Stochastic Adaptive Search for Global Optimization

STOCHASTIC ADAPTIVE SEARCH FOR GLOBAL OPTIMIZATION

ZELDA B. ZABINSKY
University of Washington
Seattle, Washington, USA

Kluwer Academic Publishers
Boston/Dordrecht/London

I dedicate this book to my parents, Joe and Helen Zabinsky, to show my love and appreciation.

Contents

List of Figures

List of Tables

Preface

The field of global optimization has been developing at a rapid pace. There is a journal devoted to the topic, as well as many publications and notable books discussing various aspects of global optimization. This book is intended to complement these other publications with a focus on stochastic methods for global optimization.

Stochastic methods, such as simulated annealing and genetic algorithms, are gaining in popularity among practitioners and engineers because they are relatively easy to program on a computer and may be applied to a broad class of global optimization problems. However, the theoretical performance of these stochastic methods is not well understood. In this book, an attempt is made to describe the theoretical properties of several stochastic adaptive search methods. Such a theoretical understanding may allow us to better predict algorithm performance and ultimately design new and improved algorithms.

This book consolidates a collection of papers on the analysis and development of stochastic adaptive search. The first chapter introduces random search algorithms. Chapters 2-5 describe the theoretical analysis of a progression of algorithms. A main result is that the expected number of iterations for pure adaptive search is linear in dimension for a class of Lipschitz global optimization problems. Chapter 6 discusses algorithms, based on the Hit-and-Run sampling method, that have been developed to approximate the ideal performance of pure random search. The final chapter discusses several applications in engineering that use stochastic adaptive search methods.

The target audience includes graduate students, researchers and practitioners in operations research, engineering, and mathematics. A background in mathematics is assumed, as well as a knowledge of probabilistic concepts, such as Markov chains and moment generating functions.

It is possible for readers to skip the proofs and technical details and still grasp the basic ideas.

I apologize ahead of time for errors and inconsistencies in this book. I would appreciate help in correcting mistakes, so please email your suggestions to me at <zelda@u.washington.edu>. I will maintain a web page of errata at http://faculty.washington.edu/zelda/.

I would like to express my thanks to Professor Robert L. Smith for his teaching, mentoring, and continued collaboration. Many of the ideas in this book originated with him. I am also grateful to Professor Graham R. Wood for his inspiration, collaboration and insightful comments. I appreciate the encouragement I received from Dr.s Reiner Horst and Panos Pardalos to undertake this project, and Mr. John Martindale's patience. Dr. Birna Kristinsdottir has been extremely helpful throughout the stages of writing this book. Dr. Mirjam Dür and Dr. David Bulger have given valuable advice during the final stages of the writing. I thank Mr. Kyle Knopp for assisting me with the figures. Many other colleagues have been extremely helpful, and I particularly want to thank (in alphabetical order): Dr.s Bill Baritompa, Vladimir Brayman, Bruce Campbell, Tibor Csendes, Charoenchai Khompatraporn, Victor Korotkich, Wen Luo, Sudipto Neogi, János Pintér, Edwin Romeijn, Vesna Saviç, Yanfang Shen, and Mark Tuttle.

Support from the Industrial Engineering Program at the University of Washington is gratefully acknowledged, as well as a sabbatical in the Department of Mathematics and Computing at Central Queensland University (1996), an Erskine Fellowship from the Department of Mathematics at the University of Canterbury (1998), two National Science Foundation grants (DMI-9622433 for 1996-1999 and DMI-9820878 for 1999-2003), and participation in the Marsden Fund administered by the Royal Society of New Zealand (1999-2002).

I do not have enough words to express my thanks to my husband, Dr. John Palmer, and our children, Rebecca and Aaron, for their patience and enduring support. John has been a reader, prodder, and inspiration for this work. Without him, this book would not have been possible.

Zelda B. Zabinsky

Seattle, Washington, April 2003

Chapter 1

INTRODUCTION

Global optimization refers to a mathematical program, which seeks a maximum or minimum objective function value over a set of feasible solutions. The adjective "global" indicates that the optimization problem may be very general in nature; the objective function may be nonconvex, nondifferentiable, and possibly discontinuous over a continuous or discrete domain. A global optimization problem with continuous variables may contain several local optima or stationary points. The problem of designing algorithms that obtain global solutions is very difficult when there is no overriding structure that indicates whether a local solution is indeed the global solution.

Even though global optimization problems are difficult to solve, applications of global optimization problems are prevalent in engineering and real world systems. Applications include engineering design in mechanical, civil, and chemical engineering, structural optimization, molecular biology and molecular architecture, VLSI chip design, image processing, and a number of combinatorial optimization problems [72, 185]. Engineering functions included in a global optimization problem are often supplied as "black box" functions, which might be a subroutine that returns a function evaluation for a specified solution. An engineering example of this type of problem is to minimize the weight of a structure, while limiting strain to be below a certain threshold [183]. Engineers must often provide some solution to their problem, even if it is a suboptimal one. Sometimes the global optimization problem may be so computationally difficult, that a practitioner is satisfied with any feasible solution. Optimization is currently being applied to complex systems where the objective function and constraints may only be evaluated using a simulation model. Not only does the problem lack structure and

fall into the category of "black box" functions, but it has the additional complication of having randomness in the function evaluation. As our computational capacity and algorithmic knowledge increases, we can apply global optimization to real world problems that previously were not even considered to be framed in optimization terms. Also, as the applications in global optimization develop, they will motivate new and improved methods. Thus there is a synergy between applications and algorithmic techniques.

The ultimate goal of global optimization techniques is to develop a single method that:

- works for a large class of problems,
- finds the global optima with an absolute guarantee, and
- uses very little computation.

We are very far from achieving this goal. Typically the optimization technique is chosen to match the structure of the relevant problem. For example, a practitioner solving a linear program would select a different algorithm than if the problem was nonlinear and convex. If the problem was a linear programming problem, there are many methods that take advantage of the linear structure of the problem [113]. Similarly, a convex problem which is twice continuously differentiable would be better solved with an algorithm that uses the Hessian to take advantage of that structure [13, 103]. However, if the problem was nonlinear and multimodal with mixed continuous and discrete variables, the practitioner would have difficulty selecting the best algorithm to use, and may experiment with a few methods that would then be tailored to the problem at hand. Thus, for global optimization it is still an open research question as to the best choice of algorithm given a particular problem. This indicates the need to develop a better understanding of the algorithm behavior on various problem types.

