm 50$ fs.

The dynamics simulations are computed for each of the stable Pareto-optimal solutions and compared to the *ab initio* simulation results. The dynamics results are averaged over 50 independent dynamics simulations and are plotted as a function of error in energy gradient in Figure 5.14. The

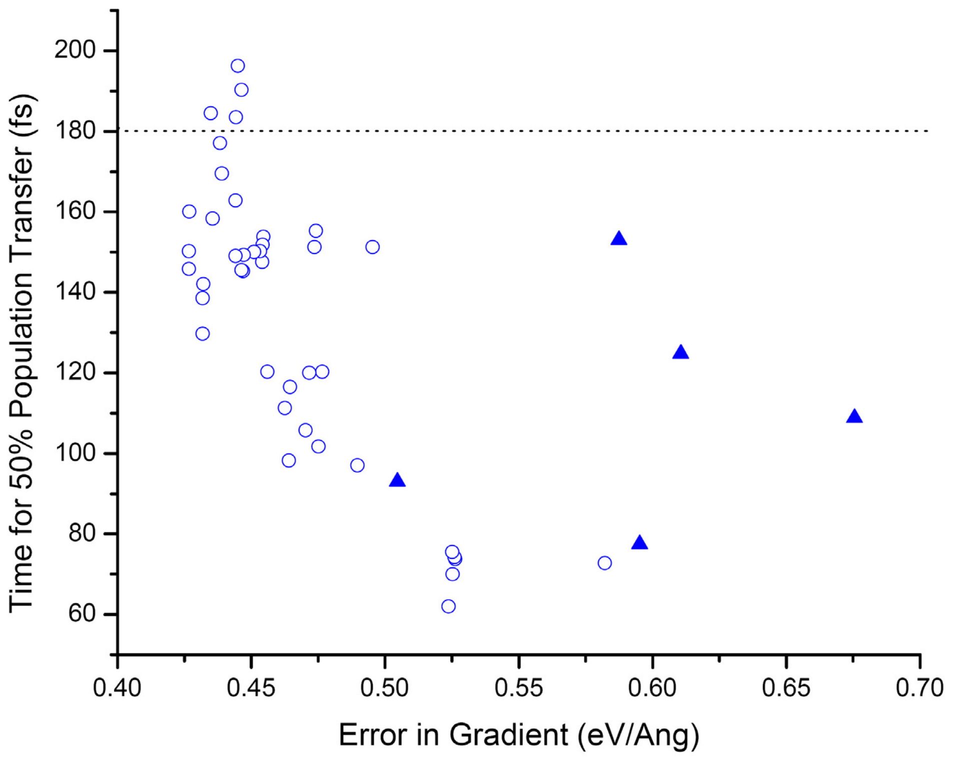


Figure 5.14: The average time for half the population to transfer from $S_1$ (excited state, pyramidalized) to $S_0$ (ground state, planar) following photoexcitation of the gas-phase ethylene for the stable Pareto-optimal semiempirical parameters. The average time for half of the population to transfer to $S_0$ from *ab initio* results is $180 \pm 50$ fs. The results are averaged over 50 independent dynamics simulations. The dynamics results for the best solutions obtained via single-objective optimization are denoted by triangles. Reproduced with permission from A. Thompson.

results show that most of the stable Pareto-optimal solutions—especially those with lower error in energy gradient—yield near-ideal, *ab initio* quality dynamics results. Moreover, the dynamics simulation results for the best solutions obtained via single-objective optimization are significantly worse than those obtained via multiobjective optimization. In essence, the dynamics simulation results clearly show that majority of the stable Pareto-optimal solutions—specifically 85 out of 100—yield dynamics results in agreement with *ab initio* simulations.

### 5.3.7 Stable, Energistically and Dynamically Accurate Parameter Sets

The results presented in the previous sections clearly show that multiobjective GA yields multiple semiempirical parameter sets that (1) are stable to small perturbations, (2) yield accurate—indeed, near ideal—energetics for untested, yet critical excited-state configurations, and (3) yield dynam-

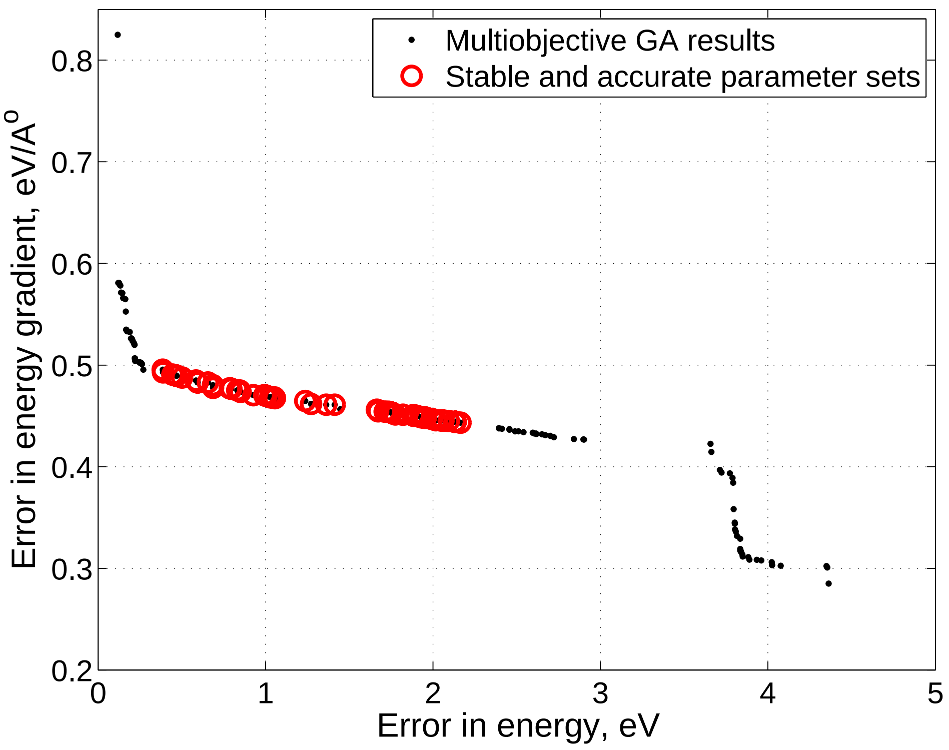


Figure 5.15: Subset of Pareto-optimal solutions for ethylene that (1) are stable to small perturbations, (2) yield accurate—indeed, near ideal—energetics for untested, yet critical configuration, and (3) yield dynamics with *ab initio* accuracy. Overall 61 out of 150 Pareto-optimal solutions obtained via multiobjective GA are found to be stable and yield accurate energetics and dynamics.

ics with *ab initio* accuracy. The subset of Pareto-optimal solutions which are stable and produce accurate energetics and dynamics are shown in Figure 5.15. Combining the results from online sensitivity analysis, energetics tests, and dynamics simulations, reveals that out of the 150 parameter sets in the Pareto-optimal front, 61 parameter sets are stable and yield accurate configurational energies and dynamics. Interestingly, parameters sets that are stable and yield accurate energetics and dynamics are not on and around the nose of the Pareto front as one might expect, but are slightly to the right of the Pareto front. The reason being that, a slight increase in error in energy leads to an improvement in error in energy gradient which controls the accuracy of dynamics.

It should also be noted that similar to ethylene, results for benzene also show that while the standard semiempirical parameter sets yield inaccurate dynamics, the multiobjective GA optimized parameter sets yield results consistent with experiments and *ab initio* computations. For example, the newly optimized parameter sets predict an $S_2$ lifetime of 100 fs, in agreement with experiment

(Radloff et al., 1997).

### 5.3.8 Semiempirical parameter interactions

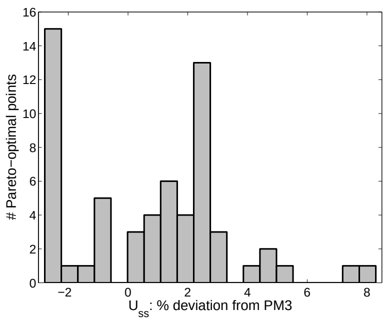
The multiobjective optimization of the semiempirical parameters for ethylene resulted in 61 different optimal parameter sets. Figure 5.16 depicts the histogram of the deviation of the 11 semiempirical parameter values from their corresponding PM3 values. The results show that the parameter values, with the exception of $\zeta_p$ are very diverse. Understanding the relationship between these optimal, stable, and accurate semiempirical parameters can yield insights into important energy relations for the given molecule. In addition to yielding physical insights into the excited-state energetics, if we can determine the relationships between the semiempirical parameters, we would have *interpretable* semiempirical methods.

Another advantage of using multiobjective GA is that the Pareto optimal solutions can be *mined* to automatically discover the relation between semiempirical parameters. That is, using the optimal, stable, and accurate parameter-set data, the relationship between the semiempirical parameters can be symbolically regressed via genetic programming. That is what we do here. First the 61 Pareto-optimal parameter sets that (1) are stable to small perturbations, (2) yield accurate configurational energies, and (3) yield *ab initio* quality excited-state dynamics are selected. The data is normalized using a z-score

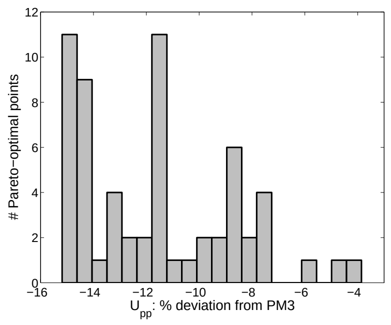
$$x'_i = \frac{x_i - \bar{x}_i}{s_{x_i}}$$

where $\bar{x}_i$ and $s_{x_i}$ are the sample mean and variance of the $i^{\text{th}}$ semiempirical parameter, respectively. The objective then is to determine the functional relationship between each of the 11 semiempirical parameters as a function of the rest of the 10 parameters. For example, we use GP to evolve a functional relationship between $U_{ss}$ in terms of $U_{pp}$, $\beta_s$, $\beta_p$, $\zeta_s$, $\zeta_p$, $G_{ss}$, $G_{sp}$, $G_{pp}$, $G_{p_2}$, and $H_{sp}$.

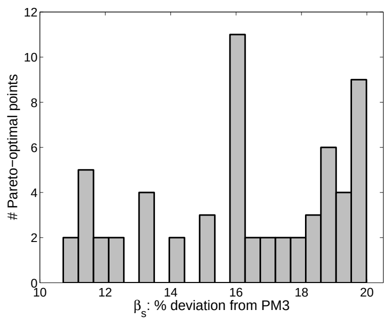
The normalized data is used to evolve relationships between the semiempirical parameters via GP. The following function set $\mathcal{F} = \{+, -, *, /, \hat{}, \exp\}$ is used for all the runs. The terminal set contains an ephemeral random constant and all the semiempirical parameters except the one for which we want to discover the functional relationship. For example, if we want to find the relationship between $U_{ss}$ and the rest of the semiempirical parameters, then the terminal set would consist of $\mathcal{T} = \{U_{pp}, \beta_s, \beta_p, \zeta_s, \zeta_p, G_{ss}, G_{sp}, G_{pp}, G_{p_2}, H_{sp}, \mathcal{R}\}$. The output of the candidate program



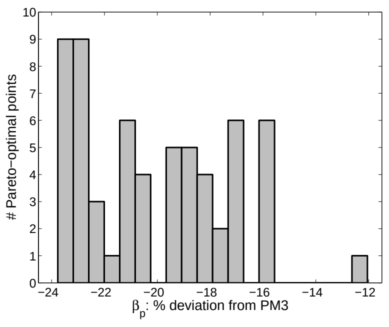
(a)



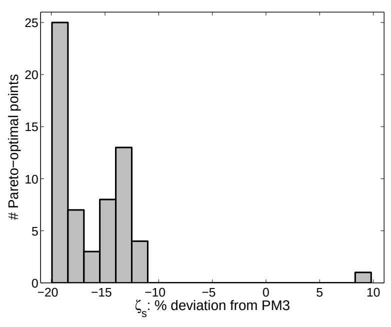
(b)



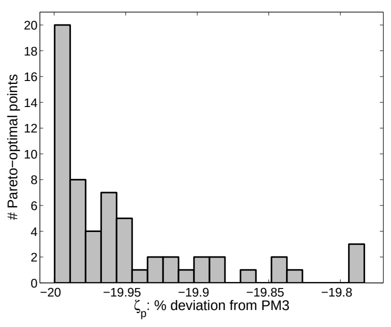
(c)



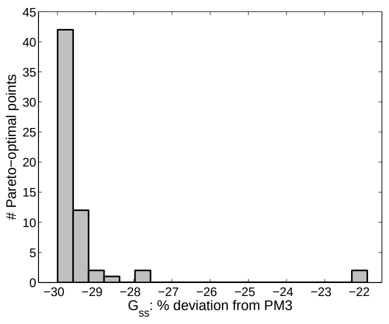
(d)



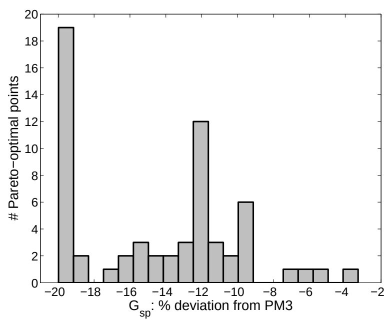
(e)



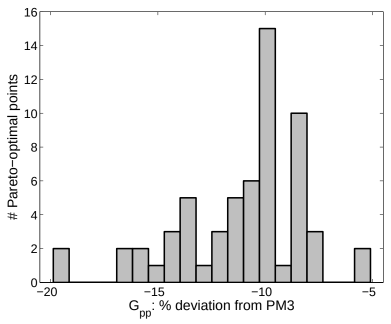
(f)



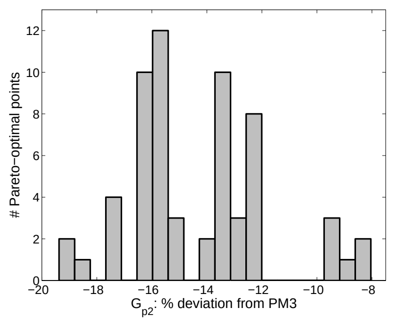
(g)



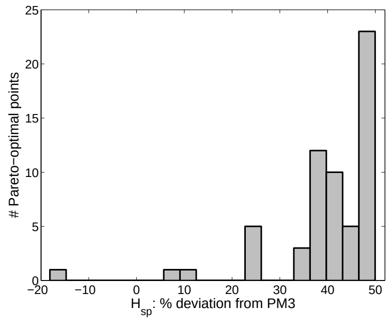
(h)



(i)



(j)



(k)

Figure 5.16: Histogram of the deviation of stable, accurate and optimal semiempirical parameter for ethylene from their corresponding PM3 parameter values.

is the normalized value of a semiempirical parameter.

The fitness of a solution is computed as the root mean square error (RMSE) between the predicted and Pareto-optimal value of the semiempirical parameter for which we are evolving the candidate program. In all GP runs, a population size of 1000 and a run duration of 100 generations were used. For each of semiempirical parameter, over 100 independent GP runs were computed. The best evolved regression function for each of the independent runs is then simplified using the symbolic math toolbox in matlab. The coefficients are then optimized using either linear or non-linear regression methods. While the highlights of the results are discussed in the remainder of this section, the best evolved solutions for each of the semiempirical parameters along with the RMSE values and the number of independent GP runs that yield the functional form are given in Appendix B.

Table 5.2: Functional relationship between semiempirical parameters symbolically regressed via GP using the stable and accurate Pareto-optimal solutions. The coefficients of the GP-regressed relations are optimized via linear or non-linear regression methods. The results show the two most frequently evolved relations and the least-complex, most-accurate relationship for each of the 11 parameters. More results are presented in Appendix B.

| | **# GP Runs** | **RMS error** | **GP-regressed relation** |
|---|---|---|---|
| $U_{ss}$ | 1 | 0.198 | $1.658U_{pp} + 2.086G_{p_2} + 0.795\beta_s + 0.476\zeta_s - 1.5G_{ss} - 0.34G_{sp}$ |
| | 37 | 0.417 | $1.968U_{pp} + 1.052G_{p_2} + 0.438\beta_s$ |
| | 24 | 0.525 | $1.671U_{pp} + 1.052G_{p_2}$ |
| $U_{pp}$ | 1 | 0.172 | $0.174U_{ss} + 0.389\beta_p - 0.3G_{sp} - 0.325G_{p_2}$ |
| | 76 | 0.218 | $0.708\beta_p - 0.474G_{sp}$ |
| | 21 | 0.324 | $0.441U_{ss} - 0.726G_{p_2}$ |
| $\beta_s$ | 1 | 0.349 | $-0.627 - 0.087G_{ss} + 0.754G_{ss}^2 + G_{ss}^3\left(0.694H_{sp} + 0.039G_{sp}\right) + G_{ss}^4H_{sp}$ |

*continued on next page*

| Table 5.2 – continued from previous page | | | |
|---|---|---|---|
| | **# GP Runs** | **RMS error** | **GP-regressed equation** |
| | 13 | 0.585 | $-0.579 + 0.583G_{ss}^2$ |
| | 38 | 0.593 | $1.248G_{ss} - 0.531G_{p_2}$ |
| $\beta_p$ | 1 | 0.199 | $0.034 - 0.737G_{ss} + G_{ss}G_{pp}\left(0.339 - 0.057G_{pp}\right) + 0.054G_{pp}e^{0.812G_{sp}}$ |
| | 21 | 0.293 | $1.275U_{pp} + 0.571G_{sp}$ |
| | 28 | 0.305 | $-0.724G_{ss} + 0.299G_{pp}$ |
| $\zeta_s$ | 1 | 0.468 | $-0.233 + 0.682\zeta_p G_{pp} U_{ss} e^{2.578\beta_p}$ |
| | 16 | 0.582 | $-0.145 + 0.39\zeta_p G_{pp} U_{ss}$ |
| | 8 | 0.598 | $-0.182 + 0.193\zeta_p^2$ |
| $\zeta_p$ | 1 | 0.146 | $-0.251 + \beta_s\left(0.59G_{ss} - 0.592G_{p_2}\right) + 0.09\zeta_s^2\beta_s^2$ |
| | 8 | 0.259 | $-0.159 + 0.063\zeta_s + 0.183\beta_s\zeta_s$ |
| | 52 | 0.262 | $-0.167 + 0.193\zeta_s^2\beta_s$ |
| $G_{ss}$ | 1 | 0.086 | $-0.315 + 0.516G_{p_2} + 0.278\zeta_p + 0.103\beta_s + \zeta_p\left(0.17G_{sp} + 0.096\beta_s\zeta_p\right)$ $+ G_{p_2}\left(0.119\beta_s + 0.172G_{p_2}\right) + G_{p_2}\zeta_p\left(0.445\beta_s - 0.605\zeta_p - 0.489G_{p_2}\right)$ |
| | 22 | 0.312 | $0.697G_{p_2} + 0.346\beta_s$ |
| | 14 | 0.313 | $0.871G_{p_2} + 0.276\zeta_p$ |
| $G_{sp}$ | 1 | 0.140 | $-0.2749 + 1.592\beta_p - 2.538U_{pp} - 0.28\zeta_s + \beta_p\left[1.433\beta_p\right.$ $-\zeta_s\left(1.652 + 0.043U_{pp}^{-1}\right) - \zeta_s G_{p_2}\left.\left(5.758 + 0.197U_{pp}^{-1}\right)\right]$ $- U_{pp}\left(2.608U_{pp} + 0.537\beta_s + 0.101\zeta_s - 1.718\zeta_s G_{p_2}\right)$ $- \zeta_s\left(0.157\beta_s - 0.366G_{p_2}\right) - \beta_p^2\left[3.16\beta_p + 2.74G_{p_2} - 3.482\beta_p G_{p_2}\right.$ $\left.-\zeta_s U_{pp}^{-1}\left(0.91 + 2.288G_{p_2}\right)\right] + U_{pp}^2\left[1.788U_{pp} + 1.248G_{ss} - 1.516\beta_s\right.$ $\left.-1.01G_{p_2} + U_{ss}\left(0.119 + 0.417G_{p_2} - 1.002\beta_s\right)\right] + U_{pp}^3\left(3.663U_{pp} +\right.$ |
| | | | *continued on next page* |

| Table 5.2 – continued from previous page | | | |
|---|---|---|---|
| | # GP Runs | RMS error | GP-regressed equation |
| | | | $1.699G_{p_2} + 0.502U_{ss} + 2.433U_{ss}G_{p_2}) - \beta_p U_{pp} [0.311 - 2.781\beta_p$ $+1.493U_{pp} - 2.455\beta_s + 9.914U_{pp}^2 - G_{p_2}(0.905 - 4.994\beta_p$ $-6.97U_{pp}) - U_{ss}(2.45U_{ss} + 0.978\beta_p + 0.858U_{pp})$ $-U_{ss}G_{p_2}(4.396 + 5.378\beta_p - 6.152U_{pp})]$ |
| | 2 | 0.356 | $-0.118 + 1.017\beta_p - 1.643U_{pp} - 1.336\beta_p U_{pp} + 0.466\beta_p^2 + 0.813U_{pp}^3$ |
| | 75 | 0.414 | $1.137\beta_p - 1.7U_{pp}$ |
| $G_{pp}$ | 1 | | $0.256 - 0.204\zeta_p + 0.762\beta_p + 1.137G_{p_2} + 0.09\beta_s$ $+ G_{p_2}\left(0.481\beta_p + 0.1\zeta_s + 0.53\zeta_p^2\right)$ |
| | 4 | 0.316 | $0.922\zeta_p + 1.286\beta_p + 1.048G_{p_2} - 0.235\zeta_s$ |
| | 90 | 0.350 | $0.74\zeta_p + 1.297\beta_p + 1.082G_{p_2}$ |
| $G_{p_2}$ | 1 | 0.174 | $-0.068 + 0.39G_{ss} + 0.343U_{ss} - 0.844U_{pp} - 0.314\beta_s + 0.289G_{pp}\beta_p$ |
| | 20 | 0.333 | $0.977G_{ss} - 0.256\zeta_s$ |
| | 29 | 0.337 | $0.51G_{ss} - 0.467U_{pp}$ |
| $H_{sp}$ | 1 | 0.456 | $-0.594 + 1.12\beta_p - 0.537G_{p_2} - 0.316\zeta_p + 0.455G_{pp}$ $+ 1.558G_{p_2}G_{pp} - 0.071G_{ss}\beta_s - G_{p_2}\beta_p [1.797 - 2.245\beta_p$ $-0.947U_{pp} + G_{p_2}(1.716\beta_p + 0.086U_{pp} + 2.783G_{pp} - 1.623G_{p_2}G_{pp})]$ $+ G_{p_2}^2 G_{pp}U_{pp}(1.798 - 0.972G_{p_2})$ |
| | 25 | 0.654 | $0.221 + 0.274G_{p_2}^3 G_{pp}$ |
| | 28 | 0.666 | $-0.013 - 0.323G_{p_2} + 0.545G_{p_2}G_{pp}$ |

Table 5.2 shows the two most frequently evolved, and statistically most-accurate and least-complex (determined via an F-test) relationships for each of the 11 semiempirical parameters. The results can be summarized as follows:

- For $U_{ss}$, 37 out of 113 independent GP runs suggest that $U_{ss}$ is linearly proportional to $U_{pp}$, $G_{p_2}$, and $\beta_s$. Moreover *all* the independent runs suggest a linear relationship between $U_{ss}$ and $U_{pp}$ which is in agreement with previously published analysis (Owens, 2004).

- For $U_{pp}$, 76 out of 113 independent GP runs suggest that $U_{pp}$ is linearly proportional to $\beta_p$ and $G_{sp}$. Additionally, 21 independent GP runs suggest that $U_{pp}$ is linearly proportional to $U_{ss}$ and $G_{p_2}$. Finally, the most-accurate, least-complex relationship is a linear combination of the two most frequently evolved symbolic expressions.

- For $\beta_s$, 38 out of 113 independent GP runs suggest that $\beta_s$ is linearly proportional to $G_{ss}$ and $G_{p_2}$. Moreover *all* the independent GP runs indicate a relationship between $\beta_s$ and $G_{ss}$.

- For $\beta_p$, 28 independent GP runs suggest a linear relationship between $\beta_p$ and $G_{ss}$ and $G_{pp}$. Additionally, 21 independent runs suggest a linear relationship between $\beta_s$ and $U_{pp}$ and $G_{sp}$.

- For $\zeta_s$, 16 independent GP runs suggest that $\zeta_s$ is related to the product of $\zeta_p$, $G_{pp}$, and $U_{ss}$. However, the high RMS error coupled with majority of independent GP runs yielding very diverse relationships for $\zeta_s$ suggest that $\zeta_s$ does not have a strong relationship with other semiempirical parameters.

- For $\zeta_p$, 52 independent GP runs suggest that $\zeta_p$ is directly proportional to $\zeta_s^2\beta_s$. Moreover, all independent GP runs indicate a relationship between $\zeta_p$ and $\zeta_s$.

- For $G_{ss}$, 22 independent GP runs reveal a linear relationship between $G_{ss}$, $G_{p2}$ and $\beta_s$ and 14 independent GP runs reveal a relationship between $G_{ss}$, $G_{p2}$ and $\zeta_p$. Moreover, majority of the independent GP runs reveal a relationship between $G_{ss}$ and $G_{p_2}$.

- For $G_{sp}$, 75 independent GP runs suggest that $G_{sp}$ is linearly proportional to $\beta_p$ and $U_{pp}$. Moreover, all the independent GP runs consist of the similar relationship between $G_{sp}$, $\beta_p$ and $G_{sp}$.

- For $G_{pp}$, 90 independent GP runs reveal that $G_{pp}$ is linearly proportional to $\zeta_p$, $\beta_p$, and $G_{p_2}$. Indeed, the best solutions evolved in all the GP runs contain the above relationship.

- For $G_{p_2}$, 29 independent GP runs suggest that $G_{p_2}$ is linearly proportional to $G_{ss}$ and $U_{pp}$ and 20 independent runs suggest that $G_{p_2}$ is linearly proportional to $G_{ss}$ and $\zeta_s$. Furthermore, all independent GP runs suggest that $G_{p_2}$ depends on $G_{ss}$.

- For $H_{sp}$, 28 independent GP runs reveal that $H_{sp}$ is related to $G_{p_2}$ and $G_{pp}$. More importantly all the runs indicate that $H_{sp}$ is related to $G_{pp}$ and $G_{p_2}$, which is in agreement with previously published analysis (Owens, 2004).

## 5.4 Summary

In this chapter, we investigated the use of multiobjective genetic algorithms in multiscaling simulations of excited state dynamics in photochemistry. Specifically, multiobjective genetic algorithms were used to bridge high-level quantum chemistry and semiempirical methods to provide an accurate representation of complex molecular excited-state and ground-state behavior, well beyond previous attempts, or expectation of human experts, and a dramatic reduction (from about 100 to 1000 times) in computational cost. Rapid reparameterization of semiempirical methods not only eliminates the need for a full-fledged *ab initio* dynamics simulation, which is prohibitively expensive for large molecules, but also eliminates drawbacks of semiempirical methods that use standard parameter sets and can yield unphysical dynamics. The results show that the evolutionary approach provides significantly better results—with up to 384% lower error in the energy and 86.5% lower error in the energy gradient—than those reported in literature. Furthermore, it also provides a large number of parameter sets, all of which yield globally accurate PESs and physical dynamics.

Even more surprising and potentially groundbreaking, the multiobjective GA results produce *transferable* potentials—that is, parameters from one molecular system can be used for similar systems. *Transferability* to chemists is analogous to building blocks to a GA researchers. Optimized semiempirical parameters of a small number of relatively simple molecules can be used to predict accurately the behavior of large complex molecules. More work needs to be done, but transferability of ethylene parameters to simulate accurately benzene, and vice versa, is strongly suggestive that GAs will enable the fast, accurate simulation of complex molecules from a standard GA-tuned database. This is an ultimate goal of this work, since it would allow direct simulation of photoinduced *cis-trans* isomerization in molecules such as stilbene and azobenzene, as well as energy transfer in dendrimeric molecules. Furthermore, this opens up the possibility of accurate simulations of photochemistry in complex environments such as proteins and condensed phases. If this pans out it will transform the way chemicals are modeled and designed radically.

In conclusion, the multiobjective GA-discovered potentials *inherit* the accuracy of the *ab initio* data, permit simulations to orders of magnitude larger time scales (multi-picoseconds) than currently possible by *ab initio* methods, even for simple molecules, and exhibit transferability in initial tests—the "Holy Grail" for materials and chemistry simulations. This multiobjective optimization approach is an enabling technology to simulate successfully, and within reasonable time frame and with sufficient accuracy, complex, multiscale biological, chemical and materials problems that are ubiquitous in science and engineering and thus impacting our ability to address critical biophysical simulations of, for example, vision and photosynthesis, and for automated design of pharmaceuticals and functional materials.

# Chapter 6

# Scalability of Genetic Programming

The previous two chapters clearly demonstrated the efficacy of genetic algorithms and genetic programming in multiscaling materials modeling. In order to systematically scale these methods and to tackle more complex problems we need to understand their scalability and *competence*. Competence here implies those GAs that can solve hard problems quickly, reliably, and accurately (Goldberg, 1999a; Goldberg, 2002). Using Goldberg's (Goldberg, 2002) design decomposition theory and facetwise modeling approaches the remainder of this thesis will address some of the scalability issues involved with GP and multiobjective GAs.

Central to the design-decomposition approach is the notion of *building blocks* or *minimal sequential superior subsolutions* of a problem (Goldberg, 2002). Based on this notion of building blocks, facetwise models for analyzing scalability of GP and multiobjective GAs (MOGAs) are considered and competent methods for each are proposed. Specifically, population sizing in GP dictated by adequate building-block supply (Sastry, O'Reilly & Goldberg, 2003) and accurate decision making between competing building blocks (Sastry, O'Reilly & Goldberg, 2004) are addressed in this chapter. The facetwise models are empirically verified on a broad class of test problems. A competent GP design—the extended compact genetic programming (eCGP) which is based on the extended compact genetic algorithm (eCGA) (Harik, 1999; Harik, Lobo & Sastry, 2006)—that solves a broad class of *adversarially-designed* boundedly-difficult problems using only polynomial (cubic) number of function evaluations is proposed (Sastry & Goldberg, 2003a) in chapter 7.

One of the requirements of multiobjective genetic algorithms is the ability to maintain a diverse set of optimal solutions, termed as Pareto-optimal solutions, in a reliable manner. The scalability of MOGAs in reliably maintaining is modeled and verified with empirical results (Sastry, Pelikan & Goldberg, 2005) in chapter 8. Along the way, based on the multiobjective Bayesian optimization algorithm (Khan, Goldberg & Pelikan, 2002; Pelikan, Sastry & Goldberg, 2005), a competent

MOGA design—the multiobjective extended compact genetic algorithm (meCGA)—is proposed that can solve boundedly-difficult problems using only a polynomial (quadratic) number of function evaluations (Pelikan, Sastry & Goldberg, 2006; Sastry, Pelikan & Goldberg, 2005).

The first topic of the analysis part of this thesis deals with population sizing in GP both from building-block supply and decision-making grounds and is discussed in the remainder of this chapter.

## 6.1 Population Sizing in GP

The growth in application of genetic programming (GP) to problems of practical and scientific importance is remarkable (Ebner et al., 2007; Keijzer et al., 2004; Koza et al., 2003; Riolo & Worzel, 2003; O'Reilly et al., 2004; Thierens et al., 2007). Yet, despite the increasing interest and empirical success, GP researchers and practitioners are often frustrated—sometimes stymied—by the lack of theory available to guide them in selecting key algorithm parameters or to help them explain empirical findings in a systematic manner. A key parameter determining performance of a GP is the population size. For example, small population sizes might lead to premature convergence and yield substandard solutions. On the other hand, large population sizes lead to unnecessary expenditure of valuable computational time. Population size should be large enough to ensure that all the raw substructures—tree segments that are potentially part of the solution to the search problem—are present in the initial population so that recombination and mutation can assemble and fine-tune them to yield high-quality solutions. Empirically, GP population sizes run from ten to a million members or more, but at present there is no practical guide to knowing when to choose which size.

The purpose here is to begin to address this lack of theory by providing an estimate of the population size necessary to solve a given GP problem. It is hoped that, like in the GA theory (Goldberg, 2002), the availability of a population sizing equation will be a valuable tool to aid GP practitioners in their efforts to understand how GP processes information, a pursuit that may eventually lead them to an understanding and design of competent GP (Looks, 2006; Sastry & Goldberg, 2003a; Shan et al., 2006). The first step towards understanding population sizing is to tackle the issue of building-block (BB) supply (Sastry, O'Reilly & Goldberg, 2003). The BB-supply population sizing model answers what population size is required to ensure the presence of all raw building blocks for a given tree size (or size distribution) in the initial population.

However, the building-block supply based population size is conservative because it does not guarantee the growth in the market share of good substructures. That is, while ensuring the building-block supply is important for a selectorecombinative algorithm's success, ensuring a growth in the market share of good building blocks by deciding correctly between competing building blocks is also critical (Goldberg, 2002). Moreover, the population sizing for GA success is usually bounded by the population size required for making good decisions between competing building blocks. Therefore, a population-sizing model to ensure good decision making between competing building blocks (Sastry, O'Reilly & Goldberg, 2004) is also presented in this chapter. The analytical approach is similar to that used by Goldberg, Deb and Clark (1992) for developing a population-sizing model based on decision-making for genetic algorithms (GAs).

## 6.2 Population Sizing for Adequate Building Block Supply

This section presents facetwise models for the supply of BBs and estimate the population size required to guarantee the presence of all raw BBs for a given tree size (or size distribution) in the initial GP population. Although ensuring BB growth supersedes BB supply in the subsequent population, BB growth will be extremely difficult if BB supply is not ensured. Although decision making usually governs the population sizing, sometimes BB supply is the dominating facet. In such cases a facetwise model of BB supply is necessary for ensuring a successful GP design. Furthermore, understanding initial supply of BBs is essential for developing a practical population-sizing model.

The section is structured as follows. We start with a brief literature review of BB supply. Section 6.2.2 provides background and states key assumptions made in this study. Details of an expression mechanism and test problems are provided in section 6.2.3, followed by facetwise models for BB supply in section 6.2.4. Section 6.2.5 outlines some thoughts on handling BB supply for real GP expressions.

### 6.2.1 Literature Review

The GP community is interested in identifying strategies to size populations, in order to estimate the computational effort required to solve particular problems with GP; however, few studies have addressed this topic, thus far. One approach has been suggested by Langdon and Poli (2002), but it has not been fully developed. The approach proposed by Langdon and Poli (2002) employs the methodology used by Poli (2000) for the sizing of populations in GA. Poli (2000) used Stephens and Waelbroeck's (1999) concept of transmission probability to develop a recursive conditional schema theory that allows for the prediction of the probability of reaching a solution to a problem in a fixed number of generations. An expression for the transmission probability for standard GP was developed by Langdon and Poli (2002). However, the expression is very difficult to evaluate.

The methodological and analytic foundation for our approach to deciphering selectorecombinative GA (Goldberg, 2002) and GP (Goldberg & O'Reilly, 1998; O'Reilly & Goldberg, 1998) has been stated before. Put succinctly, our approach is to analyze and understand GP's simple mechanisms before its complex ones. We predict that lessons learned from experimentation and theory on a simple case will lead to insight, and possibly, carry over to more complex cases. Therefore, we start by analyzing building-block supply in GP's initial population, before the activity of crossover and selection.

While building-block supply has been largely ignored in GP literature, many researchers have studied the BB supply in GAs. Holland (1975) estimated the number of BBs that receive at least a specified number of trials using Poisson distribution. A later study (Goldberg, 1989c) calculated the same quantity more exactly using binomial distribution and studied their effects on population sizing in serial and parallel computation. Reeves (1993) proposed a population sizing model for supply of alphabets with fixed cardinality. Recently, Goldberg, Sastry and Latoza (2001) developed facetwise models for ensuring BB supply in the initial population for genetic algorithms. They considered a population of fixed-length strings consisting alphabets of arbitrary cardinality $\chi$. They predicted that the population size required to ensure the presence of all competing building blocks with a tolerance of $\epsilon = 1/m$ is given by $n = \chi^k (k \log \chi + \log m)$, where $\chi$ is the alphabet cardinality, $k$ is BB size, and $m$ is the number of BBs.

This section follows a similar methodology along the lines of Goldberg, Sastry and Latoza (2001) and develops facetwise models for predicting the probability of the presence of a single schema as well as all schemas in a given partition. Before developing the models, we present some background and state assumptions used in the modeling procedure.

### 6.2.2 Preliminaries

In this section, we present definitions and concepts that underpin our analysis of BB supply in GP.

#### GP Tree Composition

Most GP implementations reported in the literature use parse trees to represent candidate programs in the population (Langdon & Poli, 2002). We have also assumed a tree representation in our analysis. To simplify the analysis further, we consider the following:

1. A primitive set of the GP tree is $\mathcal{F} \cup \mathcal{T}$, where $\mathcal{F}$ denotes the set of functions (interior nodes to a GP parse tree), and $\mathcal{T}$ denotes the set of terminals (leaf nodes in a GP parse tree).

2. The cardinality of $\mathcal{F} = \chi_f$ and the cardinality of $\mathcal{T} = \chi_t$.

3. The arity of all functions in the primitive set is two: All functions are binary and thus parse trees generated from the primitive set are binary.

We believe that our analysis could be extended to primitive sets containing functions with arity greater than two (non-binary trees). We also note that our assumption closely matches a common GP benchmark, symbolic regression, which frequently has arithmetic functions of arity two.

#### Translating GA Schemas to GP Schemas Isn't Straightforward

Schemas are similarity templates that describe sets of solutions that share a common feature. The GP literature contains several alternative definitions of schemas (Altenberg, 1994; Koza, 1992; Langdon & Poli, 2002; O'Reilly & Oppacher, 1995; Rosca, 1997; Whigham, 1995). Per O'Reilly and Oppacher (1995), a GP schema is a multiset of subtrees and tree fragments with nodes denoted as functions, terminals or "don't care" symbols. Tree fragments are trees with at least one leaf that is a "don't care" symbol which can be matched by any subtree (including subtrees with only one node).

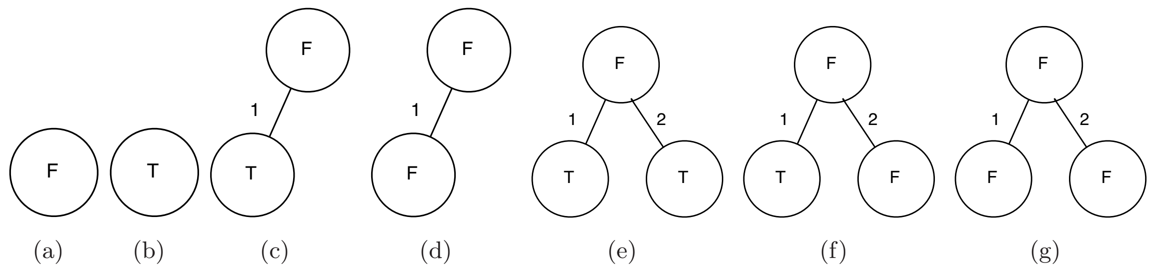


Figure 6.1: Smallest tree fragments in GP. Fragments (c) and (d) have mirrors where the child is the 2nd parameter of the function. Likewise, fragment (f) has mirror where the 1st and the 2nd parameters of the function are reversed.

**Tree Fragments**

While in general tree fragments refer to a multiset of tree patterns or tree templates, we restrict ourselves to a single tree pattern. A tree fragment pattern has each of its nodes labeled with the function symbol, $\mathcal{F}$, or terminal symbol, $\mathcal{T}$. However, it does not have an absolute position or positional anchor. Figure 6.1 shows the fragments our analysis focuses on. Along the edge between a function and its child node, a numeral denotes what parameter of the function the child node is (that is, the first or second argument in the case of a binary function). A tree fragment has a length or size; that is, its number of nodes, $k = N_t + N_f$, where $N_t$ and $N_f$ are the number of terminal and functional nodes in the tree fragment, respectively. Furthermore, the total number of possible instances of a tree fragment is given by

$$\kappa = \chi_f^{N_f} * \chi_t^{N_t}. \tag{6.1}$$

For example, for the tree fragment $P_b$ shown in Figure 6.1(b) (fragment with only terminal), $N_f = 0$, and $N_t = 1$, and therefore, the total number of instances of $P_b$ is $\chi_t$.

Since a tree fragment is not anchored to a position of a tree, it is possible that none or more than one instance of a fragment can be present in a single tree. Yet, the smallest fragments $P_a$ and $P_b$ (see Figures 6.1(a)–(b)) appear at least once or twice in a tree respectively. Assuming a single tree of size $s$[1] and the tree properties listed in section 6.2.2, Table 6.1 provides estimates (derived

[1]It should be noted here that the average tree size of a population can be calculated for popular initialization schemes (Koza, 1992), or initialization schemes such as PCT1 or PCT2 (Luke, 2000c) can be used to generate a population which conform to an expected tree size.

by probability of frequency) of the average quantity of tree-fragment instances, $\phi$. In other words, $\phi$ counts the expected number of tree-fragment instances, given the tree size (or size distribution), in the population.

Table 6.1: Designations, $P_i$, and descriptions of tree fragments considered in the BB supply models, the quantity of fragments $\phi$, and the number of competing schemas in the fragments $\kappa$, for a binary tree of size $s$. See also Figure 6.1.

| $P$ | **Description** | $\phi$ | $\kappa$ |
|---|---|---|---|
| $P_a$ | function | $\frac{1}{2}(s-1)$ | $\chi_f$ |
| $P_b$ | terminal | $\frac{1}{2}(s+1)$ | $\chi_t$ |
| $P_c$ | one terminal that is the first parameter of a binary function | $\frac{1}{4}(s+1)$ | $\chi_f \cdot \chi_t$ |
| $P_d$ | a function at the root and a function as its first parameter | $\frac{1}{4}(s-3)$ | $\chi_f^2$ |
| $P_e$ | a function at the root and 2 terminals as its parameters | $\frac{1}{8}\frac{(s+1)^2}{(s-1)}$ | $\chi_f \cdot \chi_t^2$ |
| $P_f$ | a function at the root and 1 terminal as the first parameter and one function as it second parameter. | $\frac{1}{8}\frac{(s+1)(s+3)}{(s-1)}$ | $\chi_f^3$ |
| $P_g$ | a binary function at the root and 2 functions as its parameters | $\frac{1}{8}\frac{(s-3)^2}{(s-1)}$ | $\chi_f^2 \cdot \chi_t$ |

**The Tree Fragments are not Enough: How are They Expressed?**

While tree fragments are the parts of a physical tree, and counting number of instances of tree fragments can itself be important, what is more important are those tree fragments that get *expressed*. The expression mechanism dictates what the building blocks of a problem are and therefore affects the BB supply. Specifically, we are interested in expression of small tree fragments into partially correct subfunctions. Let us consider, for example, symbolic regression of $1+x+x^2+x^3$. Early on in the GP run, it is important to get the constant and the linear part of the symbolic equation right. Therefore, all the tree fragments that contribute to the correct constant and linear subfunctions are important and their supply is critical in the initial population.

We illustrate the methodology to incorporate expression mechanism in BB supply models by using a simple expression mechanism, called `ORDER`, which is explained in the next section. We choose `ORDER` because while it models some of the GP behavior (Goldberg & O'Reilly, 1998; O'Reilly & Goldberg, 1998), the expression mechanism can be analyzed in a straightforward manner.

### 6.2.3 ORDER Expression Mechanism

ORDER is a simple, yet intuitive expression mechanism which makes it amenable to analysis and modeling (Goldberg & O'Reilly, 1998; O'Reilly & Goldberg, 1998). The primitive set of ORDER consists of the primitive JOIN of arity two and complimentary primitive pairs $(X_i, \bar{X}_i)$, $i = 0, 1, \cdots, \ell$. A candidate solution of the ORDER problem is a binary tree with JOIN primitive at the internal nodes and either $X_i$'s or $\bar{X}_i$'s at its leaves. The candidate solution's expression is determined by parsing the program tree inorder (from left to right). The program expresses the value $X_i$ if, during the inorder parse, a $X_i$ leaf is encountered before its complement $\bar{X}_i$. Furthermore, only unique primitives are expressed in ORDER during the inorder parse.

Building blocks in ORDER are the sets of primitives that are part of the subfunctions that improve fitness. In this study, we consider two test problems that use ORDER expression mechanism: 1. UNITATION: where each primitive $X_i$ is a BB, and 2. DECEPTION: where $k$ primitives form a BB. The following sections describe these two test problems.

#### UNITATION

In UNITATION, for each $X_i$ that is expressed, an equal unit of fitness value is accredited. That is,

$$f_1(x_i) = \begin{cases} 1 & \text{if } x_i \in \{X_1, X_2, \cdots, X_\ell\} \\ 0 & \textit{otherwise} \end{cases} . \tag{6.2}$$

The fitness function for ORDER is then defined as

$$F(\mathbf{x}) = \sum_{i=1}^{\ell} f_1(x_i), \tag{6.3}$$

where $\mathbf{x}$ is the set of primitives expressed by the tree. The output for optimal solution of a $\ell$-primitive UNITATION problem is $\{X_1, X_2, \cdots, X_\ell\}$, and its fitness value is $\ell$.

#### DECEPTION

In DECEPTION, the primitives are divided into $m$ subgroups, each subgroup consisting of $k$ primitives. The fitness of each subgroup is computed using the following trap function (Deb & Goldberg, 1992;

Goldberg, 1987):

$$f_k\left(u(x_1, x_2, \cdots, x_k)\right) = \begin{cases} 1.0 & u = k \\ (1.0 - \delta)\left(1 - \frac{u}{k-1}\right) & u < k \end{cases}, \tag{6.4}$$

where $u$ is the unitation, or the number of primitives, $X_i$, in a subgroup:

$$u\left(x_1, x_2, \cdots, x_k\right) = \sum_{i=1}^{k} f_1(x_i), \tag{6.5}$$

where $x_i$ is the $i$th primitive, $\delta$ is the difference in the functional value between the correct BB and its deceptive attractor. The fitness function of a candidate solution (tree) is then given by

$$F\left(\mathbf{x}\right) = f_k\left(u(x_1, x_2, \cdots, x_k)\right) + f_k\left(u(x_{k+1}, \cdots, x_{2k})\right) + \cdots + f_k\left(u(x_{(m-1)k+1}, \cdots, x_{mk})\right), \tag{6.6}$$

where $F$ is the fitness function, $\mathbf{x}$ is the expressed primitives, $m$ is the number of BBs, and $\ell = mk$.

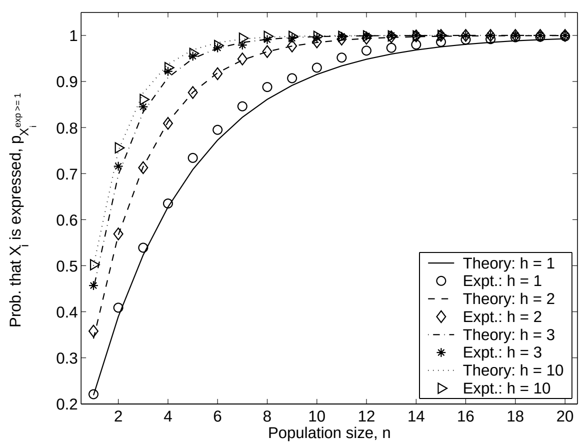
## 6.2.4 Facetwise Models of Building-Block Supply

In this section, we develop facetwise models for building-block supply for `ORDER` expression. First we start with addressing the supply of a single BB in a given partition. Then we extend the model to ensure the supply of all schemas in a partition. We then use the facetwise models to derive a population-sizing model dictated by BB supply. The models developed in this section are verified with empirical results for `UNITATION` and `DECEPTION` along the way.

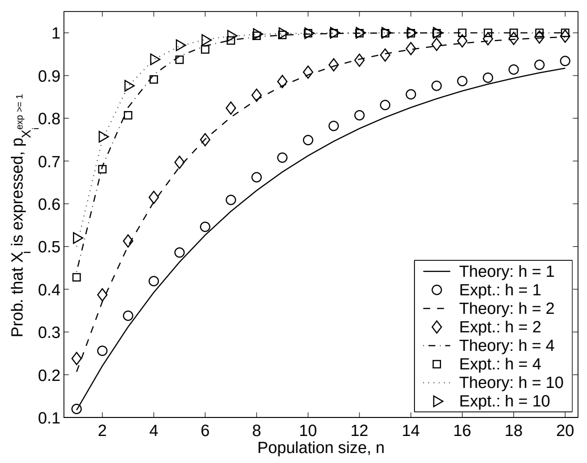
### Supply of a Single Building Block

Assuming trees of size, $s$, and that the expression mechanism used is `ORDER`, the probability that a primitive $X_i$ is expressed in a tree is given by

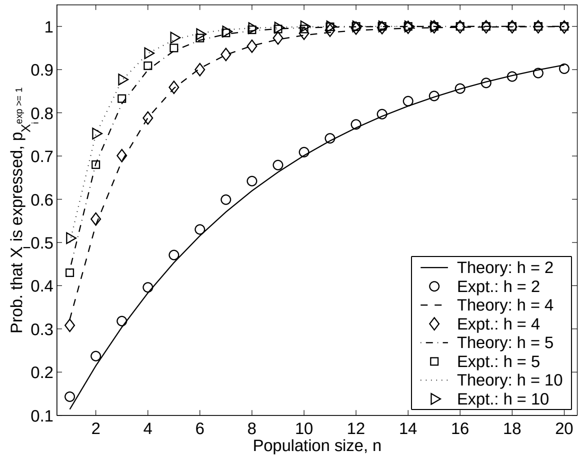
$$\begin{aligned} p_{X_i^{\text{exp}}} &= p\left(\#of X_i \geq 1, \#of \bar{X}_i = 0\right) + p\left(X_i \text{ appears before } \bar{X}_i\right), \\ &= \sum_{j=1}^{n_l} \begin{pmatrix} n_l \\ n_l - j \end{pmatrix} 2^{j-1} \left(\frac{\ell - 2}{\ell}\right)^{n_l - j} \left(\frac{1}{\ell}\right)^{j}, \\ &= \frac{1}{2\ell^{n_l}}\left[\ell^{n_l} - (\ell - 2)^{n_l}\right], \end{aligned}$$



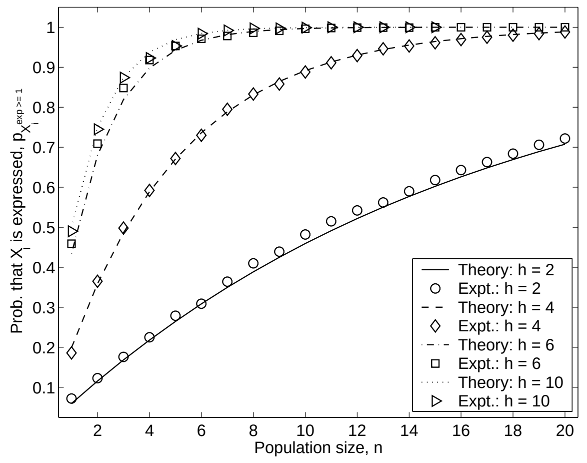
(a) Problem size, $\ell = 8$



(b) Problem size, $\ell = 16$



(c) Problem size, $\ell = 32$



(d) Problem size, $\ell = 64$

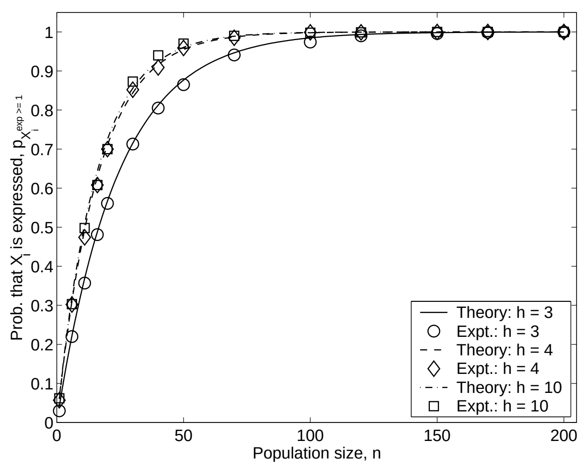
Figure 6.2: Verification of the facetwise model for a single BB supply (Equation 6.11) with empirical results for the `UNITATION` problem for different tree heights, $h$, as a function of population size, $n$. The empirical results depict the proportion of runs having at least one copy of a particular schema out of 1000 trials.

$$= \frac{1}{2}\left[1 - \left(1 - \frac{2}{\ell}\right)^{n_l}\right], \tag{6.7}$$

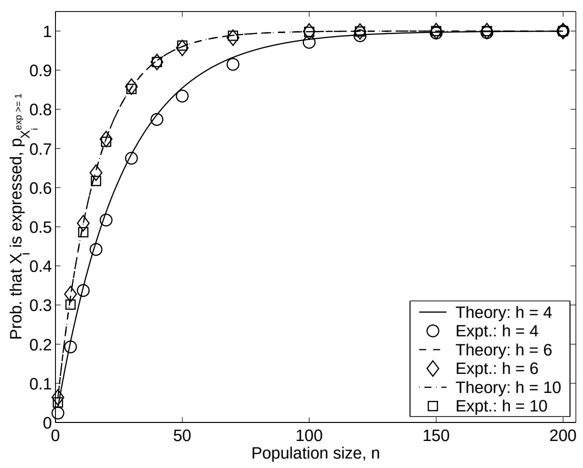
where $n_l = (s+1)/2$, is the number of leaf nodes in the tree, and $\ell = \chi_t$.

