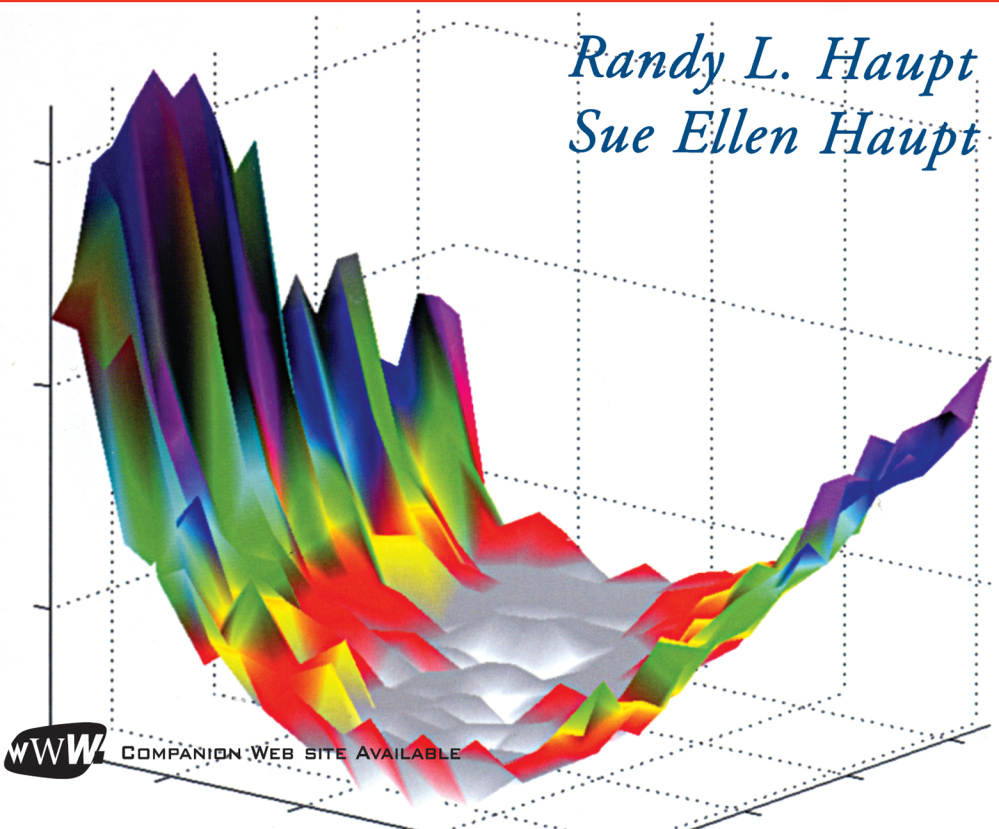


WILEY

# Practical Genetic Algorithms

*Second Edition*

*Randy L. Haupt  
Sue Ellen Haupt*





The CD no longer accompanies  
this book. The CD files can be  
found at  
[www.booksupport.wiley.com](http://www.booksupport.wiley.com)  
where you enter the ISBN  
9780471455653



# **PRACTICAL GENETIC ALGORITHMS**



---

# **PRACTICAL GENETIC ALGORITHMS**

**SECOND EDITION**

---

Randy L. Haupt  
Sue Ellen Haupt

 **WILEY-  
INTERSCIENCE**

**A JOHN WILEY & SONS, INC., PUBLICATION**

CD files can now be found by entering in  
ISBN 9780471455653  
on [booksupport.wiley.com](http://booksupport.wiley.com)

The book is printed on acid-free paper. ☺

Copyright © 2004 by John Wiley & Sons, Inc. All rights reserved.

Published by John Wiley & Sons, Inc., Hoboken, New Jersey.  
Published simultaneously in Canada.

No part of this publication may be reproduced, stored in a retrieval system, or transmitted in any form or by any means, electronic, mechanical, photocopying, recording, scanning, or otherwise, except as permitted under Section 107 or 108 of the 1976 United States Copyright Act, without either the prior written permission of the Publisher, or authorization through payment of the appropriate per-copy fee to the Copyright Clearance Center, Inc., 222 Rosewood Drive, Danvers, MA 01923, 978-750-8400, fax 978-646-8600, or on the web at [www.copyright.com](http://www.copyright.com). Requests to the Publisher for permission should be addressed to the Permissions Department, John Wiley & Sons, Inc., 111 River Street, Hoboken, NJ 07030, (201) 748-6011, fax (201) 748-6008.