1. Classification of Optimization Problems

Figure 1.1 illustrates a categorization of optimization problems. It is similar to the NEOS optimization tree [119] in the first division on continuous and discrete variables, but then the trees differ. The intent of Figure 1.1 is to depict a hierarchy of problems. In the categorization of the figure, the first division is on the type of domain, whether the variables are continuous or discrete (a mixed variable category is not included in the figure). Later in the theoretical analysis of pure adaptive search (Chapter 2) and hesitant adaptive search (Chapter 3), similarities are drawn between continuous problems and discrete problems. Further

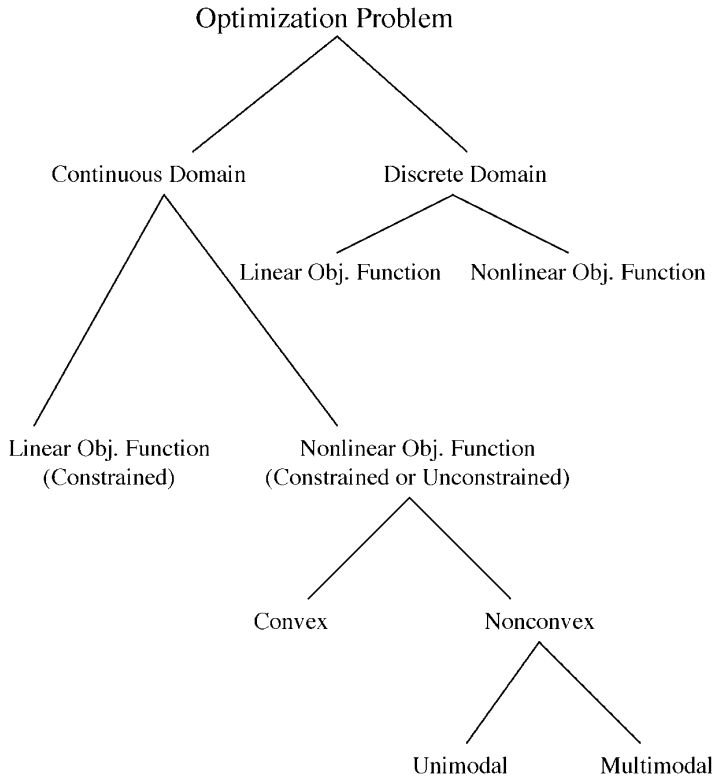

Figure 1.1. Categorization of optimization problems.

categories of the domain may include constrained (bounded) or unconstrained. When the feasible set is constrained, it could be characterized as to whether the feasible set was convex or nonconvex, and whether the equations defining the feasible set were linear or nonlinear. These characterizations are not depicted in the figure.

The second level in Figure 1.1 distinguishes between linear and nonlinear objective functions. Notice for discrete problems, the objective function distinguishes the problem as either a linear discrete or nonlinear discrete problem, whereas commonly an "integer program" presumes the objective function is linear. The characterization of nonlinear continuous problems is further broken down into convex and nonconvex, where nonconvex is typically considered *global optimization*. The nonconvex functions may be unimodal or multimodal. For example, mini-

mizing a concave function over a set of constraints, would fall into the nonconvex multimodal category. This categorization does not rely on derivatives, and hence allows a nondifferentiable convex function (e.g., a function with breakpoints) to fall into the same category as twice continuously differentiable convex functions. On the discrete domain side, there is not an analogous concept of convexity. In fact, discrete feasible sets are never convex according to Rardin [129, page 114]. However, it may be useful to extend the definition of convexity to discrete feasible regions using an algorithmic perspective based on neighborhoods of discrete points. Discrete problems with nonlinear objective functions are considered *global optimization* problems in this book. This organization of optimization problems may suggest categories of problems that share similar characteristics and thus may guide algorithmic development.

2. Types of Algorithms

Global optimization algorithms are often classified as either deterministic or stochastic. The focus of this book is on stochastic methods that can be applied to global optimization problems with little known structure, such as "black-box" functions. There are several excellent books on global optimization, including the two volume Handbook of Global Optimization [72, 122], an overview of deterministic methods by Horst and Tuy [73], and an introduction to global optimization with stochastic methods by Törn and Žilinskas [165]. A stochastic method in this book refers to an algorithm that uses some kind of randomness (typically a pseudo-random number generator), and may be called a Monte Carlo method. Examples include pure random search, simulated annealing, and genetic algorithms.

Why stochastic search as opposed to deterministic search methods? Random search methods have been shown to have a potential to solve large problems efficiently in a way that is not possible for deterministic algorithms. Dyer and Frieze [48] showed that estimating the volume of a convex body takes an exponential number of function evaluations for *any* deterministic algorithm, but if one is willing to accept a weaker claim of being correct with an estimate that has a high probability of being correct, then a stochastic algorithm can provide such an estimate in polynomial time. Thus there is a trade-off between the amount of computation and the type of guarantee of optimality. The analyses in this book also relax the requirement of providing an absolute guarantee of the global optimum and instead are satisfied with a probabilistic estimate of the global optimum. One question is whether a stochastic algorithm can be executed in polynomial time, on the average, while it is known that a deterministic method for global optimization is NP-hard

[173]. This is at the heart of the research presented in this book, and is explored in more detail in subsequent chapters.

Another advantage to stochastic methods is that they are relatively easy to implement on complex problems. Simulated annealing, genetic algorithms, tabu search and other random search methods are being widely applied to continuous and discrete global optimization problems [129]. Because the methods typically only rely on function evaluations, rather than gradient and Hessian information, they can be coded quickly, and applied to a broad class of ill-structured problems. A disadvantage to these methods is that they are currently customized to each specific problem largely through trial and error, and there is little theory to support the quality of the solution. A common experience is that the stochastic algorithms perform well and are "robust" in the sense that they give useful information quickly for ill-structured global optimization problems.