Assuming that primitives are expressed independent of each other, the probability that an order $k$ BB (without loss of generality, we will consider $X_1X_2\cdots X_k$) is expressed by a tree is given by

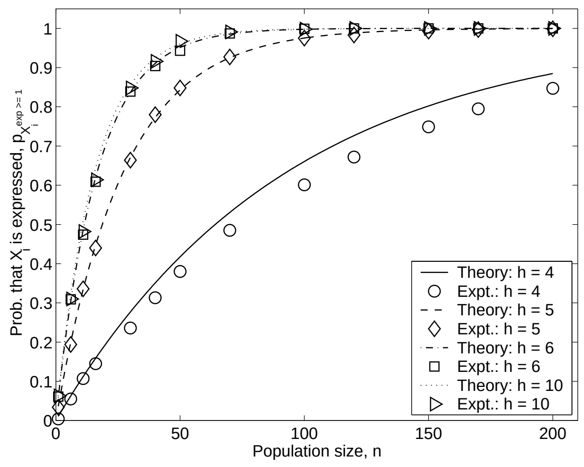
$$\begin{aligned} p_{X_{1\cdots k}^{\text{exp}}} &= p\left(X_i \text{ is expressed}\right)^k, \\ &= \left[\frac{1}{2}\left\{1 - \left(1 - \frac{2}{\ell}\right)^{n_l}\right\}\right]^k. \end{aligned} \tag{6.8}$$



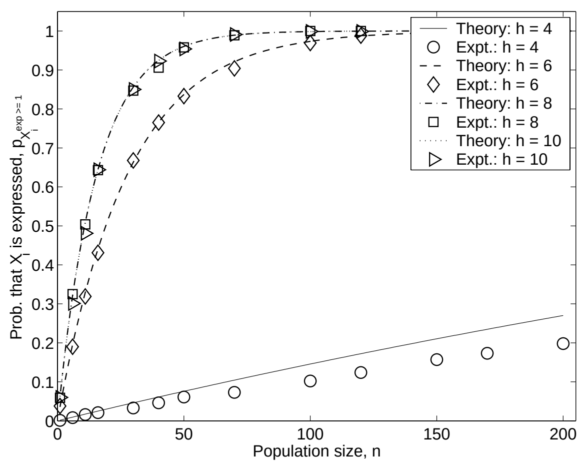
(a) Problem size, $\ell = 8$



(b) Problem size, $\ell = 16$



(c) Problem size, $\ell = 32$



(d) Problem size, $\ell = 64$

Figure 6.3: Verification of the facetwise model for a single BB supply (Equation 6.11) with empirical results for the `DECEPTION` problem for different tree heights, $h$, as a function of population size, $n$. The empirical results depict the proportion of runs having at least one copy of a particular schema out of 1000 trials.

The probability that the BB is not expressed by a tree is then given by

$$\begin{aligned} p_{X_{1\cdots k}^{\text{not exp}}} &= 1 - p_{X_{1\cdots k}^{\text{exp}}}, \\ &= 1 - \left[\frac{1}{2}\left\{1 - \left(1 - \frac{2}{\ell}\right)^{n_l}\right\}\right]^k. \end{aligned} \tag{6.9}$$

The probability that a BB is not expressed by any of the $n$ individuals in the population is given by

$$
\begin{aligned}
p_{X^{\mathrm{exp}=0}_{1\cdots k}} &= \left(p_{X^{\mathrm{not\ exp}}_{i\cdots k}}\right)^n, \\
&= \left[1 - \left[\frac{1}{2}\left\{1 - \left(1 - \frac{2}{\ell}\right)^{n_l}\right\}\right]^k\right]^n.
\end{aligned}
\tag{6.10}
$$

Therefore the probability that an order-$k$ BB is expressed by at least one individual in the population is given by

$$
\begin{aligned}
p_{X^{\mathrm{exp}\geq 1}_{1\cdots k}} &= 1 - p_{X^{\mathrm{exp}=0}_{1\cdots k}}, \\
&= 1 - \left[1 - \left[\frac{1}{2}\left\{1 - \left(1 - \frac{2}{\ell}\right)^{n_l}\right\}\right]^k\right]^n.
\end{aligned}
\tag{6.11}
$$

The model for single BB success given by Equation 6.11 is compared to empirical results for the `UNITATION` problem ($k = 1$) in Figure 6.2, and for the `DECEPTION` problem ($k = 4$) in Figure 6.3. The empirical results are for full trees, therefore, $n_l = 2^h$. The results show that the empirical results agree with the models.

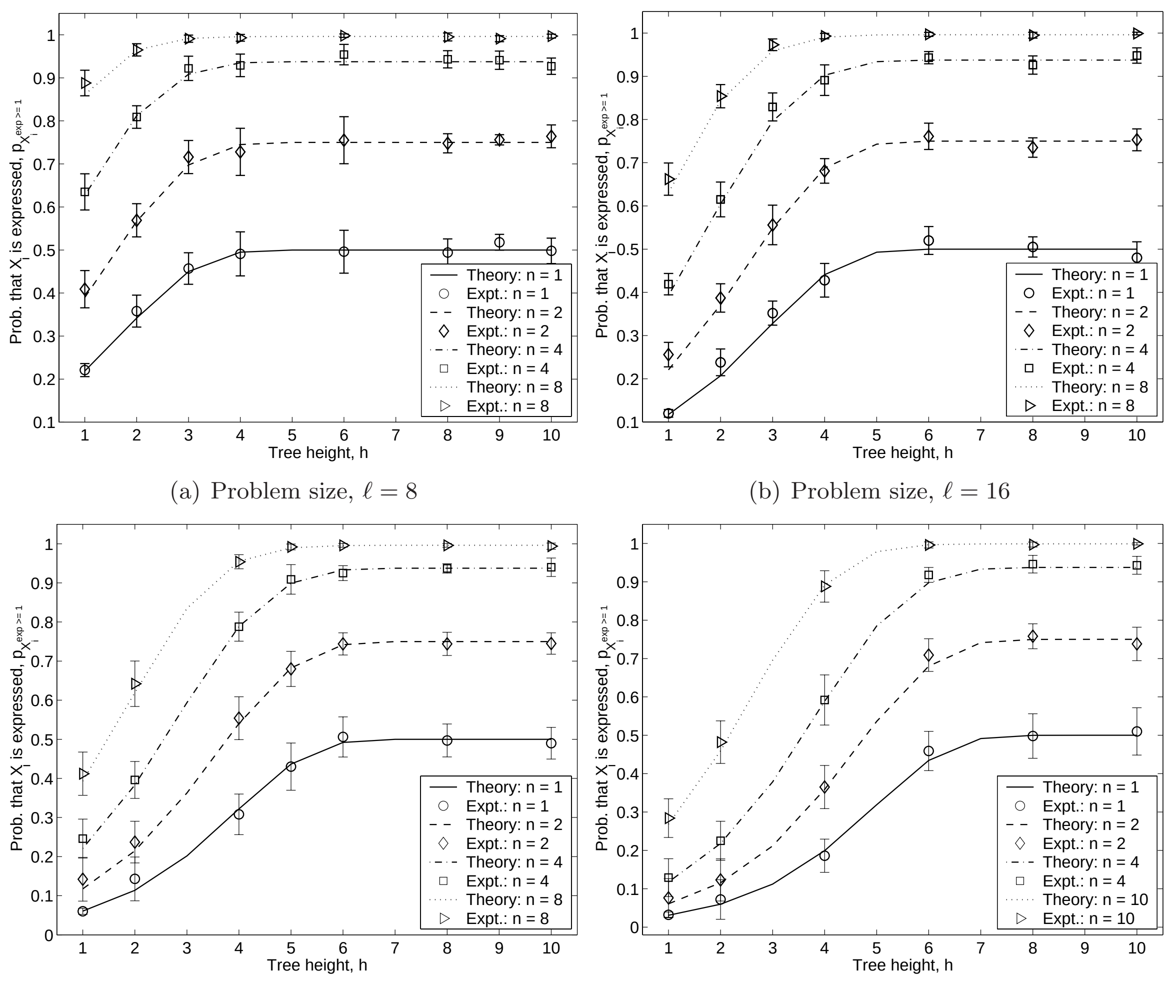
Using the approximation, $(1 - r/s)^s \approx e^{-r}$, and recognizing that this approximation is sufficiently accurate even for modest values of $s$, we can simplify Equation 6.11 as follows:

$$
p_{X^{\mathrm{exp}\geq 1}_{1\cdots k}} \approx 1 - \exp\left[-n2^{-k}\exp\left\{-k\exp\left(-\frac{2n_l}{\ell}\right)\right\}\right].
\tag{6.12}
$$

When $n_l \gg \ell$, $p_{X^{\mathrm{exp}\geq 1}_{i}} \approx 1 - \exp\left(-n2^{-k}\right)$. In other words, the probability of a BB being expressed by at least one individual, given a population size, increases with the tree size and saturates as $2^h > \ell$, as shown in Figure 6.4 for `UNITATION` problem.

### Supply for Partition Success

When solving real-world problems, one does not have prior knowledge about a particular schema being superior to others in a partition. Hence it is necessary to ensure that all competing schemas in a partition are present. The decision process would then be able to consider all the relevant alternative schemas. Therefore, in this section we extend the model developed in the previous section to ensure the presence of at least one copy of all the competing schemas (both the primitive



(a) Problem size, $\ell = 8$ (b) Problem size, $\ell = 16$

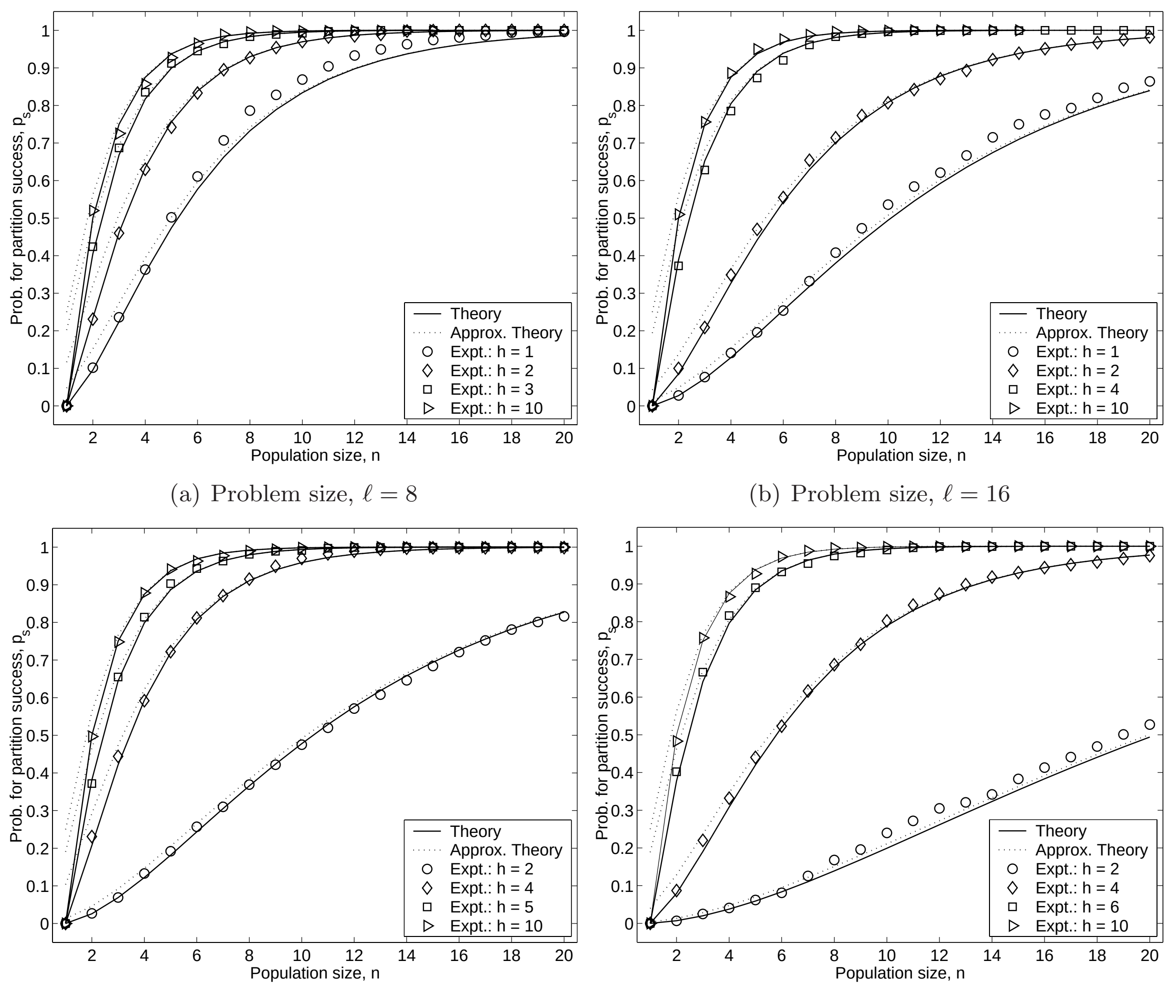
(c) Problem size, $\ell = 32$ (d) Problem size, $\ell = 64$

Figure 6.4: Verification of the facetwise model for a single BB supply (Equation 6.11) with empirical results for the UNITATION problem for different population size, $n$, as a function of tree height, $h$. The empirical results depict the proportion of runs having at least one copy of a particular schema out of 1000 trials.

and its complement) in a partition.

For ORDER, we are interested in the probability that all the $2^k$ possible schemas are present in the population. Assuming that individual schema success values are independent, the probability for partition success is given by

$$\begin{aligned} p_s &= \left(p_{X_{1\cdots k}^{\exp\geq 1}}\right)^{2^k}, \\ &= \left[1-\left[1-\left[\frac{1}{2}\left\{1+\left(1-\frac{2}{\ell}\right)^{n_l}\right\}\right]^k\right]^n\right]^{2^k}. \end{aligned} \tag{6.13}$$



(a) Problem size, $\ell = 8$

(b) Problem size, $\ell = 16$

(c) Problem size, $\ell = 32$

(d) Problem size, $\ell = 64$

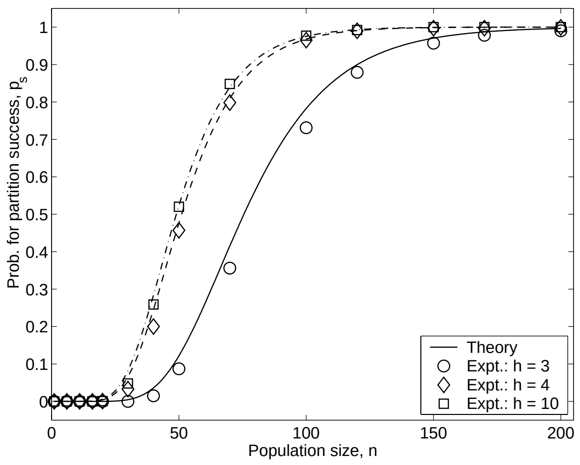
Figure 6.5: Verification of the models for BB partition success (Equations 6.16 and 6.14) with empirical results for `UNITATION` problem for different tree heights, $h$, and problem sizes, $\ell$, as a function of population size, $n$. The empirical results depict the proportion of runs having at least one copy of a primitive and its complement in the population out of 1000 trials.

Using the approximation $(1 - r/s)^s \approx e^{-r}$, the above equation can be further approximated as

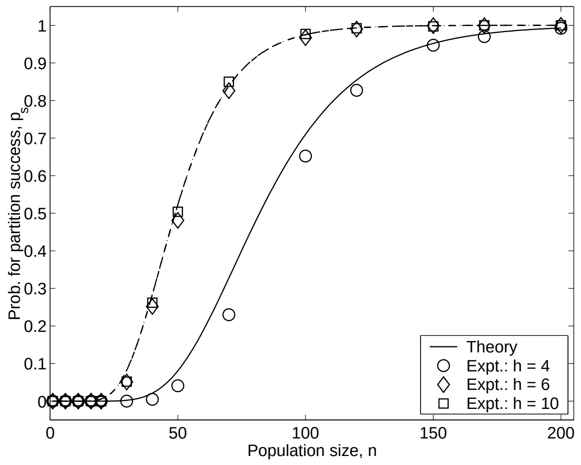
$$p_s \approx \exp\left[-2^k \exp\left[-n2^{-k} \exp\left\{-k \exp\left(-\frac{2^{h+1}}{\ell}\right)\right\}\right]\right]. \tag{6.14}$$

It should be noted that the independence assumption of individual schema success is an approximation and a more exact model can be derived which is illustrated for BB of unit size ($k = 1$).

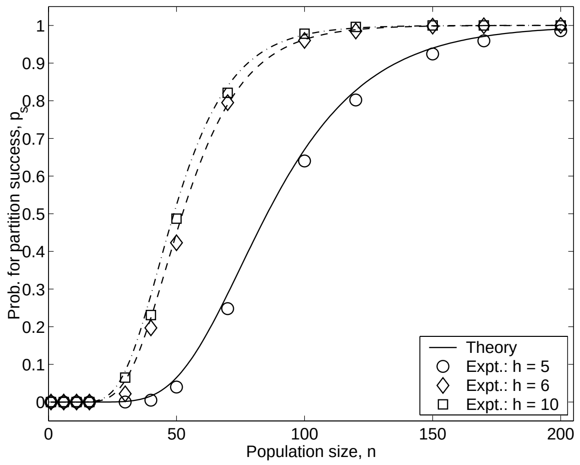
$$p_s = \sum_{i=2}^{n} \left(2^i - 2\right) \left( \begin{array}{c} n \\ n-i \end{array} \right) \left(p_{X_i^{\exp}}\right)^i \left(1 - 2p_{X_i^{\exp}}\right)^{n-i}, \tag{6.15}$$



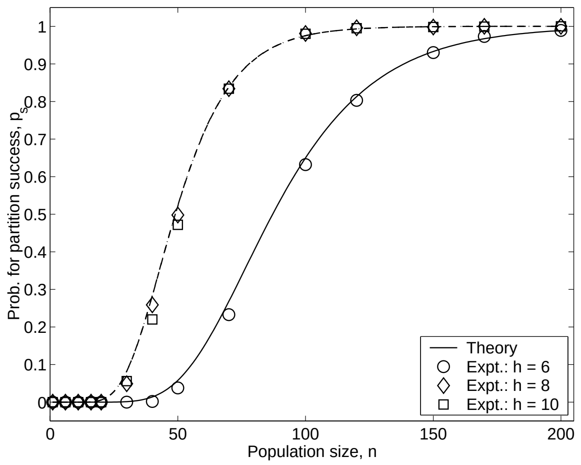
(a) Problem size, $\ell = 8$



(b) Problem size, $\ell = 16$



(c) Problem size, $\ell = 32$



(d) Problem size, $\ell = 64$

Figure 6.6: Verification of the BB partition success model (Equation 6.14) with empirical results for `DECEPTION` problem for different tree heights, $h$, and problem sizes, $\ell$, as a function of population size, $n$. The empirical results depict the proportion of runs having at least one copy of a primitive and its complement in the population out of 1000 trials.

The above equation can be rearranged as follows:

$$\begin{aligned} p_s &= \sum_{j=0}^{n-2} \begin{pmatrix} n \\ j \end{pmatrix} \left(2p_{X_i^{\text{exp}}}\right)^{n-j} \left(1-2p_{X_i^{\text{exp}}}\right)^{j} - 2\sum_{j=0}^{n-2} \begin{pmatrix} n \\ j \end{pmatrix} \left(p_{X_i^{\text{exp}}}\right)^{n-j} \left(1-2p_{X_i^{\text{exp}}}\right)^{j}, \\ &= 1 + \left(1-2p_{X_i^{\text{exp}}}\right)^{n} - 2\left(1-p_{X_i^{\text{exp}}}\right)^{n}. \end{aligned} \tag{6.16}$$

Equations (6.16) and (6.13) are compared with empirical results in Figure 6.5. The figures show that the approximate model (Equation 6.13) agrees with Equation 6.16 for higher population sizes and larger tree sizes. The partition success model (Equation 6.13) is compared with the empirical results for `DECEPTION` with $k = 4$, in Figure 6.6. Both Figures 6.5 and 6.6 clearly validate the BB supply model.

### Population Sizing for Building-Block Supply

The facetwise model derived in the previous section will be rearranged in this section to estimate the population size required to ensure the presence of all BBs of a partition for `ORDER`, given the problem size is $\ell$, and the tree height is $h$. Assuming that we can tolerate a probability $\epsilon$ of not having all BBs in a given partition, and setting $p_s$ to $1-\epsilon$, we can rewrite Equation 6.14,

$$1-\epsilon = \exp\left[-2^k \exp\left[-n2^{-k} \exp\left\{-k\exp\left(-\frac{2n_l}{\ell}\right)\right\}\right]\right]. \tag{6.17}$$

Taking logarithms on both sides of the above equation and using the approximation, $\ln(1-\epsilon) \approx -\epsilon$, we get

$$\epsilon = 2^k \exp\left[-n2^{-k} \exp\left\{-k\exp\left(-\frac{2^{h+1}}{\ell}\right)\right\}\right]. \tag{6.18}$$

After taking logarithms on both sides of the above equation and rearranging the resulting equation, we can write

$$n = 2^k\left(k\ln 2 - \ln\epsilon\right)\exp\left[-k\exp\left(-\frac{2n_l}{\ell}\right)\right]. \tag{6.19}$$

If we assume tree size to be big enough ($n_l \gg \ell$), then the above equation can be simplified as $n \approx 2^k\left(k\ln 2 - \ln\epsilon\right)$. Furthermore, if we assume that the supply error is inversely proportional to the number of BBs, $m$, that is, $\epsilon = 1/m$,

$$n \approx 2^k\left(k\ln 2 + \ln m\right). \tag{6.20}$$

It is interesting to note that the above population-sizing equation for BB supply in `DECEPTION` is identical to that developed by Goldberg, Sastry and Latoza (2001) for selectorecombinative GAs.

### 6.2.5 Some Thoughts On Modeling Realistic GP Expressions

The last section developed BB supply models for `ORDER` expression mechanism and verified it for two test problems for different parameter values. This section provides a brief outline on how to develop BB supply models for realistic GP expressions. First we start by addressing the supply of raw tree fragments, or in other words, we consider that every tree fragment in the tree is *expressed*.

#### Tree Fragment Supply

#### Single BB Success

The probability that a tree does not contain a partition, $P_i$, is given by

$$p\left(\# of P_i = 0\right) = \left(1 - \frac{1}{\kappa}\right)^{\phi}. \tag{6.21}$$

Recall that the values for $\kappa$, and $\phi$ for different partitions are given in Table 6.1. From the above equation, we can write the probability that the population contains at least one copy of the partition, $P_i$, as

$$p_k = 1 - \left[\left(1 - \frac{1}{\kappa}\right)^{\phi}\right]^n. \tag{6.22}$$

Using the approximation, $(1 - r/s)^s \approx e^{-r}$, and recognizing that this approximation is sufficiently accurate even for modest values of $s$, we can write

$$p_k \approx 1 - \exp\left(-\frac{n\phi}{\kappa}\right). \tag{6.23}$$

Furthermore, from Table 6.1, we can see that $\phi \approx 2^{-k}s$, where $k = N_t + N_f$. Substituting this approximation for $\phi$ in the above equation, we get

$$p_k \approx 1 - \exp\left(-\frac{n \cdot s}{\kappa \cdot 2^k}\right). \tag{6.24}$$

It should be noted that the approximation for $\phi$ is an underestimation for the tree fragments, $P_b$, $P_c$, $P_e$ and $P_f$, and an overestimation for the tree fragments, $P_a$, $P_d$, and $P_g$.

## Partition Success

Similar to the previous section, we assume that the schema partition success values are independent. Then, the probability of at least one success of each of the $\kappa$ schemas, $p_s$ is given by $p_s = p_k^\kappa$:

$$\begin{aligned} p_s &= \left[1 - \exp\left(-\frac{n \cdot s}{\kappa \cdot 2^k}\right)\right]^\kappa, && (6.25) \\ &\approx \exp\left[-\kappa \exp\left(-\frac{n \cdot s}{\kappa \cdot 2^k}\right)\right]. && (6.26) \end{aligned}$$

## Population Sizing for Partition Success

We now proceed to model the population size required to ensure the presence of all order-$k$ tree fragments. Assuming that we can tolerate a probability $\epsilon$ of not having all BBs in a given partition, and setting $p_s$ to $1 - \epsilon$, we can rewrite Equation 6.26,

$$1 - \epsilon = \exp\left[-\kappa \exp\left(-\frac{n \cdot s}{\kappa \cdot 2^k}\right)\right]. \tag{6.27}$$

Taking logarithm on both sides and using the approximation $\ln(1 - \epsilon) \approx -\epsilon$, for small values of $\epsilon$, gives

$$\epsilon = \kappa \exp\left(-\frac{n \cdot s}{\kappa \cdot 2^k}\right). \tag{6.28}$$

Solving the above equation for $n$ yields

$$n = \frac{1}{2} 2^k \kappa \left(\log \kappa - \log \epsilon\right). \tag{6.29}$$

Recall that $\kappa = \chi_f^{N_f} \chi_t^{N_t}$, and $k = N_f + N_t$. Then we can rewrite the above equation as

$$n = \frac{1}{s} \left(2\chi_f\right)^{N_f} \left(2\chi_t^{N_t}\right) \left[N_f \ln \chi_f + N_t \ln \chi_t - \ln \epsilon\right]. \tag{6.30}$$

This relation can be further simplified if we assume that the supply error is inversely proportional to the number of BBs, $m$, that is, $\epsilon = 1/m$. Then, the equation may be rewritten as

$$n = \frac{1}{s} \left(2\chi_f\right)^{N_f} \left(2\chi_t^{N_t}\right) \left[N_f \ln \chi_f + N_t \ln \chi_t + \ln m\right]. \tag{6.31}$$

**Incorporating Expression**

While counting the tree fragments may be useful enroute with proper expression model as in section 6.2.4, on its own it is not realistic. Therefore, we have to compute the combined probability that a tree fragment is present in the population and that it expresses a correct subfunction:

$$p(\text{BB is present}) = p(\text{fragment is present})p(\text{expression}). \tag{6.32}$$

In the above equation we assume that the events that a tree being present in the population and it being expressed are independent. It should be noted that this assumption becomes more accurate as the population size increases. The probability of a tree fragment being present in the population, $p(\text{fragment is present}) = p_k$, and is given by Equation 6.24, and the expression model is incorporated by the term $p(\text{expression})$. For example, in the symbolic regression example of $1+x+x^2+x^3$, the probability of expression incorporates the probability of different tree fragments expressing the linear and constant subfunctions.

## 6.3 Population Sizing for Good Decision Making

This section builds on the previous section wherein we considered the building block supply problem for GP. In the previous section, we asked what population size is required to ensure the presence of all raw building blocks for a given tree size (or size distribution) in the initial population. The building-block supply based population size is conservative because it does not guarantee the growth in the market share of good substructures. That is, while ensuring the building-block supply is important for a selectorecombinative algorithm's success, ensuring a growth in the market share of good building blocks by deciding correctly between competing building blocks is also critical (Goldberg, 2002). Furthermore, the population sizing for GA success is usually bounded by the population size required for making good decisions between competing building blocks. Our results herein show this to be the case, at least for the `ORDER` problem.

The purpose of this section is to derive a population-sizing model to ensure good decision making between competing building blocks. The analytical approach is similar to that used by Goldberg, Deb and Clark (1992) for developing a population-sizing model based on decision-making for genetic algorithms (GAs). In the proposed population-sizing model, we incorporate factors that are common to both GP and GAs, as well as those that are unique to GP. The population-sizing model is verified on three different test problems that span the dimension of building block *expression*—thus, modeling the phenomena of bloat at various degrees. Using `ORDER`, with `UNITATION` as its fitness function, provides a model problem where, per tree, a building block can be expressed only once despite being present multiple times. At the opposite extreme, our `LOUD` problem models a building block being expressed each time it is present in the tree. In between, the `ON-OFF` problem provides tunability of building block expression. A parameter controls the frequency with which a 'function' can suppress the expression of the subtrees below it, thus effecting how frequently a tree expresses a building block. This series of experiments not only validates the population-sizing relationship, but also empirically illustrates the relationship between population size and problem difficulty, solution complexity, bloat and tree structure.

### 6.3.1 GA Population Sizing from the Perspective of Competing Building Blocks

The derivational foundation for the proposed GP population-sizing model is the result for the selectorecombinative GAs by Goldberg, Deb and Clark (1992), wherein they consider how the GA can derive accurate estimates of BB fitness in the presence of detrimental noise. The model recognizes that, while selection is the principal decision maker, it distinguishes individuals based on their fitness and not the quality of subsolutions. Therefore, there is a possibility that an inferior building block gets selected over a better building block in a competition due to noisy observed contributions from adjoining building blocks that are also engaged in competitions.

To derive a relation for the probability of deciding correctly between competing BBs, the authors considered two individuals, one with the best BB and the other with the second best BB in the same competition (Goldberg, Deb & Clark, 1992).

Let $i_1$ and $i_2$ be these two individuals with $m$ non-overlapping BBs of size $k$ as shown in Figure 6.7. Individual $i_1$ has the best BB, $H_1$ ($111\cdots111$ in Figure 6.7) and individual $i_2$ has the second best BB, $H_2$ ($000\cdots000$ in Figure 6.7). The fitness values of $i_1$ and $i_2$ are $f_{H_1}$ and $f_{H_2}$ respectively. To derive the probability of correct decision making, we have to first recognize

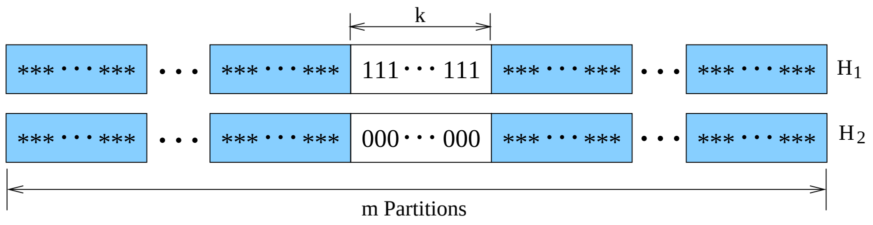

k
***···*** ··· ***···*** 111···111 ***···*** ··· ***···*** H1
***···*** ··· ***···*** 000···000 ***···*** ··· ***···*** H2
m Partitions

Figure 6.7: Two competing building blocks of size $k$, one is the best BB, $H_1$, and the other is the second best BB, $H_2$.

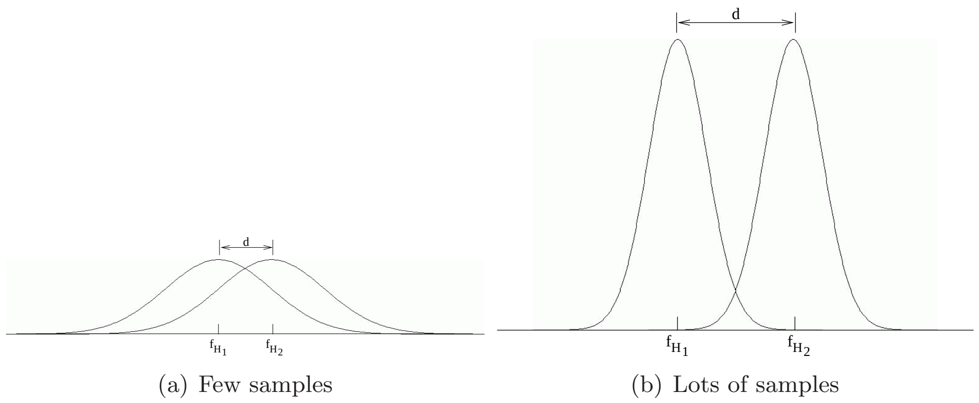


Figure 6.8: Fitness distribution of individuals in the population containing the two competing building blocks, the best BB $H_1$, and the second best BB $H_2$. When two mean fitness distributions overlap, low sampling increases the likelihood of estimation error. When sampling around each mean fitness is increased, fitness distributions are less likely to be inaccurately estimated.

that the fitness distribution of the individuals containing $H_1$ and $H_2$ is Gaussian since we have assumed an additive fitness function and the central limit theorem applies. Two possible fitness distributions of individuals containing BBs $H_1$ and $H_2$ are illustrated in Figure 6.8.

The distance between the mean fitness of individuals containing $H_1$, $\overline{f}_{H_1}$, and the mean fitness of individuals containing $H_2$, $\overline{f}_{H_2}$, is the *signal*, $d$. That is

$$d = \overline{f}_{H_1} - \overline{f}_{H_2}. \tag{6.33}$$

Recognize that the probability of deciding correctly between $H_1$ and $H_2$ is equivalent to the probability that $f_{H_1} - f_{H_2} > 0$. Also, since $f_{H_1}$ and $f_{H_2}$ are normally distributed, $f_{H_1} - f_{H_2}$ is also normally distributed with mean $d$ and variance $\sigma^2_{H_1} + \sigma^2_{H_2}$, where $\sigma^2_{H_1}$ and $\sigma^2_{H_2}$ are the fitness variances of individuals containing $H_1$ and $H_2$ respectively. That is,

$$f_{H_1} - f_{H_2} \sim \mathcal{N}(d, \sigma^2_{H_1} + \sigma^2_{H_2}). \tag{6.34}$$

The probability of correct decision making, $p_{dm}$, is then given by the cumulative density function of a unit normal variate which is the signal-to-noise ratio:

$$p_{dm} = \Phi\left(\frac{d}{\sqrt{\sigma^2_{H_1} + \sigma^2_{H_2}}}\right). \tag{6.35}$$

Alternatively, the probability of making an error on a single trial of each BB can estimated by finding the probability $\alpha$ such that

$$z^2(\alpha) = \frac{d^2}{\sigma^2_{H_1} + \sigma^2_{H_2}}, \tag{6.36}$$

where $z(\alpha)$ is the ordinate of a unit, one-sided normal deviate. Notationally $z(\alpha)$ is shortened to $z$.

Now, consider the BB variance, $\sigma^2_{H_1}$ (and $\sigma^2_{H_2}$): since it is assumed that the fitness function is the sum of $m$ independent subfunctions each of size $k$, $\sigma^2_{H_1}$ (and similarly $\sigma^2_{H_2}$) is the sum of the variance of the adjoining $m-1$ subfunctions. Also, since it is assumed that the $m$ partitions are uniformly scaled, the variance of each subfunction is equal to the average BB variance, $\sigma^2_{BB}$. Therefore,

$$\text{GA BB Variance:} \qquad \sigma^2_{H_1} = \sigma^2_{H_2} = (m-1)\sigma^2_{BB}. \tag{6.37}$$

A population-sizing equation was derived from this error probability by recognizing that as the number of trials, $\tau$, increases, the variance of the fitness is decreased by a factor equal to the number of trials:

$$z^2(\alpha) = \frac{d^2}{\frac{2(m-1)\sigma_{BB}}{\tau}}. \tag{6.38}$$

To derive the quantity of trials, $\tau$, assume a uniformly random population (of size $n$). Let $\chi$ represent the cardinality of the alphabet (2 for the GA) and $k$ the building-block size. For any individual, the probability of $H_1$ is $1/\kappa$ where $\kappa = \chi^k$. There is exactly one instance per individual of the competition, $\phi = 1$. Thus,

$$\tau = n \cdot p_{BB} \cdot \phi = n \cdot 1/\kappa \cdot 1 = n/\kappa. \tag{6.39}$$

By rearrangement and calling $z^2$ the coefficient $c$ (still a function of $\alpha$) a fairly general population-sizing relation was obtained:

$$n = 2c\chi^k(m-1)\frac{\sigma^2_{BB}}{d^2}. \tag{6.40}$$

To summarize, the decision-making based population sizing model in GAs consists of the following factors:

- **Competition complexity**, quantified by the total number of competing building blocks, $\chi^k$.
- **Subcomponent Complexity**, quantified by the number of building blocks, $m$.
- **Ease of decision making**, quantified by the signal-to-noise ratio, $d/\sigma^2_{BB}$.
- **Probabilistic safety factor**, quantified by the coefficient $c$.

### 6.3.2 GP Definitions for a Population Sizing Derivation

We use the same assumption detailed in section 6.2.2 made for developing the BB supply-based population sizing model. As in the BB supply model, the decision-making analysis adopts a definition of a GP schema (or similarity template) called a "tree fragment". A tree fragment is a tree with at least one leaf that is a "don't care" symbol. This "don't care" symbol can be matched by any subtree (including degenerate leaf only trees). As before, we are most interested in only the small set of tree fragments that are defined by three or fewer nodes.

The defining length of a tree fragment is the sum of its quantities of function symbols, $\mathcal{F}$, and terminal symbols, $\mathcal{T}$:

$$k = N_f + N_t. \tag{6.41}$$

Because a tree fragment is a similarity template, it also represents a competition. Since we are concerned with decision making, we will therefore use "competition" instead of a "tree fragment".

From Equation 6.1, we know that the size of a competition (that is, how many BBs compete) is

$$\kappa = \chi_f^{N_f} * \chi_t^{N_t}$$

As mentioned in the previous section, because a tree fragment is defined without any positional anchoring, it can appear multiple times in a single tree. We denote the number of instances of a tree fragment that are present in a tree of size $\lambda$, (that is, the quantity of a tree fragment in a tree) as $\phi$. This is equivalent to the instances of a competition as $\phi$ is used in the GA case (see Equation 6.39). For full binary trees:

$$\phi \approx 2^{-k}\lambda. \tag{6.42}$$

Later, we will explain how $\phi$ describes *potential* number of instances of a BB in a tree.

### 6.3.3 GP Population Sizing based on Decision Making

We now proceed to derive a GP population sizing relationship based on building block decision making. Preliminarily, unless noted, we also make the same assumptions as the GA population-sizing derivation outlined in Section 6.3.1.

The first manner in which the GP population-sizing derivation diverges from the GA case is how BB fitness variance (that is, $\sigma_{H_1}^2$ and $\sigma_{H_2}^2$) is estimated (for reference, see Equation 6.37). Recall that for the GA the source of a BB's fitness variance was collateral noise from the $(m-1)$ competitions of its adjoining BBs. In GP, the source of collateral noise is the average number of adjoining BBs present and expressed in each tree, denoted as $\bar{q}$. Thus:

$$\text{GP BB Variance:} \qquad \sigma_{H_1}^2 = \sigma_{H_2}^2 = [\bar{q}_{BB}^{expr}(m, \lambda) - 1]\sigma_{BB}^2. \tag{6.43}$$

Thus, the probability of making an error on a single trial of the BB can be estimated by finding the probability $\alpha$ such that

$$z^2(\alpha) = \frac{d^2}{2[\bar{q}_{BB}^{expr} - 1]\sigma_{BB}^2}. \tag{6.44}$$

The second way the GP population size derivation diverges from the GA case is in how the number of trials of a BB is estimated (for reference, see Equation 6.39). As with the GA, for GP we assume a uniformly distributed population of size $n$. In GP the probability of a trial of a particular BB must account for it being both present, $1/\kappa$, *and* expressed in an individual (or tree), which we denote as $p_{BB}^{expr}$. So, in GP:

$$\tau = \frac{1}{\kappa} \cdot p_{BB}^{expr} \cdot \phi \cdot n. \tag{6.45}$$

Thus, the population size relationship for GP is:

$$n = 2c\frac{\sigma_{BB}^2}{d^2}\kappa\left[\bar{q}_{BB}^{expr} - 1\right]\frac{1}{p_{BB}^{expr}\phi}, \tag{6.46}$$

where $c = z^2(\alpha)$ is the square of the ordinate of a one-sided standard Gaussian deviate at a specified error probability $\alpha$. For low error values, $c$ can be obtained by the usual approximation for the tail of a Gaussian distribution: $\alpha \approx \exp(-c/2)/(\sqrt{2c})$.

Obviously, it is not always possible to factor the real-world problems in the terms of this population sizing model. A practical approach would be to first approximate $\phi = 2^{-k}(\lambda)$ trials per tree (the full binary tree assumption). Then, we can estimate the size of the shortest program that will solve the problem, (one might regard this as the Kolomogorov complexity of the problem, $\lambda_k$), and choose a multiple of the shortest program size for $\lambda$ in the model. For the case where $\lambda$ is taken as a multiple of the shortest program size, $\bar{q} = c_k m_k$. To ensure the initial supply of building blocks that is sufficient to solve the problem, the initial population should be initialized with trees of size $\lambda$. Therefore, the population sizing in this case can be written as

$$n = c\frac{\sigma_{BB}^2}{d^2}\kappa\frac{(c_k m_k - 1)\,2^{k+1}}{p_{BB}^{expr}\lambda}. \tag{6.47}$$

Similar to the GA population sizing model, the decision-making based population sizing model in GP consists of the following factors:

- **Competition complexity**, quantified by the total number of competing building blocks, $\kappa$.
- **Ease of decision making**, quantified by the signal-to-noise ratio, $d/\sigma_{BB}^2$.
- **Probabilistic safety factor**, quantified by the coefficient $c$.
- **Number of subcomponents**, which unlike GA population-sizing, depends not only on the

minimum number of building blocks required to solve the problem $m_k$, but also on the tree size $\lambda$, the size of the problem primitive set and how bloat factors into trees (quantified by $p_{BB}^{expr}$).

### 6.3.4 Sizing Population for Model Problems

This section derives the components of the population-sizing model (Equation 6.46) for three test problems, `ORDER`, `LOUD`, and `ON-OFF`. We develop the population-sizing equation for each of these problems and verify them with empirical results. In all experiments we assume that $\alpha = 1/m$ and thus derive $c$. Table 6.2 shows some of these values. For all experiments, the initial population is randomly generated with either full trees or by the ramped half-and-half method. The trees were allowed to grow up to a maximum size of 1024 nodes. We used a tournament selection with tournament size of 4 in obtaining the empirical results. We used subtree crossover with a crossover probability of 1.0 and retained 5% of the best individuals from the previous population. A GP run was terminated when either the best individual was obtained or when a predetermined number of generations was exceeded. The average number of BBs correctly converged in the best individuals was computed over 50 independent runs. The minimum population size required such that $m-1$ BBs converge to the correct value is determined by a bisection method (Sastry, 2001). That is the error tolerance $\alpha$ is set to $1/m$. The results for population size and convergence time were averaged over 30 such bisection runs, while the results for the number of function evaluations were averaged over 1500 independent runs. We start with population sizing for `ORDER`, where a building block can be expressed at most once in a tree.

Table 6.2: Values of $c = z^2(\alpha)$ used in population sizing equation.

| $m$ | 8 | 16 | 32 | 64 | 128 |
|---|---|---|---|---|---|
| $c$ | .97 | 1.76 | 2.71 | 3.77 | 4.89 |

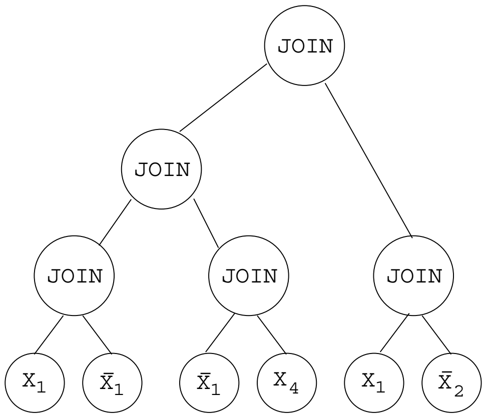


Figure 6.9: A candidate solution for a 4-primitive `ORDER` problem. The output of the program is $\{X_1, \bar{X}_2, X_4\}$ and its fitness is 2.

### `ORDER`: At most one expression per building block per tree

In `ORDER`, for each $X_i$ that is expressed, an equal unit of fitness value is accredited. That is,

$$f_1(x_i) = \begin{cases} 1 & \text{if } x_i \in \{X_1, X_2, \cdots, X_m\} \\ 0 & \textit{otherwise} \end{cases} . \tag{6.48}$$

The fitness function for `ORDER` is then defined as

$$F(\mathbf{x}) = \sum_{i=1}^{m} f_1(x_i), \tag{6.49}$$

where $\mathbf{x}$ is the set of primitives expressed by the tree. The output for an optimal solution of a $2m$-primitive `ORDER` problem is $\{X_1, X_2, \cdots, X_m\}$, and its fitness value is $m$. The building blocks in `ORDER` are the primitives, $X_i$, that are part of the subfunctions that reduce error (alternatively improve fitness). The shortest perfect program has $\lambda_k = 2m - 1$ nodes.

For example, consider a candidate solution for a 4-primitive `ORDER` problem as shown in Figure 6.9. The sequence of leaves for the tree is $\{X_1, \bar{X}_1, \bar{X}_1, X_4, X_1, \bar{X}_2\}$, the expression during inorder parse is $\{X_1, \bar{X}_2, X_4\}$, and its fitness is 2. For more details, motivations, and analysis of the `ORDER` problem, the interested reader should refer elsewhere (Goldberg & O'Reilly, 1998; O'Reilly & Goldberg, 1998).

For the `ORDER` problem, we can easily see that $\sigma^2_{BB} = 0.25$, $d = 1$, and $\phi = 1$. From the previous section, we know that

$$p_{BB}^{expr} \approx \exp\left[-k \cdot e^{-\frac{\lambda}{2m}}\right]. \tag{6.50}$$

Additionally, for ORDER, $\bar{q}_{BB}^{expr}$ is given by

$$\bar{q}_{BB}^{expr} = 1 + \sum_{i=0}^{m-1} \begin{pmatrix} m-1 \\ i \end{pmatrix} i \sum_{j=0}^{i} \begin{pmatrix} i \\ j \end{pmatrix} (-1)^j \left(\frac{i-j+1}{m}\right)^{n_l - 1}, \tag{6.51}$$

where $n_l$ is the average number of leaf nodes per tree in the population. The derivation of the above equation was involved and detailed, and is provided elsewhere (Sastry, O'Reilly & Goldberg, 2004).

Substituting the above relations (Equations 6.50 and 6.51) in the population-sizing model (Equation 6.46) we obtain the following population-sizing equation for ORDER:

$$n = 2^{k-1} z^2(\alpha) \left(\frac{\sigma_{BB}^2}{d^2}\right) \left[\bar{q}_{BB}^{expr} - 1\right] \exp\left[k \cdot e^{-\frac{\lambda}{2m}}\right]. \tag{6.52}$$

The above population-sizing equation is verified with empirical results in Figure 6.10. The initial population was randomly generated with either full trees or by the ramped half-and-half method with trees of heights, $h \in [h_k - 1, h_k + 1]$, where $h_k$ is the minimum tree height with an average of $2m$ leaf nodes.

As shown in Figure 6.11, we empirically observed that the convergence time and the number of function evaluations scale linearly and cubically with the program size of the most compact solution, $\lambda_k$, respectively. From this empirical observation, we can deduce that the population size for ORDER scales quadratically with the program size of the most-compact solution. For ORDER, $\lambda_k = 2m - 1$.

To summarize, for the ORDER problem, where a building block is expressed at most once per individual, the population size scales as $n = \mathcal{O}\left(2^k \lambda_k^2\right)$, the convergence time scales as $t_c = \mathcal{O}\left(\lambda_k\right)$, and the total number of function evaluations required to obtain the optimal solution scales as $n_{fe} = \mathcal{O}\left(2^k \lambda_k^3\right)$.

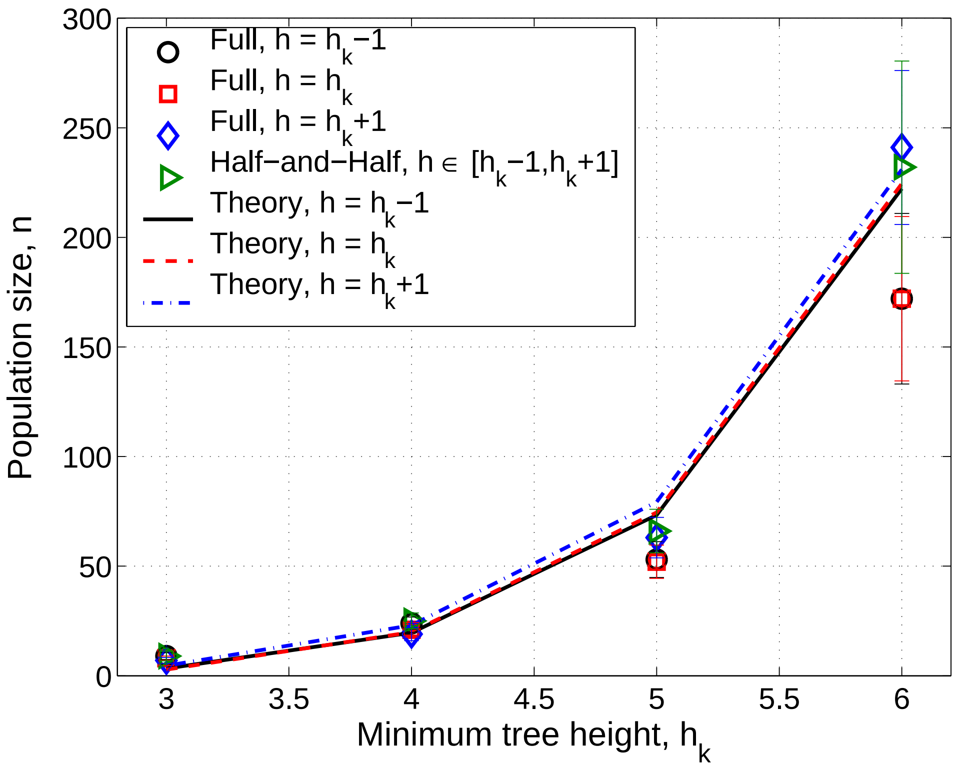


Figure 6.10: Empirical validation of the population-sizing model (Equation 6.52) for ORDER problem. Tree height $h_k$ equals $2^m$ and $\lambda = 2m - 1 = 2^{h+1} - 1$.