**Limit of Liability/Disclaimer of Warranty:** While the publisher and author have used their best efforts in preparing this book, they make no representations or warranties with respect to the accuracy or completeness of the contents of this book and specifically disclaim any implied warranties of merchantability or fitness for a particular purpose. No warranty may be created or extended by sales representatives or written sales materials. The advice and strategies contained herein may not be suitable for your situation. You should consult with a professional where appropriate. Neither the publisher nor author shall be liable for any loss of profit or any other commercial damages, including but not limited to special, incidental, consequential, or other damages.

For general information on our other products and services please contact our Customer Care Department within the U.S. at 877-762-2974, outside the U.S. at 317-572-3993 or fax 317-572-4002.

Wiley also publishes its books in a variety of electronic formats. Some content that appears in print, however, may not be available in electronic format.

***Library of Congress Cataloging-in-Publication Data:***

Haupt, Randy L.

Practical genetic algorithms / Randy L. Haupt, Sue Ellen Haupt.—2nd ed.

p. cm.

Red. ed. of: Practical genetic algorithms. c1998.

“A Wiley-Interscience publication.”

Includes bibliographical references and index.

ISBN 978-0-471-45565-3

1. Genetic algorithms. I. Haupt, S. E. II. Haupt, Randy L. Practical genetic algorithms.

III. Title.

QA402.5.H387 2004

519.6'2—dc22

2004043360



*To our parents  
Anna Mae and Howard Haupt  
Iona and Charles Slagle  
and  
our offspring,  
Bonny Ann and Amy Jean Haupt*



## CONTENTS

---

<b>Preface</b>	<b>xi</b>
<b>Preface to First Edition</b>	<b>xiii</b>
<b>List of Symbols</b>	<b>xv</b>
<b>1 Introduction to Optimization</b>	<b>1</b>
1.1 Finding the Best Solution	1
1.1.1 What Is Optimization?	2
1.1.2 Root Finding versus Optimization	3
1.1.3 Categories of Optimization	3
1.2 Minimum-Seeking Algorithms	5
1.2.1 Exhaustive Search	5
1.2.2 Analytical Optimization	7
1.2.3 Nelder-Mead Downhill Simplex Method	10
1.2.4 Optimization Based on Line Minimization	13
1.3 Natural Optimization Methods	18
1.4 Biological Optimization: Natural Selection	19
1.5 The Genetic Algorithm	22
Bibliography	24
Exercises	25
<b>2 The Binary Genetic Algorithm</b>	<b>27</b>
2.1 Genetic Algorithms: Natural Selection on a Computer	27
2.2 Components of a Binary Genetic Algorithm	28
2.2.1 Selecting the Variables and the Cost Function	30
2.2.2 Variable Encoding and Decoding	32
2.2.3 The Population	36
2.2.4 Natural Selection	36
2.2.5 Selection	38
2.2.6 Mating	41
2.2.7 Mutations	43
2.2.8 The Next Generation	44
2.2.9 Convergence	47
	<b>vii</b>