A general theory of performance of stochastic search algorithms is presented in this book. The basic measure of performance is the number of iterations until first sampling within ϵ of the global optimum. The analysis of performance is developed by investigating a series of algorithms that are theoretical in nature - because they assume properties of the sampling distribution that may not be practically implemented. However their analysis motivates algorithms that are practical. The performance analysis also provides an understanding of how the sampling distribution of an algorithm is related to the performance measure.

3. Definitions and Assumptions

The basic global optimization problem (P), used throughout the book, is defined as,

$$(P) \qquad \min_{x \in S} f(x) \tag{1.1}$$

where x is a vector of n decision variables, S is an n-dimensional feasible region and assumed to be nonempty, and f is a real-valued function defined over S. The goal is to find a value for x contained in S that minimizes f. Let the global optimal solution to (P) be denoted by (x_*, y_*) where

$$x_* = \arg\min_{x \in S} f(x) \tag{1.2}$$

and

$$y_* = f(x_*) = \min_{x \in S} f(x). \tag{1.3}$$

It will also be convenient to define

$$y^* = \max_{x \in S} f(x).$$

In order to ensure a global optimum exists, we need to assume some regularity conditions. If (P) is a continuous problem and the feasible set S is nonempty and compact, then the Weierstrass theorem from classical analysis guarantees the existence of a global solution [103]. If (P) is a discrete problem and the feasible set S is nonempty and finite, a global solution exists. The existence of a global optimum can be guaranteed under slightly more general conditions, but these conditions are sufficient for the purposes of this book. Note that the existence of a *unique* minimum at x_* is not required. If there are multiple optimal minima, let x_* be an arbitrary fixed global minimum.

A distinction is usually made between local optima and global optima. A local optimum will be defined as a feasible point $\hat{x}$ such that sufficiently small neighborhoods surrounding $\hat{x}$ contain no points that are both feasible and improving in objective function value. For continuous domains, a small neighborhood is typically a ball of radius δ, where $\delta > 0$. For discrete domains, the set of nearest neighbors to $\hat{x}$ could be used to determine whether $\hat{x}$ is a local optimum. Notice the relationship between the concept of small neighborhood and local optimum. The definition of neighborhood, especially for discrete problems, is usually associated with an algorithm, rather than the definition of the problem, which implies that a point $\hat{x}$ might be a local optimum with respect to one algorithm and neighborhood structure, but not with respect to a different algorithm and neighborhood structure. For example, the Traveling Salesperson Problem has several possible neighborhood structures that might impose different interpretations of local optima. Thus it is important to remember that a local optimum is related to a neighborhood structure. The definition of global optima (in Equations 1.2 and 1.3) is more straightforward because it is relative to the entire feasible region, not just a local region.

The contours of the objective function of a global optimization problem in two dimensions provides a useful visualization. The level set of the function may be interpreted as the set defined by a single contour. The level set at value y, denoted

$$S(y) = \{x : x \in S \text{ and } f(x) \leq y\},$$

is the set of feasible solutions whose objective function values are y or better. The level set is defined as including points with equal objective function values, but in some algorithms we also define an improving set to only include points that are strictly improving. This distinction becomes important in the analysis of discrete problems. It is discussed in more detail when describing pure adaptive search on a finite domain, in Chapter 2. Also let $N(y)$ be the number of iterations needed to first

achieve an objective function value of y or less, which corresponds to the number of iterations to first obtain a sample point in the level set $S(y)$. It is also convenient in Chapter 3 to define $M(y)$ as the number of iterations just before landing in $S(y)$, and the relationship $N(y) = 1 + M(y)$ accounts for the extra iteration to actually land in $S(y)$. The number of iterations needed to achieve an accuracy of y or less, $N(y)$, is the primary measure of performance for an algorithm in this book. It may be called the first passage time or first hitting time in the literature on stochastic processes. If y is close to the optimum, for example $y = y_* + \epsilon$ for a small positive value of ϵ, then $N(y_* + \epsilon)$ describes the number of iterations to get within ϵ of the optimum. In this book, we focus on the expected value of $N(y)$, and derive the distribution when possible.

3.1 Assumptions for Continuous Problems

Additional assumptions and restrictions on the generality of the global optimization problem (P) are made throughout this book as needed, however a brief discussion is given here.

A *continuous* global optimization problem is classified by the feasible region containing real-valued, continuous variables, $S \subset \mathbb{R}^n$. Typically, the feasible region for a continuous problem, S, is assumed to be a nonempty, compact set which is full-dimensional. The feasible region can be described by simple upper and lower bounds, $x_i^L \leq x_i \leq x_i^U$ for $i = 1, \ldots, n$ known as *box* constraints, or by more general functional constraints, $g_j(x) \leq 0$ for $j = 1, \ldots, m$. In general, S may form a nonconvex set, which can possibly be disconnected. In this case, we usually assume upper and lower bounds on the variables are known, such that S is contained in a box, or hyperrectangle.

For a continuous global optimization problem, there are several traditional ways to classify the objective function $f(x)$. The objective function $f(x)$ may be a linear function (e.g. linear programming), a nonlinear convex function, a unimodal but nonconvex function, or a multimodal function. Typically nonlinear programming assumes the objective function is nonlinear, convex and twice continuously differentiable, while global optimization often includes functions that are not differentiable everywhere, and may be discontinuous (e.g., step functions). These characteristics of the objective function may also be used to describe the constraint equations.

Another way to characterize a function of continuous variables is whether it satisfies the Lipschitz condition. A function f satisfies the *Lipschitz condition* with Lipschitz constant K, if

$$|f(x) - f(y)| \leq K\|x - y\| \tag{1.4}$$

for all x and $y \in S$, where $\| \cdot \|$ is the Euclidean norm on $\mathbb{R}^n$. A function satisfying the Lipschitz condition has a bound on the derivative, and according to Törn and Žilinskas [165, page 25], practical objective functions often have such bounds. It is less common to actually know the value of the bound. Many stochastic global optimization algorithms assume a Lipschitz constant exists, but do not use the actual value in the algorithm. This is in contrast to Lipschitz optimization which requires the value or an upper bound of the Lipschitz constant. Algorithms that require the Lipschitz constant and rely on estimates are discussed in [64]. Several analyses presented later in the book assume the objective function satisfies the Lipschitz condition.