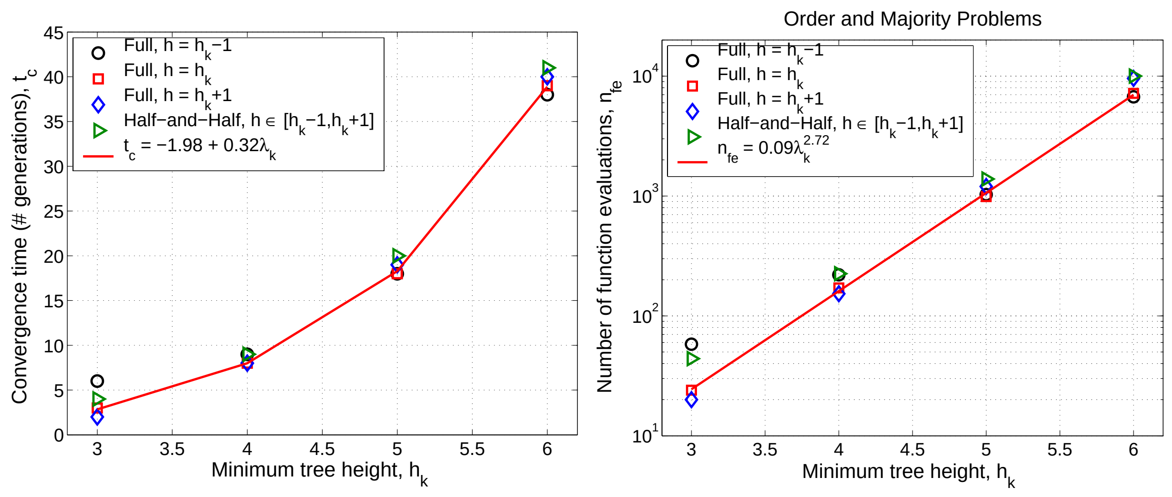
**LOUD: Every building block in a tree is expressed**

In ORDER, a building block could be expressed at most once in a tree, however, in many GP problems a building block can be expressed multiple times in an individual. Indeed, an extreme case is when every building block occurrence is expressed. One such problem is a modified version of a test problem proposed by Soule (2002) (see also (Soule, 2003; Soule & Heckendorn, 2002)), which we call as LOUD.

In LOUD, the primitive set consists of an "add" function of arity two, and three constant terminals 0, 1 and 4. The objective is to find an optimal number of fours and ones. That is, for an individual with $i$ 4s and $j$ 1s, the fitness function is given by

$$F(\mathbf{x}) = |i - m_4| + |j - m_1| , \tag{6.53}$$

where $m_4$ and $m_1$ denote the optimal number of 4s and 1s, respectively. In LOUD, even though



(a) Convergence Time, $t_c$ (b) Total number of function evaluations, $n_{fe}$

Figure 6.11: Empirical results for the convergence time and the total number of function evaluations required to obtain the global solution for ORDER problem. Note that $\lambda_k = 2m - 1$ so convergence time and the number of function evaluations scale linearly and cubically with the program size of the most compact solution or problem difficulty. The implication is that population size for ORDER problem is quadratic.

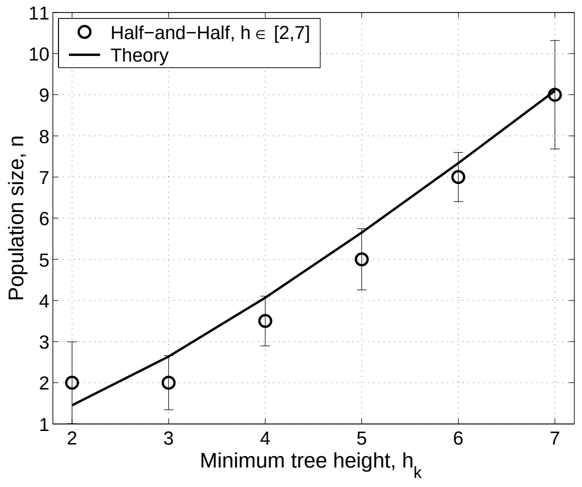
a zero is expressed it does not contribute to fitness. On the other hand, a 4 or 1 is expressed each time it appears in an individual and each occurrence contributes to the fitness value of the individual. The problem size, $m = m_4 + m_1$ and $\lambda_k = 2m - 1$ .

For the LOUD problem the building blocks are "4" and "1". It is easy to see that for LOUD, $\sigma^2_{BB} = 0.25$, $d = 1$, $\phi = \lambda/2$, and $p^{expr}_{BB} = 1/3$. Furthermore, the average number of building blocks expressed is given by $\bar{q}^{expr}_{BB} = 2n_l/3 \approx \lambda/3$. Substituting these values in the population-sizing model (Equation 6.46) we obtain

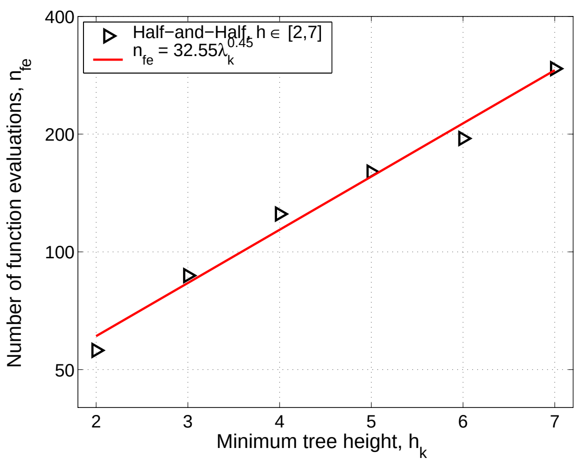
$$n = 2 \cdot 3^k z^2(\alpha) \left(\frac{\sigma^2_{BB}}{d^2}\right) \left[\frac{1}{3}\lambda - 1\right] \cdot \left(\frac{2}{\lambda}\right). \tag{6.54}$$

The above population-sizing equation is verified with empirical results in Figure 6.12. The initial population was randomly generated by the ramped half-and-half method with trees of heights, $h \in [2, 7]$ yielding an average tree size of 4.1 (this value is analytically 4.5).

We empirically observed that the convergence time was constant with respect to the problem size, and the number of function evaluations scales sub-linearly with the program size of the most-compact solution, $\lambda_k$. From this empirical observation, we can deduce that the population size for



(a) Population size, $n$



(b) Total number of function evaluations, $n_{fe}$

Figure 6.12: Empirical validation of the population-sizing model (Equation 6.54) and empirical results for the total number of function evaluations required to obtain the global solution for LOUD problem. Convergence time was constant with respect to problem size. Note that $\lambda_k = 2m - 1$ so the number of function evaluations scales sub-linearly with the program size of the most compact solution or problem difficulty. The implication is that population size for LOUD problem is sub-linear.

LOUD scales sub-linearly with the program size of the most-compact solution. For LOUD $\lambda_k = 2m-1$.

To summarize, for the LOUD problem, where a building block is expressed each time it occurs in an individual, the population size scales as $n = \mathcal{O}\left(3^k \lambda_k^{0.5}\right)$, the convergence time is almost constant with the problem size, and the total number of function evaluations required to obtain the optimal solution scales as $n_{fe} = \mathcal{O}\left(3^k \lambda_k^{0.5}\right)$.

## ON-OFF: Tunable building block expression

In the previous sections we considered two extreme cases, one where a building block could be expressed at most once in an individual and the other where every building block occurrence is expressed. However, usually in GP problems, some of the building blocks are expressed and others are not. For example, a building block in a non-coded segment is neither expressed nor contributes to the fitness. Empirically, Luke (2000a) calculated the percentage of inviable nodes in runs of the 6 and 11 bit multiplexer problems and symbolic regression over the course of a run. This value is seen to vary between problems and change over generations. Therefore, the third test function, which we call ON-OFF, is one in which the probability of a building block being expressed is tunable.

In ON-OFF, the primitive set consists of two functions EXP and $\overline{\texttt{EXP}}$ of arity two and terminals $\texttt{X}_1$,

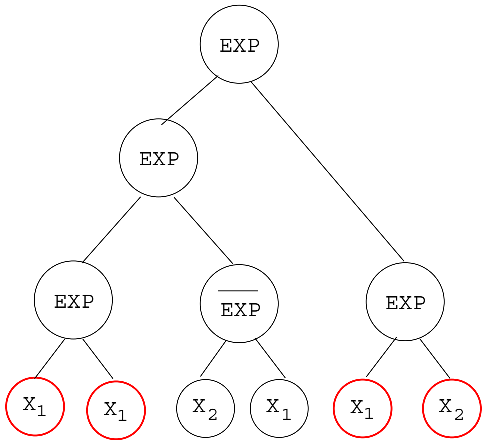


Figure 6.13: A candidate solution for a 2-primitive ON-OFF problem. The output of the program is $\{X_1, X_1, X_1, X_2\}$ and its fitness is $|3 - m_{x_1}| + |1 - m_{x_2}|$.

and $\mathtt{X}_2$. The function EXP expresses its child nodes, while $\overline{\text{EXP}}$ suppresses its child nodes. Therefore a leaf node is expressed only when all its parental nodes have the primitive EXP. This function can potentially approximate some bloat scenarios of standard GP problems such as symbolic-regression and multiplexer problems where invalidators are responsible for nullifying a building block's effect (Luke, 2000a). The probability of expressing a building block can be tuned by controlling the frequency of selecting EXP for an internal node in the initial tree.

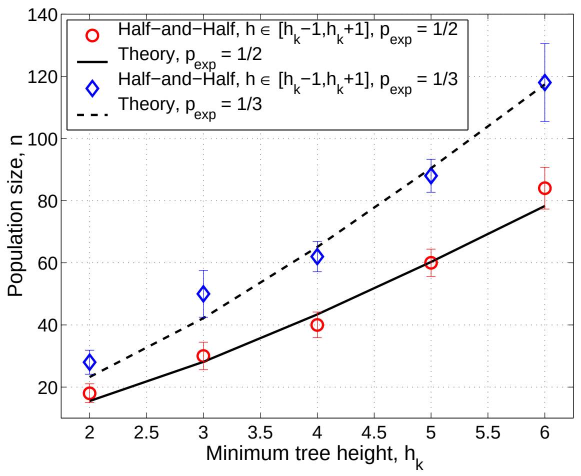
Similar to LOUD, the objective for ON-OFF is to find an optimal number of fours and ones. That is, for an individual with $i$ $\mathtt{X}_1$s and $j$ $\mathtt{X}_2$s, the fitness function is given by

$$F(\mathbf{x}) = |i - m_{X_1}| + |j - m_{X_2}| \,. \tag{6.55}$$

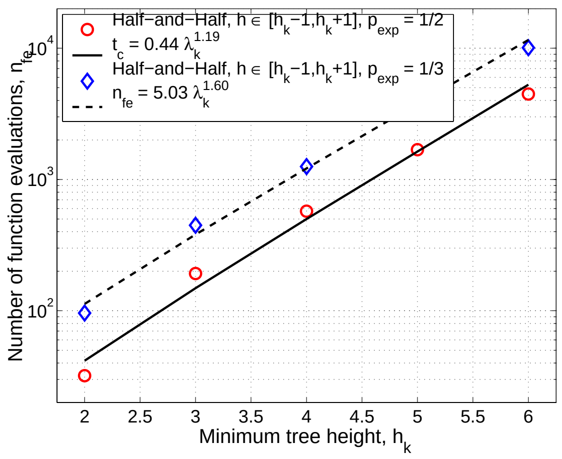
The problem size $m = m_{X_1} + m_{X_2}$ and $\lambda_k = 2m - 1$.

For example, consider a candidate solution for the LOUD problem as shown in Figure 6.13. The terminals that are expressed are $\{X_1, X_1, X_1, X_2\}$ and the fitness is given by $|3 - m_{x_1}| + |1 - m_{x_2}|$.

For the ON-OFF problem the building blocks are $X_1$ and $X_2$, $\sigma^2_{BB} = 0.25$, $d = 1$, $\phi = \lambda/2$, and $p^{expr}_{BB} = p^h_{EXP}$. Here, $p_{EXP}$ is the probability of a node being the primitive EXP. The average number of building blocks expressed is given by $\bar{q}^{expr}_{BB} = n_l \cdot p^h_{EXP} \approx \frac{s}{2} \cdot p^h_{EXP}$. Substituting these



(a) Population size, $n$



(b) Total number of function evaluations, $n_{fe}$

Figure 6.14: Empirical validation of the population-sizing model (Equation 6.56) and empirical results for the total number of function evaluations required to obtain the global solution for `On-Off` problem. Convergence time was constant with respect to problem size. Note that $\lambda_k = 2m - 1$. The convergence time scales linearly $\mathcal{O}(\lambda_k)$, and the number of function evaluations scales sub-quadratically $\mathcal{O}(\lambda_k^1.5)$ with the program size of the most compact solution or problem difficulty. Therefore, the population size for `On-Off` problem scales sub-linearly $\mathcal{O}(\lambda_k^0.5)$.

values in the population-sizing model (Equation 6.46) we obtain

$$n = 2^{k+1} z^2(\alpha) \left(\frac{\sigma_{BB}^2}{d^2}\right) \left[\frac{\lambda}{2} p_{EXP}^h - 1\right] \cdot \left(\frac{2}{\lambda p_{EXP}^h}\right). \tag{6.56}$$

The above population-sizing equation is verified with empirical results in Figure 6.14. The initial population was randomly generated by the ramped half-and-half method with trees of heights, $h \in [h_k - 1, h_k + 1]$, where $h_k$ is the minimum tree height with an average of $m$ leaf nodes. We empirically observed that the convergence time was linear with respect to the problem size, and the number of function evaluations scales sub-quadratically with the program size of the most-compact solution, $\lambda_k$. From this empirical observation, we can deduce that the population size for `On-Off` scales sub-linearly with the program size of the most-compact solution ($\lambda_k = 2m - 1$).

To summarize, for the `On-Off` problem, where a building block expression is tunable, the population size scales as $n = \mathcal{O}\left(2^k \lambda_k^{0.5}/p_{exp}\right)$, the convergence time scales linearly as $t_c = \mathcal{O}\left(2^k \lambda_k / p_{exp}\right)$, and the total number of function evaluations required to obtain the optimal solution scales as $n_{fe} = \mathcal{O}\left(2^k \lambda_k^{1.5}/p_{exp}^2\right)$.

## 6.4 Summary

In this chapter, facetwise models for scalability of genetic programming were developed. Specifically, population-sizing models were developed to ensure (1) adequate supply of raw building blocks in the initial population, and (2) accurate decision-making between competing building blocks.

First a detailed analysis of building-block supply in the initial population of GP using `ORDER` expression was presented. Two facetwise models were derived, one for ensuring the supply of a single schema in a partition, and the other for ensuring the supply of all competing schemas in a partition for problems which employ `ORDER` expression mechanism. The latter model has been employed to estimate the population size required to ensure the presence of at least one copy of all raw BBs of a partition in the initial population. The population sizing model indicates that there is a minimum tree size dependent on the problem size. Furthermore, the models suggest that when the tree size is greater than the problem size, the population size required on BB supply grounds is $2^k\left(k\ln\chi + \ln m\right)$. This study also shows that the population size required to ensure the presence of all instances of tree fragments (assuming that all of them are expressed) is $\frac{1}{s}\left(2\chi_f\right)^{N_f}\left(2\chi_t\right)^{N_t}\left[N_f\ln\chi_f + N_t\ln\chi_f + \ln m\right]$.

More importantly, in the process of deriving the building-block supply model, we gained valuable insight into a) what makes GP different from a GA in the sizing context and b) the implications of these differences. The difference of GP's larger alphabet, while influential in implying GP needs larger population sizes, was not a difficult factor to handle while bloat and the variable length individuals in GP are more complicated.

Moving to the second step, by considering a decision making model, we extended the GA decision making model along these dimensions. First, the proposed model retains a term describing collateral noise from competing BBs ($\bar{q}[m,\lambda]$) but it recognizes that the quantity of these competitors depends on tree size and the likelihood that the BB is present and expresses itself (rather than behave as an intron). Second, the proposed model, like its GA counterpart, assumes that trials decrease BB fitness variance, however, what was simple in a GA—there is one trial per population member, for the GP case is more involved. That is, the probability that a BB is present in a population member depends both on the likelihood that it is present in lieu of another BB *and* expresses itself, *plus* the number of potential trials any BB has in each population member.

The decision-making population-sizing model for GP shows that, to ensure correct decision making within an error tolerance, population size must go up as the probability of error decreases, noise increases, alphabet cardinality increases, the signal-to-noise ratio decreases, tree size decreases and bloat frequency increases. This obviously matches intuition. There is an interesting critical trade-off with tree size with respect to determining population size: pressure for larger trees comes from the need to express all correct BBs in the solution while pressure for smaller trees comes from the need to reduce collateral noise from competing BBs.

The decision-making model is conservative because "[...]it assumes that decisions are made irrevocably during any given generation. It sizes the population to ensure that the correct decision is made on average in a single generation" (Goldberg, 2002). A more accurate and different model would account for how correct decision making accumulates over the course of a run, and how, over the course of a run, improper decision making can be rectified.

The fact that the model is based on statistical decision making means that crossover does not have to be incorporated. In GAs crossover solely acts as a mixer or combiner of BBs. Interestingly, in GP, crossover also interacts with selection with the potential result that programs' size grows and structure changes. When this happens, the frequency of bloat can also change (see (Luke, 2000a; Luke, 2000b) for examples of this with multiplexer and symbolic regression). These changes in size, structure and bloat frequency imply a much more complex model if one were to attempt to account for decision making throughout a run. They also suggest that when using the model as a rule of thumb to size an initial population it may prove more accurate if the practitioner overestimates bloat in anticipation of subsequent tree growth causing more than the bloat seen in the initial population, given its average tree size.

It appears difficult to use this model with real problems where, among the GP-specific factors, the most compact solution and BB size is not known a priori and the extent of bloat cannot be estimated. In the case of the GA model, the estimation of model factors has been addressed by Reed, Minsker and Goldberg (2000), where they estimated variance with the standard deviation of the fitness of a large random population. In the GP case, this sampling population should be controlled for average tree size. If a practitioner were willing to work with crude estimates of bloat, BB size and most compact solution size, a multiple of the size of the most compact solution could be plugged in, and bloat could be used with that size to estimate the probability that a BB is expressed and present and the average number of BBs of the same size present and expressed, on average, in each tree. In the future we intend to experiment with the model and well known toy GP problems (for example, multiplexer, symbolic regression) where bloat frequency and most compact problem size are obtainable, and simple choices for BB size exist to see whether the ideal population size scales with problem size within the order of complexity the model predicts.

As in GAs, population sizing is very important for successful application of GP, because it is the principal factor in controlling ultimate solution quality. Once the quality-size relation is understood, populations can be sized to obtain a desired quality and only two things can happen in empirical trials. The quality goal can be equaled or exceeded, in which case, all is well with the design of the algorithm, or (as is more likely) the quality target can be missed, in which case there is some other obstacle to be overcome in the algorithm design. Moreover, once the population sizing is understood in this way it can be combined with an understanding of run duration, thereby yielding first estimates of GP run complexity, a key milestone in making our understanding of these processes more rigorous.

# Chapter 7

# Scalable Genetic Programming: Extended Compact Genetic Programming

The scalability analysis of the previous chapter indicated that with ideal recombination operator, genetic programming scales cubically with the problem size. This chapter presents a competent GP design that emulates properties of an ideal recombination operator and solves boundedly difficult problems quickly, reliably and accurately. While the design of competent genetic algorithms has received systematic and careful attention (Goldberg, 1999a; Goldberg, 2002), attempts in designing scalable genetic programming are limited. That is, even though the growth in application of genetic programming (GP) to problems of practical and scientific importance has been remarkable (Koza, 1992; Koza et al., 2003; Spector et al., 1999), there has been limited attention given to the development of competent operators that adapt linkage. Most of the studies on GP employ fixed crossover operators such as sub-tree crossover. Analyses of fixed recombination operators in selectorecombinative GAs suggest that fixed operators had highly inadequate and suggest operators that adapt linkage are essential for solving GA-hard problems in tractable time (Thierens, 1999; Thierens & Goldberg, 1993; Sastry & Goldberg, 2003b).

Therefore, the purpose of this chapter is to present one of the first designs of a competent GP—called the extended compact genetic programming (eCGP)—that adaptively identifies and propagates important subsolutions of a search problem. The proposed algorithm is similar to the probabilistic incremental program evolution (PIPE) (Sałustowicz & Schmidhuber, 1997). PIPE is a univariate probabilistic model building technique and is based on Baluja's PBIL (Baluja, 1994). In PIPE computer programs or mathematical functions are evolved as in genetic programming. Programs are represented by trees where each internal node represents a function/instruction and leaves represent terminals. In PIPE, probabilistic representation of the program trees is used and the probabilities of each instruction in each node in a maximal possible tree are used to model promising and generate new programs.

Similar to other univariate estimation of distribution algorithms, PIPE can only handle problems where there is no interactions among variables. However, there are large classes of problems where variables do interact and it is essential to capture those interactions and design operators that respect these higher-order interactions. Therefore, the proposed algorithm is not only capable of handling univariate variable interactions, but also can capture fairly complex multivariate interactions among the variables of the search problem. The multivariate interactions are captured in a similar fashion as in the extended compact genetic algorithm (eCGA) (Harik, 1999; Harik, Lobo & Sastry, 2006). Initial scalability results show that eCGP solves problems of bounded difficulty in polynomial time, as opposed to exponential time required by simple GP.

This chapter is structured as follows. The next section presents a synopsis of the extended compact genetic algorithm. The key features of the proposed algorithm (eCGP) are described in section 7.2 followed by an outline of test problems used to compare the performance of eCGP and simple GP in section 7.3. Section 7.4 describes the results followed by a note on the future work and key conclusions.

## 7.1 Extended Compact Genetic Algorithm (eCGA)

The extended compact GA proposed by Harik (1999) is based on a key idea that the choice of a good probability distribution is equivalent to linkage learning. The measure of a good distribution is quantified based on minimum description length (MDL) models. The key concept behind MDL models is that all things being equal, simpler distributions are better than more complex ones. The MDL restriction penalizes both inaccurate and complex models, thereby leading to an optimal probability distribution. Thus, MDL restriction reformulates the problem of finding a good distribution as an optimization problem that minimizes both the probability model as well as population representation. The probability distribution used in eCGA is a class of probability models known as marginal product models (MPMs). MPMs are formed as a product of marginal distributions on a partition of the genes and are similar to those of the compact GA (CGA) (Harik, Lobo & Goldberg, 1998) and PBIL (Baluja, 1994). Unlike the models used in CGA and PBIL, MPMs can represent probability distributions for more than one gene at a time. MPMs also facilitate a direct linkage map with each partition separating tightly linked genes. For example, the following MPM, `[1,3][2][4]`, for a four-bit problem represents that the 1st and 3rd genes are linked and 2nd and 4th genes are independent. Furthermore, the MPM consists of the following marginal probabilities $\{p(x_1 = 0, x_3 = 0)$, $p(x_1 = 0, x_3 = 1)$, $p(x_1 = 1, x_3 = 0)$, $p(x_1 = 1, x_3 = 1)$, $p(x_2 = 0)$, $p(x_2 = 1)$, $p(x_4 = 0)$, $p(x_4 = 1)\}$, where $x_i$ is the value of the ith gene.

The eCGA can be algorithmically outlined as follows:

1. Initialization: The population is usually initialized with random individuals. However, other initialization procedures can also be used.

2. Evaluate the fitness value of the individuals

3. Selection: The eCGA uses s-wise tournament selection (Goldberg, Korb & Deb, 1989; Sastry & Goldberg, 2001). However, other selection procedures can be used instead of tournament selection.

4. Build the probabilistic model: In eCGA, both the structure and the parameters of the model are searched. A greedy search heuristic is used to find an optimal model of the selected individuals in the population.

5. Create new individuals: In eCGA, new individuals are created by sampling the probabilistic model.

6. Replace the parental population with the offspring population.

7. Repeat steps 2–6 until some convergence criteria are met.

Two things need further explanation, one is the identification of MPM using MDL and the other is the creation of a new population based on MPM. The identification of MPM in every generation is formulated as a constrained optimization problem,

$$\text{Minimize} \qquad C_m + C_p \tag{7.1}$$

$$\text{Subject to} \qquad \chi^{k_i} \leq n \quad \forall i \in [1, m]. \tag{7.2}$$

Where $C_m$ is the model complexity which represents the cost of a complex model. In essence, the model complexity, $C_m$, quantifies the model representation size in terms of number of bits required to store all the marginal probabilities. Let, a given problem of size $\ell$ with alphabet cardinality $\chi$, have $m$ partitions with $k_i$ genes in the $i^{\text{th}}$ partition, such that $\sum_{i=1}^{m} k_i = \ell$. Then each partition $i$ requires $\chi^k - 1$ independent frequencies to completely define its marginal distribution. Furthermore, each frequency is of size $\log_2(n)$, where $n$ is the population size. Therefore, the model complexity (or the model representation size), $C_m$, is given by

$$C_m = \log_2(n) \sum_{i=1}^{m} \left(\chi^{k_i} - 1\right). \tag{7.3}$$

The compressed population complexity, $C_p$, represents the cost of using a simple model as against a complex one. In essence, the compressed population complexity, $C_p$, quantifies the data compression in terms of the entropy of the marginal distribution over all partitions. Therefore, $C_p$ is evaluated as

$$C_p = n \sum_{i=1}^{m} \sum_{j=1}^{\chi^{k_i}} -p_{ij} \log_2 (p_{ij}) \tag{7.4}$$

where $p_{ij}$ is the frequency of the $j^{\text{th}}$ gene sequence of the genes belonging to the $i^{\text{th}}$ partition. In other words, $p_{ij} = N_{ij}/n$, where $N_{ij}$ is the number of chromosomes in the population (after selection) possessing bit-sequence $j \in [1, \chi^{k_i}]$ [1] for $i^{\text{th}}$ partition. The constraint (Equation 7.2) arises due to finite population size.

The following greedy search heuristic is used to find an optimal or near-optimal probabilistic model:

1. Assume each variable is independent of each other. The model is a vector of probabilities.
2. Compute the model complexity and population complexity values of the current model.
3. Consider all possible $\frac{1}{2}\ell(\ell - 1)$ merges of two variables.
4. Evaluate the model and compressed population complexity values for each model structure.
5. Select the merged model with lowest combined complexity.
6. If the combined complexity of the best merged model is better than the combined complexity of the model evaluated in step 2., replace it with the best merged model and go to step 2.

[1] Note that a BB of length $k$ has $\chi^k$ possible sequences where the first sequence denotes $00\cdots0$ and the last sequence $(\chi-1)(\chi-1)\cdots(\chi-1)$

7. If the combined complexity of the best merged model is less than or equal to the combined complexity, the model cannot be improved and the model of step 2. is the probabilistic model of the current generation.

A new population is generated based on the optimal MPM as follows: A population of size $n(1 - P_c)$ where $P_c$ is the crossover probability, is filled by the best individuals in the current population. The remaining $nP_c$ individuals are generated by randomly choosing subsets from the current individuals according to the probabilities of the subsets as calculated in the probabilistic model.

## 7.2 Extended Compact Genetic Programming (eCGP)

The previous section outlined the key features of extended compact genetic algorithm. This section combines the features of eCGA (Harik, 1999) and PIPE (Sałustowicz & Schmidhuber, 1997) to create a multivariate probabilistic model building genetic programming. The proposed algorithm, called extended compact genetic programming (eCGP), adaptively identifies both the building blocks and their structure. eCGP also exchanges the building blocks from different partitions effectively, and therefore drastically improves BB mixing when compared to fixed recombination operators. The eCGP can be viewed as an extension of both eCGA and PIPE. While eCGA operates on only fixed-length bit-strings, eCGP operates on variable-size program trees. While PIPE uses only fixed-structure univariate probabilistic models to create new offspring, eCGP utilizes adaptive-structure multivariate probabilistic models.

The eCGP algorithm is similar to that of eCGA described in the previous section (Section 7.1). Similar to PIPE, the probabilistic model is built on a maximal tree. It should be noted that we use a maximal tree only to build the probabilistic model. The trees that are generated from the probabilistic model need not, and indeed are not, maximal trees. Unlike PIPE which considers each node in the tree to be independent of each other, eCGP accommodates multivariate interactions between nodes. In other words, eCGP decomposes or partitions the maximal tree into subtrees and simultaneously builds the probabilistic models for each subtree. Therefore, eCGP not only searches for the structure of the probabilistic model, but also learns the parameters of the model. The structure of the model, similar to eCGA, is MPM, and the parameters are the respective

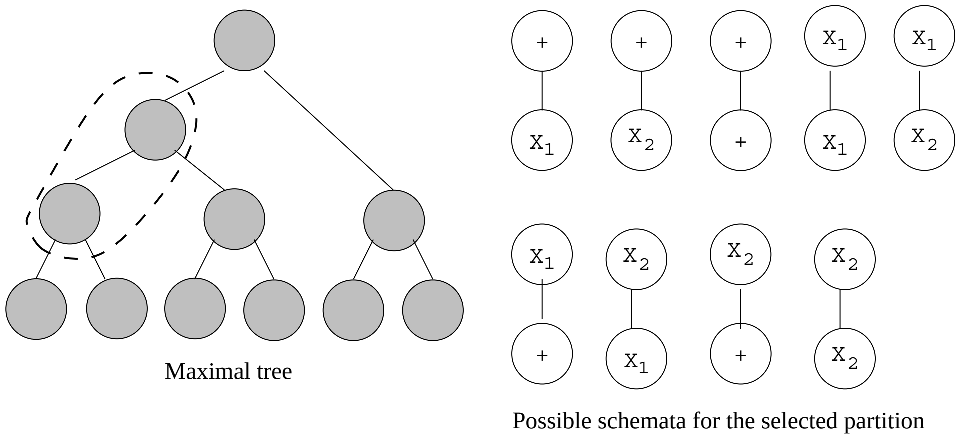


Figure 7.1: Illustration of all possible subtrees in a given partition.

frequencies of all possible subtrees in a given partition. For example, Figure 7.1 shows all possible subtrees, for a 3-variable problem (1 functional, 2 terminals) (Sastry, O'Reilly & Goldberg, 2003).

A greedy search heuristic is used to find an optimal probabilistic model and the MDL metric is used as a measure of the model quality. Finally, the "best" model is used to sample new individuals (program trees). Similar to PIPE, a maximal tree is created using the probabilistic model for each offspring, and unused portions of the tree are pruned before the evaluation of the candidate solution. We emphasize once more that even though we sample new offspring from maximal-tree probabilistic model, the final pruned tree need not be a maximal tree. This is because, during the sampling process, both functionals and terminals are candidates and if a terminal is chosen at a particular internal node all the other nodes that are below it and connected to it are discarded during the pruning process.

## 7.3 Test Problems

Our approach in designing pilot experiments for investigating the scalability of eCGP is to *design* bounding *adversarial problems* that exploit one or more dimensions of problem difficulty. Particularly, our pilot test problems should possess the following properties:

- Building-block identification should be critical for successful innovation. That is, if the BBs of the problem are not identified and exchanged, it should be impossible to attain the global solution.

- Building-block structure and interactions of the problem should be known to the researchers, but not to the problem solver (search method). Ensuring that the BB identification methods work on such problems provides assurance that they would also identify BBs of real-world problems, where the BBs are not known a priori.

- The properties such as building-block size, and problem difficulty should be tunable without significantly changing the functional.

This adversarial and systematic design method contrasts sharply with the common practices of using historical, randomly generated, or ad hoc test functions (Goldberg, 2002).

In this study we employ two classes of problems: 1. OneMax-like GP-easy problem, and 2. Deceptive trap problem. They are respectively described in the following sections.

### 7.3.1 GP-Easy Problem: `ORDER`

As mentioned in the previous chapter, `ORDER` is a GP version of the OneMax problem in GAs (Goldberg & O'Reilly, 1998; O'Reilly & Goldberg, 1998), and therefore is a GP-easy problem. The primitive set of `ORDER` consists of the primitive `JOIN` of arity two and complimentary primitive pairs $(X_i, \bar{X}_i)$, $i = 0, 1, \cdots, \ell$ of arity one. A candidate solution of the `ORDER` problem is a binary tree with `JOIN` primitive at the internal nodes and either $X_i$'s or $\bar{X}_i$'s at its leaves. The candidate solution's output is determined by parsing the program tree inorder (from left to right). The program expresses the value $X_i$ if, during the inorder parse, an $X_i$ leaf is encountered before its complement $\bar{X}_i$ and neither $X_i$ nor its complement are encountered earlier. That is, the program only expresses unique primitives during the inorder parse, and expresses the first occurrence of either $X_i$ or $\bar{X}_i$ (for each $i$), whichever comes first during the inorder parse.

For each unique $X_i$ (or $\bar{X}_i$) that a program expresses, an equal unit of fitness value is accredited. That is,

$$f_1(x_i) = \begin{cases} 1 & if\ x_i \in \{X_1, X_2, \cdots, X_\ell\} \\ 0 & otherwise \end{cases} . \tag{7.5}$$

The fitness function for `ORDER` is then defined as

$$F(\mathbf{x}) = \sum_{i=1}^{\ell} f_1\left(x_i\right) \tag{7.6}$$

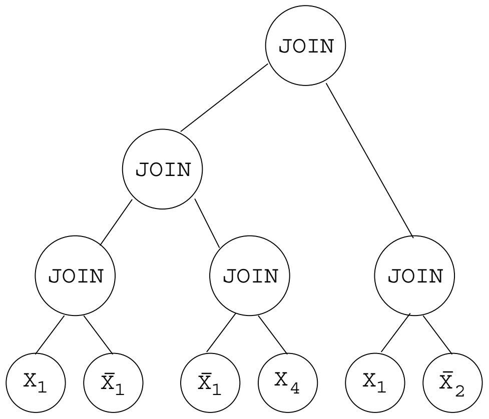


Figure 7.2: A candidate solution for a 4-primitive ORDER problem. The output of the program is $\{X_1, \bar{X}_2, X_4\}$ and its fitness is 2.

where $\mathbf{x}$ is the set of primitives output by the program. The expression for optimal solution of a $\ell$-primitive ORDER problem is $\{X_1,\ X_2,\ \cdots,\ X_\ell\}$, and its fitness value is $\ell$.

For example, consider a candidate solution for a 4-primitive ORDER problem as shown in Figure 7.2. The sequence of leaves for the tree is $\{X_1,\ \bar{X}_1,\ \bar{X}_1,\ X_4,\ X_1,\ \bar{X}_2\}$, the expression during inorder parse is $\{X_1,\ \bar{X}_2,\ X_4\}$, and its fitness is 2.

For more details, motivations, and analysis of the ORDER problem, the interested reader should refer elsewhere (Goldberg & O'Reilly, 1998; O'Reilly & Goldberg, 1998).

## 7.3.2 GP-Hard Problem: Deceptive Trap

Another test problem used in this study is the deceptive *trap* function (Deb & Goldberg, 1992; Deb & Goldberg, 1994; Goldberg, Deb & Horn, 1992) which consists of additively separable *deceptive* functions (Goldberg, 1987). Deceptive functions are designed to thwart the very mechanism of selectorecombinative search by punishing any localized hillclimbing and requiring mixing of whole building blocks at or above the order of deception. Using such *adversarially* designed functions is a stiff test—in some sense the stiffest test—of algorithm performance. The idea is that if an algorithm can beat an adversarially designed test function, it can solve other problems that are equally hard or easier than the adversary. Furthermore, if the building blocks of such deceptive functions are not identified and respected by selectorecombinative GAs, then they almost always converge to the local optimum.

The expression mechanism of the program for deceptive trap function is identical to that of

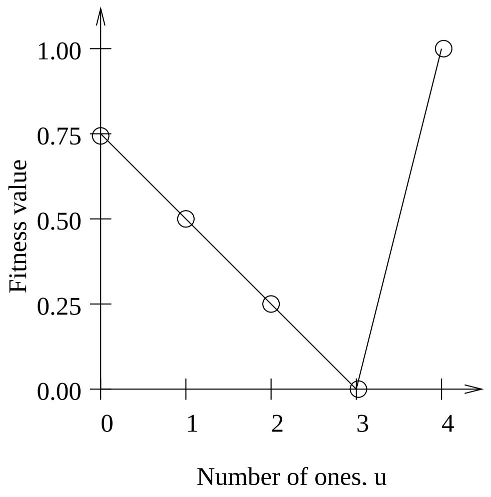


Figure 7.3: A Fully deceptive trap function with $k = 4$, and $\delta = 0.25$.

ORDER, and the difference is in the fitness evaluation procedure. Unlike ORDER, a deceptive trap function divides the expressed primitives into subgroups of $k$-primitives, and the fitness value for a $k$-primitive subgroup is defined as follows:

$$f_k\left(u(x_1, x_2, \cdots, x_k)\right) = \begin{cases} 1.0 & u = k \\ (1.0 - \delta)\left(1 - \frac{u}{k-1}\right) & u < k \end{cases}, \tag{7.7}$$

where $u$ is the unitation, or the number of primitives, $X_i$, in a portion of the tree:

$$u\left(x_1, x_2, \cdots, x_k\right) = \sum_{i=1}^{k} f_1(x_i), \tag{7.8}$$

where $x_i$ is the $i$th unique primitive, $\delta$ is the difference in the functional value between the good BB (all ones) and its deceptive attractor (all zeros). The difficulty of a trap function can be adjusted by modifying the values $k$, and $\delta$. The problem becomes more difficult as the value of $k$ is increased and that of $\delta$ is decreased. A 4-primitive deceptive trap function is illustrated in Figure 7.3.

The fitness function of the trap function is then defined as

$$\begin{aligned} F\left(\mathbf{x}\right) &= f_k\left(u(x_1, x_2, \cdots, x_k)\right) + f_k\left(u(x_{k+1}, \cdots, x_{2k})\right) + \\ &\quad \cdots + f_k\left(u(x_{(m-1)k+1}, \cdots, x_{mk})\right), \end{aligned} \tag{7.9}$$

where $F$ is the fitness function, $\mathbf{x}$ is the expressed primitives (expressed leaf nodes of the tree), $m$ is the number of BBs, and $\ell = mk$.

The important feature of additively separable trap functions is that if the good BB (all ones) in any particular partition is not identified, then the GA tends to converge to the deceptive attractor (all zeros) in that partition. Therefore, BB identification and mixing is critical to innovation success. Furthermore, notice that the problems are of bounded difficulty $k < \ell$. If the trap was of length $\ell$, nothing could work better than enumeration or random search without replacement. However, given that the difficulty is bounded, a GA that identifies BBs and mixes them well has the opportunity to solve the problem in polynomial time (Goldberg, 2002).

## 7.4 Results and Discussion

This section compares the performance of eCGP and a simple GP for both `ORDER` and 3-primitive deceptive trap function (k = 3). Specifically, we investigate how eCGP and simple GP scale-up with the problem size (number of terminals and functionals) for both `ORDER`, which is a GP-easy problem, and deceptive trap function which is a boundedly GP-hard problem.

A simple GP consists of a tree-swap crossover and s-wise tournament selection. Crossover probability of 1.0 and tournament size of 2 was used for both eCGP and simple GP. Mutation is not considered in this study. The initial population for both eCGP and simple GP was generated using the ramped half-and-half method and maximum tree depth of $2\log_2 \ell$ was used. A GP run was stopped when the fitness variance of the population was less than $\ell^{-2}$ for `ORDER` and less than $m^{-2}$ for the deceptive trap function. The number of function evaluations required in order to obtain a population with at least $m-1$ correct building blocks are empirically computed. All the results presented in this section are averaged over 50 independent runs.

Figure 7.4 compares the scalability of eCGP and simple GP for the `ORDER`. The figure plots the number of function evaluations against problem size. The results indicate that even though eCGP requires a slightly larger number of function evaluations, the slope of the curve is slightly less than that for simple GP. This result is consistent with the performance of competent GAs on GA-easy problems (Pelikan, 2005; Pelikan, Goldberg & Cantú-Paz, 2000) and is caused by the spurious dependencies that are introduced by the selection mechanism and fixed population size

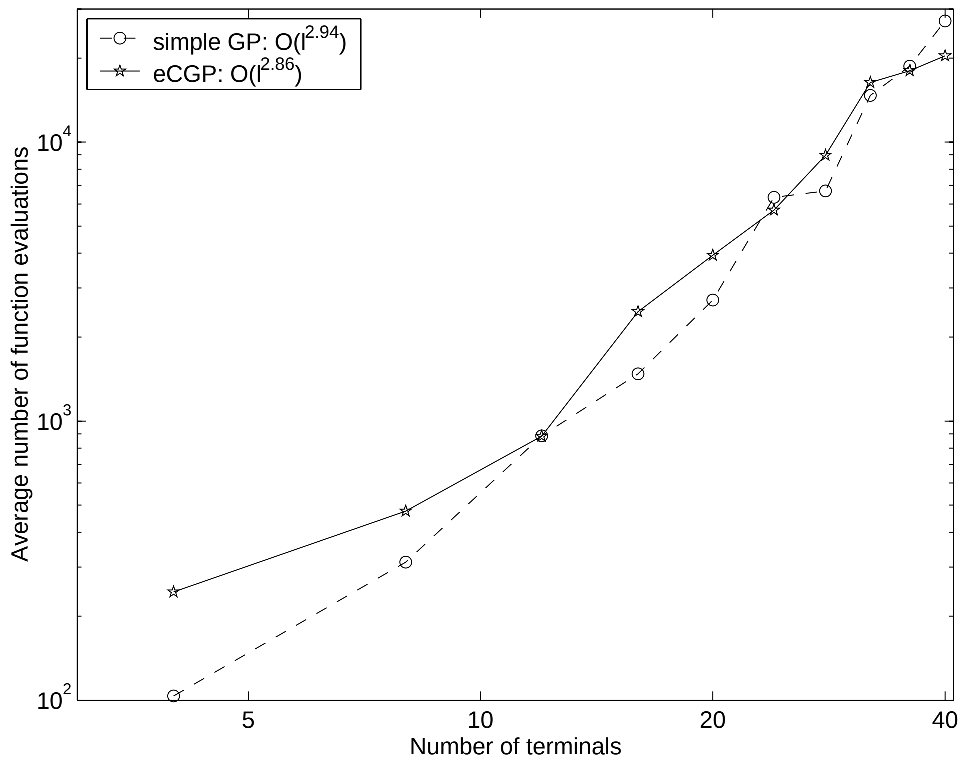


Figure 7.4: Comparison of eCGP and simple GP for `ORDER`. The figure plots the average number of function evaluations as a function of problem size. The results are averaged over 50 independent runs.

(Pelikan, Goldberg & Sastry, 2001; Pelikan, Sastry & Goldberg, 2003; Yu et al., 2007). However, as the problem size increases, simple GP will suffer from mixing effects (Sastry & Goldberg, 2003b) and the slope of the number of function evaluations will increase.

Figure 7.5 compares the scalability of the eCGP and the simple GP for the deceptive trap problem. The building block size, $k$ is set to 3 and the signal difference, $d$ is set to 0.25. The figure plots the number of function evaluations as a function of problem size. The results clearly indicate the effectiveness of linkage-adaptive recombination operator of eCGP over the fixed recombination operator of simple GP. The results indicate that eCGP scales-up polynomially (cubic) with problem size. Furthermore, the scalability of eCGP is about the same for both `ORDER` and deceptive trap function. Finally, the number of function evaluations required for simple GP appears to be very high for problems with more than 24 terminals, and therefore the problems become intractable quickly. For example, for deceptive trap problem with 30 terminals, the simple GP was not able to converge to the optimal solution even after $5 \times 10^8$ evaluations. However, for the problems sizes

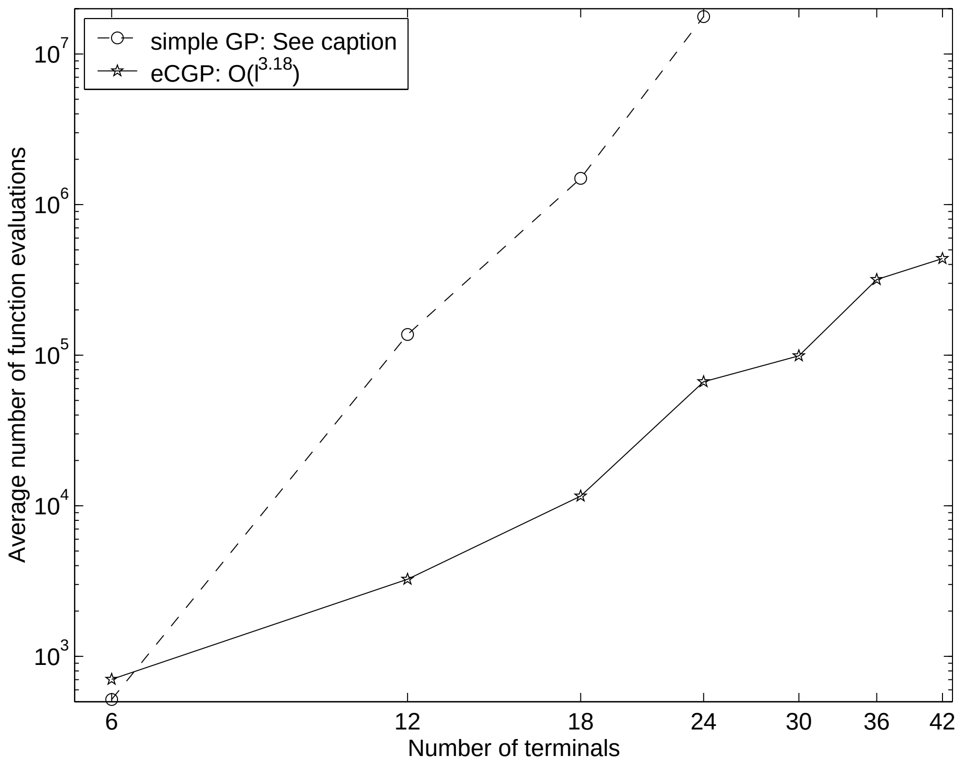


Figure 7.5: Comparison of eCGP and simple GP for the deceptive trap function (k = 3). The figure plots the average number of function evaluations as a function of problem size. The results are averaged over 50 independent runs. The simple GP was not able to solve problems beyond 24 primitives. For example, for a deceptive trap function with 30 primitive, simple GP did not converge to the optimal solution even after $5 \times 10^8$ function evaluations. For the problem sizes that we were able to obtain the optimal solution via a simple GP, the scale-up is $O(\ell^{8.91})$. But its failure to converge to the optimal solution for larger problems suggests that the scale-up is indeed much larger than $O(\ell^{8.91})$.

that simple GP was able to converge to the optimal solutions, the scale-up with problem size is $O(\ell^9)$.

We recognize that both `ORDER` and the deceptive trap problems have non-overlapping building blocks. That is, an expressed primitive is a part of only one building block. As mentioned earlier, eCGP partitions the search problem into linkage groups and therefore is very effective in identifying non-overlapping building blocks. Even though the success of eCGP in identifying, propagating, and exchanging of non-overlapping building blocks, enables it to solve a broad class of additively decomposable problems, we acknowledge that even broader class of problems can be tacked if complex interactions such as overlapping BBs and hierarchically interacting BBs can also be identified. The successful adaptation of eCGA into genetic programming domain, makes the adaptation of other powerful competent GAs such as the Bayesian optimization algorithm (BOA) (Pelikan, Goldberg & Cantú-Paz, 2000) and the hierarchical Bayesian optimization algorithm (hBOA) (Pelikan & Goldberg, 2001; Pelikan, 2005) into GP domain straightforward and promising.

## 7.5 Summary

A competent GP, called extended compact GP (eCGP), was introduced in this chapter. The extended compact GP uses and combines the ideas from extended compact genetic algorithm (eCGA) and probabilistic incremental program evolution (PIPE). The proposed algorithm adaptively identifies, propagates, and exchanges important subsolutions of a search problem. The subsolutions are identified by building a multivariate probabilistic model of promising solutions and the subsolutions are exchanged by creating new offspring by sampling the probabilistic model. The results of eCGP were compared to those of a simple GP for two test problems: 1. `ORDER`: A GP-easy problem, and 2. Deceptive trap: A GP-hard problem. The results show that eCGP scales up polynomially with the problem size (number of terminals) for both GP-easy and GP-hard problems. On the other hand, as expected, a simple GP scales-up polynomially for a GP-easy problem, and exponentially for GP-hard problems.

The chapter presented initial results of one of the first attempts at developing scalable GP designs and researchers have started to pay increasing attention to develop competent GP (Looks, 2006; Shan et al., 2006). Much work still remains to be done, some of which is listed as follows:

- The eCGP can only identify non-overlapping BBs, and attempts to develop competent operators that not only identify complex BB structures, but also tackle BBs with hierarchical interactions, and some studies are currently underway (Looks, 2006).

- Two adversarially designed test problems were considered in this study, and more tests have to be conducted on different class of problems and the scalability of eCGP has to be analyzed.

- We need to investigate the convergence time and population-sizing requirement of eCGP, both analytically through facetwise and dimensionless models, and also empirically for different classes of problems.

- This study considered problems with only one functional and many more terminals. However, many GP problems consider many functionals and only a few terminals (assuming the ephemeral random constant is one type of terminal). The performance of eCGP on such a class of problems needs to be investigated.

The results presented in this chapter demonstrates that competent GP can be designed in a principled manner to solve both GP-easy and GP-hard problems requiring polynomial (oftentimes cubical) number of function evaluations. However, an important and open question still remains as to whether there are any GP-hard problems that thwart the mechanism of selectorecombinative GP. While it is clear that in optimization hard problems exist that thwart the mechanisms of selectorecombinative process, it is not very clear whether such hard problems exist in GP domain. For example, the success stories of standard genetic programming with fixed recombination operators successfully solving complex real-world problems seem to suggest that many complex real-world domains might not be GP-hard.

This open question of existence of GP-hard problems is also related to a broader issue of problem difficulty in system identification. While addressing optimization problems, it is clear that there is premium on scalable recombination and mutation operators that automatically identify and exploit good substructures (neighborhoods). However, when performing system identification, say symbolic regression via GP or model building in estimation of distribution algorithms, we seem to get away with simple crossover methods or even local search methods.

Another related question of critical importance—especially for scalable genetic-programming design—is, if GP-hard problems do exist, then what are the principal dimensions of GP problem difficulty that bound real-world GP-hard problems? Answer to this question will help us design *adversarial* test problems that exploit one or more dimensions of problem difficulty and test GP designs on the boundary of their *design envelope*. The answer to this question will also help us understand what makes a problem hard for estimation of distribution algorithms such as the Bayesian optimization algorithm where the probabilistic model building relies on a hill-climber.

Nevertheless, this chapter and subsequent work on scalable GP by other researchers clearly demonstrate that, as with GAs, it is possible to design scalable GP designs in principled manner. Moreover, this study also demonstrates that competent GP designs can be advantageous over standard GP with fixed recombination operators.