2.3	A Parting Look	47
	Bibliography	49
	Exercises	49
<b>3</b>	<b>The Continuous Genetic Algorithm</b>	<b>51</b>
3.1	Components of a Continuous Genetic Algorithm	52
3.1.1	The Example Variables and Cost Function	52
3.1.2	Variable Encoding, Precision, and Bounds	53
3.1.3	Initial Population	54
3.1.4	Natural Selection	54
3.1.5	Pairing	56
3.1.6	Mating	56
3.1.7	Mutations	60
3.1.8	The Next Generation	62
3.1.9	Convergence	64
3.2	A Parting Look	65
	Bibliography	65
	Exercises	65
<b>4</b>	<b>Basic Applications</b>	<b>67</b>
4.1	“Mary Had a Little Lamb”	67
4.2	Algorithmic Creativity—Genetic Art	71
4.3	Word Guess	75
4.4	Locating an Emergency Response Unit	77
4.5	Antenna Array Design	81
4.6	The Evolution of Horses	86
4.5	Summary	92
	Bibliography	92
<b>5</b>	<b>An Added Level of Sophistication</b>	<b>95</b>
5.1	Handling Expensive Cost Functions	95
5.2	Multiple Objective Optimization	97
5.2.1	Sum of Weighted Cost Functions	99
5.2.2	Pareto Optimization	99
5.3	Hybrid GA	101
5.4	Gray Codes	104
5.5	Gene Size	106
5.6	Convergence	107
5.7	Alternative Crossovers for Binary GAs	110
5.8	Population	117
5.9	Mutation	121
5.10	Permutation Problems	124
5.11	Selecting GA Parameters	127

5.12	Continuous versus Binary GA	135
5.13	Messy Genetic Algorithms	136
5.14	Parallel Genetic Algorithms	137
5.14.1	Advantages of Parallel GAs	138
5.14.2	Strategies for Parallel GAs	138
5.14.3	Expected Speedup	141
5.14.4	An Example Parallel GA	144
5.14.5	How Parallel GAs Are Being Used	145
	Bibliography	145
	Exercises	148
<b>6</b>	<b>Advanced Applications</b>	<b>151</b>
6.1	Traveling Salesperson Problem	151
6.2	Locating an Emergency Response Unit Revisited	153
6.3	Decoding a Secret Message	155
6.4	Robot Trajectory Planning	156
6.5	Stealth Design	161
6.6	Building Dynamic Inverse Models—The Linear Case	165
6.7	Building Dynamic Inverse Models—The Nonlinear Case	170
6.8	Combining GAs with Simulations—Air Pollution Receptor Modeling	175
6.9	Optimizing Artificial Neural Nets with GAs	179
6.10	Solving High-Order Nonlinear Partial Differential Equations	182
	Bibliography	184
<b>7</b>	<b>More Natural Optimization Algorithms</b>	<b>187</b>
7.1	Simulated Annealing	187
7.2	Particle Swarm Optimization (PSO)	189
7.3	Ant Colony Optimization (ACO)	190
7.4	Genetic Programming (GP)	195
7.5	Cultural Algorithms	199
7.6	Evolutionary Strategies	199
7.7	The Future of Genetic Algorithms	200
	Bibliography	201
	Exercises	202
	<b>Appendix I Test Functions</b>	<b>205</b>
	<b>Appendix II MATLAB Code</b>	<b>211</b>
	<b>Appendix III High-Performance Fortran Code</b>	<b>233</b>
	<b>Glossary</b>	<b>243</b>
	<b>Index</b>	<b>251</b>



## PREFACE

---

When we agreed to edit this book for a second edition, we looked forward to a bit of updating and including some of our latest research results. However, the effort grew rapidly beyond our original vision. The use of genetic algorithms (GAs) is a quickly evolving field of research, and there is much new to recommend. Practitioners are constantly dreaming up new ways to improve and use GAs. Therefore this book differs greatly from the first edition.

We continue to emphasize the “Practical” part of the title. This book was written for the practicing scientist, engineer, economist, artist, and whoever might possibly become interested in learning the basics of GAs. We make no claims of including the latest research on convergence theory; instead, we refer the reader to references that do. We do, however, give the reader a flavor for how GAs are being used and how to fiddle with them to get the best performance.

The biggest addition is including code—both MATLAB and a bit of High-Performance Fortran. We hope the readers find these a useful start to their own applications. There has also been a good bit of updating and expanding. Chapter 1 has been rewritten to give a more complete picture of traditional optimization. Chapters 2 and 3 remain dedicated to introducing the mechanics of the binary and continuous GA. The examples in those chapters, as well as throughout the book, now reflect our more recent research on choosing GA parameters. Examples have been added to Chapters 4 and 6 that broaden the view of problems being solved. Chapter 5 has greatly expanded its recommendations of methods to improve GA performance. Sections have been added on hybrid GAs, parallel GAs, and messy GAs. Discussions of parameter selection reflect new research. Chapter 7 is new. Its purpose is to give the reader a flavor for other artificial intelligence methods of optimization, like simulated annealing, ant colony optimization, and evolutionary strategies. We hope this will help put GAs in context with other modern developments. We included code listings and test functions in the appendixes. Exercises appear at the end of each chapter. There is no solution manual because the exercises are open-ended. These should be helpful to anyone wishing to use this book as a text.