3.2 Assumptions for Discrete Problems

A *discrete* global optimization problem is classified by the feasible region containing discrete variables. The feasible region may be described in several ways; for example S could be the set of integers between 0 and 100, which retains numerical properties, or S could be a non-ordered set such as $S = \{\text{red}, \text{yellow}, \text{blue}\}$, or the list of cities on a tour for the Traveling Salesperson Problem. Typically, the feasible region for a discrete problem, S, is assumed to be a nonempty and finite, although it is possible for S to be infinite with a bounded objective function. The concept of "dimension" is not always appropriate for a discrete global optimization problem, so it is difficult to compare a discrete domain with a continuous domain. One way to construct a comparable problem is to consider the distinct points in an n-dimensional lattice, $\{1, \ldots, k\}^n$. The number of points in the domain for the lattice is k^n, and as k gets large, the discrete domain resembles a continuous domain of dimension n. Later in this book, a lattice is used to compare a discrete problem with a continuous one.

The objective function for a discrete global optimization problem is a real-valued function defined on the points in S. The objective function $f(x)$ may have a functional form, or be evaluated by the means of a subroutine or computer code on the possible points in S. For example, $f(x)$ and S for a continuous global optimization problem can be turned into a discrete problem by simply adding the constraint that x be integer valued. The objective function for a discrete problem with integer variables may be considered linear or nonlinear with respect to the variables when they are relaxed to be continuous variables. This is interesting when algorithms (such as simulated annealing) define neighborhoods that may not be a natural neighborhood on the relaxed problem. Analogous def-

initions of linearity, convexity, Lipschitz condition, and other concepts must be extended to discrete problems to better generalize methods that are appropriate for both continuous and discrete domains.

3.3 Mixed Continuous-discrete Problems

A *mixed continuous-discrete* global optimization problem is classified by the feasible region containing both continuous and discrete variables. In engineering applications, it is often convenient to define a global optimization problem for continuous variables, and then explore the effects of limiting a subset of the variables to discrete values. In Chapter 7, a 10-bar truss problem is described, where the decision variables are the diameters of the bars. While the diameter of a bar is mathematically a continuous variable, in reality, bars are only manufactured and readily available with a discrete set of standard diameters. Thus a continuous problem becomes a discrete one when taking practical considerations into account.

Another example, described in more detail in Chapter 7, is a composite structure, where the number of plies is an integer variable while the height and width of stiffeners are continuous variables. Additional variables in a composite structure are the fiber angles of the plies. The fiber angle of a ply is similar to the diameter of a truss member in that mathematically the angle could take on any value between ± 90 degrees, however manufacturing technological constraints restrict the angle to a discrete set of values, e.g., 0, ± 45, or ± 90 degrees. In order to solve these types of real-world problems, we need robust algorithms that can be applied to global optimization problems with a mixture of continuous and discrete variables.

4. Overview of Random Search Methods

One motivation for random search methods in global optimization is the potential to obtain approximate solutions quickly and easily. In addition, theoretical analysis of random search methods indicates that performance may be very good, possibly polynomial in dimension, despite the fact that global optimization problems are NP-hard for deterministic methods. A brief overview of random search methods is presented in this section to provide some context for stochastic methods for global optimization. Underlying all of these methods is a probabilistic approach to sampling the feasible region. Whether the algorithm is simulated annealing, a genetic algorithm or multistart, it has some method of generating new candidate points, which can be called its sampling distribution. The sampling distribution employed by the algorithm is

used in the subsequent analyses to characterize the performance of the method.

We start with a brief description of exhaustive search, including grid search and pure random search. Then we present a framework for sequential random search and a brief discussion of simulated annealing. This is followed by a framework for two-phase methods, including multi-start, clustering, single linkage and multi-level single linkage algorithms. Finally a brief overview of population-based algorithms is discussed, including a framework for genetic algorithms and similar methods. The common theme carried throughout the book is studying the probability distribution of the points generated by an algorithm, and the impact the sampling distribution has on the complexity and behavior of the algorithm on classes of global optimization problems.

4.1 Enumeration or Exhaustive Search

When confronted with a global optimization problem, the basic and perhaps most natural approach is to simply evaluate all points in the domain. If the domain S is finite and relatively small, an exhaustive search is a reasonable approach. As Rardin notes [129, page 627], if the domain has only a few discrete decision variables, the most effective optimization method is often the most direct: enumeration of all the possibilities. However, in most discrete global optimization, it is not practical to perform complete enumeration. Combinatorial optimization includes the Traveling Salesperson Problem, which for an N-city tour, has $(N-1)!$ points in the domain. For $N = 10,000$, this is over 10^{20} possible tours, and at one CPU second per function evaluation, brute force enumeration would exceed 10^{12} years.

Grid Search. When the global optimization problem involves continuous variables, there are an infinite number of points in the domain, and complete enumeration is impossible. A common approach is to perform a *grid search,* essentially discretizing the domain. A grid search creates an equally spaced grid of points over the feasible region, and evaluates the objective function at each point. If the objective function satisfies the Lipschitz condition with constant K and the domain is an n-dimensional hyperrectangle of maximum length D on each side, then the grid spacing can be determined (see Dixon and Szegö [44, 45]) to achieve a desired accuracy, of having the estimate within ϵ of the global optimum. If the spacing between grid points is ϵ/K on each coordinate, then there are approximately $(KD/\epsilon)^n$ grid points, and the possible error between adjacent points is bounded by ϵ. Hence, the number of evaluations needed to obtain an accuracy of ϵ is proportional to $(KD/\epsilon)^n$. Thus, the number

of function evaluations to achieve an accuracy of ϵ is exponential in the dimension n.