# Chapter 8

# Scalability of Multiobjective Genetic Algorithms

Multiobjective genetic algorithms (MOGAs) have received increased attention in the recent years and have been applied with significant success to real-world problems (Coello Coello, Van Veldhuizen & Lamont, 2002; Deb, 2001). However, studies on the theory and analysis of MOGAs have been limited in part because of the complexity of both the algorithms and the problems. For example, some aspects of problem difficulty and algorithm scalability have been studied (Chen, 2004; Deb, 1999).

Recently, there has been a growing interest in extending estimation of distribution algorithms (EDAs) (Larrañaga & Lozano, 2002; Pelikan, Goldberg & Lobo, 2002; Pelikan, Sastry & Cantú-Paz, 2006)—a class of *competent* genetic algorithms (Goldberg, 1999a) that replace traditional variation operators of GAs with probabilistic model building of promising solutions and sampling the model to generate new offspring—to solve multiobjective search problems quickly, reliably, and accurately. Such multiobjective EDAs (MOEDAs) (Ahn, 2005; Bosman & Thierens, 2002b; Khan, Goldberg & Pelikan, 2002; Ocenasek, 2002) typically combine the model-building and sampling procedures of EDAs with the selection procedure of MOGAs such as the non-dominated sorting GA (NSGA-II) (Deb et al., 2002), and a niching method such as sharing or crowding in the objective space. MOEDAs have been shown to outperform traditional MOGAs in efficiently searching and maintaining Pareto-optimal solutions on boundedly-difficult problems.

However, the scalability of the population size and the number of function evaluations required by EDAs as a function of problem size and the number of Pareto-optimal solutions has been largely ignored. This is the case even though one of the primary motives for designing MOEDAs is to carry over the polynomial (oftentimes sub-quadratic) scalability of EDAs to boundedly-difficult multiobjective search problems. However, the usual scalability approach used for single-objective GAs with one or few global solutions—where, we investigate the minimum number of function evaluations to get high-quality solutions quickly, reliably, and accurately—does not work for multiobjective problems, and it is easy to get into combinatorial difficulty. Even with two objectives, additively decomposable problems have exponentially many Pareto-optimal solutions. This massive multimodality introduces a fundamental limitation on the scalability of MOGAs in general, and MOEDAs in particular. This chapter demonstrates that even if the substructures (or linkages) are correctly identified, the combinatorial explosion of the number of Pareto-optimal solutions can overwhelm the niching capability and as expected lead to exponential scalability. This fundamental nature of multiobjective additively decomposable problems introduces a limit on the number of building blocks (or substructures) that can differ between the multiple objectives. That is, MOGAs scale polynomially (subquadratically) only if the multiple objectives share common building blocks and have a limited number of building blocks that are different. Facetwise models are used to predict the limit in the number of competing substructures between multiple objectives.

The chapter is organized as follows. The next section provides a brief background on the motivation for the chapter, followed by a brief description of multiobjective extended compact genetic algorithm (meCGA). The details on the test problems and experimental methodologies are described in the subsequent sections. Section 8.6 demonstrates how massive multimodality of the search problem can overwhelm the niching mechanism and lead to exponential scale-up of MOEDAs. In section 8.7, using facetwise models of population-sizing for EDAs and niching methods, a model to predict the limit on the growth rate of the number of competing substructures between two objectives that can lead to polynomial scalability is proposed. Finally, key conclusions of the study are summarized.

## 8.1 Related Work

Over the last few decades, there has been a growing interest in extending estimation of distribution algorithms to solve multiobjective search problems. Similar to single-objective EDAs (Larrañaga & Lozano, 2002; Pelikan, 2005; Pelikan, Goldberg & Lobo, 2002), multiobjective EDAs replace the variation operators of MOGAs with the probabilistic model building of promising solutions and sampling the model to generate new offspring. Recently, several MOEDAs have been proposed (Ahn, 2005; Bosman & Thierens, 2002a; Bosman & Thierens, 2002b; Khan, 2002; Khan, Goldberg

& Pelikan, 2002; Laumanns & Ocenasek, 2002; Ocenasek, 2002) which have combined variants of the Bayesian optimization algorithm (BOA) (Pelikan, Goldberg & Cantú-Paz, 2000) and iterated density estimation evolutionary algorithm (IDEA) (Bosman & Thierens, 1999; Bosman & Thierens, 2000) with the selection and replacement procedures of MOGAs (Coello Coello, Van Veldhuizen & Lamont, 2002; Deb, 2001). Even though MOEDAs have been shown to outperform their MOGA counterparts on different test problems, none of the studies have systematically analyzed the scalability of MOEDAs.

Therefore, the original purpose of this study was to systematically analyze the scalability of MOEDAs on a class of boundedly-difficult additively decomposable problems. We followed a methodology analogous to that used to test the scalability of single-objective EDAs with $\mathcal{O}(1)$ global solutions. In particular, the minimum number of function evaluations required to obtain and maintain all the Pareto-optimal solutions quickly, reliably, and accurately was investigated. The scalability of multiobjective Bayesian optimization algorithm (Khan, 2002) and multiobjective extended compact GA was tested on several bi-objective test problems and observed that MOEDAs scale exponentially with problem size. This is the case even when the EDAs successfully solve each of the objectives alone, requiring only sub-quadratic number of function evaluations.

Further analysis of the scalability results and the test problems, indicated a fundamental fact of such building-block-wise difficult problems: Exponential growth in the number of Pareto-optimal solutions. When considering each objective in isolation, there is only one global solution, but when considering the two objectives in multiobjective optimization, the total number of global (Pareto-optimal) solutions grow exponentially ($\mathcal{O}(2^m)$). Since we need at least one individual to maintain a Pareto-optimal solution, we need exponentially many individuals to maintain all the Pareto-optimal solutions in the population.

The following section briefly describes multiobjective extended compact genetic algorithm (meCGA), which is used as a representative algorithm of MOEDAs. The multiobjective eCGA is chosen not only because of its simplicity and ease of visualizing the probabilistic models, but also because it bounds the scalability of other binary EDAs and competent GAs such as BOA (Pelikan, Sastry & Goldberg, 2003; Sastry & Goldberg, 2004a; Yu et al., 2007).

## 8.2 Multiobjective Extended Compact Genetic Algorithm (meCGA)

In this study, multiobjective extended compact GA is used and its scalability on a class of boundedly-difficult problems is tested. The multiobjective extended compact genetic algorithm (meCGA) is similar to mBOA (Khan, Goldberg & Pelikan, 2002), except that the model building and sampling procedure of BOA is replaced with those of extended compact GA (eCGA) (Harik, 1999). The meCGA is used in this study in part because the simplicity of the probabilistic model and its direct mapping to linkage groups makes it amenable to systematic analysis. The typical steps of meCGA can be outlined as follows:

1. *Initialization:* The population is usually initialized with random individuals. However, other initialization procedures can also be used in a straightforward manner.

2. *Evaluation:* The fitness or the quality-measure of the individuals are computed.

3. *Selection:* As in mBOA, the selection procedure of NSGA-II (Deb et al., 2002) is used. That is, the non-dominated sorting ranks are computed first, followed by the crowding distance for all the individuals in the population. As in NSGA-II the individual comparison operator is used to *bias* the generation of new individuals.

4. *Probabilistic model estimation:* Unlike traditional GAs, however, EDAs assume a particular probabilistic model of the data, or a *class* of allowable models. A *class-selection metric* and a *class-search mechanism* is used to search for an optimum probabilistic model that represents the selected individuals.

   **Model representation:** The probability distribution used in eCGA is a class of probability models known as marginal product models (MPMs). MPMs partition genes into mutually independent groups and specifies marginal probabilities for each linkage group.

   **Class-Selection metric:** To distinguish between better model instances from worse ones, eCGA uses a minimum description length (MDL) metric (Rissanen, 1978). The key concept behind MDL models is that all things being equal, simpler models are better than more complex ones. The MDL metric used in eCGA is a sum of two components:

- **Model complexity** which quantifies the model representation size in terms of number of bits required to store all the marginal probabilities:

$$C_m = \log_2(n) \sum_{i=1}^{m} \left(2^{k_i} - 1\right). \tag{8.1}$$

  where $n$ is the population size, $m$ is the number of linkage groups, $k_i$ is the size of the $i^{\text{th}}$ group.

- **Compressed population complexity**, which quantifies the data compression in terms of the entropy of the marginal distribution over all partitions.

$$C_p = n \sum_{i=1}^{m} \sum_{j=1}^{2^{k_i}} -p_{ij} \log_2 \left(p_{ij}\right), \tag{8.2}$$

  where $p_{ij}$ is the frequency of the $j^{\text{th}}$ gene sequence of the genes belonging to the $i^{\text{th}}$ partition.

**Class-Search method:** In eCGA, both the structure and the parameters of the model are searched and optimized to best fit the data. While the probabilities are learnt based on the variable instantiations in the population of selected individuals, a greedy-search heuristic is used to find an optimal or near-optimal probabilistic model. The search method starts by treating each decision variable as independent. The probabilistic model in this case is a vector of probabilities, representing the proportion of individuals among the selected individuals having a value '1' (or alternatively '0') for each variable. The model-search method continues by merging two partitions that yields greatest improvement in the model-metric score. The subset merges are continued until no more improvement in the metric value is possible.

5. *Offspring creation:* New individuals are created by sampling the probabilistic model. The offspring population are generated by randomly generating subsets from the current individuals according to the probabilities of the subsets as calculated in the probabilistic model.

6. *Replacement:* Two replacement techniques are used in this study: (1) Restricted tournament replacement (RTS) (Harik, 1995) in which offspring replaces the closest individual among $w$ individuals randomly selected from the parent population, only if the offspring is better than

the closest parent. (2) Elitist replacement used in NSGA-II, in which the parent and offspring population are combined. The domination ranks and crowding distances are computed on the combined population. Individuals with increasing ranks are gradually added starting from those with the lowest rank into the new population till its size reaches $n$. However, if it is not possible to add all the solutions belonging to a particular rank without increasing the population size to greater than $n$, then individuals with greater crowding distances are preferred.

7. Repeat steps 2–6 until one or more termination criteria are met.

## 8.3 Test Problem

Our approach in verifying the performance of MOEDA is to consider bounding *adversarial problems* that exploit one or more dimensions of problem difficulty (Goldberg, 2002). Particularly, we are interested in problems where building-block identification is critical for the GA success. Additionally, the problem solver (meCGA) should not have any knowledge of the building-block structure of the test problem, but should be known to researchers for verification purposes.

One such class of problems is the m-k deceptive *trap* problem, which consists of additively separable *deceptive* functions (Ackley, 1987; Deb & Goldberg, 1992; Goldberg, 1987). Deceptive functions are designed to thwart the very mechanism of selectorecombinative search by punishing any localized hillclimbing and requiring mixing of whole building blocks at or above the order of deception.

In this study, we use a class of test problems with two objectives: (1) m-k deceptive trap, and (2) m-k deceptive inverse trap. String positions are first divided into disjoint subsets or partitions of $k$ bits each. The $k$-bit trap and inverse trap are defined as follows:

$$trap_k(u) = \begin{cases} 1 & \text{if } u = k \\ (1-d)\left[1 - \frac{u}{k-1}\right] & \text{otherwise} \end{cases}, \tag{8.3}$$

$$invtrap_k(u) = \begin{cases} 1 & \text{if } u = 0 \\ (1-d)\left[\frac{u-1}{k-1}\right] & \text{otherwise} \end{cases}, \tag{8.4}$$

where $u$ is the number of 1s in the input string of $k$ bits, and $d$ is the signal difference. Here, we use $k = 3$, 4, and 5, and $d = 0.9$, 0.75, and 0.8 respectively.

The m-k trap and inverse trap functions have conflicting objectives. Any solution that sets the bits in each partition either to 0s or 1s is Pareto optimal and thus there are a total of $2^m$ solutions in the Pareto-optimal front with $m + 1$ distinct points in the objective space.

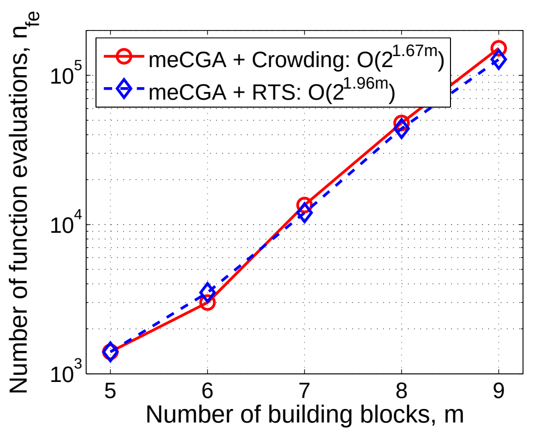
## 8.4 Experimental Methodology

The scalability of MOGAs are measured as the minimum number of function evaluations required to maintain at least one copy of all the Pareto-optimal solutions for problems of different sizes. For each problem type, problem size, and algorithm, a bisection method was used to determine a minimum population size to allocate at least one individual to each representative solution in the Pareto front. As mentioned earlier, for the test problems we consider in this study, for an $\ell$-bit problem—where $\ell = m \cdot k$—there are $2^m$ Pareto-optimal solutions with $m + 1$ distinct objective value pairs. In this study, we investigate the population size required to (1) find at least one copy of all the $2^m$ Pareto-optimal solution, and (2) find at least one copy of the $m + 1$ distinct points in the Pareto-optimal front. That is, we consider Pareto-optimal solutions with the same values of both objectives to be equivalent.

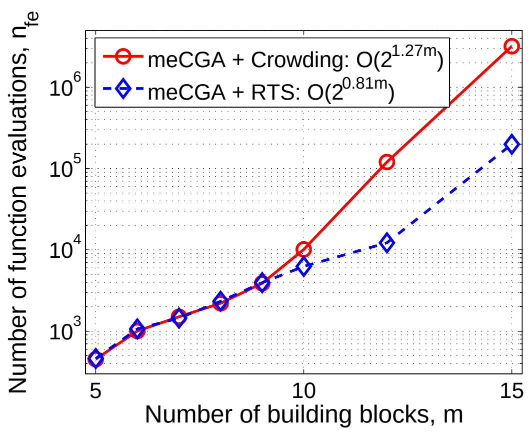
The probability of maintaining at least one copy of all the representative Pareto-optimal solutions at a given population size is computed by averaging 10–30 independent MOGA runs. The minimum population size required to maintain at least one copy of all the representative solutions in the Pareto front are averaged over 10-30 independent bisection runs. Therefore, the results for each problem type, problem size, and algorithm correspond to 100–900 independent GA runs. The number of generations for meCGA was bounded by $5\ell$, where $\ell$ is the string length.

## 8.5 Scalability of meCGA

Scalability tests were conducted on m-k deceptive trap and inverse trap functions for $k = 3$, 4, and 5, however, for brevity, only results for $k = 3$ are shown in this chapter. Moreover, the results for other values of $k$ are qualitatively similar and those for $k = 3$ are representative of the behavior of



(a)



(b)

Figure 8.1: Scalability of meCGA with crowding and with RTS for the m-3 deceptive trap and inverse trap with the problem size. Here, we plot the minimum number of function evaluations required to search and maintain at least one copy of (a) all the $2^m$ solutions in the Pareto-optimal front, and (b) only the $m+1$ solutions in the Pareto-optimal front with different objective-value pairs. Here, we treat the genotypically (and phenotypically) different Pareto-optimal solution with same values in both objectives to be equivalent.

meCGA.

Figure 8.1(a), shows the scalability of meCGA with the problem size for m-3 deceptive trap and inverse trap problem. That is, the minimum number of function evaluations required to allocate at least one copy of all the solutions in the Pareto-optimal front is plotted as a function of problem size. As shown in the figure, all algorithms scale exponentially. The scale-up does not improve even if we restricted the requirement to finding only those $m+1$ Pareto-optimal solutions with different objective-value pairs as shown in Figure 8.1(b). That is, even if we consider genotypically (and phenotypically) distinct solutions that have the same value in both objectives to be equivalent, meCGA scales exponentially. This is despite the linkage information being identified correctly by meCGA. Additionally, the scalability does not improve if the niching or speciation is performed in the objective space (as in NSGA-II) or in the variable space (as in restricted tournament selection). To reiterate, meCGA scales up exponentially on the trap and inverse trap functions, in spite of accurate identification of the building blocks. Furthermore, when considering the single-objective case of m-k trap functions or inverse trap functions eCGA scales polynomially with problem size.

## 8.6 Exponential Growth of Pareto-Optimal Solutions

As mentioned in the previous section, the exponential scale-up is not due to incorrect linkage identification and mixing (Goldberg, Thierens & Deb, 1993; Thierens, 1999; Thierens & Goldberg, 1993), but because the niching mechanism gets quickly overwhelmed due to the exponential growth in the number of Pareto-optimal solutions. Furthermore, the distribution of the $2^m$ solutions in the Pareto-optimal front is not uniform. There are exponentially as many solutions in the middle of the front than at the edges (see Table 8.1). That is, there is only one solution—a binary string with all 0s and all 1s—at each extreme of the Pareto-optimal front. In contrast, there are $\begin{pmatrix} m \\ m/2 \end{pmatrix} \approx \mathcal{O}\left(e^m\right)$ genotypically different solutions in the middle of the Pareto-optimal front with same values in both objectives.

Table 8.1: Distribution of genotypically and phenotypically different solutions in the Pareto-optimal front with same values in both objectives. $n_{1,BBs}$ refers to the number of $k$-bit partitions (substructures) with 1s and $n_{0,BBs}$ is the number of $k$-bit partitions with 0s.

| $n_{1,BBs}$ | $0$ | $1$ | $\cdots$ | $i$ | $\cdots$ | $m$ |
|---|---|---|---|---|---|---|
| $n_{0,BBs}$ | $m$ | $m-1$ | $\cdots$ | $m-i$ | $\cdots$ | $0$ |
| **# solutions** | 1 | $m$ | $\cdots$ | $\begin{pmatrix} m \\ i \end{pmatrix}$ | $\cdots$ | 1 |

This highly non-linear distribution of solutions in the Pareto-front has two effects on the niching mechanisms used in MOGAs in general, and MOEDAs in particular:

- Since the extremes of the Pareto-optimal front (maximizing most partitions or substructures with respect to one particular objective) have exponentially smaller representatives than in the middle, it takes exponentially longer time, or exponentially larger population size (Goldberg, 2002; Thierens, 1999) to search and maintain the solutions at the extremes of the Pareto-optimal front. When the population size is fixed, the probability of maintaining a solution in the middle of the Pareto-optimal front is higher than doing so in extremes of the front, as shown in Figure 8.2.

- Since there are multiple points that are genotypically and phenotypically different, but lie on the same point on the Pareto-optimal front (solutions have same values in both objectives), some of them vanish over time due to drift. The drift affects both the solutions in the middle

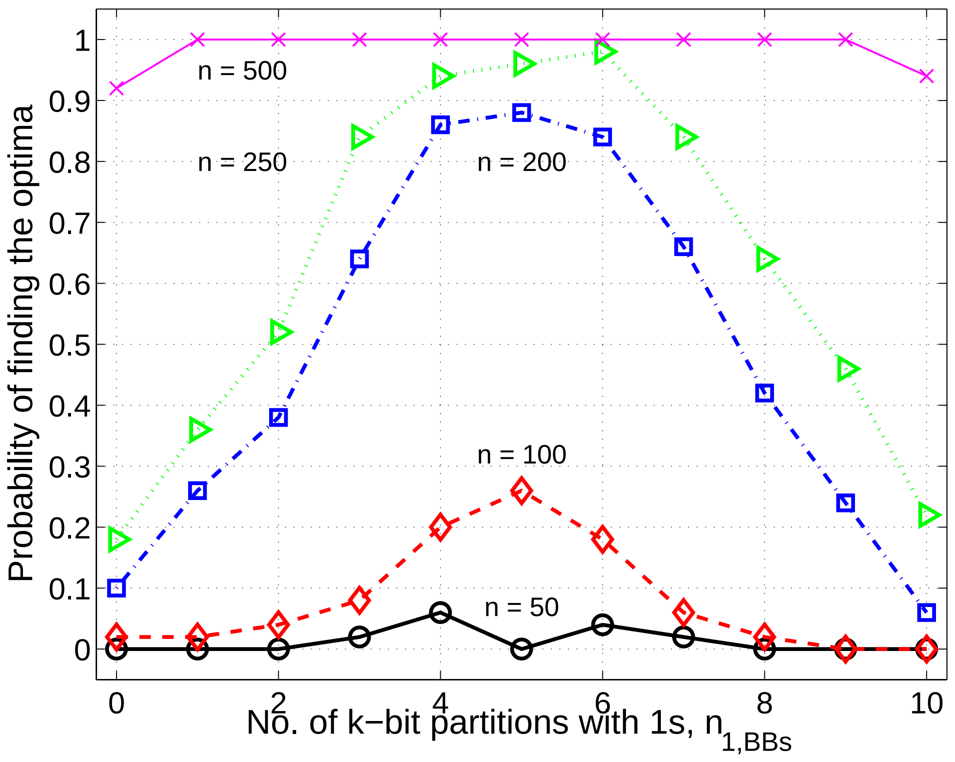


Figure 8.2: Probability of finding and maintaining different solutions on the Pareto-optimal for the 10-3 deceptive trap and inverse trap problem as a function of population size. The results are for meCGA with elitist crowding and the results are averaged over 100 independent runs.

and the extremes of Pareto front.

### 8.6.1 Overwhelming the Niching Method

To illustrate how additively decomposable problems with conflicting objectives can overwhelm the niching mechanism used in MOGAs—irrespective of linkage adaptation capabilities of the evolutionary algorithm—and lead to exponential scalability, we consider a problem where linkage learning is not required. Specifically, we consider the OneMax-ZeroMax problem which is similar to bi-criteria OneMax problem of Chen (2004). In OneMax-ZeroMax problem, the task is to maximize two objectives, one of which is the sum of all the bits with value 1, and the other is the sum of all the bits with value 0:

$$f_{\text{OneMax}}(X) = \sum_{i=1}^{\ell} x_i, \tag{8.5}$$

$$f_{\mathrm{ZeroMax}}(X) \quad = \quad \sum_{i=1}^{\ell}(1 - x_i), \tag{8.6}$$

where $\ell$ is the problem size, and $x_i$ is the value of the $i^{\mathrm{th}}$ bit of a candidate solution $X$.

The OneMax-ZeroMax problem is specifically chosen to isolate the effect of linkage identification from those of the niching methods on the scalability of the MOGAs. Unlike the m-k deceptive trap and inverse trap function, linkage identification is not necessary for the OneMax-ZeroMax problem. Furthermore, both OneMax and ZeroMax problems are GA-easy problems which a simple selectorecombinative GA with uniform crossover and tournament selection can solve in linear time (Harik et al., 1999; Mühlenbein & Schlierkamp-Voosen, 1993).

However, in a multiobjective scenario of the OneMax-ZeroMax, the *entire* search space ($2^\ell$) belongs to the Pareto-optimal front with $\ell + 1$ distinct objective-value pairs. Therefore, in order to maintain all the Pareto-optimal solutions, we would require $\mathcal{O}(2^\ell)$ population size. From the details presented in the previous sections, even if we relax the scalability requirement to finding at least one copy of all $\ell + 1$ distinct Pareto-optimal solutions, the exponential requirement in the population size (and consequently the number of function evaluations) is not relaxed. Therefore, as expected, the MOGAs, particularly multiobjective univariate marginal distribution algorithm (mUMDA) and NSGA-II, scale exponentially in solving the OneMax-ZeroMax problem as shown in Figure 8.3. The mUMDA algorithm used in this study is identical to meCGA where the probabilistic model is a univariate model where each variable/bit is considered independent to each other (Mühlenbein & Paaß, 1996), which is the ideal model for the OneMax-ZeroMax problems.

The results clearly indicate how the niching methods—both those that work in parameter space (RTS) and those that work in objective space (Crowding)—get overwhelmed due to exponentially large number of solutions in the Pareto-optimal front. That is, since the number of Pareto-optimal solutions grow exponentially, in order to maintain at least one copy of all the global solutions we would require exponentially large population sizes. Additionally, the results also show that even if the requirement is relaxed by treating all the different points that lie on the same point in the Pareto-optimal front to be equivalent, the scalability does not improve. Finally, the results suggest that in decomposable problems, if all or majority of the substructures compete in the two objectives, then the niching method fails to maintain good coverage, leading to exponential scale-up.

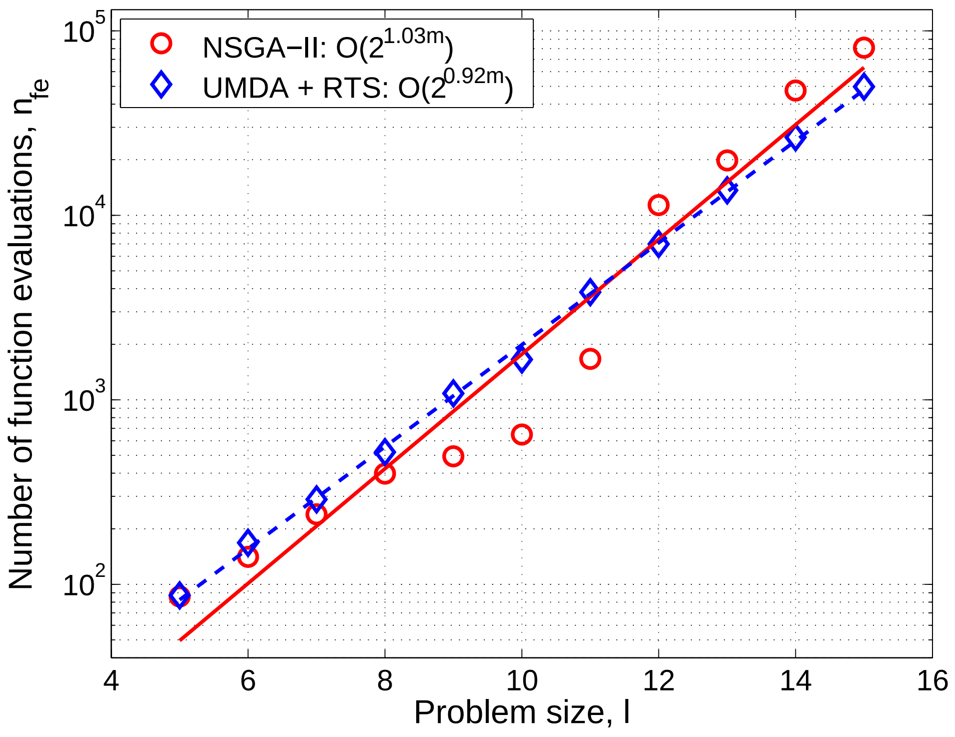


Figure 8.3: Scalability of NSGA-II and mUMDA on the OneMax-ZeroMax problem in terms of minimum number of function evaluations required to maintain at least one copy of each of the $\ell + 1$ distinct solutions in the Pareto-optimal front. Both algorithms with two different niching methods scale exponentially with the problem size.

This combinatorial growth in the number of Pareto-optimal solutions is a deal breaker for tractable solutions and the following scenarios can be envisioned to address this issue:

- Acknowledge that with practical population sizes, some of the Pareto-optimal solutions cannot be covered (especially at the edges of the Pareto front), and do the best we can. In such a scenario, an MOGA with linkage-adaptation capabilities outperforms MOGAs with fixed recombination operators (Pelikan, Sastry & Goldberg, 2005).

- Size the population appropriately in accordance with the exponential growth in the Pareto-optimal solutions. Here, Mahfoud's population-sizing model for niching methods (Mahfoud, 1994), which predicts that the population size grows linearly with the number of global solutions, is applicable.

- Understand the fundamental limits on the growth in the number of Pareto-optimal solutions and thereby the type of search problems that permit tractable search. If we want MOEDAs to scale polynomially on additively-decomposable problems, the number of Pareto-optimal solutions have to be limited. In other words, if the number of substructures that are different (or compete) between multiple objectives is limited, then MOEDAs can scale polynomially. That is, there is an imposed limit on the type of additively decomposable problems MOEDAs can solve in polynomial time. We use facetwise models to predict this limit on the growth of competing substructures between multiple objectives (and consequently, the number of Pareto-optimal solutions) in the next section.

## 8.7 Limit on the Growth of Competing Substructures

The results in the previous two sections clearly indicate that MOEDAs with either RTS or crowding mechanism of NSGA-II scale-up exponentially with problem size on boundedly-difficult additively-separable multiobjective problems. We also demonstrated that the exponential scalability is due to the niching method being overwhelmed because of exponentially large number of solutions in the Pareto-optimal front. The exponential growth in the number of Pareto-optimal solutions imposes a fundamental limitation on the type of problems which MOEDAs can solve in polynomial time. That is, MOEDAs can solve only those multiobjective problems in polynomial time that have limited growth in the number of Pareto-optimal solutions.

One way to restrict the growth of the Pareto-optimal solutions is to control the number of substructure (building blocks) that compete between the two objectives, $m_d$. That is, for a problem with $m$ substructures, the two objectives differ in only $m_d$ substructures and share the same $m-m_d$ substructures. For example, consider $4-bit$ OneMax-ZeroMax problem, $m_d = 2$, and without loss of generality, that the last two building blocks differ between the two objectives. Then there are only four Pareto-optimal solutions (as opposed to $2^4 = 16$): `1100`, `1101`, `1110`, and `1111`. Since the total number of Pareto-optimal solutions, $n_{opt} = 2^{m_d}$, by controlling the number of competing substructures, we implicitly control the total number of Pareto-optimal solutions.

The growth-rate of the competing substructures should be such that the effect of model accuracy, decision making, and building-block supply on the population sizing is dominant over the effect of niching on the population size. The effect of model accuracy, decision making and substructure supply on the population sizing of eCGA is given by (Pelikan, Sastry & Goldberg, 2003; Sastry &

Goldberg, 2004a):

$$n_{\text{eCGA}} \propto c_1 \cdot 2^k \cdot m \log m, \tag{8.7}$$

where $c_1$ is a constant. The effect of niching method on the population-sizing of GAs was modeled by Mahfoud (Mahfoud, 1994) and is reproduced below:

$$n_{\text{niching}} \propto \frac{\log\left[\left(1-\gamma^{1/t}\right)/n_{opt}\right]}{\log\left[\left(n_{opt}-1\right)/n_{opt}\right]} \approx c_2 \cdot 2^{m_d}, \tag{8.8}$$

where $\gamma$ is the probability of maintaining at least one copy of all the Pareto-optimal solutions, $t$ is the number of generations we need to maintain all the niches, and $c_2$ is a constant. While Mahfoud derived the population-sizing estimate for fitness-sharing method, it is generally applicable to other niching methods and MOGAs as well (Khan, 2002; Reed, 2002).

In order to restrict the number of Pareto-optimal solutions, and thereby to circumvent the niching method from being overwhelmed we require $n_{\text{eCGA}} \geq n_{\text{niching}}$. That is,

$$c_2 \cdot 2^{m_d} \geq c_1 \cdot 2^k \cdot m \log m. \tag{8.9}$$

The above equation can be approximated [1] to obtain a conservative estimate of the maximum number of competing substructures that circumvent the niching mechanism from being overwhelmed, which is given by:

$$m_d \approx k + \log_2(m) \tag{8.10}$$

To reiterate, MOEDAs can solve additively separable problems in polynomial time if the number of building blocks that differ between two objectives is less than that predicted in the above equation. We verify this assertion with empirical results for the OneMax-ZeroMax problem in Figure 8.4. Specifically, for the $m - m_d$ building blocks shared by both objectives we consider the OneMax function, and for the $m_d$ differing building blocks, we consider the OneMax-ZeroMax problem. The particular building blocks that differ between the two objectives are randomly chosen for a particular problem instance. As shown in Figure 8.4, the results indicate that when the limit

[1]Since $\log_2\left(\frac{c_1 \log m}{c_2}\right) \sim 1$, we neglect the term

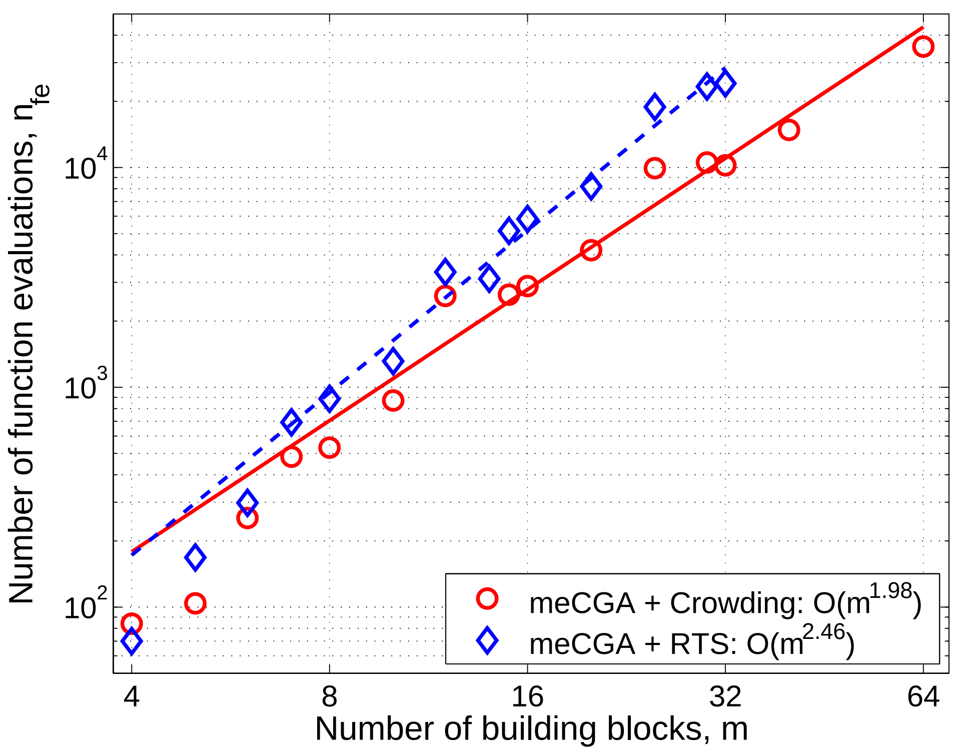


Figure 8.4: Scalability of meCGA with the crowding mechanism of NSGA-II and RTS niching for both OneMax-ZeroMax and m-3 deceptive trap and inverse trap problems. The growth rate of number of substructures that compete in the two objectives for a given problem size is controlled as given by Equation 8.10.

on the growth-rate of competing substructures is satisfied, the MOEDAs scale-up polynomially with the problem size.

## 8.8 Summary

In this chapter, we studied the scalability of multiobjective genetic algorithms, specifically multiobjective extended compact genetic algorithm (meCGA), on a class of boundedly-difficult additively separable problems. We observed that even when the linkages were correctly identified, the multiobjective genetic algorithms scaled-up exponentially with problem size due to the combinatorial growth in the number of Pareto-optimal solutions. The results demonstrated that even if the linkage is correctly identified, massive multimodality of the search problems can easily overwhelm the nicher and lead to exponential scale-up. That is, in decomposable problems, if majority or all the substructures compete in different objectives, then the number of Pareto-optimal solutions increases exponentially. This exponential increase overwhelms the nicher and causes significant problems in maintaining a good coverage of the Pareto-optimal front. This combinatorial explosion of Pareto-optimal solutions introduces a fundamental limit on the number of competing substructures between multiple objectives. Using facetwise models that incorporate the combined effects of model accuracy, decision making, and substructure supply, and the effect of niching on the population sizing, we predict this limit on the growth rate of maximum number of substructures that can compete in the two objectives to circumvent the failure of the niching method. If the number of competing substructures between the multiple objectives is less than the proposed limit, MOGAs scale-up polynomially with the problem size on boundedly-difficult problems.

The results suggest that even though the multiple objectives are conflicting, they have to share common building blocks and have a limited number of building blocks that are different in order for multiobjective GAs to scale polynomially with problem size. This study also demonstrated that while each of the objectives is tractable with single-objective GA, when considered under multiobjective optimization, they become intractable. However, as demonstrated in chapter 5, multiobjective optimizers can also render problems tractable that are intractable via a single-objective optimizer. Therefore, we need further investigation to fully understand what other facets of multiobjective problem difficulty are, and, where multiobjective approaches excel. While most competent multiobjective GAs assume that even though the building blocks differ for different objectives, the underlying substructures are the same for different objectives. However, this need not be the case and further studies are required to analyze the scalability of multiobjective GAs on problems where different objectives have different substructures (or neighborhoods).

# Chapter 9

# Summary and Conclusions

Effective multiscale modeling and simulation methods are essential in advancing both the science and synthesis in a wide array of fields including physics, chemistry, materials science, biology, biotechnology and pharmacology, where many phenomena span several orders in time and space. This thesis demonstrated the potential of using genetic algorithms (GAs)—search, optimization, and machine-learning methods based on natural selection and genetics—and genetic programming (GP)—GAs that evolve computer programs—for multiscale modeling with the help of two case studies in materials science and chemistry. In essence, genetic algorithms and genetic programming are used as effective methods for coupling modeling methods from different scales and are applicable in various multiscaling areas, for example, modeling multi-timescale kinetics, obtaining constitutive rules and finding chemical reaction pathways. This study demonstrated that GAs and GP enable modeling of more complex systems up to realistic timescales which are 2-15 orders of magnitude greater than that possible from current methods, and they do so by using 2–5 orders of magnitude less CPU time. The results show that GA- and GP-enabled multiscale modeling approach holds promise and has the potential to radically transform the way complex multiscale phenomena are modeled and analyzed and materials, chemicals and pharmaceuticals are designed and synthesized.

The first application dealt with multi-timescaling alloy kinetics, which is critical for designing functional nanomaterials. Specifically, GP was used to bridge molecular dynamics (MD) and kinetic Monte Carlo (KMC) to span simulations by orders-of-magnitude in time. On a non-trivial example of vacancy-assisted migration on a surface of a face centered cubic copper-cobalt alloy, GP predicts all barriers with 0.1% error from calculations for less than 3% of active configurations, independent of the type of potentials used to obtain the learning set of barriers via molecular dynamics. The genetic programming-based KMC approach avoids the need or expense of calculating the entire potential-energy surface, is highly accurate, and leads to a significant scale-up in real simulation time for complex cases as it enables use of KMC and, more importantly, leads to a significant reduction in CPU time needed for KMC ($>$ 7-orders of magnitude for quantum-based calculations), not possible from any other current means.

The second case study addressed multiscaling quantum chemistry simulation, with particular focus on photochemical reactions, which are fundamental in many settings such as biological (for example, photosynthesis and vision) and technological (for example, solar cells and LEDs). Multiobjective genetic algorithms were used to bridge high-level quantum chemistry and semiempirical methods to provide accurate representation of complex molecular excited-state and ground-state behavior, well beyond previous attempts, or expectation of human experts, and 2-3 orders reduction in computational cost. Rapid reparameterization of semiempirical methods not only eliminates the need for a full-fledged *ab initio* dynamics simulation, which is prohibitively expensive for large molecules, but also eliminates drawbacks of semiempirical methods that use standard parameter sets and can yield unphysical dynamics. The results show that the evolutionary approach provides significantly better results—with up to 384% lower error in the energy and 86.5% lower error in the energy gradient—than those reported in literature. Moreover multiobjective GAs yield multiple high-quality semiempirical parameter sets that (1) are stable to random perturbations, (2) yield accurate configurational energies on untested and key excited-state configurations, and (3) yield excited-state dynamics simulation results with *ab initio* accuracy. Even more surprising and potentially groundbreaking, our MOGA results produce *transferable* potentials—that is, parameters from one molecular system can be used for similar systems.

While the applications part of the thesis used a fairly straightforward flavor of genetic programming and multiobjective genetic algorithms, in order to address more complex systems, we need to have a better understanding of the scalability and the limits of these algorithms. Therefore, along with applications, the thesis also addressed and analyzed the scalability of such methods, in particular genetic programming and multiobjective genetic algorithms. It should be noted that although the scalability studies were motivated with the specific application of GAs and GP to multiscaling problems, the models developed and the lessons learned are not just limited to the multiscaling domain, but are broadly applicable to a wide range of problems of interest to both theoreticians and practitioners in genetic and evolutionary computation field.

Facetwise models of (i) population sizing required for adequate supply of raw subsolutions required to obtain optimal solutions, and (ii) population sizing required to ensure accurate decision making between competing subsolutions were presented. The building-block supply based population-sizing model indicates that there is a minimum tree size dependent on the problem size. Furthermore, the models suggest that when the tree size is greater than the problem size, the population size required on BB supply grounds is $2^k\left(k \ln \chi + \ln m\right)$, where $k$ is the order of the building block, $\chi$ is the alphabet cardinality, and $m$ is the number of building blocks. The population-sizing model based on accurate decision-making shows that, to ensure correct decision making within an error tolerance, population size must go up as the probability of error decreases, noise increases, alphabet cardinality increases, the signal-to-noise ratio decreases *and* tree size decreases and bloat frequency increases.

While simple genetic programming scales cubically on easy problems, it scales exponentially on hard problems. Therefore, this study developed a scalable design of GP, called extended compact genetic programming (eCGP), which is based on the extended compact genetic algorithm (eCGA). The proposed algorithm adaptively identifies, propagates, and exchanges important subsolutions of a search problem. The proposed competent GP solves a broad class of *adversarially designed* boundedly difficult problems using only polynomial (cubic) number of function evaluations. This is in contrast to standard GP with fixed recombination operators that scale exponentially on GP-hard problems. The extended compact GP design and subsequent work on scalable GP by other researchers clearly demonstrate that, as with GAs, it is possible to design scalable GP in a principled manner. Moreover, this study also demonstrated that competent GP designs can be advantageous over standard GP with fixed recombination operators.

Finally, this thesis also addressed the limits on the scalability of multiobjective GAs in reliably maintaining a diverse set of optimal solutions—also known as Pareto-optimal solutions. Along the way, based on the multiobjective Bayesian optimization algorithm, a competent MOGA design—the multiobjective extended compact genetic algorithm (meCGA)—was proposed that can solve boundedly-difficult problems using only a polynomial (quadratic) number of function evaluations. The results show that even when the building blocks are correctly identified, massive multimodality of the search problems can easily overwhelm the nicher and lead to exponential scale-up. Facetwise models that incorporate the combined effects of model accuracy, decision making, and sub-structure supply, and the effect of niching on the population sizing, predict a limit on the growth rate of maximum number of building blocks that can compete among multiple objectives to circumvent the failure of the niching method. If the number of competing building blocks between the multiple objectives is less than the proposed limit, MOGAs scale-up polynomially with the problem size on boundedly-difficult problems. The results suggest that even though the multiple objectives are conflicting, they have to share common building blocks and have a limited number of building blocks that are different in order for multiobjective GAs to scale polynomially with problem size.

This study shows that genetic algorithms and genetic programming are indeed valuable tools for bridging lower-level and higher-level models and enabling practical multiscale modeling of materials, chemical, physical, and biological phenomena. GAs and GP are scalable, robust, and efficient procedures that solve hard problems quickly, reliably, and accurately. The multiscale modeling approaches developed here can be applied readily to other systems similar to those addressed in this thesis. The proposed work can be enhanced in a number of ways to enable fast and accurate modeling, design and analysis of more complex materials and reaction-chemistry systems.

To extend multiscale kinetics modeling via GP for more complex, cooperative effects, such as island diffusion via surface dislocations, GP could be interfaced with temperature accelerated dynamics and/or pattern-recognition methods. For systems with long-range fields, GP could be coupled with phase field methods. Moreover, the efficiency of the GP-based multiscale modeling approach can be enhanced by coupling it with local cluster expansion methods.

To enable fast and accurate excited-state dynamics of more complex chemical systems, along with multiobjective GAs for tuning semiempirical parameters, GP could be used to evolve domain-specific potentials starting from scratch or from existing semiempirical methods. Also, the initial evidence of transferability of MOGA optimized semiempirical methods is highly promising. More systematic studies are needed to validate this initial evidence, however, this opens up the possibility of accurate simulations of photochemistry in complex environments such as proteins and condensed phases. If this pans out it will transform the way chemicals are modeled and designed radically.

Furthermore, the multiscale modeling approach proposed in this thesis should also carry over to modeling multiscale phenomena in other domains. More work needs to be done to extend the proposed approach and while the underlying mathematics and systems will be different, the methodology and the implementation should be similar to those described in this study.

For GP and MOGA researchers, this study clearly shows the power of facetwise and dimensional modeling and suggests that both GP and MOGA community would benefit from paying more attention to design-decomposition and facetwise thinking than they have so far. Similar to GAs, facetwise models can be very useful in GP and MOGA domains not only yielding practical performance bounds, but also is helping understand their limits and capabilities.

More work needs to be done to extend the population-sizing model for GP to be more broadly applicable. A tighter population-sizing bound can be obtained by extending the models along the lines of the gambler's ruin model (Harik, Cantú-Paz, Goldberg, & Miller, 1999). Additionally, in developing population sizing models for GP, we assumed a fairly straightforward representation scheme, however, GP practitioners use fairly sophisticated and advanced representations, and the population-sizing models need to be extended to handle other commonly used representations.

As with GAs, GP researchers can benefit from scalable and efficient GP designs as opposed to using fixed search operators that don't discover, exchange and explore key building blocks. Work on designing scalable GP has already progressed and initial results show promise, and they could be further enhanced by addressing, in a principled manner, scalability and model building issues. An important and open question remains however, as to whether there are any GP-hard problems that thwart the mechanism of selectorecombinative GP. This issue is also related to a broader issue of problem difficulty in system identification. While addressing optimization problems, it is clear that there is premium on scalable recombination and mutation operators that automatically identify and exploit good substructures (neighborhoods). However, when solving system-identification problems, say symbolic regression via GP or model building in estimation of distribution algorithms, we seem to get away with simple crossover methods or even hill-climbers. Another follow-up question of critical importance to scalable genetic-programming design is, what the principal dimensions of GP problem difficulty are? The answer to this question will also help us design *adversarial* problems for estimation of distribution algorithms where probabilistic models are typically built using a hill-climber.

For developing tractable multiobjective GA designs, MOGA researchers would benefit from designing scalable operators that yield representative coverage of the Pareto-optimal solution. Rather than requiring to capture the entire Pareto-front, which in many cases might not be practical and might even be unreasonable, relaxing the requirement and thinking in terms of approximate, but representative, coverage of the Pareto-optimal solutions would lead to MOGA designs that solve hard multiobjective problems quickly, reliably, and accurately.

In addressing scalability of MOGAs, we considered a class of problems where each of the objectives is tractable with single-objective GA, when treated as multiple objectives, they can become intractable. However, this is not the only possibility, and as demonstrated in chapter 5, multiobjective GAs can be more efficient than single-objective GAs and we need more studies to understand when and where multiobjective GAs excel. The scalable multiobjective GA developed in this study assumes that different objectives share the same building-block structure and builds and samples from a single probability model. However, this need not be the case and further studies are required to design scalable multiobjective GAs and to analyze their scalability on problems where different objectives have different sub-structures (or neighborhoods).

We hope that this work encourages other researchers to apply, analyze, and design genetic algorithms and genetic programming for the multiscale modeling of complex phenomena in physics, chemistry, materials science, biology and pharmacology. Such an endeavor holds promise in yielding scalable and robust multiscale modeling approaches that will be beneficial for the design, analysis, and modeling of complex systems.

# Appendix A

# GP Regressed Inline Barrier Function

The inline barrier function $f_{\text{barrier}}$ symbolically regressed via GP for 2nd n.n. jumps considering only seven surface 1$^{\text{st}}$ n.n. environmental atoms (our initial simple case) is given by

$$\begin{aligned} f_{\text{barrier}}(\vec{x}) &= x_1 - 2x_2 + x_3 + x_4 - 2x_5 + x_7 \\ &\quad - \frac{x_7}{x_6} - g_1 - g_2 + g_3 \end{aligned} \tag{A.1}$$

where $\vec{x} = \{x_1, x_2, x_3, x_4, x_5, x_6, x_7\}$, and $x_i$ is either 0 or 1 denoting a Cu or Co atom, and

$$\begin{aligned} g_1 &= \frac{1}{x_5}\left[x_4 + \frac{x_5}{(x_1 + x_4 + 2x_5 + 40x_7)}\right]^{0.025} \\ g_2 &= \frac{-g_4 \cdot [x_2 + 0.25x_5]^{\left(g_7 x_2^{g_6(1+x_3)}\right)}}{[x_1 + x_3 + x_4 + 0.945 \cdot x_4 x_6 / x_7]} \\ g_3 &= 40x_2\left[x_2^{(1+x_2+2x_3+x_5+g_5)}\right] \end{aligned} \tag{A.2}$$

The functions $g_i(\vec{x})$ are highly non-linear functions of the configurations, that is,

$$\begin{aligned} g_4 &= g_8\left[\frac{0.473 \cdot x_3 x_5 \left(x_2^{(x_7/x_4)}\right)^{2x_3}}{0.177 \cdot x_1 x_6 g_9 (x_1 + x_4)(x_2 + x_5)}\right] \\ g_5 &= \frac{(x_2 + 0.025x_5)^{g_{10}} \cdot g_{11}}{x_2 + x_3 + x_4 + g_{12}} \\ g_6 &= x_2 + \frac{x_5\left[2(x_2 + x_5) + \frac{x_7}{x_6} + 40\right]}{x_5 + x_3\left[2(x_2 + x_5) + \frac{x_7}{x_6} + 40\right]} \\ g_7 &= (g_{13} + g_{14})\left[x_2^{(x_2 + x_5/(x_2 + x_5 + x_4))}\right] \end{aligned} \tag{A.3}$$

$$
\begin{aligned}
g_8 &= \left[(x_2^{g_{14}} + g_{17})\left(x_2^{(x_2+g_{18})} + g_{17}\right)\right]^{0.025} \\
g_9 &= \frac{x_6(0.473 + x_6)}{x_4(x_2^{g_{19}})^{\frac{x_5}{x_7}}} \\
g_{10} &= g_{15} x_2^{g_{20}(2x_2x_5 + x_2x_4x_6 + x_5)} \\
g_{11} &= \frac{3x_3x_5x_2^{(2x_2)}}{g_{21}\left[x_3 - 1 - \frac{x_6}{x_7} + (0.59 - x_1)(x_5 - x_6)\right]} \\
g_{12} &= x_1 + x_2 + 3x_3 + x_4 + x_5 + x_2^{2(x_2+x_5)} \\
&\quad + 0.95\frac{x_4x_6}{x_7} + (0.59 - x_1)(x_5 - x_6) \\
&\quad + \frac{x_7}{x_6 - 0.23}
\end{aligned}
\tag{A.4}
$$

$$
\begin{aligned}
g_{13} &= x_2 + \frac{x_5}{0.473 + x_1 + x_3 + x_4 + x_2^{g_{22}}} \\
g_{14} &= x_2 + \frac{x_5\left[0.473 + x_2^{g_{23}}\right]}{x_5 + x_2\left[0.473 + x_2^{g_{23}}\right]} \\
g_{15} &= 1 + x_2 + x_5 + \frac{x_5}{x_6} + \frac{x_7}{x_4} + \frac{x_5}{x_1 + x_4} \\
&\quad + g_{16}^{(x_5/x_7)} + x_2^{\left(x_3 - 1 + \frac{x_6}{x_5} - \frac{x_6}{x_7} + \frac{x_6}{x_2}\right)} \\
g_{16} &= x_2^{\left(-1 - x_1 + x_3 - x_4 + \frac{x_6}{x_7} + \frac{x_6}{x_5} - g_{24}\right)} \\
g_{17} &= \frac{x_5\left[0.473 + 40x_7\right]}{x_5 + x_2\left[0.473 + 40x_7\right]}
\end{aligned}
\tag{A.5}
$$

$$
\begin{aligned}
g_{18} &= \frac{x_5\left[0.473 + x_2^{(x_2+x_5)}\right]}{x_5 + (x_2 + x_5)\left[0.473 + x_2^{(x_2+x_5)}\right]} \\
g_{19} &= \left[-x_1 + x_3 - x_4 + \frac{x_6}{x_5} - g_{24}\right]^{(x_5/x_7)} \\
g_{20} &= \frac{\left[x_5 + x_7/x_6\right]^{(x_2+x_5)}}{x_2x_5 + x_5 + x_4x_6} \\
g_{21} &= x_6\,(x_2 + x_5)\left[x_3 + \frac{x_5}{x_2 + x_5 + \frac{x_7}{x_6} + g_{25}}\right]
\end{aligned}
\tag{A.6}
$$

$$\begin{aligned}
g_{22} &= \frac{x_6}{x_7} + \frac{x_6}{x_5} - x_1 - x_3 - x_4 \\
&\quad - \left(\frac{x_2}{x_1}\right)^{\left(\frac{x_6}{x_3} - 0.47(x_2+x_5)\right)} \\
g_{23} &= \left[x_3 + \frac{x_6}{x_5}\right]^{(3x_3 + x_7/x_6)} \\
g_{24} &= \left(\frac{x_2}{x_1}\right)^{\left(\frac{x_6}{x_3} + 0.473(x_2+x_5)\right)} \\
g_{25} &= 0.47 + \frac{x_5}{x_2 + x_5 + \frac{x_7}{x_6} + \frac{1}{x_2+x_5} + \frac{x_4}{0.473+x_6}}
\end{aligned} \tag{A.7}$$

# Appendix B

# Semiempirical Parameter Interactions for Ethylene Symbolically Regressed via Genetic Programming

As described in chapter 5, the multiobjective genetic algorithms yields 61 Pareto-optimal semiempirical parameter sets that are (1) stable to small perturbations, (2) yield accurate configurational energies, and (3) yield *ab initio* quality excited-state dynamics are selected. The data is normalized using a z-score and the normalized data is used to evolve relationships between the semiempirical parameters via GP. In all GP runs, a population size of 1000 and a run duration of 100 generations were used. For each of the semiempirical parameter, over 100 independent GP runs were computed. The best evolved regression function for each of the independent runs are then simplified using the symbolic math toolbox in matlab. The coefficients are then optimized using either linear or non-linear regression methods. The best evolved solutions for each of the semiempirical parameters along with the RMS error values and the number of independent GP runs that yield the functional form are given in the following tables.