In addition to the people thanked in the first edition, we want to recognize the students and colleagues whose insight has contributed to this effort. Bonny Haupt did the work included in Section 4.6 on horse evolution. Jaymon Knight translated our GA to High-Performance Fortran. David Omer and Jesse

Warrick each had a hand in the air pollution problem of Section 6.8. We've discussed our applications with numerous colleagues and appreciate their feedback.

We wish the readers well in their own forays into using GAs. We look forward to seeing their interesting applications in the future.

*State College, Pennsylvania*  
*February 2004*

RANDY L. HAUPT  
SUE ELLEN HAUPT



## PREFACE TO FIRST EDITION

---

The book has been organized to take the genetic algorithm in stages. Chapter 1 lays the foundation for the genetic algorithm by discussing numerical optimization and introducing some of the traditional minimum seeking algorithms. Next, the idea of modeling natural processes on the computer is introduced through a discussion of annealing and the genetic algorithm. A brief genetics background is supplied to help the reader understand the terminology and rationale for the genetic operators. The genetic algorithm comes in two flavors: binary parameter and real parameter. Chapter 2 provides an introduction to the binary genetic algorithm, which is the most common form of the algorithm. Parameters are quantized, so there are a finite number of combinations. This form of the algorithm is ideal for dealing with parameters that can assume only a finite number of values. Chapter 3 introduces the continuous parameter genetic algorithm. This algorithm allows the parameters to assume any real value within certain constraints. Chapter 4 uses the algorithms developed in the previous chapters to solve some problems of interest to engineers and scientists. Chapter 5 returns to building a good genetic algorithm, extending and expanding upon some of the components of the genetic algorithm. Chapter 6 attacks more difficult technical problems. Finally, Chapter 7 surveys some of the current extensions to genetic algorithms and applications, and gives advice on where to get more information on genetic algorithms. Some aids are supplied to further help the budding genetic algorithmist. Appendix I lists some genetic algorithm routines in pseudocode. A glossary and a list of symbols used in this book are also included.

We are indebted to several friends and colleagues for their help. First, our thanks goes to Dr. Christopher McCormack of Rome Laboratory for introducing us to genetic algorithms several years ago. The idea for writing this book and the encouragement to write it, we owe to Professor Jianming Jin of the University of Illinois. Finally, the excellent reviews by Professor Daniel Pack, Major Cameron Wright, and Captain Gregory Toussaint of the United States Air Force Academy were invaluable in the writing of this manuscript.

RANDY L. HAAPT  
SUE ELLEN HAAPT

*Reno, Nevada  
September 1997*



## LIST OF SYMBOLS

---

$a_N$	Pheromone weighting
$\mathbf{A}_n$	Approximation to the Hessian matrix at iteration $n$
$b$	Distance weighting
$b_n$	Bit value at location $n$ in the gene
$chromosome_n$	Vector containing the variables
$cost$	Cost associated with a variable set
$cost_{min}$	Minimum cost of a chromosome in the population
$cost_{max}$	Maximum cost of a chromosome in the population
$c_n$	Cost of chromosome $n$
$C_n$	Normalized cost of chromosome $n$
$c_s$	Scaling constant
$e_N$	Unit vectors
$f(*)$	Cost function
$\hat{f}$	Average fitness of chromosomes containing schema
$\bar{f}$	Average fitness of population
$G$	Generation gap
$g_m(x, y, \dots)$	Constraints on the cost function
$gene[m]$	Binary version of $p_n$
$gene_n$	$n^{th}$ gene in a chromosome
$\mathbf{H}$	Hessian matrix
$hi$	Highest number in the variable range
$\mathbf{I}$	Identity matrix
$J_0(x)$	Zeroth-order Bessel function
$\Gamma_1$	Length of link 1
$\Gamma_2$	Length of link 2
$life_{min}$	Minimum lifetime of a chromosome
$life_{max}$	Maximum lifetime of a chromosome
$lo$	Lowest number in the variable range
$ma$	Vector containing row numbers of mother chromosomes
$mask$	Mask vector for uniform crossover
$N_{bits}$	$N_{gene} \times N_{par}$ . Number of bits in a chromosome
$N_{gene}$	Number of bits in the gene
$N_{keep}$	Number of chromosomes in the mating pool
$N_n(0, 1)$	Standard normal distribution (mean = 0 and variance = 1)
$N_{pop}$	Number of chromosomes in the population from generation to generation