Pure Random Search. A stochastic version of grid search is pure random search. *Pure random search,* discussed in Chapter 2, was first defined by Brooks [26], discussed by Anderssen [6], and later named in the classic volumes by Dixon and Szegö [44, 45]. Pure random search samples repeatedly from the feasible region S, typically according to a uniform sampling distribution. Although the points of pure random search are not evenly spaced, as in grid search, they are uniformly scattered over the feasible region. If the objective function has some regularity that coincides with the regularly spaced grid points (e.g., a sinusoidal function), then the probabilistic nature of pure random search provides an advantage. While pure random search sacrifices the guarantee of determining the optimal solution within ϵ, it can be shown that pure random search converges to the global optimum with probability one. A similarity between grid search and pure random search is that the expected number of function evaluations for pure random search to get within ϵ is also exponential in the dimension n (see Chapter 2).

Other Covering Methods. Other sampling methods that eventually cover the feasible set may also be used in a type of exhaustive search. Törn and Žilinskas discuss other covering methods [165, Chapter 2] which provide a means to sample points throughout the feasible region in a thorough manner. They include quasi-random sequences, such as introduced by Halton (see [165, page 33], or [63]) as providing a uniform covering. Unfortunately, the rate of convergence for a covering method is in general slow, as is demonstrated by the exponential complexity of both grid search and pure random search. To improve performance, some means to focus the search on promising regions is needed.

4.2 Sequential Random Search

Stochastic algorithms tend to be categorized into two classes, sequential algorithms, and two-phase methods which include multistart and clustering methods. Section 4.2 provides an overview of sequential algorithms, including simulated annealing, and Section 4.3 provides a brief description of two-phase methods. It should be noted that this categorization is blurred as new algorithms are being developed. For example, it is not clear which categorization includes genetic algorithms; here they are summarized in Section 4.4.

Sequential random search, including simulated annealing, has been applied to many "black box" global optimization problems. Sequential random search procedures can be characterized by producing a sequence of random points $\{X_k\}$ on iteration $k, k = 0, 1, \ldots$ which may depend on the previous point or several of the previous points. We provide a framework for sequential random search.

Sequential Random Search

Step 0. Initialize algorithm parameters and initial point $X_0 \in S$ and set iteration index $k = 0$.

Step 1. Generate a candidate point $V_{k+1} \in S$ according to a specific *generator*.

Step 2. Update the current point X_{k+1} based on the candidate point and previous points.

Step 3. If a stopping criterion is met, stop. Otherwise update algorithm parameters, increment k and return to Step 1.

Sequential random search depends on two basic iterations, the generator in Step 1 that produces candidate points, and the update procedure in Step 2 that may accept a candidate point to be the next point in the sequence. The sequence of points X_k provides a search path through the feasible set. If the objective function is consistently improving, $f(X_0) > f(X_1) > \cdots$ then the algorithm has an improving search. If, as in simulated annealing, non-improving points are occasionally accepted in the update procedure, the search path is not consistently improving. In all practical implementations that this author has seen, a part of Step 3 includes recording the best point found so far. In this way, an algorithm may update X_{k+1} with a non-improving point without risk of losing the value of the incumbent solution. This has implications when discussing convergence of an algorithm, and distinguishing whether the algorithm converges to the global optimum with respect to the current point, or the algorithm converges to the global optimum with respect to the best point sampled thus far.

Simulated Annealing. Simulated annealing may be viewed as a type of sequential search algorithm. Simulated annealing was motivated by the physical annealing process when slowly cooling metals, and introduced by Metropolis, et al. [110], and later by Kirkpatrick in 1983 [89]. Simulated annealing has most often been applied to combinatorial problems (see Aarts and Korst [1]), such as the Traveling Salesperson

Problem [89] and various scheduling problems [126], but has also been applied to continuous or mixed domain problems [18, 36, 140, 141, 168]. While simulated annealing had a burst of popularity in the 80's and 90's, other sequential random search algorithms predate it. Several of these sequential random search methods were reported as experiencing computation that is linear in dimension, and are discussed later in this section.

The generator, or method of generating candidate points as in Step 1, is usually specific to each algorithm and tailored for the problem at hand. The generator usually makes a move in the vicinity of the current point X_k on iteration k. In analyses, the generator is often viewed as a Markov chain characterized by the one-step transition probability. In continuous domains, the generator often involves making a step in a specified direction. Then the transition probability would correspond to the probability of selecting a particular direction and step. In discrete domains, a similar step generator may be used, however it is more common for discrete generators to be described as selecting a candidate point at random from a neighborhood, where the definition of the neighborhood is specific to the problem. A discrete neighborhood may be the set of nearest neighbors, that is the set of points that can be sampled with one transition. For the Traveling Salesperson Problem, several generators have been proposed, including a one-city swap, or a k-city swap where k links are deleted and replaced in a different way that maintains feasibility [46]. A generator based on Hit-and-Run is discussed in Chapter 6 that does not have to be tailored for specific problems, but can be defined for a general class of global optimization problems.

A typical feature of simulated annealing distinguishing it from other sequential random search algorithms is that, besides accepting points that have improvements in objective function value, it also has a probability of accepting non-improving points. The update procedure in Step 2 of sequential random search for simulated annealing is

$$X_{k+1} = \begin{cases} V_{k+1} & \text{with acceptance probability } P_{T_k}(X_k, V_{k+1}) \\ X_k & \text{otherwise} \end{cases}$$

for any iteration k. The parameter T_k is referred to as the temperature, and it is initialized at a large value. The temperature controls the probability of accepting a non-improving point; when the temperature is high there is a large probability of accepting a non-improving point, and as the temperature decreases to zero the probability of accepting a non-improving point also decreases to zero.