Table B.1: $U_{ss} = f\left(U_{pp}, \beta_s, \beta_p, \zeta_s, \zeta_p, G_{ss}, G_{sp}, G_{pp}, G_{p_2}, H_{sp}\right)$

| # runs | RMSE | GP-regressed equation |
|---|---|---|
| 1 | 0.1632 | $0.3903 + 2.4027U_{pp} + 1.2746G_{p_2} + 0.7383\beta_s - U_{pp}^2\left(0.563 + 0.4178U_{pp} - 0.791\zeta_s + 0.0712\zeta_p + 0.7951\beta_s\right) - U_{pp}G_{p_2}\left(1.2009 + 1.4634U_{pp} - 0.2316\zeta_s - 1.1747\zeta_p + 1.6616\beta_s\right) - G_{p_2}^2\left(1.3827 + 1.2974U_{pp} + 0.267\zeta_s - 0.6504\zeta_p + 0.8195\beta_s\right)$ |
| 1 | 0.1736 | $0.394 + 2.1902U_{pp} + 1.407G_{p_2} + 0.5817\beta_s - 0.1002\zeta_s - G_{p_2}\left(0.9223G_{p_2} - 0.2069H_{sp}\right) + U_{pp}\left[0.0888H_{sp} - 0.0933\zeta_s - 0.1074\zeta_p - G_{pp}\left(0.381 - 0.3299\zeta_s - 0.4472\zeta_p - 0.9065\beta_p\right)\right] - U_{pp}^2\left[0.3761 + 0.1099G_{p_2} - 0.8357\zeta_s + 0.2337\zeta_p - \beta_p\left(0.8698 + 0.1072\zeta_s + 0.0132\zeta_p\right)\right] + U_{pp}^3\left(0.0009 + 0.1958\beta_p\right)$ |
| 1 | 0.1982 | $1.6576U_{pp} + 2.0861G_{p_2} + 0.7953\beta_s + 0.4763\zeta_s - 1.4999G_{ss} - 0.3402G_{sp}$ |
| 1 | 0.1991 | $-0.0911 + 2.25U_{pp} + 2.0463G_{p_2} + 0.6927\beta_s + 0.4392\zeta_s - 1.1469G_{ss} + U_{pp}\left(0.64U_{pp} + 0.4184G_{p_2} + 0.2647G_{sp}\right) + H_{sp}\left(0.1373U_{pp} + 0.0982G_{p_2}\right)$ |
| 1 | 0.2804 | $-0.2239 + 1.8822U_{pp} + 1.7333G_{p_2} + 0.4808\zeta_s - 0.9175G_{ss} + U_{pp}\left(0.0282U_{pp} + 0.9072G_{p_2} - 1.0425G_{ss} - 0.1828\beta_s\right)$ |
| 1 | 0.2998 | $0.0502 + 2.3422U_{pp} + 1.4189G_{p_2} + 0.2755\beta_s + 0.2569\zeta_s + U_{pp}\left(0.3304U_{pp} + 0.5637G_{p_2} - 0.1205U_{pp}^2\right)$ |
| 1 | 0.3004 | $-0.0018 + 1.7687U_{pp} + 1.3788G_{p_2} + 0.357\beta_s + 0.1857\zeta_s + 0.2829\beta_p - 0.2256G_{sp} + U_{pp}\left(0.559U_{pp} + 0.6168G_{p_2}\right)$ |
| 1 | 0.3095 | $-0.0192 + 2.1736U_{pp} + 1.3796G_{p_2} + 0.3232\beta_s + 0.2317\zeta_s + U_{pp}\left(0.5314U_{pp} + 0.4779G_{p_2} + 0.0939G_{ss}\right)$ |
| 1 | 0.3098 | $-0.0268 + 2.1549U_{pp} + 1.3584G_{p_2} + 0.3008\beta_s + 0.2303\zeta_s + U_{pp}\left(0.5202U_{pp} + 0.546G_{p_2}\right)$ |
| 1 | 0.3145 | $0.1899 + 1.8145U_{pp} + 1.2289G_{p_2} + 0.6947\beta_s - 0.0536G_{ss} - 0.3880H_{sp} + U_{pp}\left(0.7601G_{ss} - 0.0339\beta_s\right) + U_{pp}^2\left(0.4768 + 0.4365G_{ss} + 0.4493\beta_p\right) + U_{pp}^3\left(0.3557 + 0.1363\beta_p\right)$ |
| 1 | 0.3216 | $0.2432 + 1.8399U_{pp} + 1.1777G_{p_2} + 0.3087\beta_s - U_{pp}\left(0.8658G_{p_2} + 0.1149\zeta_p\right) - G_{p_2}^2\left(1.1881 - 0.5985U_{pp}\right) + U_{pp}^2\left(0.0243 - 0.1025U_{pp} - 0.6153G_{p_2}\right)$ |
| 1 | 0.3301 | $-0.057 + 1.9725U_{pp} + 1.2766G_{p_2} + 0.2789\zeta_s - 0.1362H_{sp} + U_{pp}\left(0.462U_{pp} + 0.3196G_{p_2}\right) + H_{sp}\left(0.1466U_{pp} + 0.3222G_{p_2}\right)$ |
| 1 | 0.3423 | $0.0912 + 2.1373U_{pp} + 1.3311G_{p_2} + 0.4527\beta_s + U_{pp}\left(0.3350U_{pp} + 0.3619H_{sp}\right) + G_{p_2}\left(0.5317U_{pp} + 0.2509H_{sp}\right)$ |

*continued on next page*

*continued from previous page*

| # runs | RMSE | GP-regressed equation |
|---|---|---|
| 1 | 0.3627 | $2.0741U_{pp} + 1.1774G_{p_2} + 0.3599\beta_s + 0.2303\zeta_s - 0.0193G_{pp} + 0.0359G_{sp}$ |
| 2 | 0.3630 | $2.0084U_{pp} + 1.1365G_{p_2} + 0.3586\beta_s + 0.2168\zeta_s$ |
| 2 | 0.3750 | $0.0266 + 2.0859U_{pp} + 1.2892G_{p_2} + 0.4022\beta_s + U_{pp}\left(0.4584U_{pp} + 0.5371G_{p_2}\right)$ |
| 1 | 0.3754 | $1.9243U_{pp} + 1.4796G_{p_2} + 0.5228\beta_s + 0.2195\zeta_p - 0.6043G_{ss}$ |
| 1 | 0.3902 | $0.0519 + 1.7381U_{pp} + 1.2381G_{p_2} + H_{sp}\left(0.5441U_{pp} + 0.684G_{ss}\right)$ |
| 1 | 0.3907 | $0.2218 + 1.5811U_{pp} + 1.1702G_{p_2} - G_{p_2}\left(0.7951U_{pp} + 0.9851G_{p_2}\right) - U_{pp}\beta_s^2\left(0.0247 + 0.0577\zeta_p\right)$ |
| 1 | 0.4127 | $1.937U_{pp} + 1.1359G_{p_2} + 0.4576\beta_s + 0.1386\beta_p$ |
| 1 | 0.4132 | $0.2633 + 1.4686U_{pp} + 0.7189G_{p_2} + 0.4865G_{ss} - G_{ss}\left(-0.7815U_{pp} - 1.027G_{p_2}\right)$ |
| **37** | **0.4166** | $\mathbf{1.9683U_{pp} + 1.0523G_{p_2} + 0.4376\beta_s}$ |
| 2 | 0.4249 | $0.0121 + 2.0035U_{pp} + 1.0156G_{p_2} + 0.5205G_{ss} + U_{pp}\left(0.7290U_{pp} + 0.8204G_{p_2}\right)$ |
| 4 | 0.4421 | $1.7957U_{pp} + 1.1643G_{p_2} + 0.2896\zeta_s$ |
| 1 | 0.4484 | $-0.0342 + 1.4607U_{pp} + 0.78G_{p_2} - G_{ss}\left(0.8126U_{pp} + 0.9252G_{p_2}\right)$ |
| 8 | 0.4694 | $-0.0424 + 1.8938U_{pp} + 1.3082G_{p_2} + 0.6362U_{pp}^2 + 0.657U_{pp}G_{p_2}$ |
| 2 | 0.4762 | $1.6819U_{pp} + 1.0312G_{p_2} + 0.2246\zeta_p$ |
| 1 | 0.4786 | $-0.2657 + 0.6357U_{pp} - 0.4520G_{sp} + 0.4884G_{pp} - U_{pp}\left(0.2826G_{ss} + 0.1677G_{sp}\right) - 0.0942G_{ss}^2$ |
| 1 | 0.4982 | $0.6020U_{pp} + 0.3443\beta_s - 0.4692G_{pp} + 0.2957G_{sp}$ |
| 3 | 0.5052 | $1.7368U_{pp} + 0.7899G_{p_2} + 0.3539G_{ss}$ |
| 24 | 0.5253 | $1.6705U_{pp} + 1.0519G_{p_2}$ |
| 1 | 0.5551 | $0.3406U_{pp} - 0.5024G_{sp} + 0.3576G_{pp}$ |
| 6 | 0.5711 | $1.3678U_{pp} + 0.7565G_{ss}$ |

Table B.2: $U_{pp} = f\left(U_{ss}, \beta_s, \beta_p, \zeta_s, \zeta_p, G_{ss}, G_{sp}, G_{pp}, G_{p_2}, H_{sp}\right)$

| # runs | RMSE | GP-regressed equation |
|---|---|---|
| 1 | 0.1009 | $0.034 + 0.1503U_{ss} + 0.0176\beta_p + 0.0082\zeta_p - 0.3425G_{sp} - 0.5531G_{p_2} + 0.1682G_{pp} + G_{pp}\left[0.035G_{sp} + 0.1809G_{p_2} + 0.0104G_{sp}G_{p_2} - 0.0309G_{sp}G_{p_2}^2 - \zeta_s\left(0.046\zeta_s - 0.0738G_{sp} + 0.1755G_{p_2} - 0.0487G_{sp}G_{p_2} - 0.0017G_{sp}G_{p_2}^2\right)\right]$ |
| 1 | 0.1607 | $-0.0136 + 0.3889U_{ss} - 0.2372\beta_s - 0.6331G_{p_2} - U_{ss}G_{p_2}^2G_{pp}^3\left[U_{ss}\left(0.1161\beta_p + 0.2186G_{p_2} + 0.2299\beta_p^2 + 0.2654\beta_pG_{p_2}\right) + \beta_p^2\left(0.0941\beta_p + 0.0908G_{p_2} + 0.0796\beta_p^2 + 0.0754\beta_pG_{p_2}\right)\right]$ |
| 1 | 0.1717 | $0.1740U_{ss} + 0.3888\beta_p - 0.2996G_{sp} - 0.3252G_{p_2}$ |
| 1 | 0.1828 | $-0.0524 + 0.4105U_{ss} - 0.2211\beta_s - 0.6399G_{p_2} + 0.0527G_{p_2}^2$ |
| 1 | 0.1868 | $0.3957U_{ss} - 0.2176\beta_s - 0.5889G_{p_2}$ |
| 3 | 0.2027 | $0.5375\beta_p - 0.4268G_{sp} - 0.2078G_{p_2}$ |
| 1 | 0.2068 | $0.5604\beta_p - 0.1691G_{ss} - 0.4567G_{sp}$ |
| 1 | 0.2078 | $0.0999U_{ss} + 0.6774\beta_p - 0.4159G_{sp}$ |
| 1 | 0.2124 | $0.4145U_{ss} - 0.1404\zeta_s - 0.7135G_{p_2}$ |
| **76** | **0.2184** | $\mathbf{0.7076\beta_p - 0.4736G_{sp}}$ |
| 1 | 0.2390 | $0.3888U_{ss} - 0.2024G_{ss} - 0.5416G_{p_2}$ |
| 21 | 0.2533 | $0.3885U_{ss} - 0.7258G_{p_2}$ |
| 1 | 0.3058 | $-0.3467G_{sp} - 0.7290G_{p_2}$ |
| 2 | 0.3244 | $0.4412U_{ss} - 0.6732G_{ss}$ |

Table B.3: $\beta_s = f\left(U_{ss}, U_{pp}, \beta_p, \zeta_s, \zeta_p, G_{ss}, G_{sp}, G_{pp}, G_{p_2}, H_{sp}\right)$

| # runs | RMSE | GP-regressed equation |
|---|---|---|
| 1 | 0.2660 | $-0.5513 + 0.0414G_{ss} - 0.2334U_{pp} - 0.0279U_{pp}^2 + G_{ss}U_{pp}\left(2.4839 + 1.1533H_{sp} - 2.5541e^{0.5676H_{sp}}\right) +$ $G_{ss}H_{sp}\left[0.2629 - 0.0732\left(1.2234 - 0.7982G_{ss}\right)^{-1}\right] -$ $G_{ss}.^2\left[0.7477 - 1.4057e^{0.6186H_{sp}} - H_{sp}\left(0.1393 - 1.2923e^{0.6858H_{sp}}\right)\right] - 6\times10^{-4}\frac{G_{ss}}{\zeta_s} -$ $0.4122G_{ss}\left(0.2213 - 1.2160G_{ss}\right)^{-1} - 0.1754U_{pp}\left(0.2485 - 1.3568G_{ss}\right)^{-1}$ |
| 1 | 0.3488 | $-0.627 - 0.0873G_{ss} + 0.7542G_{ss}^2 + G_{ss}^3\left(0.694H_{sp} + 0.0389G_{sp}\right) + G_{ss}^4H_{sp}$ |
| 1 | 0.3815 | $1.1245G_{ss} - 1.2219G_{p_2} + 0.7681G_{sp} + 0.5544U_{ss} - 0.7602\beta_p$ |
| 1 | 0.3940 | $-0.0965 + 0.4319G_{ss} + G_{ss}G_{p_2}\left(0.2846G_{sp} + 0.5117\zeta_p\right)$ |
| 1 | 0.4162 | $-0.3411 + G_{ss}^2\left(0.6292U_{ss} - 0.6228U_{pp}\right)$ |
| 1 | 0.4202 | $-0.3906 + 0.5406U_{ss} - 0.6114U_{pp} + G_{ss}\left(0.9147U_{ss} - 0.9021U_{pp}\right)$ |
| 1 | 0.4454 | $-0.0654 + 0.7085G_{ss} + \beta_p\left(0.4011U_{pp}e^{0.2952\zeta_p} - 0.0656e^{-6.6553\zeta_p}\right)$ |
| 1 | 0.4515 | $0.215 + 1.0824G_{ss} + H_{sp}\left(0.6158U_{pp} - 0.8862\beta_p\right)$ |
| 1 | 0.4540 | $0.0568 + 0.8329G_{ss} + 0.064H_{sp} + G_{ss}\left[0.264H_{sp} + G_{sp}\left(0.0818 + 0.1694U_{pp}G_{pp}\right)\right]$ |
| 1 | 0.4543 | $0.125 + 1.0033G_{ss} - G_{ss}^2\beta_p\frac{0.0032G_{ss} - 0.0002U_{ss}}{\zeta_s}$ |
| 1 | 0.4545 | $0.1658 + 0.9494G_{ss} + G_{ss}H_{sp}\left(0.4071 - 0.2175G_{sp}\right)$ |
| 1 | 0.4559 | $-0.4112 + \zeta_p^2\left(0.3190 + 1.4573\beta_p\right) - U_{pp}\zeta_p\left(0.4582 - 1.8866\beta_p\right) + U_{pp}^2\left(0.2638 - 0.3682\beta_p\right)$ |
| 1 | 0.4569 | $-0.2867 - 0.4414U_{pp} - G_{ss}\left(0.6588U_{pp} - 0.5122H_{sp}\right)$ |
| 1 | 0.4729 | $-0.1178 - 0.7004G_{ss}G_{p_2} + 0.5634G_{ss}^3$ |
| 1 | 0.4849 | $0.1111 + 0.9987G_{ss}e^{-0.0219\frac{H_{sp}}{\zeta_s}}$ |
| 1 | 0.4924 | $0.0703 + 0.9472G_{ss} + 0.0284\zeta_s^{-1}$ |
| 1 | 0.4978 | $0.0342 + 0.8486G_{ss} + G_{ss}\beta_p\left(0.7410U_{pp} + 0.7571G_{p_2}\right)$ |

*continued on next page*

*continued from previous page*

| # runs | RMSE | GP-regressed equation |
|---|---|---|
| 1 | 0.5034 | $-0.2312 + 0.392G_{ss}e^{0.8757U_s - 0.929U_{pp}}$ |
| 1 | 0.5045 | $-0.0383 + 1.2735G_{ss} - 0.2092U_{pp} - 0.6684G_{p_2} + U_{pp}\left(0.669U_{pp} + 0.6979G_{p_2}\right)$ |
| 2 | 0.5178 | $1.3712G_{ss} - 0.8048G_{p_2} + 0.3411G_{sp}$ |
| 2 | 0.5192 | $0.0344 + 0.6140G_{ss} + G_{p_2}\left(1.3288G_{ss} - 1.2377G_{p_2}\right)$ |
| 1 | 0.5197 | $-0.1347 + 0.5683G_{ss} + \beta_p U_{pp}\left(1.0231G_{ss} - 0.9389G_{p_2}\right)$ |
| 1 | 0.5389 | $-0.133 + 0.6256G_{ss} + 1.2373G_{sp}\zeta_p\beta_p$ |
| 1 | 0.5411 | $-0.0896 + 0.5541G_{ss} + 0.1908G_{ss}^2G_{sp}$ |
| 1 | 0.5520 | $0.0696 + 0.9275G_{ss} - \zeta_p G_{p_2}\left(1.0015G_{sp} + 0.8862U_{ss}\right)$ |
| 1 | 0.5523 | $0.0003 + 0.9083G_{ss} + U_{pp}\left(0.7586U_{pp} + 0.7586G_{p_2}\right)$ |
| 3 | 0.5603 | $1.1550G_{ss} - 0.8560G_{p_2} - 0.4642U_{pp}$ |
| 1 | 0.5608 | $0.1451 + 0.9888G_{ss} + 0.2343G_{p_2}H_{sp}$ |
| 1 | 0.5621 | $0.117 + 0.9131G_{ss}e^{0.3144H_{sp}}$ |
| 1 | 0.5682 | $-0.3905 - 0.2893G_{ss} - 0.247G_{ss}^2 + 0.4773G_{ss}^3$ |
| 1 | 0.5719 | $0.0605 + 1.0173G_{ss} + 0.1365H_{sp} - 0.2272H_{sp}U_{ss}$ |
| 1 | 0.5726 | $-0.2530 - 0.2826G_{ss}^2 + 0.4006G_{ss}^3$ |
| 1 | 0.5727 | $-0.0759 + 0.6984G_{ss} - 0.8896G_{p_2}G_{sp}\zeta_p$ |
| 1 | 0.5732 | $1.0966G_{ss} + 0.4674G_{sp} + 0.5924U_{pp}$ |
| 4 | 0.5740 | $-0.3612 + 0.2711G_{ss}^3$ |
| 1 | 0.5780 | $-0.0516 + 0.7778G_{ss} - 0.4289\zeta_p\beta_p$ |
| 1 | 0.5808 | $0.0772 + 0.9909G_{ss} - 0.2899U_{ss}G_{ss}$ |
| 3 | 0.5846 | $-0.6586 - 0.1153G_{ss} + 0.663G_{ss}^2$ |

*continued on next page*

*continued from previous page*

| **# runs** | **RMSE** | **GP-regressed equation** |
|---|---|---|
| 13 | 0.5854 | $-0.5790 + 0.5828G_{ss}^2$ |
| 1 | 0.5875 | $0.0876 + 0.945G_{ss} + 0.0502G_{ss}^2 H_{sp}$ |
| 1 | 0.5902 | $0.1020 + 0.8703G_{ss} - 0.6282G_{ss}U_{pp} - 0.7042G_{ss}G_{p_2}$ |
| **38** | **0.5932** | $\mathbf{1.2476G_{ss} - 0.5208G_{p_2}}$ |
| 1 | 0.5952 | $-0.0644 + 0.7973G_{ss} + 0.4697G_{p_2}\zeta_p$ |
| 1 | 0.5962 | $-0.0539 + 1.0277G_{ss} + 0.0967\frac{G_{ss}}{\zeta_p}$ |
| 8 | 0.5970 | $0.9051G_{ss} + 0.2439H_{sp}$ |
| 2 | 0.6029 | $0.6934G_{ss} + 0.204G_{sp}$ |

Table B.4: $\beta_p = f\left(U_{ss}, U_{pp}, \beta_s, \zeta_s, \zeta_p, G_{ss}, G_{sp}, G_{pp}, G_{p_2}, H_{sp}\right)$

| # runs | RMSE | GP-regressed equation |
|---|---|---|
| 1 | 0.1449 | $0.3102 - 0.1547G_{ss} - 0.2364G_{p_2} - 0.0024\frac{H_{sp}}{\zeta_s} + G_{ss}\left(0.0315H_{sp} - 0.1771\zeta_p + 0.2691G_{ss}G_{pp}\right) -$ $G_{pp}\left[0.0158U_{ss}e^{-4.8114G_{ss}} + 0.0171G_{pp}e^{1.5582G_{ss}} - 0.1618G_{ss}e^{0.2066G_{ss}}\right]$ |
| 1 | 0.1993 | $0.0337 - 0.7371G_{ss} + G_{ss}G_{pp}\left(0.3385 - 0.0568G_{pp}\right) + 0.0541G_{pp}e^{0.8123G_{sp}}$ |
| 1 | 0.2127 | $-0.0602 - 1.0024G_{ss} + G_{ss}G_{pp}\left[0.0072\beta_s - \frac{G_{ss}^3 G_{pp}}{U_{pp}}\left(0.0251G_{ss} + G_{sp}\left(0.0359U_{pp} + 0.0149\beta_s\right)\right)\right]$ |
| 1 | 0.2327 | $-0.0135 - 0.8805G_{ss} + 0.0761G_{pp} + 0.2080G_{ss}G_{pp}$ |
| 1 | 0.2360 | $-0.0161 - 0.8836G_{ss} + 0.0347G_{sp} + 0.2485G_{ss}G_{pp}$ |
| 1 | 0.2369 | $-0.0068 - 0.8394G_{ss} + 0.0086H_{sp} + H_{sp}\left(0.2526G_{ss} - 0.0373G_{pp}\right)$ |
| 6 | 0.2381 | $-0.0161 - 0.87G_{ss} + 0.2477G_{ss}G_{pp}$ |
| 1 | 0.2628 | $0.0282 - 0.6897G_{ss} - 0.4764G_{pp}U_{pp}$ |
| 6 | 0.2658 | $-0.5944G_{ss} + 0.2859G_{pp} - 0.3611G_{p_2}$ |
| 1 | 0.2686 | $-0.0947 - 1.0906G_{ss} + 0.0903G_{ss}G_{pp}^2$ |
| 9 | 0.2695 | $-0.01 - 0.7005G_{ss} + 0.4195G_{pp}G_{p_2}$ |
| 3 | 0.2787 | $-0.2202G_{ss} + 1.0176U_{pp} + 0.474G_{sp}$ |
| 1 | 0.2824 | $1.0427U_{pp} + 0.5104G_{sp} - 0.2096G_{p_2}$ |
| 21 | 0.2932 | $1.2751U_{pp} + 0.5708G_{sp}$ |
| **28** | **0.3048** | $\mathbf{-0.9237G_{ss} + 0.2987G_{pp}}$ |
| 1 | 0.3148 | $-0.0146 - 1.0401G_{ss} + 0.1982\zeta_p + \zeta_p\left(0.1219G_{pp} - 0.1884G_{ss}\right)$ |
| 1 | 0.3196 | $-0.0666 - 1.0043G_{ss} + 0.0915G_{pp}\zeta_p$ |
| 1 | 0.3367 | $0.0643 - 1.0544G_{ss} + 0.5456\zeta_p - 0.1636G_{ss}\zeta_p$ |
| 13 | 0.3392 | $-1.0155G_{ss} + 0.2809\zeta_p$ |

*continued on next page*

*continued from previous page*

| **# runs** | **RMSE** | **GP-regressed equation** |
|---|---|---|
| 1 | 0.3485 | $-0.0089 - 0.6892G_{p_2} + 0.373G_{p_2}G_{pp}$ |
| 7 | 0.3533 | $-0.9695G_{ss} + 0.247\zeta_s$ |
| 1 | 0.3563 | $-0.0571 - 1.002G_{ss} + 0.0363\frac{G_{ss}G_{pp}}{G_{sp}}$ |
| 1 | 0.3580 | $-0.0611 - 0.8703G_{ss} + 0.7764G_{sp}\zeta_p$ |
| 3 | 0.3615 | $0.2601G_{pp} - 0.9012G_{p_2}$ |
| 1 | 0.4182 | $0.0385 - 0.9478G_{ss} + 0.0433G_{ss}^{-1}$ |
| 1 | 0.4256 | $-0.9042G_{ss}$ |

Table B.5: $\zeta_s = f\left(U_{ss}, U_{pp}, \beta_s, \beta_p, \zeta_p, G_{ss}, G_{sp}, G_{pp}, G_{p_2}, H_{sp}\right)$

| # runs | RMSE | GP-regressed equation |
|---|---|---|
| 1 | 0.3509 | $-0.0783 + G_{pp}\left[0.3312G_{pp} + 0.1028G_{ss} + 0.4088H_{sp} + U_{ss}\left(0.9397G_{pp} + 0.4543G_{ss} + 0.6618H_{sp}\right) + U_{ss}^2\left(0.9712G_{pp} + 0.7137G_{ss} + 0.2808H_{sp} - 2.0981\zeta_p\right)\right] - U_{ss}^2\zeta_p\left(0.8819G_{ss} - 0.2315H_{sp}\right) + G_{pp}G_{ss}^{-1}\left[0.0163G_{pp} - G_{pp}U_{ss}\left(0.1633 - 1.0354\zeta_p^3U_{ss}\right) - G_{pp}^2\zeta_p\left(0.3893\zeta_p + 0.4220G_{pp}\right) + G_{pp}^3\zeta_pU_{ss}\left(0.1347 + 2.6003U_{ss}\right) + G_{pp}^2\zeta_p^2U_{ss}\left(3.4145 - 7.1298U_{ss}\right) + U_{ss}^2\left(0.5421G_{pp} - 0.7414\zeta_p\right)\right]$ |
| 1 | 0.4308 | $-0.3051 + 0.2467\zeta_p - 0.0298G_{pp} - \left(G_{pp}G_{sp}\right)^{-1}\left(2.186G_{sp} + 0.0516\beta_s\right)\left[0.2701 + 0.0536G_{pp} - G_{pp}\left(0.31G_{pp} - 0.411G_{sp}\right) + \zeta_pG_{pp}\left(0.0997G_{pp} - 0.2970G_{ss}\right)\right]\left(0.0076\zeta_p + 0.2111G_{pp} - 0.0057G_{pp}e^{7.3345\beta_p}\right)$ |
| 1 | 0.4313 | $0.2173 + H_{sp}\zeta_p\left(0.1084 - 0.6765\beta_p + 0.2738\beta_p^{-1}\right) - 0.0512H_{sp}^2\beta_p^{-1} - \zeta_p^2\left(4.3158 + 0.9807\beta_p^{-1}\right) - 1.6925\beta_p\zeta_p$ |
| 1 | 0.4499 | $-0.2216 - U_{ss}G_{pp}\left[0.1978 - 0.4456\zeta_p + 0.1735U_{ss} + 0.0020e^{6.4046U_{pp}} - \zeta_p^2U_{pp}H_{sp}^3U_{ss}\left(0.0699G_{p_2} + 0.0286G_{ss}H_{sp}\right)\left(3.8971U_{ss}^2 - 1.4948\beta_p^2 + 1.8337Uss\beta_p\right)\left(0.0324U_{pp} - 0.8219H_{sp}U_{ss}^2 + 0.5941H_{sp}\beta_p^2 - 0.1601H_{sp}\beta_pU_{ss}\right)\right]$ |
| 1 | 0.4510 | $-0.3760 - 0.7127\zeta_p - 0.1488G_{pp} - \beta_s\left(0.6861U_{ss} + 0.4178U_{pp}\right) + G_{ss}\left(0.3334U_{ss} - 0.1773U_{pp}\right) + \zeta_pH_{sp}U_{ss}\left[1.9611 - \beta_p\left(5.7965 + 0.2908U_{ss}\right) + \zeta_p\left(0.9280G_{ss} + 12.4121U_{pp}\right)\right] - \zeta_pH_{sp}U_{pp}\left[1.9636 - \beta_p\left(3.1087 + 0.9108U_{ss}\right) + \zeta_p\left(8.2983G_{ss} + 12.7249U_{pp}\right)\right]$ |
| 1 | 0.4679 | $-0.2326 + 0.6816G_{pp}\zeta_pU_{ss}e^{2.5779\beta_p}$ |
| 1 | 0.4699 | $0.2193 + 0.1925\beta_p - 0.1700\beta_p^{-1} - 0.0229G_{pp}\zeta_p\beta_p^{-1} + 0.0056\zeta_p\beta_p^{-2}e^{1.1456U_{ss}}$ |
| 1 | 0.4773 | $0.3340 - 0.4057H_{sp} - H_{sp}^2\left(0.9850 + 1.4241\zeta_p + 0.0520\beta_s\right) - \zeta_p^2\left[5.5133 - H_{sp}\left(12.2195 + 0.3065G_{p_2}\right) - H_{sp}^2\left(2.6545 - 11.3926H_{sp}\right)\right] + \zeta_p^3H_{sp}\left(14.3453 - 22.5559H_{sp}^2\right) - H_{sp}^4\left(0.0125G_{ss} + 0.0675H_{sp}\zeta_p\right) - H_{sp}\zeta_p^4G_{sp}\left(4.2547 - 5.2721H_{sp}\right)$ |
| 1 | 0.4820 | $-0.2974 - 0.1630G_{pp}e^{1.5318U_{ss}} + 0.1352G_{pp}^2e^{1.9126U_{ss}}$ |
| 1 | 0.4841 | $-0.1971 + 0.8483G_{pp}U_{ss}\zeta_pe^{1.734U_{pp}}$ |

*continued on next page*

*continued from previous page*

| # runs | RMSE | GP-regressed equation |
|---|---|---|
| 1 | 0.4882 | $-0.1854 + U_{ss}G_{pp}\left[1.8559\zeta_p - 0.4018U_{pp} + \beta_s\zeta_p H_{sp}^2\left(0.1005 - 0.3284\zeta_p\right)\right]$ |
| 1 | 0.4915 | $0.1911 - H_{sp}\zeta_p\left(0.6996 + 0.1587\beta_p - 0.3327\beta_p^{-1}\right) - 0.0648H_{sp}^2\beta_p^{-1} - \zeta_p^2\left(2.6389 + 0.8117\beta_p + 0.6428\beta_p^{-1}\right)$ |
| 1 | 0.5005 | $-0.1507 + 1.4415H_{sp}\zeta_p + 1.2889H_{sp}\zeta_p U_{pp}G_{ss}^{-1}$ |
| 1 | 0.5144 | $-0.1404 + 1.6936H_{sp}\zeta_p e^{3.2308U_{pp}G_{p_2}}$ |
| 1 | 0.5153 | $-0.1448 + U_{pp}\left(0.9719\beta_p + 1.1958G_{ss}\right) + U_{ss}\zeta_p\left(0.0031\beta_p + 0.9959G_{ss}\right)$ |
| 1 | 0.5222 | $-0.1461 + 0.1827G_{ss}G_{pp}e^{2.1006U_{ss}}$ |
| 1 | 0.5336 | $-0.1774 + 0.9960H_{sp}\zeta_p + 0.3795G_{ss} - 0.5962G_{p_2}$ |
| 1 | 0.5349 | $-0.3087 + 1.4135H_{sp}\zeta_p - 0.4746G_{p_2} + 0.0424\zeta_p - 0.0878G_{p_2}U_{pp} - G_{ss}G_{pp}\left(0.0034 + 0.0809\zeta_p - 0.0323G_{ss}G_{pp}\right)$ |
| 1 | 0.5463 | $-0.1949 - \zeta_p\left[0.1618U_{pp} + \zeta_p\left(0.2244 + 0.0164\beta_p\right) + {\beta_p}^{-1}\left(0.1065\beta_s + 0.2785H_{sp}\right)\right]$ |
| 2 | 0.5470 | $-0.2019 + 1.1338H_{sp}\zeta_p - 0.2861G_{p_2}$ |
| 2 | 0.5475 | $-0.1831 - 0.2888G_{pp} + 0.4909G_{pp}\zeta_p U_{pp}$ |
| 2 | 0.5516 | $-0.2560 + 1.2820H_{sp}^2\zeta_p\beta_s e^{1.8376\beta_p}$ |
| 3 | 0.5519 | $0.0765 - 0.1125\beta_p^{-1} + 0.7837H_{sp}\zeta_p$ |
| 1 | 0.5522 | $-0.1237 + \zeta_p\beta_s\beta_p^{-1}\left(0.8325U_{pp} + 0.8226G_{p_2}\right)$ |
| 1 | 0.5550 | $0.0286 - 0.5555\zeta_p G_{p_2}\beta_p^{-1}$ |
| 5 | 0.5554 | $-0.1770 + H_{sp}^2\zeta_p\left(1.0631G_{ss} - 1.0208G_{p_2}\right)$ |
| 1 | 0.5554 | $-0.1553 + 0.1694U_{ss}G_{pp}^2 e^{1.4343H_{sp}}$ |
| 2 | 0.5571 | $-0.1976 + H_{sp}^2\zeta_p\left(0.9463\beta_p + 0.9819G_{ss}\right)$ |
| 1 | 0.5616 | $-0.1515 + 0.0484H_{sp} + 0.9275H_{sp}^2\zeta_p + 0.5210H_{sp}^3\zeta_p$ |
| 1 | 0.5621 | $-0.1025 + 1.1181\zeta_p - 0.2140H_{sp} + 0.4638\beta_p - 0.4455H_{sp}^2\zeta_p - 0.2043H_{sp}\zeta_p\beta_p$ |
| 1 | 0.5646 | $-0.1731 + H_{sp}^2\zeta_p\left(1.0482G_{ss} + 0.981U_{pp}\right)$ |

*continued on next page*

*continued from previous page*

| # runs | RMSE | GP-regressed equation |
|---|---|---|
| 2 | 0.5658 | $-0.1963 + 1.1022 H_{sp}\zeta_p + 0.2383\beta_p$ |
| 1 | 0.5665 | $-0.1207 + \zeta_p \left(0.7940 G_{pp} U_{ss} - 0.0234\zeta_p H_{sp}^2 G_{pp}^2 - 0.2556 G_{sp} e^{-0.0354\beta_s}\right)$ |
| 3 | 0.5682 | $-0.1017 - U_{ss} G_{pp} (0.1975 - 0.4235\zeta_p)$ |
| 1 | 0.5687 | $-0.1845 - 0.2723 G_{pp} + 0.3308 G_{pp}\zeta_p + 0.3364 G_{pp}\zeta_p\beta_p U_{ss}^2$ |
| 1 | 0.5706 | $-0.1230 - G_{pp} [0.3044 U_{ss} - \zeta_p (0.3478 - 0.0187 G_{pp})]$ |
| 1 | 0.5709 | $-0.1190 + G_{pp} (0.2813\zeta_p - 0.3007 U_{ss})$ |
| 1 | 0.5710 | $-0.1085 + 1.2458 H_{sp}\zeta_p - 0.3943 H_{sp}^2 \zeta_p U_{ss}$ |
| 1 | 0.5743 | $-0.0954 + U_{ss} (0.3149 + 1.101\zeta_p) + Uss.^2 (0.7893 G_{ss} - 0.7890 G_{p_2})$ |
| 1 | 0.5744 | $-0.2577 + H_{sp}\zeta_p (0.2966\zeta_p - 0.3675 H_{sp})$ |
| 1 | 0.5744 | $-0.0696 + G_{sp} (0.5661 G_{p_2} - 0.6368 G_{ss}) - G_{pp} (0.8743 G_{p_2} - 0.8853 G_{ss})$ |
| 2 | 0.5755 | $-0.2485 - H_{sp}^2 \zeta_p (0.2329 - 0.3439\zeta_p)$ |
| 2 | 0.5762 | $-0.1842 + 0.151\zeta_p - 0.2421 G_{pp} + 0.2532\zeta_p G_{pp}$ |
| 1 | 0.5785 | $-0.2071 + \zeta_p (0.8795 G_{ss} - 0.6322 G_{p_2}) - G_{ss} (0.7925 G_{ss} - 0.8138 G_{p_2})$ |
| 1 | 0.5791 | $-0.1105 + 1.01 H_{sp}\zeta_p + 0.1299 H_{sp} G_{ss}$ |
| 1 | 0.5815 | $-0.1037 - 0.1937 U_{pp} + 0.4948 G_{pp}^2 U_{ss}$ |
| 1 | 0.5816 | $-0.1381 + \beta_s\zeta_p (0.6772 U_{ss} + 0.5709 G_{sp})$ |
| **16** | **0.5817** | $\mathbf{-0.1453 + 0.3896\zeta_p G_{pp} U_{ss}}$ |
| 1 | 0.5833 | $-0.1883 + 0.3164 H_{sp}\zeta_p + 0.2074 H_{sp}^2 \zeta_p^2$ |
| 1 | 0.5873 | $0.1684 - 0.2601\beta_p^{-1} - 0.2478\beta_s\beta_p^{-1}$ |
| 3 | 0.5883 | $-0.1860 + 0.2924 H_{sp}^2 \zeta_p^2$ |
| 1 | 0.5954 | $-0.1480 + 0.1873 G_{pp}\beta_s H_{sp}\zeta_p$ |

*continued on next page*

*continued from previous page*

| **# runs** | **RMSE** | **GP-regressed equation** |
|---|---|---|
| 2 | 0.5964 | $-0.1855 + 0.1290H_{sp} + 1.0414H_{sp}\zeta_p$ |
| 1 | 0.5970 | $-0.1080 - G_{pp}\left(0.2164U_{ss} - 0.3329H_{sp}\zeta_p\right)$ |
| 6 | 0.5974 | $-0.1613 + U_{ss}\zeta_p\left(0.6985U_{pp} + 1.0158G_{ss}\right)$ |
| 1 | 0.5976 | $-0.1625 + 0.0484H_{sp}\zeta_p^3$ |
| 8 | 0.5977 | $-0.1817 + 0.1829\zeta_p^2$ |
| 1 | 0.5979 | $-0.1631 + 0.0385\zeta_p^3$ |
| 1 | 0.5991 | $-0.1778 + 0.2444\zeta_p G_{pp}$ |
| 1 | 0.5998 | $-0.1937 - \zeta_p\left(0.1585 - 0.2726H_{sp}\zeta_p\right)$ |
| 1 | 0.6011 | $-0.1625 + 0.2286H_{sp}\zeta_p^2$ |
| 1 | 0.6042 | $-0.0936 + 0.9957H_{sp}\zeta_p\beta_s G_{ss}^{-1}$ |
| 2 | 0.6047 | $-0.0346 + G_{pp}\left(0.7887G_{ss} - 0.6914G_{p_2}\right)$ |
| 1 | 0.6090 | $H_{sp}\zeta_p e^{-e^{\zeta_p}}$ |
| 2 | 0.6098 | $-0.1806 + 1.0141H_{sp}\zeta_p$ |

Table B.6: $\zeta_p = f\left(U_{ss}, U_{pp}, \beta_s, \beta_p, \zeta_s, G_{ss}, G_{sp}, G_{pp}, G_{p_2}, H_{sp}\right)$

| # runs | RMSE | GP-regressed equation |
|---|---|---|
| 1 | 0.0704 | $0.0746 + 0.0752\zeta_s - 0.0474\beta_s + G_{ss}\left[0.4924 + 0.2758\beta_p + 0.0433U_{ss} - \beta_s\left(0.0649\beta_p + 0.0428\beta_s\right)\right] +$ $\zeta_s\left[\beta_s\left(0.2600 + 0.1248\beta_s + 0.1342U_{ss} - \beta_p\left(0.3548 - 0.2379\beta_p G_{p_2}\right)\right) + \beta_p^2\left(0.2447G_{ss} + 0.5103G_{pp}\right) +\right.$ $\left.U_{ss}H_{sp}G_{pp}\left(0.0987 - 0.0069\frac{U_{ss}}{G_{ss}}\right)\right] + \zeta_s^2\left[0.0270 - 0.0062G_{sp}G_{ss} +\right.$ $\left.\beta_p^2\left(0.1845U_{pp} - 0.3581\beta_s + 0.1200\beta_s^2 G_{ss}\right)\right] - \zeta_s^3\beta_s\left(0.0196 - 0.0293G_{sp}\right)$ |
| 1 | 0.0893 | $-0.2364 - 0.0306\zeta_s - 0.6070\beta_s + 0.0646G_{ss} + 0.1001U_{pp}G_{ss} - \beta_s\left[G_{ss}\left(0.5942 - 0.4178U_{pp} - 0.920G_{ss}\right) +\right.$ $\left.G_{p_2}\left(0.0534 + 0.3628U_{pp} + 0.3820G_{ss}\right)\right] + \zeta_s\left[0.0499U_{pp} - \beta_s\left(0.1348 + 0.0124U_{pp}\right)\right] +$ $\zeta_s^2\left(0.2018 + 0.1810\beta_s + 0.0807G_{p_2}\right) - \zeta_s^3\left[0.0304 - \beta_s\left(0.1746 + 0.0568G_{p_2}\right)\right] - 0.0513\zeta_s^4\beta_s$ |
| 1 | 0.1264 | $-0.14 - 0.6498\zeta_s + 0.1178\beta_s - 5.2109\times10^{-4}G_{ss} - 0.0101G_{pp} - 0.0451\zeta_s\beta_s + U_{ss}\left[0.1261G_{ss} -\right.$ $\left.0.0622\beta_s + 0.0844G_{ss}\beta_s\right] - \zeta_s G_{ss}\left[0.7147 - 0.5828G_{ss} - \beta_s\left(0.0765 + 0.0708\zeta_s\beta_s\right)\right] + 0.1368G_{ss}e^{-0.0887U_{pp}}$ |
| 1 | 0.1463 | $-0.2514 + \beta_s\left(0.5902G_{ss} - 0.5921G_{p_2}\right) + 0.0899\zeta_s^2\beta_s^2$ |
| 1 | 0.1525 | $-0.1457 + 0.3075\beta_s + \beta_s\zeta_s\left(0.37 - 0.4125G_{pp}\right) + \beta_s^2\left(0.1226U_s - 0.0312\zeta_s\right) +$ $\beta_s^2\zeta_s^5U_{pp}\left(0.0022G_{pp} - 4.2134\times10^{-4}\zeta_s + 4.8523\times10^{-5}\beta_s\zeta_s\right)$ |
| 1 | 0.1654 | $-0.1906 + 0.057\zeta_s + 0.0438\beta_s - 0.0973G_{pp} + \beta_s\left[0.0629G_{ss} + \zeta_s\left(0.5529 - 0.1251\zeta_s + 0.0905\zeta_s G_{ss}\right)\right] -$ $0.0351\zeta_s\beta_p\left(0.2687 + 0.3297\zeta_s\right)^{-1}$ |
| 1 | 0.1834 | $-0.1065 + G_{ss}\left[0.2278 + 0.0316\zeta_s + 0.0354\beta_s + 0.1974U_{ss} + \beta_s\zeta_s\left(0.0982 + 0.0712\zeta_s\right)\right]$ |
| 1 | 0.1911 | $-0.1846 + \zeta_s\beta_s G_{p_2}\left(0.3385 + 0.3202\zeta_s\beta_s\right)$ |
| 1 | 0.1990 | $-0.1770 - 0.0397\zeta_s + 0.0888\beta_s + \zeta_s G_{ss}\left[-0.0243\zeta_s + \beta_s\left(0.2406 + 0.0168G_{ss} - 0.0491\zeta_s^2 + 0.1578\beta_s\zeta_s\right)\right]$ |
| 1 | 0.2011 | $-0.1927 + 0.0044\zeta_s + 0.1099\beta_s + \zeta_s\beta_s\left[0.1871 - 0.0464\beta_s + G_{ss}\left(0.1925 - \zeta_s\beta_s\left(0.0898 - 0.0727\beta_s\right)\right)\right]$ |
| 1 | 0.2043 | $-0.0595 - 0.0712\zeta_s + 0.1877\beta_s + 0.0536G_{pp} + \zeta_s\beta_s\left(0.0647 + 0.0137\zeta_s + 0.1307G_{p_2}\right) -$ $U_{ss}G_{pp}\left[0.1011 - \zeta_s\beta_s\left(0.1857 + 0.0124\zeta_s\beta_s\right)\right] - 0.067e^{0.9234U_{pp}}$ |

*continued on next page*

*continued from previous page*

| **# runs** | **RMSE** | **GP-regressed equation** |
|---|---|---|
| 1 | 0.2078 | $-0.2138 + \zeta_s^2 \beta_s^2 G_{ss} G_{p_2} (0.175 G_{ss} - 0.173 G_{p_2})$ |
| 1 | 0.2136 | $-0.2106 + \zeta_s \beta_s^2 G_{p_2} (0.2283 + 0.1882 \zeta_s \beta_s)$ |
| 1 | 0.2175 | $-0.2087 + \zeta_s^2 \beta_s^2 G_{p_2} (0.2827 G_{ss} - 0.269 G_{p_2})$ |
| 1 | 0.2213 | $-0.2056 + 0.1148 \beta_s + 0.1899 \zeta_s \beta_s + 0.0854 \zeta_s^2 \beta_s^2$ |
| 1 | 0.2256 | $-0.2171 + 0.1340 \zeta_s \beta_s + 0.0370 \zeta_s^2 \beta_s^2 G_{ss}$ |
| 1 | 0.2320 | $-0.1134 + 0.1454 \zeta_s^2 \beta_s + 0.0655 \zeta_s \beta_s^2 G_{ss} + 0.0141 \beta_s G_{ss} (0.2174 \beta_s + 0.2561 \zeta_s \beta_s)^{-1}$ |
| 1 | 0.2332 | $-0.1821 + 0.1902 \zeta_s^2 \beta_s - \beta_s^2 U_{pp} [0.0412 + G_{ss} (0.0003 - \zeta_s (0.24 - 0.149 G_{ss}))]$ |
| 1 | 0.2350 | $-0.1996 + \zeta_s \beta_s^2 G_{ss} (0.0227 + 0.0675 \zeta_s)$ |
| 1 | 0.2354 | $-0.1969 + \zeta_s \beta_s G_{ss} (0.0560 + 0.0645 \zeta_s \beta_s)$ |
| 1 | 0.2389 | $-0.2118 + 0.0727 \zeta_s^2 \beta_s^3$ |
| 2 | 0.2390 | $-0.2069 + 0.0732 \zeta_s^2 \beta_s^2 G_{ss}$ |
| 1 | 0.2403 | $-0.2264 - 0.0954 \zeta_s G_{pp} H_{sp} + 0.1451 \zeta_s^2 \beta_s^2$ |
| 2 | 0.2405 | $-0.2248 - 0.0322 \zeta_s + 0.0798 \zeta_s \beta_s + 0.1107 \zeta_s^2 \beta_s^2$ |
| 1 | 0.2411 | $-0.2183 + 0.0963 \zeta_s \beta_s + 0.1049 \zeta_s^2 \beta_s^2$ |
| 1 | 0.2422 | $-0.2194 - 0.0384 \zeta_s \beta_p + \zeta_s^2 \beta_s (0.0316 + 0.1036 \beta_s)$ |
| 1 | 0.2432 | $-0.2331 - 0.0626 \zeta_s + 0.126 \zeta_s^2 \beta_s^2$ |
| 1 | 0.2437 | $-0.2083 + \zeta_s^2 \beta_s (0.0498 + 0.0892 \beta_s)$ |
| 6 | 0.2460 | $-0.2216 + 0.1197 \zeta_s^2 \beta_s^2$ |
| 1 | 0.2369 | $-0.1495 + 0.1283 \zeta_s + 0.1331 \beta_s + 0.2541 \zeta_s \beta_s + 0.0991 \zeta_s^2 \beta_s$ |
| 1 | 0.2482 | $-0.1768 + 0.1926 \frac{\zeta_s^2 \beta_s^2}{G_{ss}}$ |
| 1 | 0.2532 | $-0.1873 + 0.2402 \zeta_s + 0.2059 \beta_s + \zeta_s \left[0.1112 G_{ss} + \beta_s \left(0.4703 - 0.0233 G_{ss}^2\right)\right] + G_{ss} \left(0.0192 \beta_p + 0.1091 G_{ss} e^{0.6459 U_{pp}}\right)$ |