$N_{var}$	Number of variables
$offspring_n$	Child created from mating two chromosomes
$pa$	Vector containing row numbers of father chromosomes
$parent_n$	A parent selected to mate
$p_{dn}$	Continuous variable $n$ in the father chromosome
$p_{mn}$	Continuous variable $n$ in the mother chromosome
$p_{m,n}^{\text{local best}}$	Best local solution
$p_{m,n}^{\text{global best}}$	Best global solution
$p_n$	Variable $n$
$p_{new}$	New variable in offspring from crossover in a continuous GA
$p_{norm}$	Normalized variable
$p_{quant}$	Quantized variable
$p_{lo}$	Smallest variable value
$p_{hi}$	Highest variable value
$P$	Number of processors
$P_c$	Probability of crossover
$P_m$	Probability of mutation of a single bit
$P_n$	Probability of chromosome $n$ being selected for mating
$P_{opt}$	Number of processors to optimize speedup of a parallel GA
$P_t$	Probability of the schema being selected to survive to the next generation
$P_0$	Initial guess for Nelder-Mead algorithm
$q_n$	Quantized version of $P_n$
$Q$	Quantization vector = $[2^{-1} \ 2^{-2} \ \dots \ 2^{-N_{gene}}]$
$Q_i$	Number of different values that variable $i$ can have
$Q_t$	Probability of the schema being selected to mate
$r$	Uniform random number
$R_t$	Probability of the schema not being destroyed by crossover or mutation
$s$	$\sin \phi$
$s_t$	Number of schemata in generation $t$
$T$	Temperature in simulated annealing
$T_c$	CPU time required for communication
$T_f$	CPU time required to evaluate a fitness function
$T_0$	Beginning temperature in simulated annealing
$T_N$	Ending temperature in simulated annealing
$T_P$	Total execution time for a generation of a parallel GA
$u$	$\cos \phi$
$V$	Number of different variable combinations
$v_n$	Search direction at step $n$
$v_{m,n}$	Particle velocity
$w_n$	Weight $n$
$X_{rate}$	Crossover rate
$\hat{x}, \hat{y}, \hat{z}$	Unit vectors in the $x$ , $y$ , and $z$ directions
$\alpha$	Parameter where crossover occurs

$\alpha_k$	Step size at iteration $k$
$\beta$	Mixing value for continuous variable crossover
$\delta$	Distance between first and last digit in schema
$\nabla f$	$\frac{\partial f}{\partial x} \hat{x} + \frac{\partial f}{\partial y} \hat{y} + \frac{\partial f}{\partial z} \hat{z}$
$\nabla^2 f$	$\frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2} + \frac{\partial^2 f}{\partial z^2}$
$\varepsilon$	Elite path weighting constant
$\gamma_n$	Nonnegative scalar that minimizes the function in the direction of the gradient
$\kappa$	Lagrange multiplier
$\lambda$	Wavelength
$\zeta$	Number of defined digits in schema
$\eta$	$\frac{1}{2}(life_{max} - life_{min})$
$\mu$	Mutation rate
$\sigma$	Standard deviation of the normal distribution
$\tau$	Pheromone strength
$\tau_{mn}^k$	Pheromone laid by ant $k$ between city $m$ and city $n$
$\tau_{mn}^{elite}$	Pheromone laid on the best path found by the algorithm to this point
$\xi$	Pheromone evaporation constant



# Introduction to Optimization

Optimization is the process of making something better. An engineer or scientist conjures up a new idea and optimization improves on that idea. Optimization consists in trying variations on an initial concept and using the information gained to improve on the idea. A computer is the perfect tool for optimization as long as the idea or variable influencing the idea can be input in electronic format. Feed the computer some data and out comes the solution. Is this the only solution? Often times not. Is it the best solution? That's a tough question. Optimization is the math tool that we rely on to get these answers.