The acceptance probability P_{T_k} of accepting a candidate point V_{k+1}, given the current iteration point X_k and the current temperature T_k, is

typically given by,

$$P_{T_k}(X_k, V_{k+1}) = \begin{cases} 1 & \text{if } \textit{improving,} \\ & \text{i.e., } f(V_{k+1}) < f(X_k) \\ e^{\left[\frac{f(X_k)-f(V_{k+1})}{T_k}\right]} & \text{otherwise,} \\ & \text{i.e., } f(V_{k+1}) \geq f(X_k) \end{cases} \tag{1.5}$$

The acceptance criterion based on P_{T_k} is also known as the Metropolis criterion [110]. The Metropolis criterion was introduced to model the cooling of metals, and the difference in objective function values $f(X_k) - f(V_{k+1})$ was referred to as the energy difference, and the temperature T_k involved the temperature of the heat bath as well as a physical constant known as the Boltzmann constant. The temperature is gradually reduced, and if the lowering of the temperature is sufficiently slow, the metal can reach thermal equilibrium at each temperature. The sequence of points generated while holding temperature constant T converges to the Boltzmann distribution, which characterizes the thermal equilibrium (page 14, [1]). The Boltzmann distribution plays an important role in characterizing the underlying probability of the points sampled and accepted. This is discussed further in Chapter 4.

The rate at which temperature is gradually reduced is critical to the annealing process, and if lowered too quickly results in an inferior metal. The optimization analogy is that if the temperature is lowered too quickly, the algorithm gets trapped at a local optimum. The control mechanism to reducing the temperature is called the cooling schedule. Let $\tau(T_k)$ be a function that gradually reduces the temperature T_k on iteration k. A simple geometric cooling schedule is $\tau(T_k) = 0.9T_k$. Initially when the temperature T_k is large, many non-improving points will be accepted, but as T_k approaches zero, mostly improving points will be accepted. This feature allows the algorithm to escape from local optima when T_k is large, but allows convergence to, hopefully the global optimum, as T_k becomes small. A cooling schedule often attempts to allow the optimization process to reach thermal equilibrium (e.g., the Boltzmann distribution) by lowering the temperature after a minimum number of iterations N_{T_k} at each temperature step. Often the cooling schedule is fine-tuned for a particular problem, but recent results derive an analytically motivated cooling schedule [86, 140]. Some of these results are summarized in Chapter 4.

We now summarize the simulated annealing algorithm.

Simulated Annealing

Step 0. Initialize algorithm parameters, including temperature T_0 and initial point $X_0 \in S$ and set iteration index $k = 0$.

Step 1. Generate a candidate point $V_{k+1} \in S$ according to a specific *generator*.

Step 2. Update the current point X_{k+1} using

$$X_{k+1} = \begin{cases} V_{k+1} & \text{with probability } P_{T_k}(X_k, V_{k+1}) \\ X_k & \text{otherwise} \end{cases}$$

where $P_{T_k}(X_k, V_{k+1})$ is given in Equation 1.5. Update algorithm parameters, including $T_{k+1} = \tau(T_k)$.

Step 3. If a stopping criterion is met, stop. Otherwise increment k and return to Step 1.

The convergence properties of simulated annealing have been analyzed in [1] for combinatorial optimization problems. The algorithm is analyzed using Markov chain theory, where iterations correspond to transitions and the state space is the finite set of outcomes. It is shown that simulated annealing asymptotically converges to the set of global optimal solutions with probability one. This is done by proving that the Markov chain describing the algorithm converges to a stationary distribution. The number of transitions required to approximate the stationary distribution depends on the second largest eigenvalue of the transition matrix. This can be used to show that the stationary distribution is approximated arbitrarily closely, only if the number of transitions is at least quadratic in the size of the solution space. For instance if the solution space S is an n dimensional binary lattice, then there are 2^n possible solutions in the solution space where n denotes the dimension of the problem. Therefore it will take at least $(2^n)^2$ transitions to verify the global optimal solution. This means that approximating the stationary distribution arbitrarily closely results in an exponential time execution of the simulated annealing algorithm. Notice that this involves verifying the global optimal solution. The authors do not give the expected number of iterations to find the global optimal solution for the first time. Other analyses by Locatelli [99, 100] and Trouvé [167] provide conditions for which simulated annealing converges in probability to the global optimum. Romeijn, et al. [141] provide convergence results for simulated annealing with both continuous and discrete variables. This is discussed in more detail in Chapter 4.

Step Size Algorithms. A common method to generate a candidate point on problems with continuous variables is take a step size in a vector direction, called a direction-step paradigm in [129]. In continuous problems, the direction of movement may be based on gradient information, although not necessarily. Sequential random search, in Step 1, typically generates a candidate point by taking a step of length S_k in a specified direction D_k:

$$X_{k+1} = X_k + S_k D_k$$

on an iteration k. In a gradient search type algorithm, the direction is based on local information by evaluating the gradient at the current point, and the step length may be the result of a line search. Quasi-Newton methods take advantage of an approximation of the Hessian to provide a search direction. As an alternative, a direction is often generated according to a distribution (often uniform on a hypersphere), and the step length may also be randomly generated.

A collection of sequential step size algorithms, including those of Rastrigin, et al. [114, 131, 132], Steiglitz, et al. [97, 152], Schrack, et al. [150, 151], and Solis and Wets [159] fit into this category of sequential random search, and all obtain a direction vector by sampling from a uniform distribution on a unit hypersphere. The method of choosing the is specific to each algorithm, such as shrinking or expanding the step length based on previously chosen points. Several of these sequential random search methods have been reported as experiencing computation that is linear in dimension.

After a candidate point is generated, Step 2 of sequential random search specifies a procedure to update the current point. Algorithms that are strictly improving have a simple procedure, update the current point only if the candidate point is improving,

$$X_{k+1} = \begin{cases} V_{k+1} & \text{if } f(V_{k+1}) < f(X_k) \\ X_k & \text{otherwise.} \end{cases}$$

This type of improving algorithm may get trapped in a local optimum if the generator does not sample over the entire domain. If the neighborhood, or procedure for generating candidate points is too restricted, it is difficult to find the global optimum. One remedy is to sample a large neighborhood, possibly draw from the entire feasible set, and another remedy is to accept non-improving points (as in simulated annealing, Section 4.2). The algorithms based on Hit-and-Run sampling methods presented in Chapters 2, 3 and 6 use a global reaching search strategy so there is a positive probability of sampling anywhere in the entire feasible region. Recent research in Very Large Scale Neighborhood Search

is pursuing the benefit of enlarging the neighborhood search. The computational tradeoffs between sampling over a very large neighborhood and possibly the entire feasible region, versus sampling over a restricted neighborhood but accepting non-improving points is explored in subsequent chapters.