*continued on next page*

*continued from previous page*

| **# runs** | **RMSE** | **GP-regressed equation** |
|---|---|---|
| 1 | 0.2540 | $-0.1619 + 0.1552\zeta_s^2\beta_s + 0.092\zeta_s G_{ss}^2$ |
| 2 | 0.2573 | $-0.1572 + G_{ss}\zeta_s\left(0.1551\zeta_s + 0.0919\beta_s\right)$ |
| 1 | 0.2585 | $-0.1344 + 0.2219\beta_s + 0.2365\zeta_s + 0.5381\beta_s\zeta_s$ |
| 8 | 0.2592 | $-0.1588 + 0.0628\zeta_s + 0.1831\zeta_s^2\beta_s$ |
| 2 | 0.2605 | $-0.1459 + 0.2329\zeta_s^2 G_{ss} e^{0.424U_{pp}}$ |
| 2 | 0.2610 | $-0.1581 + 0.2143\zeta_s^2\beta_s e^{0.2396U_{pp}}$ |
| **52** | **0.2623** | $\mathbf{-0.1671 + 0.1926\zeta_s^2\beta_s}$ |
| 1 | 0.2623 | $-0.1889 + \zeta_s^2\beta_s U_{ss}\left(0.0796 + 0.1395G_{ss}\right)$ |
| 1 | 0.2654 | $-0.1918 + 0.0338\zeta_s^2\beta_s G_{pp}G_{ss}$ |
| 2 | 0.2667 | $-0.1657 + 0.1928\zeta_s^2 G_{ss}$ |
| 1 | 0.2805 | $-0.1045 + 0.252\beta_s + 0.4034\beta_s\zeta_s G_{ss}$ |

Table B.7: $G_{ss} = f\left(U_{ss}, U_{pp}, \beta_s, \beta_p, \zeta_s, \zeta_p, G_{sp}, G_{pp}, G_{p_2}, H_{sp}\right)$

| # runs | RMSE | GP-regressed equation |
|---|---|---|
| 1 | 0.0860 | $-0.3153 + 0.5160G_{p_2} + 0.2783\zeta_p + 0.1033\beta_s + \zeta_p\left(0.1698G_{sp} + 0.0961\beta_s\zeta_p\right) + G_{p_2}\left(0.1193\beta_s + 0.1723G_{p_2}\right) + G_{p_2}\zeta_p\left(0.4451\beta_s - 0.6046\zeta_p - 0.4878G_{p_2}\right)$ |
| 1 | 0.1555 | $-0.3931 + 0.6119G_{p_2} + \zeta_p^2 G_{p_2}\left(4.1131G_{p_2} + 0.5790\zeta_p\right) - \zeta_p^2 G_{p_2}^2\left(0.6733G_{p_2} + 2.4695\zeta_p\right)$ |
| 1 | 0.1759 | $-0.2657 + 0.5126G_{p_2} + 0.3595\zeta_p - \beta_p\left(0.2970G_{p_2} + 0.0143\zeta_p\right)$ |
| 1 | 0.1787 | $-0.0549 + 0.9090G_{p_2} + \zeta_s\left[0.1763G_{p_2} + 0.1228\beta_p - 0.2161\beta_s + 0.1503U_{pp} + 0.2838e^{1.1118G_{p_2}+0.2156\beta_p} + G_{p_2}\zeta_s\beta_s\left(G_{p_2}\left(1.8636 - 0.5613e^{0.3458G_{p_2}-0.5198\beta_p}\right) - \beta_p\left(0.9468 - 0.0053\beta_p e^{-4.0626G_{p_2}+3.1266\beta_p}\right) + H_{sp}\left(0.5589 - 0.3036e^{1.5253G_{p_2}+0.9429\beta_p}\right)\right)\right]$ |
| 1 | 0.1971 | $-0.0004 + 0.5312G_{p_2} - 0.2921\beta_p + 0.1971\zeta_s + 0.2102\beta_s - 0.0051U_{ss} + 0.1440G_{p_2}\zeta_s$ |
| 1 | 0.2023 | $-0.0652 + 0.4622G_{p_2} + 0.1467\zeta_p + 0.1528\beta_s - G_{p_2}\zeta_p\left(0.9806 - 0.9016G_{p_2}\zeta_p\right)$ |
| 1 | 0.2093 | $0.8444G_{p_2} + 0.5314\zeta_s - 0.3442G_{pp}$ |
| 1 | 0.2319 | $0.1321 + 0.7341G_{p_2} + 0.2410\beta_s - 0.0690\beta_p^{-1} + 0.1832G_{p_2}\zeta_s$ |
| 2 | 0.2374 | $0.4624G_{p_2} + 0.2768\zeta_p - 0.4569\beta_p$ |
| 2 | 0.2385 | $-0.1418 - 0.8758\beta_p + 0.1428G_{pp}^2$ |
| 1 | 0.2508 | $-0.0619 + 0.7406G_{p_2} + 0.2475\zeta_p + 0.0985\beta_s + 0.1015G_{p_2}\beta_s$ |
| 4 | 0.2520 | $0.7447G_{p_2} + 0.1953\zeta_s + 0.2677\beta_s$ |
| 4 | 0.2537 | $0.5113G_{p_2} - 0.4442\beta_p + 0.2590\zeta_s$ |
| 1 | 0.2560 | $-0.2152 - 0.9184\beta_p + 0.1432e^{0.7138G_{pp}}$ |
| 1 | 0.2571 | $-0.0682 + 0.6317G_{p_2} - 0.4032\beta_p - 0.5116U_{pp}\zeta_p$ |
| 1 | 0.2592 | $0.7328G_{p_2} + 0.1944\zeta_p + 0.2442\beta_s$ |
| 2 | 0.2829 | $0.2229G_{pp} - 0.9559\beta_p + 0.1532\zeta_s$ |

*continued on next page*

*continued from previous page*

| # runs | RMSE | GP-regressed equation |
|---|---|---|
| 1 | 0.2837 | $-0.0675 - 0.8697\beta_p + 0.2939G_{p_2}\zeta_p G_{pp}$ |
| 1 | 0.2854 | $-0.1015 + 0.7545G_{p_2} + U_{ss}\left(0.0540 + 0.1366H_{sp} + 0.4636\zeta_p\right)$ |
| 1 | 0.2863 | $-0.0730 + 0.6395G_{p_2} - 0.4007\beta_p + 0.5320G_{p_2}\zeta_p$ |
| 3 | 0.2896 | $0.4794G_{p_2} - 0.2797\beta_p + 0.2929\beta_s$ |
| 2 | 0.2904 | $-0.0753 + 0.8991G_{p_2} + 0.1843\beta_s\zeta_p$ |
| 1 | 0.2991 | $-0.0475 + 0.8914G_{p_2} + 0.1041G_{pp}\zeta_s$ |
| 2 | 0.3027 | $-0.0840 + 0.9438G_{p_2}e^{0.3625\zeta_p}$ |
| 1 | 0.3052 | $-0.0640 + 0.8884G_{p_2} + 0.0645\zeta_p^2$ |
| **22** | **0.3124** | $\mathbf{0.6971G_{p_2} + 0.346}\boldsymbol{\beta}_\mathbf{s}$ |
| 14 | 0.3125 | $0.8714G_{p_2} + 0.2756\zeta_p$ |
| 1 | 0.3127 | $0.2630\zeta_s + 0.9527G_{p_2} - 0.0872G_{sp}$ |
| 2 | 0.3131 | $-0.0724 + 0.9977G_{p_2} - 0.5430U_{pp}\zeta_p$ |
| 9 | 0.3131 | $0.2978G_{pp} - 0.9752\beta_p$ |
| 12 | 0.3138 | $0.2907\zeta_p - 0.869\beta_p$ |
| 1 | 0.3205 | $-0.0457 + 0.8768G_{p_2}e^{0.2032\beta_s\zeta_p}$ |
| 10 | 0.3216 | $0.9088G_{p_2} + 0.2622\zeta_s$ |
| 2 | 0.3407 | $0.265\zeta_s - 0.9018\beta_p$ |

Table B.8: $G_{sp} = f\left(U_{ss}, U_{pp}, \beta_s, \beta_p, \zeta_s, \zeta_p, G_{ss}, G_{pp}, G_{p_2}, H_{sp}\right)$

| # runs | RMSE | GP-regressed equation |
|---|---|---|
| 1 | 0.1396 | $-0.2749 + 1.5920\beta_p - 2.5377U_{pp} - 0.2799\zeta_s + \beta_p\left[1.4329\beta_p - \zeta_s\left(1.6518 + 0.0433U_{pp}^{-1}\right) - \zeta_s G_{p_2}\left(5.7581 + 0.1967U_{pp}^{-1}\right)\right] - U_{pp}\left(2.6077U_{pp} + 0.5373\beta_s + 0.1010\zeta_s - 1.7178\zeta_s G_{p_2}\right) - \zeta_s\left(0.1566\beta_s - 0.3663G_{p_2}\right) - \beta_p^2\left[3.1601\beta_p + 2.7403G_{p_2} - 3.4819\beta_p G_{p_2} - \zeta_s U_{pp}^{-1}\left(0.9097 + 2.2878G_{p_2}\right)\right] + U_{pp}^2\left[1.7884U_{pp} + 1.2482G_{ss} - 1.5156\beta_s - 1.0100G_{p_2} + U_{ss}\left(0.1188 + 0.4173G_{p_2} - 1.0016\beta_s\right)\right] + U_{pp}^3\left(3.6625U_{pp} + 1.6990G_{p_2} + 0.5022U_{ss} + 2.4334U_{ss}G_{p_2}\right) - \beta_p U_{pp}\left[0.3113 - 2.7812\beta_p + 1.4930U_{pp} - 2.4548\beta_s + 9.9139U_{pp}^2 - G_{p_2}\left(0.9051 - 4.9944\beta_p - 6.9695U_{pp}\right) - U_{ss}\left(2.4496U_{ss} + 0.9780\beta_p + 0.8578U_{pp}\right) - U_{ss}G_{p_2}\left(4.3962 + 5.3784\beta_p - 6.1522U_{pp}\right)\right]$ |
| 1 | 0.1799 | $0.0579 + 0.7813\beta_p - 1.6868U_{pp} - \left[0.0771 + 0.8769U_{pp} - 0.6662U_{ss} - 0.7230G_{pp} + U_{pp}\left(0.1036U_{pp} - 0.1452U_{ss} + 0.2354G_{pp}\right)\right]\left[0.2750\beta_p + 2.6870U_{pp} - 0.7534U_{ss} + 3.6732G_{pp} + \zeta_p G_{pp}U_{ss}\left(9.8001\beta_p - 4.2388U_{pp}\right)\right]\left[0.0861U_{ss} + G_{p_2}\left(0.1033U_{ss} + 0.7956\beta_p + 0.4161\beta_s + U_{pp}\left(1.0148\beta_p + 1.2660\beta_s\right) + U_{ss}\left(0.4416\beta_p + 0.1806\beta_s\right) + U_{pp}U_{ss}\left(0.8478\beta_p + 0.5110\beta_s\right) + \beta_s\left(0.1583\beta_p + 0.0036\beta_s\right) + \beta_s U_{pp}\left(0.2628\beta_p - 0.0890\beta_s\right)\right)\right]$ |
| 1 | 0.1908 | $1.0916 - 1.1651\beta_p - 1.5853U_{pp} + \beta_p^2\left(1.3074 - 0.3417\beta_s^2\right) - U_{pp}^2\left(0.7635 - 0.4814U_{pp} + 0.7077U_{ss}\right) - \beta_p U_{pp}\left[0.5839 - 0.8092\beta_p - 0.4544U_{pp} - 0.5203U_{ss} + \beta_s^2\left(0.6082\beta_p - 0.4016U_{pp}\right)\right] + \left[0.5737\beta_p - 0.9393U_{pp} - 0.9789G_{ss} + U_{pp}\left(0.0939\beta_p - 0.7562U_{pp} - 1.1189G_{ss}\right)\right]\left[1.6935 - 2.1970\beta_p - 0.8657U_{pp} + 1.3896U_{ss} + \beta_p\left(2.3274\beta_p - 2.4362U_{ss}\right) - U_{pp}\left(0.0492U_{pp} + 0.3254U_{ss}\right) + U_{pp}^2\left(1.8134U_{pp} - 1.2473U_{ss}\right) + \beta_p U_{pp}\left(1.4141 + 1.9058\beta_p - 3.1806U_{pp} - 0.0276U_{ss}\right) - 1.4841e^{-0.1328U_{pp}}\right]$ |
| 1 | 0.2156 | $0.2560 + 0.6692\beta_p - 2.4299U_{pp} - 0.3993\beta_s + 0.2703U_{ss} + \beta_p\left(0.4582\beta_p + 1.3736\beta_s - 0.3953U_{ss}\right) - U_{pp}\left(1.0022\beta_s + 0.0478U_{ss} - 0.0251\beta_s^2 - 0.2915\beta_s U_{ss}\right) - \beta_s\left(0.1145\beta_s + 0.1197U_{ss}\right) - U_{pp}^2\left(0.2550 - 0.8084U_{pp} + 1.0729\beta_s + 0.6135U_{ss}\right) + \beta_p U_{pp}\left(0.3801 + 0.8214\beta_p - 1.0367U_{pp} + 1.4385\beta_s + 0.5426U_{ss}\right)$ |

*continued on next page*

*continued from previous page*

| # runs | RMSE | GP-regressed equation |
|---|---|---|
| 1 | 0.2391 | $-0.2602+0.6078\beta_p-1.6607U_{pp}+0.0735U_{ss}-0.0929G_{ss}+0.2176G_{pp}-0.0493\zeta_s-0.3331\zeta_p-$ $G_{p_2}\left(0.0768U_{pp}-0.5286U_{ss}\right)-U_{pp}\left(0.1101\beta_p-0.3854U_{pp}-0.0816U_{ss}+0.4381G_{pp}+0.0592\zeta_s+\right.$ $\left.0.0864H_{sp}\right)+\left(0.0549-0.2047U_{pp}-0.1155G_{pp}+0.0593\zeta_p\right)\left(0.1373+1.2181\beta_p-\right.$ $\left.1.9738U_{pp}+0.4897G_{ss}-0.7853G_{pp}\right)^{-1}$ |
| 1 | 0.2414 | $-0.0722+1.5099\beta_p-1.9176U_{pp}+0.3864\zeta_p-\beta_s\left[0.0697U_{ss}+\zeta_p\left(0.5909\beta_p-0.9159U_{pp}\right)\right]+$ $H_{sp}\left[0.4974\beta_p-0.3496U_{pp}-U_{ss}\left(0.0025-0.3492\beta_p+0.8923U_{pp}\right)-\beta_p^2\left(0.4808+0.7250H_{sp}\right)+\right.$ $\left.U_{pp}^2\left(0.5082-0.6936H_{sp}\right)+\beta_pU_{pp}\left(0.0605-0.0948\beta_p-1.1749U_{pp}+1.1342U_{ss}\right)+0.2171\beta_p^3+1.0435U_{pp}^3\right]$ |
| 1 | 0.2496 | $0.2087+0.6857\beta_p-1.7667U_{pp}+\left(0.2634H_{sp}+0.1834U_{ss}-1.4774G_{pp}\right)\left[0.0691H_{sp}-\right.$ $\left.U_{pp}\left(0.1184\beta_p-0.1255U_{pp}\right)-U_{ss}\left(0.2597\beta_p-0.1258U_{pp}\right)\right]\left[0.6654-0.4861U_{ss}+0.0365U_{pp}-\right.$ $\left.0.1555G_{pp}+0.7354\zeta_pe^{0.4388U_{pp}}-H_{sp}\left(1.9865\beta_p-3.2507U_{pp}\right)-G_{p_2}\left(0.0490\beta_p+0.8418U_{pp}\right)\right]$ |
| 1 | 0.2650 | $0.3158+0.5824\beta_p-1.4330U_{pp}++0.0218G_{pp}+0.1980U_{ss}+0.0576H_{sp}+U_{ss}\left(0.0988\beta_p-0.4242U_{pp}-\right.$ $\left.0.2713G_{pp}-0.0033U_{ss}-0.1932H_{sp}\right)-U_{pp}^2\left(0.0178+0.2391U_{ss}-0.2048U_{ss}^2\right)-$ $U_{pp}\beta_p\left(0.1811+0.0610U_{ss}+0.2065U_{ss}^2\right)$ |
| 1 | 0.2663 | $0.3062+0.5568\beta_p-1.5200U_{pp}+0.0498G_{p_2}+0.0913\zeta_s-U_{pp}\left(0.9533\beta_p-0.5453U_{pp}+0.2069U_{ss}-0.1409H_{sp}\right)+$ $\beta_p\left(0.3022\beta_p-0.0534U_{ss}-0.2030H_{sp}\right)-U_{pp}H_{sp}\left[\zeta_p\left(1.6232\beta_p-0.8166U_{pp}-0.7716U_{ss}-0.2471H_{sp}\right)-\right.$ $\left.\zeta_s\left(0.4036\beta_p-0.3371U_{pp}-0.2588U_{ss}+0.1373H_{sp}\right)\right]$ |
| 1 | 0.2664 | $-0.1311+1.9345\beta_p-1.6611U_{pp}+0.0298U_{ss}+0.0892H_{sp}-0.0624\beta_pH_{sp}-\beta_p^2\left(0.5193+0.4640\beta_p-\right.$ $\left.0.0089H_{sp}-0.7536U_{ss}+0.1294\beta_p^2\right)+U_{pp}^2\left(0.7674-0.0672U_{ss}\right)-\beta_pU_{pp}\left(0.6002-0.2508\beta_p+\right.$ $\left.0.5394U_{pp}+0.7887U_{ss}+0.0035H_{sp}-0.8512\beta_pU_{pp}-0.0628\beta_p^2+0.9465U_{pp}^2\right)$ |
| 1 | 0.2688 | $-0.0844+1.3169\beta_p-1.7438U_{pp}+U_{pp}\left[0.3148\beta_p+1.9964U_{pp}-\beta_pU_{pp}\left(0.9082\beta_p-2.0875U_{pp}\right)-\right.$ $\left.\beta_pH_{sp}\left(0.3817\beta_p-0.0823U_{pp}\right)-U_{pp}^2\left(0.9858U_{pp}+0.4352H_{sp}\right)\right]\left[0.2724U_{pp}-\beta_pU_{pp}\left(0.9056-\right.\right.$ |

*continued on next page*

*continued from previous page*

| **# runs** | **RMSE** | **GP-regressed equation** |
|---|---|---|
| | | $0.4689U_{pp} + 1.4211U_{pp}^2\big) + U_{pp}^2\left(0.8215 - 0.7743U_{pp} - 0.7812U_{pp}G_{p_2} + 0.1141U_{pp}^2\right) -$ $\beta_p^2U_{pp}\left(0.0439 - 1.1252U_{pp}\right) + \beta_p^3\left(0.3631G_{p_2} + 0.0980U_{pp}\right) - \beta_pG_{p_2}U_{pp}\left(0.9266\beta_p - 1.3706U_{pp}\right)\big]$ |
| 1 | 0.2760 | $0.2010 + 0.8001\beta_p - 1.4812U_{pp} - \beta_p^2\left(0.1886 - 0.1273\beta_p - 0.2968U_{ss} + 0.1023G_{p_2}\right) - U_{pp}^2\left(0.0287 + \right.$ $\left.0.1982U_{ss} - 0.0307G_{p_2}\right) - \beta_pU_{pp}\left(0.1637 + 1.0010\beta_p - 0.2624U_{pp} + 0.1783U_{ss} + 0.4798G_{p_2}\right)$ |
| 1 | 0.2821 | $0.1647 + 0.7054\beta_p - 1.3557U_{pp} - U_{pp}\left[1.4271\beta_p + 0.2663U_{ss} - \beta_p\left(0.2318U_{ss} + 0.2702\beta_p\right)\right] +$ $\beta_p\left(0.5290\beta_p + 0.3007U_{ss} + 0.2373\beta_pU_{ss}\right) + U_{pp}^2\left(0.5420 - 0.6656\beta_p - 0.4394U_{ss}\right) + 0.2860U_{pp}^3$ |
| 1 | 0.2895 | $0.3471 + 0.4496\beta_p - 1.6591U_{pp} - \beta_p\left(0.1403\beta_p + 0.2562U_{pp} - 0.0099U_{ss} - 0.1735G_{pp}\right) -$ $\beta_pH_{sp}G_{pp}\left(0.1005 + 0.6858\beta_p + 0.1471U_{pp} - 0.2613G_{pp} + 0.0485\beta_p^2 + 0.2463U_{pp}\beta_p\right)$ |
| 1 | 0.2940 | $0.2046 + 0.6751\beta_p - 1.4385U_{pp} - \left[0.5765U_{pp} + G_{p_2}U_{pp}^2\left(0.0383\beta_p + 0.2697U_{pp} - G_{pp}\left(0.1528\beta_p - 0.3350U_{pp}\right)\right)\right]$ $\left[0.1512U_{pp} - U_{ss}\left(0.3397\beta_p - 0.4232U_{pp}\right)\left(1.6496U_{pp} + G_{p_2}\left(0.3689\beta_p - 1.0484U_{pp}\right)\right)\right.$ $\left.\left(0.6797 + 0.9683U_{pp} - H_{sp}\left(1.3265\beta_p - 1.0574U_{pp}\right)\right)\right]$ |
| 1 | 0.2950 | $0.1673 + 0.6266\beta_p - 1.3349U_{pp} - 0.1217U_{ss} + 0.1681\beta_p^2 + 0.1775U_{pp}^2 - G_{p_2}G_{pp}^{-1}\left(0.0745\beta_p - 0.0424U_{pp}\right) -$ $\beta_pU_{pp}\left[0.5827 + \zeta_p^{-1}\left(0.1180\beta_p - 0.1258U_{pp}\right)\right]$ |
| 1 | 0.3037 | $0.1511 + 0.7549\beta_p - 1.2130U_{pp} + \beta_p^2\left(0.0984 - 0.1108\beta_p\right) + U_{pp}^2\left(0.2106 - 0.7039U_{pp} - 0.0945U_{pp}^3H_{sp}\right) -$ $\beta_pU_{pp}\left[0.7757 + 0.0910\beta_p - 0.6941U_{pp} + U_{pp}^2H_{sp}\left(0.0817\beta_p - 0.1743U_{pp}\right)\right]$ |
| 1 | 0.3066 | $0.1398 + 0.9324\beta_p - 1.7501U_{pp} + \beta_p^2\left(0.0772\beta_p - 0.1877U_{pp} + 0.6140U_{ss}\right) -$ $U_{pp}^2\left(0.1094\beta_p - 0.3160U_{pp} + 0.2600U_{ss}\right) - 0.3831\beta_pU_{pp}U_{ss}$ |
| 1 | 0.3143 | $0.1219 + 0.7176\beta_p - 1.3902U_{pp} - 0.1843U_{ss} + G_{p_2}H_{sp}\left(0.1712 + 0.4003G_{p_2} + 0.4024U_{pp} - 0.0113U_{ss}\right)$ |
| 1 | 0.3168 | $0.1304 + 0.7741\beta_p - 1.3883U_{pp} - U_{ss}\left[0.2107\beta_p + 0.0647U_{pp} - U_{pp}\left(0.3591\beta_p - 0.4104U_{pp}\right)\right]$ |
| 1 | 0.3209 | $0.1969 + 0.8969\beta_p - 1.6301U_{pp} - U_{pp}^2\left(0.0983 + 0.1281U_{pp} + 1.2712\beta_p + 0.2410G_{ss}\right) -$ $\beta_p^2\left(0.1651\beta_p - 1.2694U_{pp} - 0.4931G_{ss}\right) - 0.6244\beta_pU_{pp}G_{ss}$ |

*continued on next page*

*continued from previous page*

| # runs | RMSE | GP-regressed equation |
|---|---|---|
| 1 | 0.3220 | $0.1781 + 0.7433\beta_p - 1.5412U_{pp} - \beta_p U_{pp}^2 (1.2666 + 2.5902U_{pp}H_{sp}) +$ $\beta_p^2 U_{pp} \left(1.3539 + 0.4100U_{pp}^2 + 5.1044U_{pp}H_{sp}\right) - \beta_p^3 \left(0.1361 + 0.3907U_{pp}^2 + 3.8608U_{pp}H_{sp}\right) +$ $\beta_p^4 (1.0477H_{sp} - 0.047U_{pp}) + 0.0383\beta_p^5 + 0.3188U_{pp}^4 H_{sp}$ |
| 1 | 0.3297 | $0.2287 + 0.8273\beta_p - 1.6037U_{pp} - 0.0879U_{ss} + 0.1322G_{pp}U_{pp} + \beta_p G_{p_2} \left(0.1937 + 0.0899G_{pp} + 0.2505U_{ss}U_{pp}^{-1}\right)$ |
| 1 | 0.3364 | $0.1691 + 0.7236\beta_p - 1.4583U_{pp} - 0.1259U_{ss} + G_{ss}U_{pp} (0.3765\beta_p - 0.5079U_{pp})$ $(0.0033 + 2.4413\beta_p - 3.5580U_{pp} + 0.0320U_{ss})^{-1}$ |
| 1 | 0.3409 | $0.1327 + 0.7854\beta_p - 1.5128U_{pp} - 0.2357\beta_p^3 - 0.5975U_{pp}^3 + U_{pp}\beta_p (0.5131\beta_p + 0.4215U_{pp})$ |
| 1 | 0.3473 | $0.0585 + 1.2004\beta_p - 1.8721U_{pp} - 0.0868H_{sp}U_{pp} + \zeta_p G_{pp} (0.0429 + 0.3345\beta_p)$ |
| 1 | 0.3487 | $0.1216 + 0.8957\beta_p - 1.6844U_{pp} + 0.0517H_{sp} + 0.2678G_{p_2}U_{ss}$ |
| 1 | 0.3523 | $0.0974 + 0.6642\beta_p - 1.3279U_{pp} - U_{pp} \left(0.3897\beta_p^2 + 0.9765U_{pp}^2 - 1.4475\beta_p U_{pp}\right)$ |
| 1 | 0.3544 | $-0.1146 + 0.9803\beta_p - 1.5787U_{pp} - 0.0514U_{ss} + 0.8153U_{pp}^2 + 0.4578\beta_p^2 - 1.3338U_{pp}\beta_p$ |
| 2 | 0.3559 | $-0.1179 + 1.0169\beta_p - 1.6425U_{pp} - 1.3358\beta_p U_{pp} + 0.4656\beta_p^2 + 0.8125U_{pp}^3$ |
| 1 | 0.3747 | $-0.0113 + 1.2513\beta_p - 1.7967U_{pp} - \zeta_p (0.4705\beta_p - 0.3399U_{pp})$ |
| 1 | 0.3782 | $0.0778 + 0.8326\beta_p - 1.7093U_{pp} - 0.2009G_{ss} + 0.1890G_{ss}U_{ss}$ |
| 1 | 0.3829 | $0.1060 + 0.9889\beta_p - 1.4869U_{pp} - U_{pp}U_{ss} (0.2117 + 0.1337U_{ss})$ |
| 1 | 0.3940 | $1.0396\beta_p - 1.7124U_{pp} + 0.1669H_{sp}$ |
| 1 | 0.3982 | $1.1866\beta_p - 1.6360U_{pp} + 0.1578\beta_s$ |
| 1 | 0.4059 | $1.0050\beta_p - 1.7585U_{pp} - 0.2020G_{ss}$ |
| **75** | **0.4138** | $\mathbf{1.137\beta_p - 1.7002U_{pp}}$ |

Table B.9: $G_{pp} = f\left(U_{ss}, U_{pp}, \beta_s, \beta_p, \zeta_s, \zeta_p, G_{ss}, G_{sp}, G_{p_2}, H_{sp}\right)$

| # runs | RMSE | GP-regressed equation |
|---|---|---|
| 1 | 0.2551 | $0.2556 - 0.2040\zeta_p + 0.7617\beta_p + 1.1368G_{p_2} + 0.0898\beta_s + G_{p_2}\left(0.4814\beta_p + 0.1\zeta_s + 0.53\zeta_p^2\right)$ |
| 1 | 0.2602 | $0.2897 + 0.0142\zeta_p + 0.6151\beta_p + 1.0871G_{p_2} + G_{p_2}\left(0.4721\beta_p - 0.3809\zeta_p\right) + 0.4632\zeta_p G_{ss}$ |
| 1 | 0.2628 | $0.3338 + 0.2772\zeta_p + 0.7351\beta_p + 0.9715G_{p_2} + \zeta_p G_{p_2}^2\left(2.0404 - 2.9421\zeta_p G_{p_2}\right) - 2.3965\zeta_p^2\beta_p^3 +$ $\zeta_p G_{p_2}\beta_p\left[0.1669 - \zeta_p\left(10.3005\beta_p + 11.8231G_{p_2}\right)\right]$ |
| 1 | 0.2704 | $0.1893 + 0.8824\zeta_p + 1.3158\beta_p + 0.8287G_{p_2} + G_{p_2}\left(0.5145H_{sp} - 1.1331\zeta_p\right) -$ $H_{sp}\zeta_p\left[0.8854 - 0.1671G_{p_2} - \beta_p\left(1.1465 - 1.3632G_{ss} - 1.4019\zeta_p^2\right) + \zeta_p\left(0.9543 - 0.2230H_{sp} + 0.0045\zeta_p^3\right)\right]$ |
| 1 | 0.2794 | $0.3583 + 0.6217\zeta_p + 0.6705\beta_p + 1.0121G_{p_2} + 0.4031\beta_p G_{p_2}$ |
| 1 | 0.3078 | $0.9329\zeta_p + 0.9779\beta_p + 1.4046G_{p_2} - 0.6973G_{ss}$ |
| 1 | 0.3083 | $-0.0294 + 0.7456\zeta_p + 1.2536\beta_p + 1.0565G_{p_2} + 0.0744G_{sp} + 0.1881\zeta_p\beta_p U_{ss}G_{sp} +$ $0.0090\zeta_s G_{ss}\left(0.0079 - 1.4197\zeta_p^2 + 0.7628\zeta_p G_{ss}\right)^{-1}$ |
| 1 | 0.3151 | $0.1471 + 0.5837\zeta_p + 0.83236\beta_p + 0.8334G_{p_2} + \zeta_p G_{p_2}\left(0.8580\beta_p + 1.555G_{p_2}\right)$ |
| 4 | 0.3160 | $0.9222\zeta_p + 1.2859\beta_p + 1.0480G_{p_2} - 0.2345\zeta_s$ |
| 1 | 0.3179 | $0.0341 + 0.3950\zeta_p + 1.1245\beta_p + 1.0352G_{p_2} + H_{sp}\left(0.4014\zeta_p + 0.1705G_{p_2}\right)$ |
| 2 | 0.3211 | $0.0781 + 0.5470\zeta_p + 1.0111\beta_p + 1.0434G_{p_2} + H_{sp}\zeta_p\left(0.3471 + 0.6265\beta_p\right)$ |
| 1 | 0.3220 | $0.1476 + 0.7977\zeta_p + 1.1183\beta_p + 1.1076G_{p_2} + 0.8508\zeta_p\beta_p U_{ss}$ |
| 1 | 0.3374 | $0.1784 + 1.3150\zeta_p e^{-0.125\zeta_p} + 1.4402\beta_p e^{-0.2463\zeta_p} + 0.9355G_{p_2}e^{-1.2296\zeta_p}$ |
| 1 | 0.3473 | $0.0208 + 0.8222\zeta_p + 1.2829\beta_p + 1.0846G_{p_2} - 0.1478\zeta_p U_{ss}$ |
| **90** | **0.3495** | $\mathbf{0.74\zeta_p + 1.2965\beta_p + 1.0821G_{p_2}}$ |
| 2 | 0.4516 | $1.1848\zeta_p - 1.6024G_{ss} + 1.3178G_{p_2}$ |
| 1 | 0.5451 | $0.0841 + \zeta_p\left(1.9293G_{sp}^2 + 0.9650U_{ss}^2 + 2.1676G_{sp}U_{ss}\right)$ |

*continued on next page*

*continued from previous page*

| **# runs** | **RMSE** | **GP-regressed equation** |
|---|---|---|
| 1 | 0.5631 | $0.5976\zeta_p + 0.8549\beta_p + 0.6017G_{ss}$ |

Table B.10: $G_{p_2} = f\left(U_{ss}, U_{pp}, \beta_s, \beta_p, \zeta_s, \zeta_p, G_{ss}, G_{sp}, G_{pp}, H_{sp}\right)$

| # runs | RMSE | GP-regressed equation |
|---|---|---|
| 1 | 0.1078 | $0.489 - 0.7247U_{pp} + [0.4133 + (0.344 + 0.3379G_{ss})(0.9878 - 0.6662G_{ss})]\left[0.2042U_{pp} + 3.0799G_{ss} + G_{ss}^6(0.1021-0.0968G_{ss})\left(0.9377 + 0.0238G_{ss}^2\zeta_s\right) - U_{ss}(0.0933G_{ss} + U_{pp}(0.3044U_{pp} - 0.0844\beta_p)) - 0.5679G_{ss}e^{0.702U_{ss}+0.5837U_{ss}G_{pp}}\right]$ |
| 1 | 0.1352 | $-0.9909 + 1.7721G_{ss} + (1.0076 - 0.8254G_{ss} - 0.0716\beta_p - 0.1441U_{pp})\left[0.2912G_{pp}+\left(2.0671 - 0.0583U_pp - 0.3488G_{ss}^2\beta_p\right)\left(1.7364 - 0.2708\zeta_p + 0.0003e^{2.0002G_{sp}}\right)^{-1}\right]$ |
| 1 | 0.1739 | $-0.0684 + 0.3902G_{ss} + 0.343U_{ss} - 0.8439U_{pp} - 0.3139\beta_s + 0.2886G_{pp}\beta_p$ |
| 1 | 0.2128 | $-0.0325 + 0.9052G_{ss} - 1.3566\zeta_s + 0.1997\zeta_s^2 - 0.1424G_{ss}G_{pp} + \zeta_s G_{ss}\left[0.3574 + 3.6668G_{ss} - 1.8766G_{ss}^2+\zeta_s(0.1539 - 0.2436G_{ss}) + \beta_p\left(0.2197\zeta_s - 0.4294G_{ss} + 0.1767G_{ss}^2 - 0.0396\zeta_s G_{ss}\right)\right]$ |
| 1 | 0.2138 | $0.0198 + 0.6546G_{ss} + 0.3177G_{pp} - 0.2242U_{pp} - 0.3057G_{ss}G_{pp}$ |
| 1 | 0.2318 | $0.0319 + 0.8510G_{ss} + 0.3411G_{pp} - G_{pp}G_{ss}(0.3754 - 0.0112G_{ss}G_{sp}\beta_s)$ |
| 4 | 0.2365 | $0.0232 + 0.8374G_{ss} + 0.3473G_{pp} - 0.3577G_{ss}G_{pp}$ |
| 1 | 0.2403 | $1.1131G_{ss} + 0.3731G_{pp} - 0.5760\zeta_p$ |
| 1 | 0.2426 | $0.1476 - 1.6676e^{0.0368e^{3.4135H_{sp}}}\left[-0.409G_{ss} + 0.1779U_{pp} - 0.1554G_{ss}\zeta_s^2(0.2510 + 0.8962G_{ss} + 0.6078U_{pp}G_{ss}\zeta_s)(0.6093 + 1.2801G_{pp} - 0.4092G_{ss})\right]$ |
| 1 | 0.2452 | $0.0834 + 1.0127G_{ss} - 0.5488\zeta_p - 0.8503U_{pp}\zeta_p G_{ss}^{-1}$ |
| 1 | 0.2526 | $0.3494 + 0.8020G_{ss} + 1.1743\zeta_p - 0.8883G_{ss}\zeta_p$ |
| 1 | 0.2603 | $0.0631 + 0.757G_{ss} - 0.3172U_{pp} - 0.1462\zeta_s - 0.0657\zeta_s G_{ss} - \zeta_s^2\beta_s^2G_{ss}^2\left(0.6632 + 0.0305G_{ss} - 0.7315G_{ss}^2 + 0.2845G_{ss}^3\right)$ |
| 1 | 0.2611 | $0.6041G_{pp} - 0.4366\zeta_p - 1.0919\beta_p$ |
| 1 | 0.2700 | $0.0256 + 0.7197G_{ss} - 0.2823U_{pp} - 0.1569\zeta_s + 0.0082\zeta_s^2 + G_{ss}^2\zeta_p(0.1950 - 0.1428G_{ss})$ |
| 1 | 0.2727 | $0.0322 + 0.7180G_{ss} - 0.2970U_{pp} - 0.1130\zeta_s - 0.0818G_{ss}\zeta_p$ |

*continued on next page*

*continued from previous page*

| **# runs** | **RMSE** | **GP-regressed equation** |
|---|---|---|
| 1 | 0.2750 | $-0.1138 + 0.4825G_{ss} - 0.6629U_{pp}G_{pp}\beta_p G_{ss}^{-1}$ |
| 1 | 0.2818 | $0.7037G_{ss} - 0.3740U_{pp} + 0.0640\beta_p - 0.2136\zeta_s$ |
| 5 | 0.2979 | $0.0731 + 0.9034G_{ss} + 0.5482U_{pp}\zeta_p$ |
| 1 | 0.2951 | $-0.0788 + 0.4033G_{ss} - 0.4555U_{pp} - 0.1913G_{ss}U_{ss}$ |
| 1 | 0.2981 | $0.3280U_{ss} - 0.8964U_{pp} - 0.2802\beta_p$ |
| 4 | 0.2984 | $0.3012G_{ss} + 0.2756U_{ss} - 0.8440U_{pp}$ |
| 3 | 0.2999 | $0.6859G_{ss} - 0.2452\beta_s - 0.4829U_{pp}$ |
| 1 | 0.3011 | $0.7168G_{ss} - 0.3285U_{pp} - 0.1879\zeta_p + 0.0156\beta_p$ |
| 1 | 0.3070 | $0.0774 + 1.0493G_{ss}\left[0.0094 - 0.0688\zeta_p + 1.0876e^{0.3978\zeta_p}\right]^{-1}$ |
| 2 | 0.3071 | $0.0773 + 0.9463G_{ss}e^{-0.3671\zeta_p}$ |
| 1 | 0.3072 | $0.0750 + 1.0064G_{ss} - 0.1834\beta_s\zeta_p$ |
| 1 | 0.3096 | $0.0673 + 0.9831G_{ss} - 0.1503G_{ss}\zeta_p - 0.0041G_{ss}^5\zeta_p$ |
| 5 | 0.3100 | $0.0720 + 0.9831G_{ss} - 0.1830G_{ss}\zeta_p$ |
| 1 | 0.3208 | $0.3821U_{ss} - 1.1415U_{pp} - 0.0640H_{sp}$ |
| 1 | 0.3247 | $0.4021U_{ss} - 1.1929U_{pp}$ |
| 1 | 0.3265 | $0.0323 + 0.9326G_{ss} - G_{pp}\left(0.0348\zeta_p + 0.1083G_{ss}\right)$ |
| 1 | 0.3291 | $0.3939G_{ss} - 0.1919\beta_p - 0.3996U_{pp}$ |
| 20 | 0.3334 | $0.9772G_{ss} - 0.2559\zeta_s$ |
| 10 | 0.3371 | $1.0138G_{ss} - 0.2633\zeta_p$ |
| **29** | **0.3373** | $\mathbf{0.5096G_{ss} - 0.4671U_{pp}}$ |
| 1 | 0.3375 | $-0.1895 + 1.0466G_{ss} + 0.0987\beta_p^{-1}$ |

*continued on next page*

*continued from previous page*

| **# runs** | **RMSE** | **GP-regressed equation** |
|---|---|---|
| 1 | 0.3610 | $0.0414 + 0.9064G_{ss} + 0.3442\beta_p\zeta_p$ |

Table B.11: $H_{sp} = f\left(U_{ss}, U_{pp}, \beta_s, \beta_p, \zeta_s, \zeta_p, G_{ss}, G_{sp}, G_{pp}, G_{p_2}\right)$

| # runs | RMSE | GP-regressed equation |
|---|---|---|
| 1 | 0.1311 | $0.8634 - 2.3347\beta_p + 2.0892\zeta_p - 0.0972U_{pp} - 1.2273G_{p_2}G_{pp} + \beta_s \left[U_{pp}\left(3.3968 + 0.6634U_{pp} + 19.3375\beta_p\right) - G_{ss}\left(10.5846 - 7.3190\beta_p - 15.4872U_{pp}\right) - \beta_p\left(11.2351 + \zeta_s\left(46.0275U_{pp}^2 + \beta_p\left(15.3614 - 15.7488U_{pp} - 7.7635\beta_p - 65.8880U_{pp}\right)\right)\right) + G_{p_2}\left(1.6071G_{sp} - 1.7287G_{pp} + 5.4204U_{pp} + 12.2091\beta_s\beta_p\zeta_s + 2.9899G_{pp}U_{pp} - 12.8781G_{sp}U_{pp} + 4.5104U_{pp}^2 + G_{ss}\left(8.5366 + 1.0498G_{pp} - 7.8379G_{sp} + 4.9423U_{pp}\right)\right) + U_{pp}^{-1}\left(G_{ss}\left(1.5681\beta_p + G_{p_2}\left(2.0822G_{sp} - 3.5293G_{pp}\right)\right) + \beta_s\beta_p\left(12.4536\beta_s\beta_p + \zeta_s G_{p_2}\left(7.3430G_{sp} - 13.1694G_{pp}\right)\right)\right) + \beta_p\zeta_s G_{p_2}\left(21.8512\beta_s G_{sp} - 21.0161\beta_s G_{pp} + U_{pp}\left(12.4458\beta_s - 24.5023G_{pp} + 45.542G_{sp} + 36.1331U_{pp} - 40.9307U_{pp}^2 + 46.4520\beta_p U_{pp} - 17.5856G_{p_2}G_{pp}U_{pp} + 26.6614G_{p_2}G_{sp}U_{pp} + 14.8431G_{p_2}U_{pp}^2 + 2.6654U_{pp}G_{sp} - 3.4174\beta_p G_{sp} - 20.6153G_{p_2}G_{pp}G_{sp} + 8.8248G_{p_2}G_{sp}^2\right)\right)\right] - U_{pp}^2\left[10.2893U_{pp} - 2.4347\beta_p + G_{p_2}\left(4.0292G_{pp} - 8.7243G_{sp} + 11.4747U_{pp} - 18.1473U_{pp}^2 - 5.4413G_{sp}U_{pp} + 25.9172\beta_p U_{pp} + 6.8632\beta_p G_{sp}\right) - G_{p_2}^2\left(2.4704G_{pp}U_{pp} + 26.8454G_{sp}U_{pp} + 6.7784U_{pp}^2 - 10.7222G_{pp}G_{sp} + 12.0712G_{sp}^2\right)\right] - G_{ss}U_{pp}\left[19.0150U_{pp} + 5.2218\beta_p + G_{p_2}\left(23.5543G_{pp} - 21.7133G_{sp} + 12.6281U_{pp} - 13.2268U_{pp}^2 + 1.8494U_{pp}G_{sp} + 33.4042\beta_p U_{pp} - 0.8648\beta_p G_{sp}\right) + G_{p_2}^2\left(4.2766G_{pp}U_{pp} - 27.6151G_{sp}U_{pp} - 10.8459U_{pp}^2 + 14.2356G_{pp}G_{sp} - 10.5913G_{sp}^2\right)\right]$ |
| 1 | 0.4653 | $-0.5938 + 1.1198\beta_p - 0.5374G_{p_2} - 0.3158\zeta_p + 0.4552G_{pp} + 1.5583G_{p_2}G_{pp} - 0.0712G_{ss}\beta_s - G_{p_2}\beta_p\left[1.7971 - 2.2452\beta_p - 0.9469U_{pp} + G_{p_2}\left(1.7158\beta_p + 0.0860U_{pp} + 2.7831G_{pp} - 1.6231G_{p_2}G_{pp}\right)\right] + G_{p_2}^2 G_{pp}U_{pp}\left(1.7978 - 0.9724G_{p_2}\right)$ |
| 1 | 0.5899 | $0.1988 - G_{ss}\beta_p\left[0.2166 - 0.4057G_{p_2} + 0.1219\beta_s - G_{p_2}\beta_p\zeta_s\left(0.0366 + 0.0284G_{p_2} - 0.0213\beta_s\right)\right]$ |
| 1 | 0.6002 | $0.2216 + G_{p_2}^2\left[0.3698U_{pp} + \beta_s\left(0.0830G_{sp} - 0.0479G_{p_2}G_{pp}\right)\right]$ |
| 1 | 0.6124 | $0.1262 + \left(0.4548\beta_p + 0.6067\beta_s + 0.8073U_{pp}\right)e^{-1.0903e^{4.7974G_{pp}}}$ |

*continued on next page*

*continued from previous page*

| # runs | RMSE | GP-regressed equation |
|---|---|---|
| 1 | 0.6199 | $0.0978 + 0.6280G_{p_2}^2 - 0.2518G_{p_2}^4$ |
| 1 | 0.6207 | $0.1592 - 0.0758G_{p_2} - G_{p_2}\left(0.5727\beta_p - 0.7887G_{pp} + 0.6918G_{p_2}\right)$ |
| 1 | 0.6230 | $0.2510 + 0.0984\beta_p e^{-1.9720G_{p_2}^2\zeta_p}$ |
| 1 | 0.6237 | $0.1248 + G_{p_2}^2\left(0.2201 - 0.0708G_{pp}\right) - G_{p_2}^3\left(0.2059 - 0.1859G_{pp}\right)$ |
| 2 | 0.6307 | $0.2105 + 0.4169G_{pp}G_{p_2} - 0.4457\beta_p U_{ss}$ |
| 1 | 0.6310 | $0.1246 + 0.4553G_{p_2}\zeta_p + 0.2321G_{p_2}^3 G_{pp}$ |
| 1 | 0.6325 | $0.2832 + 0.0691G_{pp}G_{p_2}^5$ |
| 1 | 0.6365 | $-0.1059 + 0.3753\beta_p + G_{p_2}\left(0.7274\zeta_p + 0.2543G_{pp}\right)$ |
| 1 | 0.6379 | $-0.0145 - 0.5531G_{ss} + 0.3993\beta_s + 0.6044G_{pp}G_{p_2}$ |
| 1 | 0.6380 | $0.0614 + 0.0530\beta_p\left[2.1127e^{12.8064\zeta_p} + 0.0071e^{21.9966G_{pp}}\right]^{-1}$ |
| 2 | 0.6410 | $0.1130 + 0.2973G_{p_2}^2 - 0.3618G_{p_2}^3$ |
| 1 | 0.6448 | $0.7194\beta_p + 0.6327\zeta_p - 0.4804\zeta_s$ |
| 1 | 0.6462 | $0.1617 + 0.6349\beta_p G_{p_2}\left(0.9832G_{ss} + 1.8664\zeta_p\right)^{-1}$ |
| 1 | 0.6468 | $0.1675 + 0.2052G_{p_2}^2 + 0.3181G_{p_2}^2\beta_p$ |
| 15 | 0.6499 | $-0.1251 + 0.5294\beta_p + 0.9119G_{p_2}\zeta_p$ |
| 2 | 0.6528 | $0.2742 + 0.2348G_{p_2}^2\beta_p$ |
| 25 | 0.6535 | $0.2207 + 0.2738G_{p_2}^3 G_{pp}$ |
| 4 | 0.6539 | $0.0774 + 0.4227G_{pp}G_{p_2}e^{-3.1435\zeta_p}$ |
| 1 | 0.6543 | $0.0924 + 0.6121\beta_p + 0.4752\zeta_p e^{-0.8847G_{pp}G_{p_2}}$ |
| 1 | 0.6548 | $0.0643 - 0.7693G_{p_2} + 0.7963G_{p_2}\zeta_p G_{ss}^{-1}$ |
| 3 | 0.6554 | $0.0901 + 0.8137G_{p_2}\zeta_p e^{-1.0138U_{ss}}$ |

*continued on next page*

*continued from previous page*

| # runs | RMSE | GP-regressed equation |
|---|---|---|
| **28** | **0.6663** | $\mathbf{-0.0130 - 0.3230G_{p_2} + 0.5452G_{p_2}G_{pp}}$ |
| 1 | 0.6674 | $0.0757 - 0.9904G_{p_2}^2\zeta_p + 1.2166G_{p_2}^3\zeta_p$ |
| 1 | 0.6745 | $-0.0713 + 0.6319\beta_p - 0.5921\zeta_p\beta_p$ |
| 1 | 0.6767 | $0.0444 + 0.1418U_{pp} - 0.7509U_{pp}G_{pp}$ |
| 3 | 0.6780 | $0.0548 - 0.1047G_{pp} - 0.9276U_{pp}G_{pp}$ |
| 1 | 0.6781 | $0.0320 - 0.6360G_{pp}U_{pp} + 0.2328G_{pp}G_{p_2}$ |
| 2 | 0.6842 | $0.0514 - 0.8690U_{pp}G_{pp}$ |
| 1 | 0.6908 | $-0.0125 + 0.2810\beta_p + 0.5236G_{pp}G_{p_2}$ |
| 1 | 0.7147 | $-0.018 + 0.7531G_{pp}G_{p_2}$ |

# References

Ackley, D. H. (1987). *A connectionist machine for genetic hill climbing*. Kluwer Academic Publishers.

Ahn, C. W. (2005). *Theory, design, and application of efficient genetic and evolutionary algorithms*. Doctoral dissertation, Gwangju Institute of Science and Technology, Korea.

Albert, L. A., & Goldberg, D. E. (2001). Efficient evaluation relaxation under integrated fitness functions. *Intelligent Engineering Systems Through Artificial Neural Networks*, *11*, 165–170. (Also IlliGAL Report No. 2001024).

Altenberg, L. (1994). Emergent phenomena in genetic programming. *Evolutionary Programming — Proceedings of the Third Annual Conference*, 233–241.

Andersson, K., Barysz, M., Bernhardsson, A., Blomberg, M. R. A., Carissan, Y., Cooper, D. L., Flscher, M. P., Gagliardi, L., de Graaf, C., Hess, B. A., Hagberg, D., Karlstrm, G., Lindh, R., Malmqvist, P.-Å., Nakajima, T., Neogrády, P., Olsen, J., Raab, J., Roos, B. O., Ryde, U., Schimmelpfennig, B., Schütz, M., Seijo, L., Serrano-Andrés, L., Siegbahn, P. E. M., Stålring, J., Thorsteinsson, T., Veryazov, V., & Widmark, P.-O. (2001). MOLCAS version 5.2. Lund University, Sweden.

Angeline, P. J. (1996). Genetic programming and emergent intelligence. In Kinnear, K. E. (Ed.), *Advances in Genetic Programming* (Chapter 4, pp. 75–98). Cambridge, MA: MIT Press.

Asoh, H., & Mühlenbein, H. (1994). On the mean convergence time of evolutionary algorithms without selection and mutation. *Parallel Problem Solving from Nature*, *3*, 98–107.

Babovic, V., & Keijzer, M. (2000). Genetic programming as a model induction engine. *Journal of Hydroinformatics*, *2*(1), 35–60.

Bäck, T. (1994). Selective pressure in evolutionary algorithms: A characterization of selection mechanisms. *Proceedings of the First IEEE Conference on Evolutionary Computation*, 57–62.

Bäck, T. (1995). Generalized convergence models for tournament—and $(\mu, \lambda)$—selection. *Proceedings of the Sixth International Conference on Genetic Algorithms*, 2–8.

Bäck, T. (1996). *Evolutionary algorithms in theory and practice*. New York: Oxford University Press.

Baker, J. E. (1985). Adaptive selection methods for genetic algorithms. *Proceedings of the International Conference on Genetic Algorithms and Their Applications*, 101–111.