This chapter begins with an elementary explanation of optimization, then moves on to a historical development of minimum-seeking algorithms. A seemingly simple example reveals many shortfalls of the typical minimum seekers. Since the local optimizers of the past are limited, people have turned to more global methods based upon biological processes. The chapter ends with some background on biological genetics and a brief introduction to the genetic algorithm (GA).

## 1.1 FINDING THE BEST SOLUTION

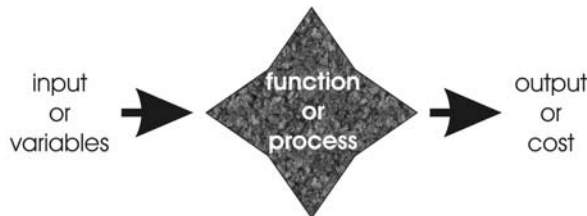
The terminology “best” solution implies that there is more than one solution and the solutions are not of equal value. The definition of best is relative to the problem at hand, its method of solution, and the tolerances allowed. Thus the optimal solution depends on the person formulating the problem. Education, opinions, bribes, and amount of sleep are factors influencing the definition of best. Some problems have exact answers or roots, and best has a specific definition. Examples include best home run hitter in baseball and a solution to a linear first-order differential equation. Other problems have various minimum or maximum solutions known as optimal points or extrema, and best may be a relative definition. Examples include best piece of artwork or best musical composition.

### 1.1.1 What Is Optimization?

Our lives confront us with many opportunities for optimization. What time do we get up in the morning so that we maximize the amount of sleep yet still make it to work on time? What is the best route to work? Which project do we tackle first? When designing something, we shorten the length of this or reduce the weight of that, as we want to minimize the cost or maximize the appeal of a product. Optimization is the process of adjusting the inputs to or characteristics of a device, mathematical process, or experiment to find the minimum or maximum output or result (Figure 1.1). The input consists of variables; the process or function is known as the cost function, objective function, or fitness function; and the output is the cost or fitness. If the process is an experiment, then the variables are physical inputs to the experiment.

For most of the examples in this book, we define the output from the process or function as the cost. Since cost is something to be minimized, optimization becomes minimization. Sometimes maximizing a function makes more sense. To maximize a function, just slap a minus sign on the front of the output and minimize. As an example, maximizing  $1 - x^2$  over  $-1 \leq x \leq 1$  is the same as minimizing  $x^2 - 1$  over the same interval. Consequently in this book we address the maximization of some function as a minimization problem.

Life is interesting due to the many decisions and seemingly random events that take place. Quantum theory suggests there are an infinite number of dimensions, and each dimension corresponds to a decision made. Life is also highly nonlinear, so chaos plays an important role too. A small perturbation in the initial condition may result in a very different and unpredictable solution. These theories suggest a high degree of complexity encountered when studying nature or designing products. Science developed simple models to represent certain limited aspects of nature. Most of these simple (and usually linear) models have been optimized. In the future, scientists and engineers must tackle the unsolvable problems of the past, and optimization is a primary tool needed in the intellectual toolbox.



**Figure 1.1** Diagram of a function or process that is to be optimized. Optimization varies the input to achieve a desired output.