Convergence. A convergence proof for sequential random search algorithms was provided by Solis and Wets [159], where convergence means that, with probability 1, the sequence $f(X_k)$ converges to the infimum of f on S as the iteration counter k tends towards infinity. The convergence theorem for global search [159, page 20] makes two assumptions. The first assumption is roughly that the update procedure (Step 2 in the sequential random search algorithm) chooses the best of the points found thus far, and the second, more restrictive assumption, is that given any subset A of S with positive "volume," the probability of repeatedly missing the set A when generating the random samples (in Step 1) must be zero. It basically says that, as long as the method of generating a subsequent point does not consistently ignore any region, then the algorithm will converge with probability one. This is particularly relevant to the step-size algorithms of Rastrigin, Schumer and Steiglitz, and others [97, 131, 132, 114, 152, 150, 151].

A second convergence proof due to Bélisle [14] says that, even if the generator of an algorithm cannot reach any point in the domain in one iteration, if there is a means such as an acceptance probability to allow the algorithm to reach any point in a finite number of iterations, then the algorithm still converges with probability one to the global optimum.

Algorithms for global optimization must either sample the entire set or use the structure of the problem to guarantee convergence. Stephens and Baritompa [161] formalize this by proving that convergence requires global information. Examples of global information that may improve performance of an algorithm include the Lipschitz constant, bounds on derivatives, bounds on the function as in interval methods, information on the level sets, number of local optima, functional form, and the global optimum itself. Stephens and Baritompa show that deterministic algorithms must sample a dense set to find the global optimum value, and proved analogous results for stochastic algorithms. They show [161, Theorem 3.2] that for any deterministic sequential sampling algorithm on a sufficiently rich class of functions F, there exists a function in F for which the algorithm fails to detect the global optimum. The analogous stochastic result [161, Theorem 3.4] is that, for any stochastic sequential sampling algorithm and any $\epsilon > 0$, there exists a function in F such that the probability that the algorithm detects the global op-

timum is less than ϵ. While these results show that attempts to find the global optima on *all* functions is doomed to failure, Stephens and Baritompa conclude that using "global optimization heuristics is often far more practical than running general algorithms until the mathematically proven stopping criteria are satisfied" [161, page 587]. They point to the need to quantify the 'niceness' of realistic functions so that practical problems and algorithms may be combined with mathematical confidence in the results. By relaxing the criteria of guaranteeing a global solution to a weaker claim of probabilistically detecting the global optimum, the analyses in the book hope to be useful in bridging the gap between mathematical confidence and practicality.

Why is it that sometimes random search algorithms appear to find a global optimum quickly, when other times they appear to get trapped at a local optimum? The rate of convergence is one way to characterize performance. With random search algorithms, the measure of performance should include both the speed of the algorithm as well as the accuracy of the final solution. This is explored in subsequent chapters.

There is experimental evidence in the literature that suggests sequential random search algorithms are efficient for large dimensional quadratic programs. Schumer and Steiglitz [152] provide experimental evidence that the number of function evaluations increases linearly with dimension for their adaptive step size algorithm on the following three test functions: $\sum_{i=1}^{n} x_i^2$, $\sum_{i=1}^{n} x_i^4$, and $\sum_{i=1}^{n} a_i x_i^2$. They also prove that the average number of function evaluations for an optimum relative step size random search restricted to an unconstrained quadratic objective function is asymptotically linear in dimension. Schrack and Borowski [150] report experimental results on a quadratic test function, $\sum_{i=1}^{n} x_i^2$, that doubling the dimension doubles the number of function evaluations required for their random search algorithm. Solis and Wets [159] experimentally verified a linear correlation between the number of function evaluations and dimension for their own variation of the step size algorithm on a quadratic test function. They provided a justification of this linearity condition based on the tendency of these algorithms to maintain a constant probability of successful improvement.

There is also theoretical justification that sequential random search algorithms are efficient for a larger class of global optimization problems. In Zabinsky and Smith [188], complexity is measured as the expected number of iterations needed to get arbitrarily close to the solution with a specified degree of certainty, and this measure is used throughout the book. An analysis of a random search procedure called pure adaptive search [125, 188] proves that it is theoretically possible for a sequential random search procedure to achieve linear complexity (in improving

iterates) for global optimization problems satisfying the Lipschitz condition. If sequential random search algorithms behave similarly to pure adaptive search, the analysis would explain why they appear efficient. The linearity performance of pure adaptive search is discussed in detail in Chapter 2.

4.3 Two-Phase Methods

Many global optimization algorithms may be thought of as having two phases, a global phase when sampling occurs over the entire feasible region, and a local phase when sampling occurs over a restricted, or focused, portion of the feasible region. A common example of combining a global phase with a local phase is a multistart method. Multistart samples starting points (often uniformly) from the entire set during its global phase, and uses them to initiate deterministic gradient search type algorithms as a part of its local phase. It is also useful to consider other algorithms as a part of a global and local phase when considering the performance or behavior of the algorithms. For instance, a multistart scheme may be used where simulated annealing can be viewed as the local phase. The global phase can be viewed as an exploratory phase aimed at exploring the entire feasible region, while the local phase can be viewed as an exploitation phase aimed at exploiting the location and/or local information (e.g. gradient) to improve on the objective function.

Schoen [149] provides a general scheme for a two-phase method, and Wood, et al. [178] also states a generic stochastic optimization algorithm. A generic two-phase algorithm is stated as follows.

Basic Two-Phase Stochastic Global Optimization Algorithm

Step 0. Initialize algorithm parameters and set iteration index $k = 0$.

Step 1. In the global phase, generate $X_k \in S$ according to a sampling distribution over S.

Step 2. In the local phase, generate a candidate point $V_k \in S$ according to a local sampling distribution, or local descent algorithm, and update information on the best solution found so far as well as other parameters of interest.

Step 3. If a stopping criterion is met, stop. Otherwise update algorithm parameters, increment k and return to Step 1.