Baluja, S. (1994). *Population-based incremental learning: A method of integrating genetic search based function optimization and competitive learning* (Technical Report CMU-CS-94-163). Carnegie Mellon University.

Banzhaf, W., Nordin, P., Keller, R., & Francone, F. (1998). *Genetic programming-An introduction on the automatic evolution of computer programs and its applications.* San Francisco, CA: Morgan Kaufmann.

Barkema, G. T., & Mousseau, N. (1996). Event-based relaxation of continuous disordered systems. *Physical Review Letters*, *77*(21), 4358–4361.

Barkema, G. T., & Mousseau, N. (2001). The activation-relaxation technique: An efficient algorithm for sampling energy landscapes. *Computational Materials Science*, *20*, 285–292.

Barthelemy, J.-F. M., & Haftka, R. T. (1993). Approximation concepts for optimum structural design—a review. *Structural Optimization*, *5*, 129–144.

Beaudoin, A., & Cassada, W. (1998). Investigation of texture and fracture toughness in AA7050 plate. *Proceedings of the TMS Annual Spring Meeting*, .

Ben-Nun, M., & Martinez, T. J. (2000). Photodynamics of ethylene: *Ab initio* studies of conical intersections. *Chemical Physics*, *259*, 237–248.

Ben-Nun, M., & Martinez, T. J. (2002). *Ab Initio* quantum molecular dynamics. *Advances in Chemical Physics*, *121*, 439–512. (Preprint: http://mtzweb.scs.uiuc.edu/reprints/2002/ACP.pdf).

Ben-Nun, M., Quenneville, J., & Martinez, T. J. (2000). *Ab Initio* multiple spawning: Photochemistry from first principles quantum molecular dynamics. *Journal of Physical Chemistry A*, *104*(22), 5161–5175. (Preprint: http://mtzweb.scs.uiuc.edu/reprints/2000/FeatureJPC.pdf).

Bethke, A. D. (1976). *Comparison of genetic algorithms and gradient-based optimizers on parallel processors: Efficiency of use of processing capacity* (Tech. Rep. No. 197). Ann Arbor, MI: University of Michigan, Logic of Computers Group.

Beyer, H.-G. (1996). Toward a theory of evolution strategies: Self-adaptation. *Evolutionary Computation*, *3*(3), 311–347.

Binder, K. (Ed.) (1986). *Monte Carlo methods in statistical physics.* Berlin: Springer.

Blickle, T., & Thiele, L. (1995). A mathematical analysis of tournament selection. *Proceedings of the Sixth International Conference on Genetic Algorithms*, 9–16.

Blickle, T., & Thiele, L. (1996). A comparison of selection schemes used in evolutionary algorithms. *Evolutionary Computation*, *4*(4), 361–394.

Bocquet, J. L. (2002). On-the-fly evaluation of diffusional parameters during a Monte Carlo simulation of diffusion in alloys: A challenge. *Defect and Diffusion Forum*, *203–205*, 81–112.

Boisvert, G., & Lewis, L. (1997). Self-diffusion of adatoms, dimers, and vacancies on Cu(100). *Physical Review B*, *56*(12), 7643–7655.

Booker, A. J., Dennis, J. E., Frank, P. D., Serafini, D. B., Torczon, V., & Trosset, M. W. (1998). *A rigorous framework for optimization of expensive functions by surrogates* (Technical Report). Hampton, VA: National Aeronautics and Space Administration (NASA). ICASE Report No. 98-47.

Booker, L. B., Fogel, D. B., Whitley, D., & Angeline, P. J. (1997). Recombination. In Bäck, T., Fogel, D. B., & Michalewicz, Z. (Eds.), *The Handbook of Evolutionary Computation* (Chapter E3.3, pp. C3.3:1–C3.3:27). Philadelphia, PA: IOP Publishing Ltd. and Oxford University Press.

Bosman, P. A. N., & Thierens, D. (1999). Linkage information processing in distribution estimation algorithms. *Proceedings of the 1999 Genetic and Evolutionary Computation Conference*, 60–67. (Also Technical Report No. UU-CS-1999-10).

Bosman, P. A. N., & Thierens, D. (2000). *Mixed ideas* (Technical Report UU-CS-2000-45). Utrecht University.

Bosman, P. A. N., & Thierens, D. (2002a). The balance between proximity and diversity in multiobjective evolutionary algorithms. *IEEE Transactions on Evolutionary Computation*, *7*(2), 174–188.

Bosman, P. A. N., & Thierens, D. (2002b). Multiobjective optimization with diversity preserving mixture-based iterated density estimation evolutionary algorithms. *International Journal of Approximate Reasoning*, *31*(3), 259–289. (Also Technical Report No. UU-CS-2001-59).

Bosworth, J. L., Foo, N., & Zeigler, B. P. (1972). *Comparison of genetic algorithms with conjugate gradient methods* (ORA Tech. Report No. 00312-1-T). Ann Arbor, MI: University of Michigan, Department of Computer and Communication Sciences.

Bouar, Y. L., & Soisson, F. (2002). Kinetic pathways from embedded-atom-method potentials: Influence of the activation barriers. *Physical Review B*, *65*(9), 094103.

Brothers, E. N., & Merz, Jr., K. M. (2002). Sodium parameters for AM1 and PM3 optimized using a modified genetic algorithm. *Journal of Physical Chemistry, B*, *106*(10), 2779–2785.

Bulmer, M. G. (1985). *The mathematical theory of quantitative genetics*. Oxford: Oxford University Press.

Cai, W., Kalos, M. H., de Koning, M., & Bulatov, V. V. (2002). Importance sampling of rare transition events in Markov processes. *Physical Review E*, *66*(4), 046703.

Cantú-Paz, E. (1997, January). *A summary of research on parallel genetic algorithms* (IlliGAL Report No. 97003). Urbana, IL: University of Illinois at Urbana-Champaign.

Cantú-Paz, E. (1998). Designing efficient master-slave parallel genetic algorithms. pp. 455. (Also IlliGAL Report No. 97004).

Cantú-Paz, E. (1999). Migration policies and takeover times in parallel genetic algorithms. *Proceedings of the 1999 Genetic and Evolutionary Computation Conference*, 775. (Also IlliGAL Report No. 99008).

Cantú-Paz, E. (2000a). *Efficient and accurate parallel genetic algorithms*. Boston, MA: Kluwer Academic Pub.

Cantú-Paz, E. (2000b). Selection intensity in genetic algorithms with generation gaps. *Proceedings of the 2000 Genetic and Evolutionary Computation Conference*, 911–918.

Cantú-Paz, E., & Goldberg, D. E. (1999). On the scalability of parallel genetic algorithms. *Evolutionary computation*, *7*(4), 429–449.

Cantú-Paz, E., & Goldberg, D. E. (2000). Efficient parallel genetic algorithms: Theory and practice. *Computer Methods in Applied Mechanics and Engineering*, *186*, 221–238.

Chen, J.-H. (2004). *Theory and applications of efficient multi-objective evolutionary algorithms*. Doctoral dissertation, Feng Chia University, Taichung, Taiwan.

Chen, Y. (1999, September). *Model order reduction for nonlinear systems*. Master's thesis, Massachusetts Institute of Technology, Cambridge, MA.

Chen, Y.-p. (2005). *Extending the scalability of linkage learning genetic algorithms: Theory and practice*. Berlin: Springer.

Chen, Y.-p., Yu, T.-L., Sastry, K., & Goldberg, D. E. (2007, April). *A survey of genetic linkage learning techniques* (IlliGAL Report No. 2007014). Urbana, IL: University of Illinois at Urbana-Champaign.

Cleri, F., & Rosato, V. (1993). Tight-binding potentials for transition metals and alloys. *Physical Review B*, *48*, 22–33.

Coello Coello, C. A., Van Veldhuizen, D. A., & Lamont, G. B. (2002). *Evolutionary algorithms for solving multi-objective problems*. Boston, MA: Kluwer Academic Publishers.

Coffin, D. J., & Clack, C. D. (2006). gLINC: Identifying composability using group perturbation. *Proceedings of ACM SIGEVO Genetic and Evolutionary Computation Conference 2006 (GECCO-2006)*, 1133–1140.

Corne, D. W., Jerram, N. R., Knowles, J. D., & Oates, M. J. (2001). PESA-II: Region-based selection in evolutionary multiobjective optimization. *Proceedings of the 2001 Genetic and Evolutionary Computation Conference 2001*, 283–290.

Crow, J. F., & Kimura, M. (1970). *An introduction of population genetics theory*. Harper and Row.

Cundari, T. R., Deng, J., & Fu, W. (2000). PM3(tm) parameterization using genetic algorithms. *International Journal of Quantum Chemistry*, *77*, 421–432.

Davis, L. (Ed.) (1991). *Handbook of genetic algorithms*. New York, NY: Van Nostrand Reinhold.

De Jong, K. A. (1975). *An analysis of the behavior of a class of genetic adaptive systems*. Doctoral dissertation, University of Michigan, Ann Arbor, MI. (University Microfilms No. 76-9381).

Deb, K. (1999). Multi-objective genetic algorithms: Problem difficulties and construction of test problems. *Evolutionary Computation*, *7*(3), 205–230.

Deb, K. (2001). *Multi-objective optimization using evolutionary algorithms*. Chichester, UK: John Wiley and Sons.

Deb, K., & Agarwal, R. (1995). Simulated binary crossover for continuous search space. *Complex Systems*, *9*, 115–148.

Deb, K., & Goldberg, D. E. (1992). Analyzing deception in trap functions. *Foundations of Genetic Algorithms*, *2*, 93–108. (Also IlliGAL Report No. 91009).

Deb, K., & Goldberg, D. E. (1994). Sufficient conditions for deceptive and easy binary functions. *Annals of Mathematics and Artificial Intelligence*, *10*, 385–408. (Also IlliGAL Report No. 92001).

Deb, K., & Kumar, A. (1995). Real-coded genetic algorithms with simulated binary crossover: Studies on multimodal and multiobjective problems. *Complex Systems*, *9*, 431–454.

Deb, K., Pratap, A., Agrawal, S., & Meyarivan, T. (2002). A fast elitist non-dominated sorting genetic algorithm for multi-objective optimization: NSGA-II. *IEEE Transactions on Evolutionary Computation*, *6*(2), 182–197. (Also KanGAL Report No. 2000001).

Dennis, J. E., & Torczon, V. (1997). Managing approximate models in optimization. In Alexandrov, N. M., & Hussaini, M. Y. (Eds.), *Multidisciplinary Design Optimization: State-of-the-Art* (pp. 330–347). Philadelphia, PA: SIAM.

Deshpande, N. (1998). Relationship between fracture toughness, fracture path, and microstructure of 7050 aluminum alloy: Part I. Quantitative characterization. *Metal Transactions A*, *29A*, 1191–1201.

Dewar, M. J. S., & Thiel, W. (1977). The MNDO method. Approximations and parameters. *Journal of The American Chemical Society*, *99*(15), 4899–4907.

Dewar, M. J. S., Zoebisch, E. G., Healy, E. F., & Stewart, J. J. P. (1985). AM1: A new general purpose quantum mechanical molecular model. *Journal of The American Chemical Society*, *107*(13), 3902–3909.

Diaz De La Rubia, T., Caturla, M. J., Alonso, E., Fluss, M. J., & Perlado, J. M. (1998). Self-decay-induced damage production and micro-structure evolution in fcc metals: An atomic-scale computer simulation approach. *Journal of Computer-Aided Materials Design*, *5*, 243–264.

Ebner, M., O'Neill, M., Eká, A., Vanneschi, L., & Esparcia-Alcázar, A. (Eds.) (2007). *Genetic programming, proceedings of the 10th European conference, EuroGP 2007*, Volume 4445 of *Lecture Notes in Computer Science*. Berlin: Springer.

Erickson, M., Mayer, A., & Horn, J. (2001). The niched Pareto genetic algorithm 2 applied to the design of groundwater remediation systems. *Proceedings of the First International Conference on Evolutionary Multi-Criterion Optimization*, 681–695.

Famanara, P., Stert, V., & Radloff, W. (1998). Ultrafast internal conversion and fragmentation in electronically excited $c_2h_4$ and $c_2h_3cl$ molecules. *Chemical Physics Letters*, *288*(2), 518–522.

Fichthorn, K. A., & Weinberg, W. H. (1991). Theoretical foundations of dynamical Monte Carlo simulations. *Journal of Chemical Physics*, *95*(2), 1090–1096.

Fish, J., & Schwab, C. (2003). Towards constitutive models based on atomistics. *International Journal for Multiscale Computational Engineering*, *1*(1), 43–56.

Fitzpatrick, J. M., Grefenstette, J. J., & Van Gucht, D. (1984). Image registration by genetic search. In *Proceedings of IEEE Southeast Conference* (pp. 460–464). Piscataway, NJ: IEEE press.

Fleurent, C., & Ferland, J. (1994). Genetic hybrids for the quadratic assignment problem. In *DIMACS Series in Mathematics and Theoretical Computer Science*, Volume 16 (pp. 190–206).

Fonseca, C. M., & Fleming, P. J. (1993). Genetic algorithms for multiobjective optimization: Formulation, discussion, and generalization. *Proceedings of the Fifth International Conference on Genetic Algorithms*, 416–423.

Girifalco, L., & Weizer, V. (1959). Application of the Morse potential function to cubic metals. *Physical Review*, *114*(3), 687–690.

Goldberg, D. E. (1983). *Computer-aided pipeline operation using genetic algorithms and rule learning*. Doctoral dissertation, University of Michigan, Ann Arbor, MI.

Goldberg, D. E. (1987). Simple genetic algorithms and the minimal deceptive problem. In Davis, L. (Ed.), *Genetic algorithms and simulated annealing* (Chapter 6, pp. 74–88). Los Altos, CA: Morgan Kaufmann.

Goldberg, D. E. (1989a). Genetic algorithms and Walsh functions: Part II, deception and its analysis. *Complex Systems*, *3*, 153–171. (Also IlliGAL Report No. 89001).

Goldberg, D. E. (1989b). *Genetic algorithms in search optimization and machine learning*. Reading, MA: Addison-Wesley.

Goldberg, D. E. (1989c). Sizing populations for serial and parallel genetic algorithms. *Proceedings of the Third International Conference on Genetic Algorithms*, 70–79. (Also IlliGAL Report No. 88004).

Goldberg, D. E. (1991). *Six steps to GA happiness.* Paper presented at Oregon Graduate Institute, Beaverton, OR.

Goldberg, D. E. (1999a). The race, the hurdle, and the sweet spot: Lessons from genetic algorithms for the automation of design innovation and creativity. In Bentley, P. (Ed.), *Evolutionary Design by Computers* (Chapter 4, pp. 105–118). San Mateo, CA: Morgan Kaufmann.

Goldberg, D. E. (1999b). Using time efficiently: Genetic-evolutionary algorithms and the continuation problem. *Proceedings of the 1999 Genetic and Evolutionary Computation Conference*, 212–219. (Also IlliGAL Report No. 99002).

Goldberg, D. E. (2002). *Design of innovation: Lessons from and for competent genetic algorithms.* Boston, MA: Kluwer Academic Publishers.

Goldberg, D. E., & Deb, K. (1991). A comparative analysis of selection schemes used in genetic algorithms. *Foundations of Genetic Algorithms*, 69–93.

Goldberg, D. E., Deb, K., & Clark, J. H. (1992). Genetic algorithms, noise, and the sizing of populations. *Complex Systems*, *6*, 333–362. (Also IlliGAL Report No. 91010).

Goldberg, D. E., Deb, K., & Horn, J. (1992). Massive multimodality, deception, and genetic algorithms. *Parallel Problem Solving from Nature*, *2*, 37–46. (Also IlliGAL Report No. 92007).

Goldberg, D. E., Deb, K., Kargupta, H., & Harik, G. (1993). Rapid, accurate optimization of difficult problems using fast messy genetic algorithms. *Proceedings of the Fifth International Conference on Genetic Algorithms*, 56–64. (Also IlliGAL Report No. 93004).

Goldberg, D. E., Korb, B., & Deb, K. (1989). Messy genetic algorithms: Motivation, analysis, and first results. *Complex Systems*, *3*(5), 493–530. (Also IlliGAL Report No. 89003).

Goldberg, D. E., & Liepens, G. (1991). Theory tutorial. (Tutorial presented at the 1991 International Conference on Genetic Algorithms, La Jolla, CA).

Goldberg, D. E., & O'Reilly, U.-M. (1998). Where does the good stuff go, and why? How contextual semantics influences program structure in simple genetic programming. *Genetic Programming, Proceedings of the First European Conference, EuroGP'98*, 16–36.

Goldberg, D. E., & Rudnick, M. (1991). Genetic algorithms and the variance of fitness. *Complex Systems*, *5*(3), 265–278. (Also IlliGAL Report No. 91001).

Goldberg, D. E., & Sastry, K. (2001). A practical schema theorem for genetic algorithm design and tuning. *Proceedings of the 2001 Genetic and Evolutionary Computation Conference*, 328–335. (Also IlliGAL Report No. 2001017).

Goldberg, D. E., Sastry, K., & Latoza, T. (2001). On the supply of building blocks. *Proceedings of the 2001 Genetic and Evolutionary Computation Conference*, 336–342. (Also IlliGAL Report No. 2001015).

Goldberg, D. E., Sastry, K., & Llorà, X. (2007). Toward routine billion-variable optimization using genetic algorithms. *Complexity*, *12*(3), 27–29.

Goldberg, D. E., & Segrest, P. (1987). Finite Markov chain analysis of genetic algorithms. *Proceedings of the Second International Conference on Genetic Algorithms*, 1–8.

Goldberg, D. E., Thierens, D., & Deb, K. (1993). Toward a better understanding of mixing in genetic algorithms. Journal of the Society of Instrument and Control Engineers, *32*(1), 10–16. (Also IlliGAL Report No. 92009).

Goldberg, D. E., & Voessner, S. (1999). Optimizing global-local search hybrids. *Proceedings of the 1999 Genetic and Evolutionary Computation Conference*, 220–228. (Also IlliGAL Report No. 99001).

Gorges-Schleuter, M. (1989a). ASPARAGOS: A population genetics approach to genetic algorithms. *Evolution and Optimization '89*, 86–94.

Gorges-Schleuter, M. (1989b). ASPARAGOS: An asynchronous parallel genetic optimization strategy. *Proceedings of the Third International Conference on Genetic Algorithms*, 422–428.

Gorges-Schleuter, M. (1997). ASPARAGOS96 and the traveling salesman problem. *Proceedings of the IEEE International Conference on Evolutionary Computation*, 171–174.

Granucci, G., & Toniolo, A. (2000). Molecular gradients for semiempirical CI wavefunctions with floating occupation molecular orbitals. *Chemical Physics Letters*, *325*, 79–85.

Grasemann, U., & Miikkulainen, R. (2005). Effective image compression using evolved wavelets. *Proceedings of the 2005 Genetic and Evolutionary Computation Conference*, *2*, 1961–1968.

Grefenstette, J. J. (1981). *Parallel adaptive algorithms for function optimization* (Tech. Rep. No. CS-81-19). Nashville, TN: Vanderbilt Univeristy, Computer Science Department.

Grefenstette, J. J., & Baker, J. E. (1989). How genetic algorithms work: A critical look at implicit parallelism. *Proceedings of the Third International Conference on Genetic Algorithms*, 20–27.

Grefenstette, J. J., & Fitzpatrick, J. M. (1985). Genetic search with approximate function evaluations. *Proceedings of the International Conference on Genetic Algorithms and Their Applications*, 112–120.

Grosso, P. B. (1985). *Computer simulations of genetic adaptation: Parallel subcomponent interaction in a multilocus model.* Doctoral dissertation, University of Michigan, Ann Arbor, MI. (University microfilms no. 8520908).

Gruau, F. (1994). *Neural network synthesis using cellular encoding and the genetic algorithm.* Doctoral dissertation, L'Universite Claude Bernard-Lyon I.

Halhal, D., Walters, G. A., Savic, D. A., & Ouazar, D. (1999). Putting more genetics into genetic algorithms. *Evolutionary Computation*, *7*(3), 311–329.

Hamilton, J. C., Daw, M. S., & Foiles, S. M. (1995). Dislocation mechanism for island diffusion on fcc (111) surfaces. *Physical Review Letters*, *74*(14), 2760–2763.

Hansen, N., & Ostermeier, A. (2001). Completely derandomized self-adaptation in evolution strategies. *Evolutionary Computation*, *9*(2), 159–195.

Harik, G. (1999, January). *Linkage learning via probabilistic modeling in the ECGA* (IlliGAL Report No. 99010). Urbana, IL: University of Illinois at Urbana-Champaign.

Harik, G., Cantú-Paz, E., Goldberg, D. E., & Miller, B. L. (1999). The gambler's ruin problem, genetic algorithms, and the sizing of populations. *Evolutionary Computation*, *7*(3), 231–253. (Also IlliGAL Report No. 96004).

Harik, G., & Goldberg, D. E. (1997). Learning linkage. *Foundations of Genetic Algorithms*, *4*, 247–262. (Also IlliGAL Report No. 96006).

Harik, G., Lobo, F., & Goldberg, D. E. (1998). The compact genetic algorithm. *Proceedings of the IEEE International Conference on Evolutionary Computation*, 523–528. (Also IlliGAL Report No. 97006).

Harik, G. R. (1995). Finding multimodal solutions using restricted tournament selection. *Proceedings of the Sixth International Conference on Genetic Algorithms*, 24–31. (Also IlliGAL Report No. 94002).

Harik, G. R. (1997). *Learning linkage to efficiently solve problems of bounded difficulty using genetic algorithms*. Doctoral dissertation, University of Michigan, Ann Arbor, MI. (Also IlliGAL Report No. 97005).

Harik, G. R., Lobo, F. G., & Sastry, K. (2006). Linkage learning via probabilistic modeling in the ECGA. In Pelikan, M., Sastry, K., & Cantú-Paz, E. (Eds.), *Scalable Optimization via Probabilistic Modeling: From Algorithms to Applications* (Chapter 3, pp. 39–61). Berlin: Springer. (Also IlliGAL Report No. 99010).

Hart, W. E. (1994). *Adaptive global optimization with local search*. Doctoral dissertation, University of California, San Diego, San Diego, CA.

Hart, W. E., & Belew, R. K. (1996). Optimization with genetic algorithm hybrids using local search. In Belew, R. K., & Mitchell, M. (Eds.), *Adaptive Individuals in Evolving Populations* (pp. 483–494). Reading, MA: Addison-Wesley.

Hartmann, A. K., & Rieger, H. (2001). *Optimization algorithms in physics* (Chapter 9, pp. 192–203). Berlin: Wiley-VCH.

Haynes, T. D., Schoenfeld, D. A., & Wainwright, R. L. (1996). Type inheritance in strongly typed genetic programming. In Angeline, P. J., & Kinnear, K. E. (Eds.), *Advances in Genetic Programming 2* (Chapter 18, pp. 359–376). Cambridge, MA: MIT Press.

Heckendorn, R. B., & Wright, A. H. (2004). Efficient linkage discovery by limited probing. *Evolutionary Computation*, *12*(4), 517–545.

Henkelman, G., & Jónsson, H. (1999). A dimer method for finding saddle points on high dimensional potential surfaces using only first derivatives. *Journal of Chemical Physics*, *111*(15), 7010–7022. *ibid* **113**, 9978 (2000); *ibid* **115**, 9657 (2001).

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*. Ann Arbor, MI: University of Michigan Press.

Horn, J., Nafpliotis, N., & Goldberg, D. E. (1994). A niched Pareto genetic algorithm for multiobjective optimization. *Proceedings of the First IEEE Conference on Evolutionary Computation, IEEE World Congress on Computational Intelligence*, 82–87. (Also IlliGAL Report No. 93005).

Hutter, M. C., Reimers, J. R., & Hush, N. S. (2002). Modeling the bacterial photosynthetic reaction center. 1. Magnesium parameters for the semiempirical AM1 method developed using a genetic algorithm. *Journal of Physical Chemistry, B*, *102*(41), 8080–8090.

Ibaraki, T. (1997). Combinations with other optimization methods. In Bäck, T., Fogel, D. B., & Michalewicz, Z. (Eds.), *Handbook of Evolutionary Computation* (pp. D3:1–D3:2). Bristol and New York: Institute of Physics Publishing and Oxford University Press.

Jacobsen, J., Cooper, B. H., & Sethna, J. P. (1998). Simulations of energetic beam deposition: From picoseconds to seconds. *Physical Review B*, *58*(23), 15847–15865.

Jin, Y. (2005). A comprehensive survey of fitness approximation in evolutionary computation. *Soft Computing Journal*, *9*(1), 3–12.

Kargupta, H. (1996). The gene expression messy genetic algorithm. *Proceedings of the International Conference on Evolutionary Computation*, 814–819.

Kassner, M. E., & Pérez-Prado, M.-T. (2000). Five-power-law creep in single phase metals and alloys. *Progress in Materials Science*, *45*, 1–102.

Keijzer, M., O'Reilly, U.-M., Lucas, S. M., Costa, E., & Soule, T. (Eds.) (2004). *Genetic programming, proceedings of the 7th European conference, EuroGP 2004*, Volume 3003 of *LNCS*. Berlin: Springer-Verlag.

Khan, N. (2002). *Bayesian optimization algorithms for multiobjective and hierarchically difficult problems*. Master's thesis, University of Illinois at Urbana-Champaign, Urbana, IL. (Also IlliGAL Report No. 2003021).

Khan, N., Goldberg, D. E., & Pelikan, M. (2002). Multiobjective Bayesian optimization algorithms. *Proceedings of the 2002 Genetic and Evolutionary Computation Conference*, 684. (Also IlliGAL Report No. 2002009).

King, H. W. (1966). Quantitative size-factors for metallic solid solutions. *Journal of Materials Science*, *1*(1), 79–90.

Kinnear, K. E. (1996a). Alternatives in automatic function definition: A comparison of performance. In Kinnear, K. E. (Ed.), *Advances in Genetic Programming* (Chapter 6, pp. 119–141). Cambridge, MA: MIT Press.

Kinnear, K. E. (1996b). A perspective on the work in this book. In Kinnear, K. E. (Ed.), *Advances in Genetic Programming* (Chapter 1, pp. 3–19). Cambridge, MA: MIT Press.

Knjazew, D. (2002). *OmeGA: A competent genetic algorithm for solving permutation and scheduling problems*. Boston, MA: Kluwer Academic Publishers.

Koza, J. R. (1989). Hierarchical genetic algorithms operation on populations of computer programs. *Proceedings of the 11th International Joint Conference on Artificial Intelligence*, *1*, 768–774.

Koza, J. R. (1992). *Genetic programming: On the programming of computers by means of natural selection*. Cambridge, MA: MIT Press.

Koza, J. R. (1994). *Genetic programming II: Automatic discovery of reusable programs*. Cambridge, MA: MIT Press.

Koza, J. R., Bennett III, F. H., Andre, D., & Keane, M. A. (1999). *Genetic programming III: Darwinian invention and problem solving*. San Francisco, CA: Morgan Kaufmann.

Koza, J. R., Keane, M. A., Streeter, M. J., Mydlowec, W., Yu, J., & Lanza, G. (2003). *Genetic programming IV: Routine human-competitive machine intelligence*. Boston, MA: Kluwer Academic Publishers.

Krasnogor, N. (2002). *Studies on the theory and design space of memetic algorithms*. Doctoral dissertation, University of the West of England, Bristol, England.

Land, M. (1998). *Evolutionary algorithms with local search for combinatorial optimization*. Doctoral dissertation, University of California at San Diego, San Diego, CA.

Langdon, W. B., & Poli, R. (2002). *Foundations of genetic programming*. Springer-Verlag.

Larrañaga, P., & Lozano, J. A. (Eds.) (2002). *Estimation of distribution algorithms*. Boston, MA: Kluwer Academic Publishers.

Laumanns, M., & Ocenasek, J. (2002). Bayesian optimization algorithm for multi-objective optimization. *Parallel Problem Solving from Nature*, 298–307.

Levanov, N. A., Stepanyuk, V. S., Hergert, W., Bazhanov, D. I., Dederichs, P. H., Katsnelson, A., & Massobrio, C. (2000). Energetics of Co adatoms on the Cu(001) surface. *Physical Review B*, *61*(3), 2230.

Lima, C. F., Pelikan, M., Sastry, K., Butz, M. V., Goldberg, D. E., & Lobo, F. (2006). Substructural neighborhoods for local search in the Bayesian optimization algorithm. *Parallel Problem Solving from Nature (PPSN IX)*, 232–241. (Also IlliGAL Report No. 2006021).

Lima, C. F., Sastry, K., Goldberg, D. E., & Lobo, F. G. (2005). Combining competent crossover and mutation operators: A probabilistic model building approach. *Proceedings of the 2005 Genetic and Evolutionary Computation Conference*, 735–742. (Also IlliGAL Report No. 2005002).

Lin, S.-C., Goodman, E. D., & Punch, W. F. (1997). Investigating parallel genetic algorithms on job shop scheduling problem. *Sixth International Conference on Evolutionary Programming*, 383–393.

Looks, M. (2006). *Competent program evolution*. Doctoral dissertation, Washington University in St. Louis, St. Louis, MO. http://metacog.org/main.pdf.

Luke, S. (2000a, 8 July). Code growth is not caused by introns. In Whitley, D. (Ed.), *Late Breaking Papers at the 2000 Genetic and Evolutionary Computation Conference* (pp. 228–235). Las Vegas, Nevada, USA. (Preprint: http://www.cs.gmu.edu/~sean/papers/intronpaper.pdf).

Luke, S. (2000b). *Issues in scaling genetic programming: Breeding strategies, tree generation, and code bloat*. Doctoral dissertation, Department of Computer Science, University of Maryland, A. V. Williams Building, University of Maryland, College Park, MD 20742 USA. http://www.cs.gmu.edu/~sean/papers/thesis2p.pdf.

Luke, S. (2000c, September). Two fast tree-creation algorithms for genetic programming. *IEEE Transactions on Evolutionary Computation*, *4*(3), 274.

Mahfoud, S. W. (1994). Population size and genetic drift in fitness sharing. *Foundations of Genetic Algorithms*, *3*, 185–224. (Also IlliGAL Report No. 94005).

Manderick, B., & Spiessens, P. (1989). Fine-grained parallel genetic algorithms. *Proceedings of the Third International Conference on Genetic Algorithms*, 428–433.

Mazzone, G., & Rosato, V. (1997). Molecular-dynamics calculations of thermodynamic properties of metastable alloys. *Physical Review B*, *55*, 837–842.

Miller, B. L. (1997, May). *Noise, sampling, and efficient genetic algorithms*. Doctoral dissertation, University of Illinois at Urbana-Champaign, Urbana, IL. (Also IlliGAL Report No. 97001).

Miller, B. L., & Goldberg, D. E. (1995). Genetic algorithms, tournament selection, and the effects of noise. *Complex Systems*, *9*(3), 193–212. (Also IlliGAL Report No. 95006).

Miller, B. L., & Goldberg, D. E. (1996a). Genetic algorithms, selection schemes, and the varying effects of noise. *Evolutionary Computation*, *4*(2), 113–131. (Also IlliGAL Report No. 95009).

Miller, B. L., & Goldberg, D. E. (1996b). Optimal sampling for genetic algorithms. *Intelligent Engineering Systems through Artificial Neural Networks*, *6*, 291–297.

Montana, D. J. (1995). Strongly typed genetic algorithm. *Evolutionary Computation*, *3*(2), 199–230. (Also BBN Technical Report No. 7866).

Moscato, P. (1989). *On evolution, search, optimization, genetic algorithms and martial arts: Towards memetic algorithms* (Technical Report C3P 826). Pasadena, CA: Caltech Concurrent Computation Program, California Institute of Technology.

Mühlenbein, H., & Paaß, G. (1996). From recombination of genes to the estimation of distributions I. Binary parameters. *Parallel Problem Solving from Nature*, *4*, 178–187.

Mühlenbein, H., & Schlierkamp-Voosen, D. (1993). Predictive models for the breeder genetic algorithm: I. continuous parameter optimization. *Evolutionary Computation*, *1*(1), 25–49.

Mukherjee, A. K. (1975). High-temperature creep. In Asenault, R. J. (Ed.), *Treatise on Materials Science and Technology*, Volume 6 (pp. 164–221). Academic Press.

Mukherjee, T., Fedder, G., Ramaswamy, D., & White, J. (2000). Emerging simulation approaches for micromachined devices. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Devices*, *19*(12), 1572–1589.

Munetomo, M., & Goldberg, D. E. (1999). Linkage identification by non-monotonicity detection for overlapping functions. *Evolutionary Computation*, *7*(4), 377–398.

Ocenasek, J. (2002). *Parallel estimation of distribution algorithms.* Doctoral dissertation, Brno University of Technology, Brno, Czech Republic.

O'Reilly, U.-M., & Goldberg, D. E. (1998). How fitness structure affects subsolution acquisition in genetic programming. *Genetic Programming 1998: Proceedings of the Third Annual Conference*, 269–277.

O'Reilly, U.-M., & Oppacher, F. (1995). The troubling aspects of a building block hypothesis for genetic programming. *Foundations of Genetic Algorithms*, *3*, 73–88.

O'Reilly, U.-M., Yu, T., Riolo, R. L., & Worzel, B. (Eds.) (2004). *Genetic programming theory and practice II.* Boston, MA, USA: Springer.

Orvosh, D., & Davis, L. (1993). Shall we repair? Genetic algorithms, combinatorial optimization, and feasibility constraints. *Proceedings of the Fifth International Conference on Genetic Algorithms*, 650.

Owens, J. M. (2004). *Theoretical studies of the solvation, dynamics, and photochemistry of ethylene, retinal protonated Schiff base, oligocellulose, and GD(III) clusters.* Doctoral dissertation, University of Illinois at Urbana-Champaign, Department of Chemistry, Urbana, IL. (Preprint:).

Padilla, H. A., Harnish, S. F., Gore, B. E., Beaudoin, A. J., Dantzig, J. A., Robertson, I. M., & Weiland, H. (2003). High temperature deformation and hot rolling of AA7055. *Proceedings of the First International Symposium on Metallurgical Modeling of Aluminum Alloys*, 1–8.

Pelikan, M. (2005). *Hierarchical Bayesian optimization algorithm: Toward a new generation of evolutionary algorithm.* Berlin: Springer Verlag.

Pelikan, M., & Goldberg, D. E. (2001). Escaping hierarchical traps with competent genetic algorithms. *Proceedings of the 2001 Genetic and Evolutionary Computation Conference*, 511–518. (Also IlliGAL Report No. 2000020).

Pelikan, M., Goldberg, D. E., & Cantú-Paz, E. (2000). Linkage learning, estimation distribution, and Bayesian networks. *Evolutionary Computation*, *8*(3), 314–341. (Also IlliGAL Report No. 98013).

Pelikan, M., Goldberg, D. E., & Lobo, F. (2002). A survey of optimization by building and using probabilistic models. *Computational Optimization and Applications*, *21*, 5–20. (Also IlliGAL Report No. 99018).

Pelikan, M., Goldberg, D. E., & Sastry, K. (2001). Bayesian optimization algorithm, decision graphs, and Occam's razor. *Proceedings of the 2001 Genetic and Evolutionary Computation Conference*, 519–526. (Also IlliGAL Report No. 2000020).

Pelikan, M., Kvasnicka, V., & Pospichal, J. (1997). Read's linear codes and genetic programming. *Genetic Programming 1997: Proceedings of the Second Annual Conference*, 268.

Pelikan, M., & Sastry, K. (2004). Fitness inheritance in the Bayesian optimization algorithm. *Proceedings of the 2004 Genetic and Evolutionary Computation Conference*, *2*, 48–59. (Also IlliGAL Report No. 2004009).

Pelikan, M., Sastry, K., & Cantú-Paz, E. (Eds.) (2006). *Scalable optimization via probabilistic modeling: Algorithms to applications*. Berlin: Springer.

Pelikan, M., Sastry, K., & Goldberg, D. E. (2003). Scalability of the Bayesian optimization algorithm. *International Journal of Approximate Reasoning*, *31*(3), 221–258. (Also IlliGAL Report No. 2001029).

Pelikan, M., Sastry, K., & Goldberg, D. E. (2005). Multiobjective hBOA, clustering, and scalability. *Proceedings of the 2005 Genetic and Evolutionary Computation Conference*, 663–670. (Also IlliGAL Report No. 2005005).

Pelikan, M., Sastry, K., & Goldberg, D. E. (2006). Multiobjective estimation of distribution algorithms. In Pelikan, M., Sastry, K., & Cantú-Paz, E. (Eds.), *Scalable Optimization via Probabilistic Modeling: From Algorithms to Applications* (Chapter 10, pp. 223–248). Berlin: Springer.

Pettey, C. C., Leuze, M. R., & Grefenstette, J. J. (1987). A parallel genetic algorithm. *Proceedings of the Second International Conference on Genetic Algorithms*, 155–161.

Picu, R. C. (2003). Foreword: Multiscale computational materials science—linking discrete and continuum models. *International Journal for Multiscale Computational Engineering*, *1*(1), vii–viii.

Poli, R. (2000). Recursive conditional schema theorem, convergence and population sizing in genetic algorithms. *Foundations of Genetic Algorithms, 6*.

Prügel-Bennet, A., & Shapiro, J. L. (1994). An analysis of a genetic algorithm using statistical mechanics. *Physics Review Letters*, *72*(9), 1305–1309.

Quenneville, J., Ben-Nun, M., & Martinez, T. J. (2001). Photochemistry from first principles—advances and future prospects. *Journal of Photochemistry and Photobiology A: Chemistry*, *144*, 229–235.

Radloff, W., Stert, V., Freudenberg, T., Hertel, I. V., Jouvet, C., Dedonder-Lardeux, C., & Solgadi, D. (1997). Internal conversion in highly excited benzene and benzene dimer: Femtosecond time-resolved photoelectron spectroscopy. *Chemical Physics Letters*, *21*(1–3), 20–26.

Ramsey, C. L., & Grefenstette, J. J. (1993). Case-Based initialization of genetic algorithms. *Proceedings of the Fifth International Conference on Genetic Algorithms*, 84–91.

Ratle, A., & Sebag, M. (2001). Grammar-guided genetic programming and dimensional consistency: Application to non-parametric identification in mechanics. *Applied Soft Computing*, *1*, 105–118.

Rattray, M., & Shapiro, J. L. (1997). Noisy fitness evaluation in genetic algorithms and the dynamics of learning. *Foundations of Genetic Algorithms*, *4*, 117–139.

Rechenberg, I. (1973). *Evolutionsstrategie: Optimierung technischer systeme nach prinzipien der biologischen evolution*. Stuttgart: Frommann-Holzboog.

Reed, P. (2002). *Striking the balance: Long-term groundwater monitoring design for multiple conflicting objectives*. Doctoral dissertation, University of Illinois, Urbana, IL.

Reed, P., Minsker, B. S., & Goldberg, D. E. (2000). Designing a competent simple genetic algorithm for search and optimization. *Water Resources Research*, *36*(12), 3757–3761.

Reeves, C. (1993). Using genetic algorithms with small populations. *Proceedings of the Fifth International Conference on Genetic Algorithms*, 92–99.

Riolo, R. L., & Worzel, B. (Eds.) (2003). *Genetic programming theory and practice*. Genetic Programming Series. Boston, MA, USA: Kluwer. Series Editor - John Koza.

Rissanen, J. J. (1978). Modeling by shortest data description. *Automatica*, *14*, 465–471.

Robertson, G. G. (1987). Parallel implementation of genetic algorithms in a classifier system. *Proceedings of the Second International Conference on Genetic Algorithms*, 140–147.

Rogers, A., & Prügel-Bennet, A. (1999). Modeling the dynamics of a steady state genetic algorithm. *Foundations of Genetic Algorithms*, *5*, 57–68.

Rosca, J. P. (1997). Analysis of complexity drift in genetic programming. *Genetic Programming 1997: Proceedings of the Second Annual Conference*, 286–294.

Rosca, J. P., & Ballard, D. H. (1996). Discovery of subroutines in genetic programming. In Angeline, P. J., & Kinnear, K. E. (Eds.), *Advances in Genetic Programming 2* (Chapter 9, pp. 177–202). Cambridge, MA: MIT Press.

Rossi, I., & Truhlar, D. G. (1995). Parameterization of NDDO wavefunctions using genetic algorithms. An evolutionary approach to parameterizing potential energy surfaces and direct dynamics calculations for organic reactions. *Chemical Physics Letters*, *233*, 231–236.

Rothlauf, F. (2002). *Representations for genetic and evolutionary algorithms*. Berlin: Springer Verlag.

Roussel, J.-M. (2005, April). Private communication.

Rudolph, G. (2000). Takeover times and probabilities of non-generational selection rules. *Proceedings of the 2000 Genetic and Evolutionary Computation Conference*, 903–910.

Ryan, C., Collins, J., & O'Neill, M. (1998). Grammatical evolution: Evolving computer programs for an arbitrary language. *Genetic Programming, Proceedings of the First European Conference, EuroGP'98*, *1391*, 83–96. (LNCS).

Sakamoto, Y., & Goldberg, D. E. (1997). Takeover time in a noisy environment. *Proceedings of the Seventh International Conference on Genetic Algorithms*, 160–165.

Salman, A. A., Mehrotra, K., & Mohan, C. K. (2000). Linkage crossover operator. *Evolutionary Computation*, *8*(3), 341–370.

Sałustowicz, R. P., & Schmidhuber, J. (1997). Probabilistic incremental program evolution. *Evolutionary Computation*, *5*(2), 123–141.

Sastry, K. (2001). *Evaluation-relaxation schemes for genetic and evolutionary algorithms.* Master's thesis, University of Illinois at Urbana-Champaign, Urbana, IL. (Also IlliGAL Report No. 2002004).

Sastry, K., & Goldberg, D. E. (2001). Modeling tournament selection with replacement using apparent added noise. *Intelligent Engineering Systems Through Artificial Neural Networks*, *11*, 129–134. (Also IlliGAL Report No. 2001014).

Sastry, K., & Goldberg, D. E. (2002). *Analysis of mixing in genetic algorithms: A survey* (IlliGAL Report No. 2002012). Urbana, IL: University of Illinois at Urbana-Champaign.

Sastry, K., & Goldberg, D. E. (2003a). Probabilistic model building and competent genetic programming. In Riolo, R. L., & Worzel, B. (Eds.), *Genetic Programming Theory and Practice* (Chapter 13, pp. 205–220). Kluwer. (Also IlliGAL Report No. 2003013).

Sastry, K., & Goldberg, D. E. (2003b). Scalability of selectorecombinative genetic algorithms for problems with tight linkage. *Proceedings of the 2003 Genetic and Evolutionary Computation Conference*, 1332–1344. (Also IlliGAL Report No. 2002013).

Sastry, K., & Goldberg, D. E. (2004a). Designing competent mutation operators via probabilistic model building of neighborhoods. *Proceedings of the 2004 Genetic and Evolutionary Computation Conference*, *2*, 114–125. (Also IlliGAL Report No. 2004006).

Sastry, K., & Goldberg, D. E. (2004b). Let's get ready to rumble: Crossover versus mutation head to head. *Proceedings of the 2004 Genetic and Evolutionary Computation Conference*, *2*, 126–137. (Also IlliGAL Report No. 2004005).

Sastry, K., & Goldberg, D. E. (2007). Let's get ready to rumble redux: Crossover versus mutation head to head on exponentially scaled problems. *Proceedings of the 2007 Genetic and Evolutionary Computation Conference*, 1380–1387. (Also IlliGAL Report No. 2007005).

Sastry, K., Goldberg, D. E., & Kendall, G. (2005). Genetic algorithms: A tutorial. In Burke, E., & Kendall, G. (Eds.), *Introductory Tutorials in Optimization, Search, and Decision Support Methodologies* (Chapter 4, pp. 97–125). Berlin: Springer. http://www.asap.cs.nott.ac.uk/publications/pdf/gxk_introsch4.pdf.

Sastry, K., Goldberg, D. E., & Llorà, X. (2007). Towards billion bit optimization via efficient estimation of distribution algorithms. *Proceedings of the 2007 Genetic and Evolutionary Computation Conference*, 577–584. (Also IlliGAL Report No. 2007007).

Sastry, K., Goldberg, D. E., & Pelikan, M. (2001). Don't evaluate, inherit. *Proceedings of the 2001 Genetic and Evolutionary Computation Conference*, 551–558. (Also IlliGAL Report No. 2001013).

Sastry, K., Johnson, D. D., Goldberg, D. E., & Bellon, P. (2004). Genetic programming for multiscale modeling. *International Journal of Multiscale Computational Engineering*, *2*(2), 239–256.

Sastry, K., Johnson, D. D., Goldberg, D. E., & Bellon, P. (2005). Genetic programming for multi-timescale modeling. *Physical Review B*, *72*, 085438.

Sastry, K., Johnson, D. D., Thompson, A. L., Goldberg, D. E., Martinez, T. J., Leiding, J., & Owens, J. (2006). Multiobjective genetic algorithms for multiscaling excited state direct dynamics in photochemistry. *Proceedings of the 2006 Genetic and Evolutionary Computation Conference*, 1745–1752. (Also IlliGAL Report No. 2006005).

Sastry, K., Johnson, D. D., Thompson, A. L., Goldberg, D. E., Martinez, T. J., Leiding, J., & Owens, J. (2007). Optimization of semiempirical quantum chemistry methods via multi-objective genetic algorithms: Accurate photochemistry for larger molecules and longer time scales. *Materials and Manufacturing Processes*, *22*(5), 553–561.

Sastry, K., Lima, C. F., & Goldberg, D. E. (2006). Evaluation relaxation using substructural information and linear estimation. *Proceedings of the 2006 Genetic and Evolutionary Computation Conference*, 419–426. (Also IlliGAL Report No. 2006003).

Sastry, K., O'Reilly, U.-M., & Goldberg, D. (2003). Building-block supply in genetic programming. In Riolo, R., & Worzel, B. (Eds.), *Genetic Programming Theory and Practice* (Chapter 9, pp. 155–172). Boston, MA: Kluwer Academic Publishers. (Also IlliGAL Report No. 2003012).

Sastry, K., O'Reilly, U.-M., & Goldberg, D. (2004). Population sizing for genetic-programming based on decision-making. In O'Reilly, U.-M., Yu, T., Riolo, R., & Worzel, B. (Eds.), *Genetic Programming Theory and Practice II* (Chapter 4, pp. 49–66). Boston, MA: Kluwer Academic Publishers. (Also IlliGAL Report No. 2004028).

Sastry, K., Pelikan, M., & Goldberg, D. E. (2004). Efficiency enhancement of genetic algorithms via building-block-wise fitness estimation. *Proceedings of the IEEE International Conference on Evolutionary Computation*, 720–727. (Also IlliGAL Report No. 2004010).

Sastry, K., Pelikan, M., & Goldberg, D. E. (2005). Limits of scalability of multiobjective estimation of distribution algorithms. *Proceedings of the Congress on Evolutionary Computation*, 217–2224. (Also IlliGAL Report No. 2005004).

Sastry, K., Pelikan, M., & Goldberg, D. E. (2006). Efficiency enhancement of estimation of distribution algorithms. In Pelikan, M., Sastry, K., & Cantú-Paz, E. (Eds.), *Scalable Optimization via Probabilistic Modeling: From Algorithms to Applications* (Chapter 7, pp. 161–185). Berlin: Springer.

Schwefel, H.-P. (1977). Numerische optimierung von computer-modellen mittels der evolutionsstrategie. *Interdisciplinary Systems Research*, *26*, .

Shan, Y., McKay, R. I., Essam, D., & Abbass, H. A. (2006). A survey of probabilistic model building genetic programming. In Pelikan, M., Sastry, K., & Cantú-Paz, E. (Eds.), *Scalable Optimization via Probabilistic Modeling: From Algorithms to Applications* (Chapter 6, pp. 121–160). Berlin: Springer.

Shapiro, J. L., Prügel-Bennet, A., & Rattray, M. (1994). A statistical mechanical formulation of the dynamics of genetic algorithms. In *Lecture Notes in Computer Science*, Volume 865 (pp. 17–27). Berlin: Springer-Verlag.

Sinha, A. (2003a). *Designing efficient genetic and evolutionary hybrids*. Master's thesis, University of Illinois at Urbana-Champaign, Urbana, IL. (Also IlliGAL Report No. 2003020).

Sinha, A. (2003b). *A survey of hybrid genetic and evolutionary algorithms* (IlliGAL Report No. 2003004). Urbana, IL: University of Illinois at Urbana-Champaign, Illinois Genetic Algorithms Laboratory.

Smith, J., & Vavak, F. (1999). Replacement strategies in steady state genetic algorithms: Static environments. *Foundations of Genetic Algorithms*, *5*, 219–234.

Smith, R., Dike, B., & Stegmann, S. (1995). Fitness inheritance in genetic algorithms. In *Proceedings of the ACM Symposium on Applied Computing* (pp. 345–350). New York, NY, USA: ACM.

Sørensen, M. R., & Voter, A. F. (2000). Temperature-accelerated dynamics for simulation of infrequent events. *Journal of Chemical Physics*, *112*(21), 9599–9606.

Soule, T. (2002). Exons and code growth in genetic programming. *Genetic Programming, Proceedings of the 5th European Conference, EuroGP 2002*, *2278*, 142–151.

Soule, T. (2003). Operator choice and the evolution of robust solutions. In Riolo, R. L., & Worzel, B. (Eds.), *Genetic Programming Theory and Practice* (Chapter 16, pp. 257–270). Kluwer.

Soule, T., & Heckendorn, R. B. (2002, September). An analysis of the causes of code growth in genetic programming. *Genetic Programming and Evolvable Machines*, *3*(3), 283–309.

Spears, W. (1997). Recombination parameters. In Bäck, T., Fogel, D. B., & Michalewicz, Z. (Eds.), *The Handbook of Evolutionary Computation* (Chapter E1.3, pp. E1.3:1–E1.3:13). Philadelphia, PA: IOP Publishing Ltd. and Oxford University Press.

Spector, L., Langdon, W. B., O'Reilly, U.-M., & Angeline, P. J. (Eds.) (1999). *Advances in genetic programming 3*. Cambridge, MA: MIT Press.

Srinivas, N., & Deb, K. (1995). Multiobjective function optimization using nondominated sorting genetic algorithms. *Evolutionary Computation*, *2*(3), 221–248.

Srivastava, R. (2002). *Time continuation in genetic algorithms*. Master's thesis, University of Illinois at Urbana-Champaign, Urbana, IL.

Srivastava, R., & Goldberg, D. E. (2001). Verification of the theory of genetic and evolutionary continuation. *Proceedings of the 2001 Genetic and Evolutionary Computation Conference*, 551–558. (Also IlliGAL Report No. 2001007).

Steiner, M. M., & Genilloud, P.-A. (1998). Simple bias potential for boosting molecular dynamics with the hyperdynamics scheme. *Physical Review B*, *57*(17), 10236–10239.

Stepanyuk, V. S., Bazhanov, D. I., Baranov, A. N., Hergert, W., Dederichs, P. H., & Kirschner, J. (2000). Strain relief and island shape evolution in heteroepitaxial metal growth. *Physical Review B*, *62*, 15398–15401.

Stepanyuk, V. S., Bazhanov, D. I., Hergert, W., & Kirschner, J. (2001). Strain and adatom motion on mesoscopic islands. *Physical Review B*, *63*, 153406.