### 1.1.2 Root Finding versus Optimization

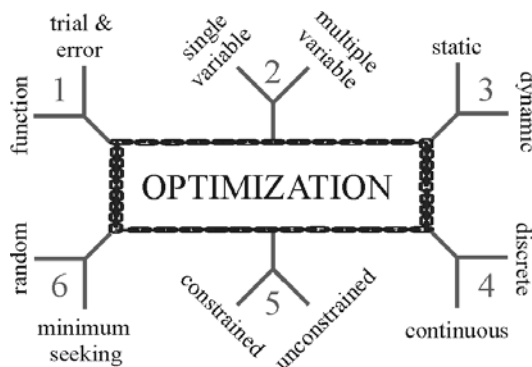
Approaches to optimization are akin to root or zero finding methods, only harder. Bracketing the root or optimum is a major step in hunting it down. For the one-variable case, finding one positive point and one negative point brackets the zero. On the other hand, bracketing a minimum requires three points, with the middle point having a lower value than either end point. In the mathematical approach, root finding searches for zeros of a function, while optimization finds zeros of the function derivative. Finding the function derivative adds one more step to the optimization process. Many times the derivative does not exist or is very difficult to find. We like the simplicity of root finding problems, so we teach root finding techniques to students of engineering, math, and science courses. Many technical problems are formulated to find roots when they might be more naturally posed as optimization problems; except the toolbox containing the optimization tools is small and inadequate.

Another difficulty with optimization is determining if a given minimum is the best (global) minimum or a suboptimal (local) minimum. Root finding doesn't have this difficulty. One root is as good as another, since all roots force the function to zero.

Finding the minimum of a nonlinear function is especially difficult. Typical approaches to highly nonlinear problems involve either linearizing the problem in a very confined region or restricting the optimization to a small region. In short, we cheat.

### 1.1.3 Categories of Optimization

Figure 1.2 divides optimization algorithms into six categories. None of these six views or their branches are necessarily mutually exclusive. For instance, a dynamic optimization problem could be either constrained or



**Figure 1.2** Six categories of optimization algorithms.

unconstrained. In addition some of the variables may be discrete and others continuous. Let's begin at the top left of Figure 1.2 and work our way around clockwise.

1. Trial-and-error optimization refers to the process of adjusting variables that affect the output without knowing much about the process that produces the output. A simple example is adjusting the rabbit ears on a TV to get the best picture and audio reception. An antenna engineer can only guess at why certain contortions of the rabbit ears result in a better picture than other contortions. Experimentalists prefer this approach. Many great discoveries, like the discovery and refinement of penicillin as an antibiotic, resulted from the trial-and-error approach to optimization. In contrast, a mathematical formula describes the objective function in function optimization. Various mathematical manipulations of the function lead to the optimal solution. Theoreticians love this theoretical approach.

2. If there is only one variable, the optimization is one-dimensional. A problem having more than one variable requires multidimensional optimization. Optimization becomes increasingly difficult as the number of dimensions increases. Many multidimensional optimization approaches generalize to a series of one-dimensional approaches.

3. Dynamic optimization means that the output is a function of time, while static means that the output is independent of time. When living in the suburbs of Boston, there were several ways to drive back and forth to work. What was the best route? From a distance point of view, the problem is static, and the solution can be found using a map or the odometer of a car. In practice, this problem is not simple because of the myriad of variations in the routes. The shortest route isn't necessarily the fastest route. Finding the fastest route is a dynamic problem whose solution depends on the time of day, the weather, accidents, and so on. The static problem is difficult to solve for the best solution, but the added dimension of time increases the challenge of solving the dynamic problem.

4. Optimization can also be distinguished by either discrete or continuous variables. Discrete variables have only a finite number of possible values, whereas continuous variables have an infinite number of possible values. If we are deciding in what order to attack a series of tasks on a list, discrete optimization is employed. Discrete variable optimization is also known as combinatorial optimization, because the optimum solution consists of a certain combination of variables from the finite pool of all possible variables. However, if we are trying to find the minimum value of  $f(x)$  on a number line, it is more appropriate to view the problem as continuous.

5. Variables often have limits or constraints. Constrained optimization incorporates variable equalities and inequalities into the cost function. Unconstrained optimization allows the variables to take any value. A constrained variable often converts into an unconstrained variable through a transforma-

tion of variables. Most numerical optimization routines work best with unconstrained variables. Consider the simple constrained example of minimizing  $f(x)$  over the interval  $-1 \leq x \leq 1$ . The variable converts  $x$  into an unconstrained variable  $u$  by letting  $x = \sin(u)$  and minimizing  $f(\sin(u))$  for any value of  $u$ . When constrained optimization formulates variables in terms of linear equations and linear constraints, it is called a linear program. When the cost equations or constraints are nonlinear, the problem becomes a nonlinear programming problem.