The multistart algorithm fits into this framework, where the global phase generates a point using a sampling distribution, and the local

phase performs a type of local search. Typically the local phase in multistart is a deterministic gradient search type algorithm, although experiments using simulated annealing and other sequential random search methods within multistart have promising results [42]. A multistart algorithm may repeatedly find a local optimum from many starting points, and clustering methods have attempted to modify the multistart idea by reducing the number of times a local search is initiated. Clusters are formed (often grown around a seed point) to predict whether a starting point is likely to lead to a local optimum already discovered, and local searches are only initiated at promising candidate points (see [165, 149]). A related idea has led to linkage methods, which "link" points in the sample and essentially view clusters as trees, instead of spherical or ellipsoidal clusters. The most well known linkage method is Multi Level Single Linkage [137, 138] and several variants are summarized in [149].

4.4 Genetic Algorithms

The division between the global phase and the local phase blurs when combining algorithms. For instance, when a simulated annealing is coupled with multistart the local phase has a global aspect to it. Population-based algorithms, including genetic algorithms and evolutionary programming [91], maintain a set of current points called the population. The current population is used to generate candidate points for a new population, typically by combining pairs of points with specific rules of crossover, mutation and reproduction. The current and new points are then evaluated and compared to update the population. The motivation behind population-based algorithms is to parallel the process of biological evolution.

Several details must be specified to completely define the algorithm. First, the population size must be chosen. The *elitist strategy* subdivides the population into three categories and maintains p_e elite (best) solutions, p_i immigrant solutions (added to maintain diversity), and p_c crossover solutions. So the total population size is $p = p_e + p_i + p_c$ (described in [129]). Computational experience indicates that the population size impacts performance, if it is too small, the algorithm has difficulty finding the global optimum, and if it is too large, then it is inefficient and essentially pure random search. The right population size for a particular problem is usually found through experimentation.

Given a current population of p points, the next step is to generate candidate points to use in a new population. The primary mechanisms are crossover, mutation, and reproduction. Crossover implies that two "parent" solutions in a population are used to generate two "children" solutions by breaking both parent solutions and reassembling the first

part of one with the second part of the other and visa versa. This is usually done with binary variables, but variations on continuous variables have also been demonstrated. It appears that direct crossover on continuous variables may be more efficient than encoding a real-valued number into binary and then performing crossover. Mutation is used to randomly alter a single solution. Reproduction is used to duplicate promising solutions which modifies the composition of the population.

The final step is to merge the current population with the new candidate points. A naive approach would be to simply rank order the population on objective function values and select the best, however experience has shown that some diversity should be maintained, to prevent premature convergence to a local optimum. Various fitness measures have been proposed, and different strategies to merge the current and new populations by selecting "survivors" and "immigrants" to maintain the population size. As with simulated annealing, the best solution is always recorded, but unlike simulated annealing, a population of alternative solutions is provided. Of course, practitioners could keep track of alternative solutions in a simulated annealing algorithm too, and keep a list of the p best solutions if that is of interest.

In keeping with the theme of the book, the method of generating candidate points may be viewed as a probability distribution. Researchers have used Markov chain analysis [40] to analyze the behavior, such as the expected waiting time, of genetic algorithms [92]. This is similar to analyses presented in subsequent chapters.

4.5 Other Stochastic Methods

There are many other stochastic methods that have been proposed for global optimization with discrete and/or continuous variables that will not be mentioned in this book. Some interesting methods include, tabu search [55], nested partitioning method [156], Lipschitz optimization [127], controlled random search [4], localization search [11, 177], and uniform covering using raspberries [69]. There are even more deterministic methods, and hybrid algorithms combining deterministic and stochastic methods (e.g. [93]). This book is not able to review them all, but instead strives to present a uniform way of viewing stochastic algorithms.

5. Overview of this Book

In Chapter 2, we present a theoretical development of the performance of two stochastic methods that can be viewed as extremes. These two random search algorithms, pure random search (PRS) and pure adaptive search (PAS), are not intended to be practical algorithms, but rather are used to describe the performance of random search algorithms based on an underlying sampling distribution. It is shown that, under certain conditions, the expected number of iterations of pure adaptive search is *linear* in dimension, while pure random search is exponential in dimension.

Chapters 3, 4 and 5 discuss relaxations of PAS: hesitant adaptive search, annealing adaptive search, and backtracking adaptive search. Hesitant adaptive search is a generalization of PAS that includes an expression of the extra computation associated with not being able to always generate improving points. Adaptive search is a relaxation of PAS by changing the sampling distribution in a way that maintains the linearity result for improving points, and is discussed in Chapter 4. Backtracking adaptive search extends the analysis to include a sequence of points that include non-improving points, backtracking in terms of the objective function. This concludes the analyses of theoretical algorithms in this book.

Chapter 6 presents an initial attempt to realize PAS by generating points that are approximately uniform. The Improving Hit-and-Run algorithm is defined in Section 2, and is shown to have a polynomial computation on average for the class of positive definite quadratic programs. Section 3 presents the Hide-and-Seek algorithm, which is motivated by Adaptive Search in the same manner that Improving Hit-and-Run was motivated by Pure Adaptive Search. Both algorithms are based on the Hit-and-Run generator, and variations are summarized later in the chapter, as well as initial results on a discrete form of Hit-and-Run. Finally, several engineering design applications, particularly the optimal design of composite structures, are discussed in Chapter 7.

6. Summary

Engineers using gradient-based local optimization methods have commented that their problems tend to have many local optima [62, 185]. The most common global optimization method found in practice is multistart, a combination of random starting points and local searches [62]. However newer methods are quite powerful, and with the advances in computer technology, are becoming practical. Simulated annealing and genetic algorithms are useful for new ill-structured applications. The

primary structure these random search methods have in common is a probabilistic way of sampling the next point. Characterizing performance based on the sampling distribution is a step towards better understanding the algorithms. This can assist practitioners in selecting an appropriate algorithm for their problem, as well as developing new applications and new algorithms. Hopefully, this book and others like it can help bridge the gap between the advances in global optimization and the practical needs of the engineers.