Stepanyuk, V. S., Tsivline, D. V., Bazhanov, D. I., Hergert, W., & Katsnelson, A. A. (2001). Burrowing of Co clusters on the Cu(001) surface: Atomic-scale calculations. *Physical Review B*, *63*, 235406.

Stephens, C., & Waelbroeck, H. (1999). Schemata evolution and building blocks. *Evolutionary Computation*, *7*(2), 109–124.

Stewart, J. J. P. (1989). Optimization of parameters for semiempirical methods 1. Method. *Journal of Computational Chemistry*, *10*(2), 209–220.

Stewart, J. J. P. (1999). MOPAC 2000. Fujitsu Limited, Tokyo, Japan.

Tanese, R. (1989). *Distributed genetic algorithms for function optimization.* Doctoral dissertation, University of Michigan, Ann Arbor, MI. (University microfilms no. 8520908).

Thierens, D. (1999). Scalability problems of simple genetic algorithms. *Evolutionary Computation*, *7*(4), 331–352.

Thierens, D., Beyer, H.-G., Bongard, J., Branke, J., Clark, J. A., Cliff, D., Congdon, C. B., Deb, K., Doerr, B., Kovacs, T., Kumar, S., Miller, J. F., Moore, J., Neumann, F., Pelikan, M., Poli, R., Sastry, K., Stanley, K. O., Stutzle, T., Watson, R. A., & Wegener, I. (Eds.) (2007). *Gecco 2007: Proceedings of the 9th annual conference on genetic and evolutionary computation*. New York, NY: ACM Press.

Thierens, D., & Goldberg, D. E. (1993). Mixing in genetic algorithms. Proceedings of the Fifth International Conference On Genetic Algorithms, 38–45.

Thierens, D., & Goldberg, D. E. (1994a). Convergence models of genetic algorithm selection schemes. *Parallel Problem Solving from Nature*, *3*, 116–121.

Thierens, D., & Goldberg, D. E. (1994b). Elitist recombination: An integrated selection recombination GA. *Proceedings of the First IEEE Conference on Evolutionary Computation*, 508–512.

Thierens, D., Goldberg, D. E., & Pereira, A. G. (1998). Domino convergence, drift, and the temporal-salience structure of problems. *Proceedings of the IEEE International Conference on Evolutionary Computation*, 535–540.

Tiley, J., Banerjee, R., Searles, T., Kar, S., & Fraser, H. (2004). Modeling the relationships between microstructural parameters and tensile properties in $Ti_6Al_4V$ using neural networks and fuzzy-logic models. *Titatnium 2003: Proceedings of the Tenth World Conference on Titanium*, (To appear).

Toniolo, A., Levine, B., Thompson, A. L., Quenneville, J., Ben-Num, M., Owens, J., Olsen, S., Manohar, L., & Martinez, T. J. (2005). Photochemistry from first principles and direct dynamics. In Kutateladze, A. (Ed.), *Computational methods in organic photochemistry* (pp. 167–234). New York, NY: Marcel-Dekker.

Toniolo, A., Thompson, A. L., & Martinez, T. J. (2004). Excited state direct dynamics of benzene with reparameterized multireference semiempirical configuration interaction methods. *Chemical Physics*, *304*, 133–145.

Trushin, O., Karim, A., Kara, A., & Rahman, T. S. (2005). Self-learning kinetic monte carlo method: Application to Cu(111). *Physical Review B*, *72*, 115401.

Tsuji, M., Munetomo, M., & Akama, K. (2006). Linkage identification by fitness difference clustering. *Evolutionary Computation*, *14*(4), 383–409.

Van der Ven, A., & Ceder, G. (2001). First-principles theory of ionic diffusion with nondilute carriers. *Physical Review B*, *64*(18), 184307.

Van Veldhuizen, D. A., & Lamont, G. B. (2000). Multiobjective optimization with messy genetic algorithms. *Proceedings of the 2000 ACM Symposium on Applied Computing*, 470–476.

Voigt, H.-M., Mühlenbein, H., & Schlierkamp-Voosen, D. (1996). The response to selection equation for skew fitness distributions. *Proceedings of the International Conference on Evolutionary Computation*, 820–825.

Voter, A. F. (1997). Hyperdynamics: Accelerated molecular dynamics of infrequent events. *Physical Review Letters*, *78*(20), 3908–3911.

Voter, A. F. (1998). Parallel replica method for dynamics of infrequent events. *Physical Review B*, *57*(22), R13985–R13988.

Voter, A. F., Montalenti, F., & Germann, T. C. (2002). Extending the time scale in atomistic simulation of materials. *Annals of Review in Materials Research*, *32*, 321–346.

Watson, R. A., & Pollack, J. B. (1999). Incremental commitment in genetic algorithms. *Proceedings of Genetic and Evolutionary Computation Conference 1999 (GECCO-1999)*, 710–717.

Werner, H.-J., Knowles, P. J., Lindh, R., Manby, F. R., Schtz, M., Celani, P., Korona, T., Rauhut, G., Amos, R. D., Bernhardsson, A., Berning, A., Cooper, D. L., Deegan, M. J. O., Dobbyn, A. J., Eckert, F., Hampel, C., Hetzer, G., Lloyd, A. W., McNicholas, S. J., Meyer, W., Mura, M. E., Nicklaß, A., Palmieri, P., Pitzer, R., Schumann, U., Stoll, H., Stone, A. J., Tarroni, R., & Thorsteinsson, T. (2002). MOLPRO, version 2002.2, a package of *ab initio* programs. http://www.molpro.net.

Whigham, P. A. (1995). A schema theorem for context-free grammars. *Proceedings of the 1995 IEEE Conference on Evolutionary Computation*, *1*, 178–181.

Yu, T.-L. (2006). *A matrix approach for finding extrema: Problems with modularity, hierarchy, and overlap*. Doctoral dissertation, University of Illinois at Urbana-Champaign, Urbana, IL. (Also IlliGAL Report No. 2007012).

Yu, T.-L., Sastry, K., Goldberg, D. E., & Pelikan, M. (2007). Population sizing for entropy-based model building in genetic algorithms. *Proceedings of the 2007 Genetic and Evolutionary Computation Conference*, 601–608. (Also IlliGAL Report No. 2006020).

Zitzler, E., Laumanns, M., & Thiele, L. (2001). SPEA2: Improving the strength Pareto evolutionary algorithm. *Proceedings of Evolutionary Methods for Design, Optimization, and Control*, 95–100. (Also TIK-ETH Report No. 103).

Zydallis, J. B., Van Veldhuizen, D. A., & Lamont, G. B. (2001). A statistical comparison of multiobjective evolutionary algorithms including the MOMGA-II. *Proceedings of the First International Conference on Evolutionary Multi-Criterion Optimization*, 226–240.

# Vita

**KUMARA SASTRY**
*Industrial and Enterprise Systems Engineering*
*University of Illinois at Urbana-Champaign*
*604 E. White St. # 4, Champaign IL 61820*
*Phone: 217-355-1122* • *Mobile: 217-417-0560*
*E-mail:* `kumara@kumarasastry.com` • *WWW:* `http://www.kumarasastry.com`

## Research Interests

Genetic algorithms, multiscale modeling in materials science and chemistry, principled efficiency enhancement, large-scale optimization, stochastic optimization, machine learning.

## Education

**PhD in Systems and Entrepreneurial Engineering,** University of Illinois at Urbana-Champaign, Urbana IL, anticipated graduation date: October 2007.
*Dissertation Title:* "Genetic algorithms and genetic programming for multiscale modeling: Applications in materials science and chemistry and advances in scalability."
*Thesis Advisors:* David E. Goldberg and Duane D. Johnson.

**MS in General Engineering,** University of Illinois at Urbana-Champaign, Urbana IL, 2002.
*Thesis Title:* "Evaluation relaxation schemes in genetic and evolutionary algorithms."
*Thesis Advisor:* David E. Goldberg.

**MSc (Hons) in Chemistry and ME in Chemical Engineering,** Birla Institute of Technology and Science, Pilani, INDIA, 1999.
*Thesis Advisor:* I. J. Nagrath.

## Research Experience

**Graduate Research Assistant: Jan 2000 - present**
*Illinois Genetic Algorithms Laboratory, University of Illinois at Urbana-Champaign, IL.*

- Research on the analysis, design and development of *competent* and *efficient* genetic algorithms and genetic programming with particular application to search, optimization, and machine learning problems in materials science and chemistry, especially in the field of multiscale materials modeling.

- Developed *evaluation-relaxation* schemes and other principled *efficiency-enhancement* methods that yield *super-multiplicative* speedups for genetic algorithms and other optimization methods.
- Implemented a SIMD-based fully-parallelized efficient genetic algorithm that could solve very large scale problems with up to **32 million** variables to full convergence, and **over a billion** variables to relaxed convergence (Altivec for IBM powerpc, SSE for Intel and AMD processors, MPI, C/C++).
- Derived population-sizing and scalability models for genetic programming.
- Integrated genetic algorithms—both serial and parallel versions—with FOX, an automated army course of action tool-box (C++, MPI).
- Developed a generic optimization tool-box specifically tailored for cooling-system design optimization for Caterpillar (C++, Matlab). Conducted an exhaustive analysis of the behavior of simple genetic algorithms in the design of simple cooling systems.

**Graduate Research Assistant: Jan 2000 - present**

*Materials Computation Center, University of Illinois at Urbana-Champaign, IL.*

- Developed a multiscaling approach for *fast* and *accurate* quantum-chemistry simulations. Used multiobjective genetic algorithms for optimizing parameters for semiempirical methods using limited *ab initio* and experimental data.
- Developed a multiscaling approach for materials kinetics simulation. Used genetic programming for symbolically regressing an *inline barrier-energy function* to bridge kinetic Monte Carlo and molecular dynamics.

**Graduate Research Assistant: Jan 1999 - Dec 1999**

*Department of Chemical Engineering, State University of New York at Buffalo, NY.*
Non-linear optimization (MINOS, GAMS, NPSOL) of hybrid power cycles.

**Project Assistant: Jul 1996 - Jul 1998**

*Center for Robotics and Intelligent Systems, Birla Institute of Tech. and Science, Pilani.*

- Developed a genetic algorithms and fuzzy-logic based pH control system (C).
- Developed a solution manual for a widely used textbook: *Control Systems* by I. J. Nagrath (Matlab).

## Teaching Experience

**Graduate Teaching Assistant: Jan 2006 - May 2006**

*Industrial and Enterprise Systems Engineering, University of Illinois at Urbana-Champaign.*
Graded homework, projects, and exams, handled problem and project sessions, and held office hours for the undergraduate course *Analytical Methods for Uncertainty/Analysis of Data* (GE 331/IE 300).

**Teaching Assistant: Aug 1998 - Dec 1998**

*Birla Institute of Technology & Science, Pilani.*
Handled the laboratory for process control course (senior-level course).

## Professional Experience

**Consultant: Apr 2007 - present**

*Quelab Laboratories Inc., Montreal (Quebec), Canada.*

Quelab produces and distributes bacteriological culture media, as well as laboratory materials used by health professionals. Its goal is to develop new culture media to facilitate clinical diagnoses and enhance the quality of bacterial analyses.

**Consultant: Sep 2004 - Sep 2007**

*Nextumi Inc., Enterprise Works, Champaign IL.*

Nextumi is a web 2.0 company that simplifies sharing of photos, videos, and other contents and contacts among people across different devices. As one of the first employees of Nextumi, Set up the research group and facilities, hired and supervised programmers and web designers, and planned and coordinated projects for the proof-of-concept prototype. Instrumental in design an development of innovations including the core technology of Nextumi. Currently, coordinating research efforts including projects related to the upcoming alpha an beta releases.

**Consultant: Aug 2002 - Jan 2004**

*Schema Inc., New Jersey. Schema LTD, Herzlia, Israel.*

Schema is a leading solution provider to wireless optimization problems. Consulted on competent and efficient genetic and evolutionary algorithms for practical solutions to wireless search and optimization problems. Invented new recombination and mutation operators that sped up GAs by a factor of 30–50 and improved solution quality by factor of 2–5.

**Summer Trainee: May 1995 - Jul 1995**

*Indira Gandhi Center for Atomic Research, Kalpakkam, India.*

Developed a Model of Crossflow Ultrafiltration for the removal of Uranium from radioactive waste (C, Matlab).

## Administrative Experience

**Student Lab Director: Aug 2002 - present**

*Illinois Genetic Algorithms Laboratory, University of Illinois at Urbana-Champaign, IL.*

Managing daily operation of Illinois genetic algorithms laboratory. Interacting with graduate students, visiting scholars and professors, and industrial contacts on a regular basis. Designed and led the assembling and upgrading efforts of a 93-node diskless PC cluster. Responsible for regular upgrading, and maintaining of the cluster, workstations, and servers. Coordinating and supervising system- and web- administrators, and librarian.

**Student team leader: Jan 2001 - Aug 2001**

*Illinois Genetic Algorithms Laboratory, University of Illinois at Urbana-Champaign, IL.*

Led a team of 7 that developed advanced cooling system design-optimization tool-box for Caterpillar. Developed a GUI-based application-independent advanced genetic algorithm tool-box which included advanced features such as niching methods, multiobjective operators, and local-search methods (C++).

## Awards & Grants

- **Best paper award** (with D. E. Goldberg and X. Llorà), estimation of distribution algorithms track, Genetic and Evolutionary Computation Conference, 2007.
- **Bronze "Humies" award** (with J. Bacardit, M. Stout, J. D. Hirst, X. Llorà, and N. Krasnogor), Human Competitive Results at the Genetic and Evolutionary Computation Conference (ACM SIG conference), 2007.
- **Best paper award nominee** (with M. Pelikan and D. E. Goldberg), genetic algorithms track, Genetic and Evolutionary Computation Conference, 2007.
- **Best paper award nominee** (with T.-L. Yu, D. E. Goldberg and M. Pelikan), estimation of distribution algorithms track, Genetic and Evolutionary Computation Conference, 2007.
- **Finalist, Lemelson-Illinois student prize**. Annual award for the most inventive student at the University of Illinois, 2007.
- **Silver "Humies" award** (with D.D. Johnson, A. L. Thompson, D. E. Goldberg, T. J. Martinez, J. Leiding, and J. Owens), Human Competitive Results at the Genetic and Evolutionary Computation Conference (ACM SIG conference), 2006.
- **Best paper award** (with D.D. Johnson, A. L. Thompson, D. E. Goldberg, T. J. Martinez, J. Leiding, and J. Owens), real world applications track, Genetic and Evolutionary Computation Conference (ACM SIG conference), 2006.
- **Research Grant FA9550-06-1-0096** (with D. E. Goldberg and M. Pelikan). Air Force Office of Scientific Research, Air Force Materiel Command, USAF.
- The paper "Genetic programming for multiscale modeling" co-authored with D. D. Johnson, D. E. Goldberg, and P. Bellon was **chosen by American Institute of Physics (AIP) editors as a focused article of frontier research** in the *Virtual Journal of Nanoscale Science and Technology*, 12(9), 2005.
- **Best paper award nominee** (with H. A. Abbass, D. E. Goldberg, and D.D. Johnson), estimation of distribution algorithms track, Genetic and Evolutionary Computation Conference, 2005.
- **Best paper award nominee** (with D. E. Goldberg), genetic algorithms track, Genetic and Evolutionary Computation Conference, 2003.
- **Best paper award** (with M. V. Butz, and D. E. Goldberg), learning classifier systems track, Genetic and Evolutionary Computation Conference, 2003.
- **Computational Science and Engineering Fellow**, University of Illinois, 2002-2003.
- **William A. Chittenden Award for outstanding master of science graduate in General Engineering**, 2001.
- Best theme oriented model award. APOGEE, all India science exhibition, 1997.
- Best exhibition award, APOGEE, all India science exhibition, 1997.

## Patents

**Methods for efficient solution to large-scale search and optimization problems.**
Inventors: Sastry, K., Goldberg, D. E., Llorà, X.
Status: Filed invention disclosure to office of technology management.

**Quantum chemistry simulations using optimization methods.**
Inventors: Sastry, K., Thompson, A., Johnson, D.D., Martinez, T. J., Goldberg, D. E.
Status: Pending.

**Methods and systems for interactive computing.**
Inventors: Llorà, X., Sastry, K., Goldberg, D. E.
Status: Pending.

**Adaptive optimization methods.**
Inventors: Lima, C. F., Sastry, K., Goldberg, D. E., Lobo, F. G.
Status: Pending.

**Methods for efficient solution set optimization.**
Inventors: Sastry, K., Pelikan, M., Goldberg, D. E.
Status: Pending (US patent application 20060212279).

## Current Research

- Practical understanding of genetic algorithms (GAs), genetic programming (GP), and genetics-based machine learning algorithms (GBML).
- Design and analysis of *competent* Genetic and evolutionary algorithms that solve *hard* problems, quickly, reliably, and accurately.
  - Extensions to non-binary problem domains such as integer, real, and program domains.
  - Competent search methods for evolving rules in learning classifier systems
- Solving search, optimization, and machine-learning problems in material science and chemistry, especially in the area of multiscale modeling.
- Principled design of efficiency-enhancement techniques such as *evaluation relaxation*, *time continuation*, *parallelization*, and *hybridization*.
- Solving large-scale optimization problems with millions to billion variables.

## Professional Activities

- **Chair** Estimation of Distribution Algorithms Track, Genetic and Evolutionary Computation Conference (ACM SIGEVO conference), 2008 (Atlanta).
- **Co-Chair** Genetic Algorithms Track, Genetic and Evolutionary Computation Conference (ACM SIGEVO conference), 2007 (London).

- **Program committee member**, Genetic and Evolutionary Computation Conference, 2002 (New York, NY), 2003 (Chicago, IL), 2004 (Seattle, WA), 2005 (Washington, DC), 2006 (Seattle, WA), 2007 (London).
- **Co-organizer**, Workshop on Optimization by Building & Using Probabilistic Models (OBUPM), 2001 (San Francisco, CA), 2004 (Seattle, WA), 2005 (Washington, DC), 2006 (Seattle, WA), 2007 (London).
- Reviewer, Evolutionary Computation Journal
- Reviewer, IEEE Transactions on Evolutionary Computation
- Reviewer, IEEE Transactions on Systems, Man, and Cybernetics
- Reviewer, Journal of Heuristics
- Reviewer, Journal of Global Optimization
- **Electronic publicity chair**, Genetic and Evolutionary Computation Conference (GECCO-2002), New York, NY.
- **Project Coordinator**, APOGEE-97, an all India science exhibition.

## Professional Memberships

- Student Member, ACM SIGEVO, Special interest group for genetic and evolutionary computation

## Invited Talks & Tutorials

- *A Billion Bits or Bust* (with D. E. Goldberg). NCSA private sector program annual meeting, May 2007.
- *Efficiency Enhancement Techniques in Estimation of Distribution Algorithms*. Grand opening of Missouri estimation of distribution laboratory (MEDAL). University of Missouri St. Louis, July 2006.
- *Principled Efficiency Enhancement Techniques*. Tutorial at Genetic and Evolutionary Computation Conference. June 2005.
- *Population Sizing for Genetic Programming Based On Decision Making*. Workshop on Parameter Setting in Evolutionary Algorithms. Genetic and Evolutionary Computation Conference. June 2005.
- *Understanding Complex Systems: A Design Decomposition Approach*. Nonlinear Dynamics and Complex Systems Seminar. Department of Physics. University of Illinois at Urbana-Champaign. April 2005.
- *Inducing Competent Neighborhood Operators: Probabilistic Model Building Approach*. Annual INFORMS meeting. Special session on Genetic Algorithms. October 2004.

- *Facetwise Understanding of Genetic Programming and Design of Competent Genetic Programming*. Department of Mathematics and Computer Science. University of Missouri at St. Louis. March 2004.

- *Genetic Programming for Multi-timescale Modeling*. Understanding Complex Systems. University of Illinois at Urbana-Champaign. May 2003.

## Collaborators

Hussein A. Abbass (CS, University of Canberra, Australia) • B. V. Babu (Chem. Eng., BITS Pilani, India) • Jaume Bacardit (CSIT, University of Nottingham, UK) • L. Behera (ECE, IIT Kanpur, India) • Pascal Bellon (MSE, University of Illinois, USA) • Martin Butz (Psychology, University of Würzburg, Germany) • Erick Cantú-Paz (Yahoo! Inc., USA) • Chhanda Chakraborti (Philosophy, IIT Kharagpur, India) • Jian-Hung Chen (CS, Chung Hua University, Taiwan) • Ying-ping Chen (CS, National Chiao Tung University, Taiwan) • David E. Goldberg (IESE, University of Illinois, USA) • Georges Harik • Duane D. Johnson (MSE, University of Illinois, USA) • Graham Kendall (CSIT, University of Nottingham, UK) • Pier Luca Lanzi (CS, Politecnico di Milano, Italy) • Claudio F. Lima (CS, University of Algarve, Portugal) • Fernando Lobo (CS, University of Algarve, Portugal) • Xavier Llorà (NCSA, University of Illinois, USA) • Todd Martinez (Chem, University of Illinois, USA) • I. J. Nagrath (ECE, BITS Pilani, India) • Kei Ohnishi (CSE, Kyushu Institute of Technology, Japan) • Yukio Ohsawa (Systems Eng., University of Tokyo, Japan) • Una-May O'Reilly (CSAIL, MIT, USA) • Albert Orriols-Puig (CS, Ramon Llull University, Spain) • Luis de la Ossa (CS, University of Castilla la Mancha, Spain) • Martin Pelikan (Math & CS, University of Missouri St. Louis, USA) • Alexis L. Thompson (Chem, University of Illinois, USA) • Shigeyoshi Tsutsui (CS, Hannan University, Japan) • Noriko Imafuji Yasui (IESE, University of Illinois, USA) • Tian-Li Yu (ECE, National Taiwan University, Taiwan).

## Publications

**Summary:** *h-index:* 14 ⋄ *Total citations:* 595

### Books

Goldberg, D. E., Sastry, K. *Genetic algorithms: The design of innovation.* (In preparation). 2nd edition. Berlin: Springer.

Thierens, D., Beyer, H.-G., Birattari, M., Bongard, J., Branke, J., Clark, J. A., Cliff, D., Congdon, C. B., Deb, K., Doerr, B., Kovacks, T., Kumar, S., Miller, J. F., Moore, J., Neumann, F., Pelikan, M., Poli, R., Sastry, K., Stanley, K. O., Stützle, T., Watson, R. A., Wegener, I. (2007). *Proceedings of the 2007 Genetic and Evolutionary Computation Conference.* New York: ACM Press.

Pelikan, M., Sastry, K., Cantú-Paz, E. (Eds.). (2006) *Scalable optimization via probabilistic modeling: From algorithms to applications.* Berlin: Springer.

### Refereed Journal Papers

Goldberg, D. E., Sastry, K., Llorà, X. (2007). Toward routine billion-variable optimization using genetic algorithms. *Complexity*, *12*(3), 27–29.

Chen, Y.-p., Yu, T.-L., Sastry, K., Goldberg, D. E. (submitted). A Survey of linkage learning techniques in genetic and evolutionary algorithms. *Evolutionary Computation Journal*. (Preprint: IlliGAL report no. 2007014).

Pelikan, M., Sastry, K., Butz, M. V., Goldberg, D. E. (submitted). Genetic and evolutionary algorithms on random additively decomposable problems. *Evolutionary Computation Journal*.

Pelikan, M., Sastry, K., Goldberg, D. E. (accepted). Sporadic model building for efficiency enhancement of the hierarchical BOA. *Genetic Programming and Evolvable Machines*.

Sastry, K., Johnson, D.D., Goldberg, D. E. (2007). Scalability of a hybrid extended compact genetic algorithm for ground state optimization of clusters. *Materials and Manufacturing Processes*, *22*(5), 570–576.

Sastry, K., Johnson, D.D., Thompson, A. L., Goldberg, D. E., Martinez, T. J., Leiding, J., Owens, J. (2007). Optimization of Semiempirical Quantum Chemistry Methods via Multiobjective Genetic Algorithms: Accurate Photochemistry for Larger Molecules and Longer Time Scales *Materials and Manufacturing Processes*, *22*(5), 553–561.

Sastry, K., Pelikan, M., Goldberg, D. E. (submitted). Efficiency enhancement of genetic algorithms by building an internal probabilistic model of fitness. *Evolutionary Computation Journal*.

Butz, M.V., Goldberg, D.E., Lanzi, P.L., Sastry, K. (2007) Problem Solution Sustenance in XCS: Markov Chain Analysis of Niche Support Distributions and Consequent Computational Complexity. *Genetic Programming and Evolvable Machines*, *8*(1), 5-57 (Preprint: IlliGAL report no. 2004033).

Sastry, K., Johnson, D. D., Goldberg, D. E., Bellon, P. (2005). Genetic programming for multitimescale modeling. *Physical Review B*, *72*, 085438. [Selected by AIP editors as focused article of frontier research in *Virtual Journal of Nanoscale Science and Technology*, *12*(9), 2005].

Butz, M. V., Sastry, K., Goldberg, D. E. (2005). Strong, stable, and reliable fitness pressure in XCS due to tournament selection. *Genetic Programming and Evolvable Machines*, *6*(1), 53–77. (Preprint: IlliGAL report no. 2003027).

Sastry, K., Johnson, D. D., Goldberg, D. E., Bellon, P. (2004). Genetic programming for multiscale modeling. *International Journal for Multiscale Computational Engineering*, *2*(2), 239–256.

Pelikan, M., Sastry, K., Goldberg, D. E. (2002). Scalability of the Bayesian optimization algorithm. *International Journal of Approximate Reasoning*, *31*(3), 221–258. (Preprint: IlliGAL report no. 2001029).

Babu, B.V., Sastry, K. K. N. (1999). Estimation of heat transfer parameters using differential evolution and orthogonal collocation. *Computers and Chemical Engineering*, *23*, 327–339.

Sastry, K. K. N., Behera, L., Nagrath, I. J. (1999). Differential evolution based fuzzy logic controller for non-linear process control. *Fundamenta Informaticae: Special Issue on Soft Computing*, *37*(1-2), 121–136.

## Book Chapters

Yu, T.-L., Sastry, K., Goldberg, D. E. (2007). Population sizing to go: Online adaptation using noise and substructural measurement. In Lobo, F., Lima, C., Michalewicz, Z. (Eds.), *Parameter Settings in Evolutionary Algorithms*. Berlin: Springer.

Pelikan, M., Sastry, K., Goldberg, D. E. (2006). Multiobjective estimation of distribution algorithms. In Pelikan, M., Sastry, K., Cantú-Paz, E. (Eds.), *Scalable Optimization via Probabilistic Modeling: From Algorithms to Applications*. Berlin: Springer.

Sastry, K., Pelikan, M., Goldberg, D. E. (2006). Efficiency enhancement of estimation of distribution algorithms. In Pelikan, M., Sastry, K., Cantú-Paz, E. (Eds.), *Scalable optimization via Probabilistic Modeling: From Algorithms to Applications*. Berlin: Springer.

Harik, G. R., Lobo, F. G., Sastry, K. (2006). Linkage learning via probabilistic modeling in the ECGA. In Pelikan, M., Sastry, K., Cantú-Paz, E. (Eds.), *Scalable Optimization via Probabilistic Modeling: From Algorithms to Applications*. Berlin: Springer.

Llorà, X., Sastry, K., Goldberg, D. E., de la Ossa, L. (in press). The $\chi$-ary extended compact classifier system: Linkage learning in Pittsburgh LCS. In Kovacs, T., Llorà, X., and Takadama, K. (Eds.), *Advances at the frontier of LCS*. Berlin: Springer.

Llorà, X., Sastry, K., Goldberg, D. E. (2007). Binary Rule Encoding Schemes: A Study Using The Compact Classifier System. In Kovacs, T., Llorà, X., Takadama, K., Lanzi, P. L., Stolzmann, W., Wilson, S. W. (Eds.), *Learning Classifier Systems*, 41–60. Berlin: Springer.

Ondas, R., Pelikan, M., Sastry, K. (2006). Genetic programming, probabilistic incremental program evolution, and scalability. In Tiwari, A., Knowles, J., Avineri, E., Dahal, K., Roy, R. (Eds.) *Applications of Soft Computing: Recent Trends*. Berlin: Springer.

Sastry, K., Goldberg, D.E., Kendall, G. (2005). Genetic algorithms: A tutorial. In Burke, E. and Kendall, G. (Eds), *Introductory Tutorials in Optimization, Search and Decision Support Methodologies*. Berlin: Springer.Preprint.

Sastry, K., O'Reilly, U.-M., Goldberg, D. E., (2004). Population sizing for genetic programming based upon decision making. In O'Reilly, U.-M., et al (Eds.), *Genetic Programming Theory and Practice II*, 49–66. Boston, MA: Kluwer Academic Publishers. (Preprint: IlliGAL report no. 2004028).

Sastry, K., Goldberg, D. E. (2003). Probabilistic Model Building and Competent Genetic Programming. In Riolo, R., Worzel, B. (Eds.), *Genetic Programming Theory and Practice*, 205–220. Boston, MA: Kluwer Academic Publishers. (Preprint: IlliGAL report no. 2003013).

Sastry, K., O'Reilly, U.-M., Goldberg, D. E., Hill, D. (2003). Building-Block Supply in Genetic Programming. In Riolo, R., Worzel, B. (Eds.), *Genetic Programming Theory and Practice*, 155–172. Boston, MA: Kluwer Academic Publishers. (Preprint: IlliGAL report no. 2003012).

Goldberg, D. E., Sastry, K., Ohsawa, Y. (2003). Discovering deep building blocks for competent genetic algorithms using chance discovery via KeyGraphs. In Ohsawa, Y., McBurney, P. (Eds.), *Chance Discovery*, 276–302. Berlin: Springer-Verlag. (Preprint: IlliGAL report no. 2002026).

## Refereed Conference Papers

Fossati, L., Lanzi, P. L., Sastry, K., Goldberg, D. E., Gomez, O. (2007). A simple real-coded extended compact genetic algorithm. *Proceedings of the Congress on Evolutionary Computation (CEC 2007)*.

Lima, C. F., Pelikan, M., Goldberg, D. E., Lobo, F. G., **Sastry, K.**, Hauschild, M. (2007). Influence of selection and replacement strategies on linkage learning in BOA. *Proceedings of the Congress on Evolutionary Computation (CEC 2007)*. (Preprint: IlliGAL report no. 2007013).

Sastry, K., Goldberg, D. E. (2007). Let's get ready to rumble redux: Crossover versus mutation head to head on exponentially scaled problems. *Genetic and Evolutionary Computation Conference (GECCO 2007)*, 1380–1387. (Preprint: IlliGAL report no. 2007006).

Sastry, K., Goldberg, D. E., Llorà, X. (2007). Towards billion bit optimization via parallel estimation of distribution algorithm. *Genetic and Evolutionary Computation Conference (GECCO 2007)*, 577–584. (Preprint: IlliGAL report no. 2007007). [Best paper in Estimation of Distribution Algorithms track].

Sastry, K., Pelikan, M., Goldberg, D. E. (2007). Empirical Analysis of ideal recombination on random decomposable problems. *Genetic and Evolutionary Computation Conference (GECCO 2007)*, 1388–1395. (Preprint: IlliGAL report no. 2006016). [Best paper award nominee in Genetic Algorithms track].

Orriols-Puig, A., Sastry, K., Lanzi, P. L., Goldberg, D. E., Bernadó-Mansilla, E. (2007). Modeling selection pressure in XCS for proportionate and tournament selection. *Genetic and Evolutionary Computation Conference (GECCO 2007)*, 1846–1853. (Preprint: IlliGAL report no. 2007004).

Llorà, X., Sastry, K., Yu, T.-L., Goldberg, D. E. (2007). Do not match, Inherit: Fitness surrogates for genetics-based machine learning techniques. *Genetic and Evolutionary Computation Conference (GECCO 2007)*, 1798–1805. (Preprint: IlliGAL report no. 2007011).

Orriols-Puig, A., Goldberg, D. E., Sastry, K., Bernadó-Mansilla, E. (2007). Modeling XCS in class imbalances: Population sizing and parameter settings. *Genetic and Evolutionary Computation Conference (GECCO 2007)*, 1838–1845. (Preprint: IlliGAL report no. 2007001).

Bacardit, J., Stout, M., Hirst, J. D., Sastry, K., Llorà, X., Krasnogor, N. (2007). Automated alphabet reduction method with evolutionary algorithms for protein structure prediction. *Genetic and Evolutionary Computation Conference (GECCO 2007)*, 346–353. (Preprint: IlliGAL report no. 2007015). [Bronze "Humies" award at the Human Competitive Results Competition].

Hauschild, M., Pelikan, M., Lima, C. F., Sastry, K. (2007). Analyzing probabilistic models in hierarchical BOA on traps and spin glasses. *Genetic and Evolutionary Computation Conference (GECCO 2007)*, 523–530. (Preprint: Medal report no. 2007001).

Yu, T.-L., Sastry, K., Goldberg, D. E., Pelikan, M. (2007). Population sizing for entropy-based model building in genetic algorithms. *Genetic and Evolutionary Computation Conference (GECCO 2007)*, 601–608. (Preprint: IlliGAL report no. 2006020). [Best paper award nominee in Estimation of Distribution Algorithms track].

Pelikan, M., Hartmann, A. K., Sastry, K. (2006). Hierarchical BOA, Cluster Exact Approximation, and Ising Spin Glasses. *Parallel Problem Solving from Nature (PPSN IX)*, 121–131.

Lima, C. F., Pelikan, M., Sastry, K., Butz, M. V., Goldberg, D. E., Lobo, F. G. (2006). Substructural neighborhoods for local search in the Bayesian optimization algorithm. *Parallel Problem Solving from Nature (PPSN IX)*. 232–241. (Preprint: IlliGAL report no. 2006021).

Pelikan, M., Sastry, K., Butz, M. V., Goldberg, D. E. (2006). Performance of evolutionary algorithms on random decomposable problems. *Parallel Problem Solving from Nature (PPSN IX)*, 788–797. (Preprint: IlliGAL report no. 2006002).

Sastry, K., Johnson, D.D., Thompson, A. L., Goldberg, D. E., Martinez, T. J., Leiding, J., Owens, J. (2006). Multiobjective genetic algorithms for multiscaling excited state direct dynamics in photochemistry. *Genetic and Evolutionary Computation Conference (GECCO 2006)*. 1745–1752. (Preprint: IlliGAL report no. 2006005). [Best paper award in Real World Applications track] [Silver "Humies" award at the Human Competitive Results Competition].

Sastry, K., Lima, C. F., Goldberg, D. E. (2006). Evaluation relaxation using substructural information and linear estimation. *Genetic and Evolutionary Computation Conference (GECCO 2006)*. 419–426. (Preprint: IlliGAL report no. 2006003).

Llorà, X., Sastry, K. (2006). Fast rule matching for learning classifier systems via vector instructions. *Genetic and Evolutionary Computation Conference (GECCO 2006)*. 1513–1520. (Preprint: IlliGAL report no. 2006001).

Pelikan, M., Sastry, K., Goldberg, D. E. (2006). Sporadic model building for efficiency enhancement of hBOA. *Genetic and Evolutionary Computation Conference (GECCO 2006)*. 405–412. (Preprint: IlliGAL report no. 2005026).

Alías, F. , Llorà, X., Formiga, L., Sastry, K., Goldberg, D. E. (2006). Efficient interactive weight tuning for TTS synthesis: Reducing user fatigue by improving user consistency. *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP 2006)*, *1*, 865–868. (Preprint: IlliGAL report no. 2005022).

Sastry, K., Winward, P., Goldberg, D. E., Lima, C. F. (2006). Fluctuating crosstalk as a source of deterministic noise and its effects on GA scalability. *Applications of Evolutionary Computing EvoWorkshops2006: EvoBIO, EvoCOMNET, EvoHOT, EvoIASP, EvoInteraction, EvoMUSART, EvoSTOCK*, 740–751. (Preprint: IlliGAL report no. 2005025).

Llorà, X., Sastry, K., Goldberg, D. E. (2005). The compact classifier system: Motivation, analysis and first results. *Proceedings of the 2005 Congress on Evolutionary Computation*, *1*, 596–603. (Preprint: IlliGAL report no. 2005019).

Yu, T.-L., Sastry, K., Goldberg, D. E. (2005). Online population size adjusting using noise and substructural measurements. *Proceedings of the 2005 Congress on Evolutionary Computation Conference*, *3*, 2491–2498. (Preprint: IlliGAL report no. 2005017).

Sastry, K., Pelikan, M., Goldberg, D. E. (2005). Limits of scalability of multiobjective estimation of distribution algorithms. *Proceedings of the 2005 Congress on Evolutionary Computation*, *3*, 2217–2224. (Preprint: IlliGAL report no. 2005004).

Yu, T.-L., Sastry, K., Goldberg, D. E. (2005). Linkage learning, overlapping building blocks, and a systematic strategy for scalable recombination. *Genetic and Evolutionary Computation Conference (GECCO 2005)*, 1217–1224. (Preprint: IlliGAL report no. 2005016).

Llorà, X., Sastry, K., Goldberg, D. E., Gupta, A., Lakshmi, L. (2005). Combating user fatigue in iGAs: Partial ordering, support vector machines, and synthetic fitness. *Genetic and Evolutionary Computation Conference (GECCO 2005)*, 1363–1370. (Preprint: IlliGAL report no. 2005009).

Pelikan, M., Sastry, K., Goldberg, D. E. (2005). Multiobjective hBOA, clustering, and scalability. *Genetic and Evolutionary Computation Conference (GECCO 2005)*, 663–670. (Preprint: IlliGAL report no. 2005005).

Sastry, K., Abbass, H. A., Goldberg, D. E., Johnson, D. D. (2005). Sub-structural niching in estimation of distribution algorithms. *Genetic and Evolutionary Computation Conference (GECCO 2005)*, 671–678. (Preprint: IlliGAL report no. 2005003). [Best paper award nominee in Estimation of Distribution Algorithms track].

Lima, C., Sastry, K., Goldberg, D. E., Lobo, F. (2005). Combining competent crossover and mutation operators: A probabilistic model building approach. *Genetic and Evolutionary Computation Conference (GECCO 2005)*, 735–742. (Preprint: IlliGAL report no. 2005002).

Sastry, K., Abbass, H. A., Goldberg, D. E. (2004). Sub-structural niching in non-stationary environments. *Proceedings of the Australian Artificial Intelligence Conference*, 873–885. (Preprint: IlliGAL report no. 2004035).

Sastry, K., Pelikan, M., Goldberg, D. E. (2004). Efficiency enhancement of genetic algorithms via building-block-wise fitness estimation. *Proceedings of the 2004 Congress on Evolutionary Computation*, 720–727. (Preprint: IlliGAL report no. 2004010).

Pelikan, M., Sastry, K. (2004). Fitness inheritance in the Bayesian optimization algorithm. *Genetic and Evolutionary Computation Conference (GECCO 2004)*, *2*, 48–59. (Preprint: IlliGAL report no. 2004009).

Ohnishi, K., Sastry, K., Chen, Y.-p., Goldberg, D. E. (2004). Inducing sequentiality using grammatical genetic codes. *Genetic and Evolutionary Computation Conference (GECCO 2004)*, *2*, 1426–1437. (Preprint: IlliGAL report no. 2004007).

Sastry, K., Goldberg, D. E. (2004). Designing competent mutation operators via probabilistic model building of neighborhoods. *Genetic and Evolutionary Computation Conference (GECCO 2004)*, *2*, 114–125. (Preprint: IlliGAL report no. 2004006).

Sastry, K., Goldberg, D. E. (2004). Lets get ready to rumble: Crossover versus mutation head to head. *Genetic and Evolutionary Computation Conference (GECCO 2004)*, *2*, 126–137. (Preprint: IlliGAL report no. 2004005).

Yu, T.-L., Goldberg, D. E., Sastry, K. (2003). Optimal sampling and speed-up for genetic algorithms on the sampled OneMax problem. *Genetic and Evolutionary Computation Conference (GECCO 2003)*, 1554–1565. (Preprint: IlliGAL report no. 2003008).

Butz, M. V., Sastry, K., Goldberg, D. E. (2003). Tournament selection in XCS. *Genetic and Evolutionary Computation Conference (GECCO 2003)*, 1857–1869. (Preprint: IlliGAL report no. 2002020). [Best paper award in Learning Classifier Systems track].

Sastry, K., Goldberg, D. E. (2003). Scalability of selectorecombinative genetic algorithms for problems with tight linkage. *Genetic and Evolutionary Computation Conference (GECCO 2003)*, 1332–1344. (Preprint: IlliGAL report no. 2002013). [Best paper award nominee in Genetic Algorithms track].

Chen, J.-H., Goldberg, D. E., Ho, S.-Y., Sastry, K. (2002). Fitness inheritance in multi-objective optimization. *Genetic and Evolutionary Computation Conference (GECCO 2002)*, 319–326. (Preprint: IlliGAL report no. 2002017).

Sastry, K., Goldberg, D. E. (2002). Genetic algorithms, efficiency enhancement, and deciding well with fitness functions with differing variances. *Genetic and Evolutionary Computation Conference (GECCO 2002)*, 528–535. (Preprint: IlliGAL report no. 2002003).

Sastry, K., Goldberg, D. E. (2002). Genetic algorithms, efficiency enhancement, and deciding well with fitness functions with differing bias values. *Genetic and Evolutionary Computation Conference (GECCO 2002)*, 536–543. (Preprint: IlliGAL report no. 2002002).

Sastry, K., Goldberg, D. E. (2001). Modeling tournament selection with replacement using apparent added noise. *Intelligent Engineering Systems Through Artificial Neural Networks*, *11*, 129–134. (Preprint: IlliGAL report no. 2001014).

Tsutsui, S., Goldberg, D. E., Sastry, K. (2001). Linkage learning in real-coded GAs with simplex crossover. *Proceedings of the 5th International Conference on Artificial Evolution*, 51–58. (Preprint: IlliGAL report no. 2000033).

Sastry, K., Goldberg, D. E., Pelikan, M. (2001). Dont evaluate, inherit. *Genetic and Evolutionary Computation Conference (GECCO 2001)*, 551–558. (Preprint: IlliGAL report no. 2001013).

Goldberg, D. E., Sastry, K., Latoza, T. (2001). On the supply of building blocks. *Genetic and Evolutionary Computation Conference (GECCO 2001)*, 336–342. (Preprint: IlliGAL report no. 2001015).

Goldberg, D. E., Sastry, K. (2001). A practical schema theorem for genetic algorithm design and tuning. *Genetic and Evolutionary Computation Conference (GECCO 2001)*, 328–335. (Preprint: IlliGAL report no. 2001017).

Pelikan, M., Goldberg, D. E., Sastry, K. (2001). Bayesian optimization algorithm, decision graphs, and Occams razor. *Genetic and Evolutionary Computation Conference (GECCO 2001)*, 519–526. (Preprint: IlliGAL report no. 2000020).

Chakraborti, C., Sastry, K. K. N. (1998). Testing the validity of logical arguments using genetic algorithms. *Proceedings of the International Conference on Knowledge based Computer Systems (KBCS 98)*, 117–126.

Chakraborti, C., Sastry, K. K. N. (1998). Genetic algorithms approach for proving logical arguments in natural language. *Genetic Programming 1998: Proceedings of the Third Annual Conference*, 463–470.

## Refereed Conference Posters

Pelikan, M., Sastry, K., Butz, M. V., Goldberg, D. E. (2006). Hierarchical BOA on random decomposable problems. *Genetic and Evolutionary Computation Conference (GECCO 2006)*. 431–432. (Preprint: IlliGAL report no. 2006002).

Llorà, X., Sastry, K., Alías, F., Goldberg, D. E., Welge, M. (2006). Analyzing active interactive genetic algorithms using visual analytics. *Genetic and Evolutionary Computation Conference (GECCO 2006)*. 1417–1418. (Preprint: IlliGAL report no. 2006004).

Ondas, R., Pelikan, M., Sastry, K. (2005). Scalability of genetic programming and probabilistic incremental program evolution. *Genetic and Evolutionary Computation Conference (GECCO 2005)*, 1785–1786. (Preprint: arXiv:cs.NE/0502029).

Llorà, X., Sastry, K., Goldberg, D. E. (2005). The compact classifier system: Motivation, analysis and first results. *Genetic and Evolutionary Computation Conference (GECCO 2005)*, 1893–1894. (Preprint: IlliGAL report no. 2005019).

Sastry, K., Goldberg, D. E. (2001). Modeling tournament selection with replacement using apparent added noise. *Genetic and Evolutionary Computation Conference (GECCO 2001)*, 781.

## Workshops & Non-Refereed Conferences

Orriols-Puig, A., Sastry, K., Goldberg, D. E., Bernadó-Manzilla, E. (2007). Substructural surrogates for learning decomposable classification problems: Implementation and first results. *International Workshop on Learning Classifier Systems*. 2875–2882. Preprint: IlliGAL report no. 2007010.

Sastry, K., Pelikan, M., Goldberg, D. E. (2004). Efficiency enhancement of probabilistic model building genetic algorithms. *Optimization by Building and Using Probabilistic Models: Workshop at the Genetic and Evolutionary Computation Conference*. (Preprint: IlliGAL report no. 2004020).

Sastry, K., Goldberg, D. E. (2002). How well does a single-point crossover mix building blocks with tight linkage? *Proceedings of the International Symposium on Computer and Information Science*. (Preprint: IlliGAL report no. 2002013).

Sastry, K. (2001). Efficient cluster optimization using a hybrid extended compact genetic algorithm with a seeded population, *Workshop Proceedings of the Genetic and Evolutionary Computation Conference*, 222–225. (Preprint: IlliGAL report no. 2001018).

Sastry, K., Goldberg, D. E. (2000). On extended compact genetic algorithm. *Late Breaking Paper in Genetic and Evolutionary Computation Conference*, 352–359. (Preprint: IlliGAL report no. 2000026).

Chakraborti, C., Sastry, K. K. N. (1997). Genetic algorithms: An efficient alternative for 'proving logical arguments. *Evonews*, 17–18.

Sastry, K. K. N., Behera, L., Nagrath, I. J. (1997). A self organizing fuzzy controller design using differential evolution. *Proceedings of the Sixth Symposium on Intelligent Systems*, 166–177.

## Technical Reports

Sastry, K. (2007). *Single and multiobjective genetic algorithm toolbox for matlab in C++*. IlliGAL report no. 2007017. University of Illinois at Urbana-Champaign, Urbana, IL.

Sastry, K. (2007). *Single and multiobjective genetic algorithm toolbox in C++*. IlliGAL report no. 2007016. University of Illinois at Urbana-Champaign, Urbana, IL.

Sastry, K., Orriols-Puig, A. (2007). *Extended compact genetic algorithm in matlab*. IlliGAL report no. 2007009. University of Illinois at Urbana-Champaign, Urbana, IL.

Ueda, T., Imafuji, N., Llorà, X., Sastry, K., Goldberg, D. E. (2007). *Toward context-aware semantic building block identification for text streams using EDAs*.

Pelikan, M., Sastry, K., Butz, M. V., Goldberg, D. E. (2006). *Generator and interface for random decomposable problems in C*. MEDAL report no. 2006003. University of Missouri-St. Louis, St. Louis, MO.

Sastry, K., de la Ossa, L., Lobo, F. G. (2006). *$\chi$-ary extended compact genetic algorithm for matlab in C++*. IlliGAL report no. 2006014. University of Illinois at Urbana-Champaign, Urbana, IL.

de la Ossa, L., Sastry, K., Lobo, F. G. (2006). *$\chi$-ary extended compact genetic algorithm in C++*. IlliGAL report no. 2006013. University of Illinois at Urbana-Champaign, Urbana, IL.

Lobo, F. G., Sastry, K., Harik, G. R. (2006). *Extended compact genetic algorithm in C++: Version 1.1*. IlliGAL report no. 2006012. University of Illinois at Urbana-Champaign, Urbana, IL.

Llorà, X., Alías, F. , Formiga, L., Sastry, K., Goldberg, D. E. (2005). *Evaluation consistency in iGAs: User contradictions as cycles in partial-ordering graphs*. IlliGAL report no. 2005022. University of Illinois at Urbana-Champaign, Urbana, IL.

Abbass, H. A., Sastry, K., Goldberg, D. E. (2004). *Oiling the wheels of change: The role of adaptive automatic problem decomposition in non-stationary environments*. IlliGAL report no. 2004029. University of Illinois at Urbana-Champaign, Urbana, IL.

Sastry, K., Goldberg, D. E. (2002). *Analysis of mixing in genetic algorithms: A survey*. IlliGAL report no. 2002012. University of Illinois at Urbana-Champaign, Urbana, IL.

Sastry, K. (2002). *Evaluation-Relaxation Schemes for Genetic and Evolutionary Algorithms*. Masters Thesis. Department of General Engineering. University of Illinois at Urbana-Champaign, Urbana, IL. [William A. Chittenden Award for outstanding master of science graduate in General Engineering] (Preprint: IlliGAL report no. 2002004).

Pelikan, M., Sastry, K., Goldberg, D. E. (2001). *Evolutionary algorithms + graphical models = scalable black-box optimization*. IlliGAL report no. 2001029. University of Illinois at Urbana-Champaign, Urbana, IL.

Sastry, K., Xiao, G. (2001). *Silicon cluster optimization using extended compact genetic algorithm*. IlliGAL report no. 2001016. University of Illinois at Urbana-Champaign, Urbana, IL.

## Source Code

**Single and multiobjective genetic algorithm toolbox for matlab in C++**
Sastry, K.
Documentation: http://www.illigal.uiuc.edu/pub/papers/IlliGALs/2007017.pdf
Source: http://www.illigal.uiuc.edu/pub/src/GA/GAtoolbox_matlab.tgz

**Single and multiobjective genetic algorithm toolbox in C++**
Sastry, K.
Documentation: http://www.illigal.uiuc.edu/pub/papers/IlliGALs/2007016.pdf
Source: http://www.illigal.uiuc.edu/pub/src/GA/GAtoolbox.tgz

**Extended compact genetic algorithm in matlab**
Sastry, K., Orriols-Puig, A.
Documentation: http://www.illigal.uiuc.edu/pub/papers/IlliGALs/2007009.pdf
Source: http://www.illigal.uiuc.edu/pub/src/ECGA/eCGAmatlab.zip

**Generator and interface for random decomposable problems in C.**
Pelikan, M., Sastry, K., Butz, M. V., Goldberg, D. E.
Documentation: http://medal.cs.umsl.edu/files/2006003.pdf
Source: http://medal.cs.umsl.edu/files/decomposable-problems.tar.gz

**$\chi$-ary extended compact genetic algorithm for matlab in C++.**
Sastry, K., de la Ossa, L., Lobo, F. G.
Documentation: http://www.illigal.uiuc.edu/pub/papers/IlliGALs/2006014.pdf
Source: http://www.illigal.uiuc.edu/pub/src/ECGA/chiECGA_matlab.tgz

**$\chi$-ary extended compact genetic algorithm in C++.**
de la Ossa, L., Sastry, K., Lobo, F. G.
Documentation: http://www.illigal.uiuc.edu/pub/papers/IlliGALs/2006013.pdf
Source: http://www.illigal.uiuc.edu/pub/src/ECGA/chiECGA.tgz

**Extended compact genetic algorithm in C++: Version 1.1.**
Lobo, F. G., Sastry, K., Harik, G.
Documentation: http://www.illigal.uiuc.edu/pub/papers/IlliGALs/2006012.pdf
Source: http://www.illigal.uiuc.edu/pub/src/ECGA/ECGA_1_1.tgz