6. Some algorithms try to minimize the cost by starting from an initial set of variable values. These minimum seekers easily get stuck in local minima but tend to be fast. They are the traditional optimization algorithms and are generally based on calculus methods. Moving from one variable set to another is based on some determinant sequence of steps. On the other hand, random methods use some probabilistic calculations to find variable sets. They tend to be slower but have greater success at finding the global minimum.

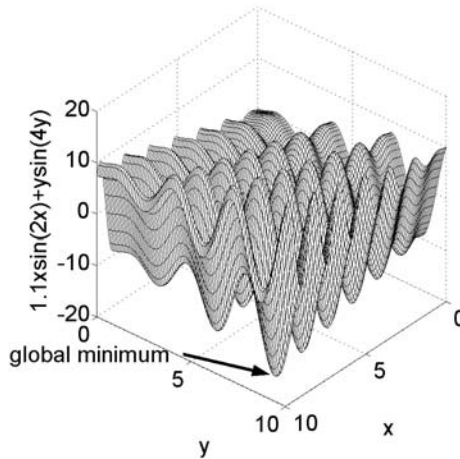
## 1.2 MINIMUM-SEEKING ALGORITHMS

Searching the cost surface (all possible function values) for the minimum cost lies at the heart of all optimization routines. Usually a cost surface has many peaks, valleys, and ridges. An optimization algorithm works much like a hiker trying to find the minimum altitude in Rocky Mountain National Park. Starting at some random location within the park, the goal is to intelligently proceed to find the minimum altitude. There are many ways to hike or glissade to the bottom from a single random point. Once the bottom is found, however, there is no guarantee that an even lower point doesn't lie over the next ridge. Certain constraints, such as cliffs and bears, influence the path of the search as well. Pure downhill approaches usually fail to find the global optimum unless the cost surface is quadratic (bowl-shaped).

There are many good texts that describe optimization methods (e.g., Press et al., 1992; Cuthbert, 1987). A history is given by Boyer and Merzbach (1991). Here we give a very brief review of the development of optimization strategies.

### 1.2.1 Exhaustive Search

The brute force approach to optimization looks at a sufficiently fine sampling of the cost function to find the global minimum. It is equivalent to spending the time, effort, and resources to thoroughly survey Rocky Mountain National Park. In effect a topographical map can be generated by connecting lines of equal elevation from an interpolation of the sampled points. This exhaustive search requires an extremely large number of cost function evaluations to find the optimum. For example, consider solving the two-dimensional problem



**Figure 1.3** Three-dimensional plot of (1.1) in which  $x$  and  $y$  are sampled at intervals of 0.1.

$$\text{Find the minimum of: } f(x, y) = x \sin(4x) + 1.1y \sin(2y) \quad (1.1)$$

$$\text{Subject to: } 0 \leq x \leq 10 \quad \text{and} \quad 0 \leq y \leq 10 \quad (1.2)$$

Figure 1.3 shows a three-dimensional plot of (1.1) in which  $x$  and  $y$  are sampled at intervals of 0.1, requiring a total of  $101^2$  function evaluations. This same graph is shown as a contour plot with the global minimum of  $-18.5547$  at  $(x, y) = (0.9039, 0.8668)$  marked by a large black dot in Figure 1.4. In this case the global minimum is easy to see. Graphs have aesthetic appeal but are only practical for one- and two-dimensional cost functions. Usually a list of function values is generated over the sampled variables, and then the list is searched for the minimum value. The exhaustive search does the surveying necessary to produce an accurate topographic map. This approach requires checking an extremely large but finite solution space with the number of combinations of different variable values given by

$$V = \prod_{i=1}^{N_{var}} Q_i \quad (1.3)$$

where

$V$  = number of different variable combinations

$N_{var}$  = total number of different variables

$Q_i$  = number of different values that variable  $i$  can attain