PART I

INTRODUCTION TO EVOLUTIONARY OPTIMIZATION

CHAPTER 1

Introduction

But ask the animals, and they will teach you, or the birds of the air, and they will tell you; or speak to the earth, and it will teach you, or let the fish of the sea inform you.
—Job 12:7–9

This book discusses approaches to the solution of optimization problems. In particular, we[1] discuss evolutionary algorithms (EAs) for optimization. Although the book includes some mathematical theory, it should not be considered a mathematics text. It is more of an engineering or applied computer science text. The optimization approaches in this book are all given with the goal of eventual implementation in software. The aim of this book is to present evolutionary optimization algorithms in the most clear yet rigorous way possible, while also providing enough advanced material and references so that the reader is prepared to contribute new material to the state of the art.

[1]This book uses the common practice of referring to a generic third person with the word *we*. Sometimes, the book uses *we* to refer to the reader and the author. Other times, the book uses *we* to indicate that it is speaking on behalf of the general population of teachers and researchers in the areas of evolutionary algorithms and optimization. The distinction should be clear from the context. Do not read too much into the use of the word *we*; it is a matter of writing style rather than a claim to authority.

Overview of the Chapter

This chapter begins in Section 1.1 with an overview of the mathematical notation that we use in this book. The list of acronyms starting on page xxiii might also be useful to the reader. Section 1.2 gives some reasons why I decided to write this book about EAs, what I hope to accomplish with it, and why I think that it is distinctive in view of all of the other excellent EA books that are available. Section 1.3 discusses the prerequisites the are expected from a reader of this book. Section 1.4 discusses the philosophy of the homework assignments in this book, and the availability of the solution manual. Section 1.5 summarizes the mathematical notation that we use in this book. The reader is encouraged to regularly remember that section when encountering unfamiliar notation, and also to begin using it himself in homework assignments and in his own research. Section 1.6 gives a descriptive outline of the book. This leads into Section 1.7, which gives some important pointers to the instructor regarding some ways that he could teach a course from this book. That section also gives the instructor some advice about which chapters are more important than others.

1.1 TERMINOLOGY

Some authors use the term *evolutionary computing* to refer to EAs. This emphasizes the point that EAs are implemented in computers. However, evolutionary computing could refer to algorithms that are not used for optimization; for example, the first genetic algorithms (GAs) were not used for optimization per se, but were intended to study the process of natural selection (see Chapter 3). This book is geared towards evolutionary optimization algorithms, which are more specific than evolutionary computing.

Others use the term *population-based optimization* to refer to EAs. This emphasizes the point that EAs generally consist of a population of candidate solutions to some problem, and as time passes, the population evolves to a better solution to the problem. However, many EAs can consist of only a single candidate solution at each iteration (for example, hill climbing and evolution strategies). EAs are more general than population-based optimization because EAs include single-individual algorithms.

Some authors use the term *computer intelligence* or *computational intelligence* to refer to EAs. This is often done to distinguish EAs from expert systems, which have traditionally been referred to as *artificial intelligence*. Expert systems model deductive reasoning, while evolutionary algorithms model inductive reasoning. However, sometimes EAs are considered a type of artificial intelligence. Computer intelligence is a more general term than evolutionary algorithm, and includes technologies like neural networks, fuzzy systems, and artificial life. These technologies can be used for applications other than optimization. Therefore, depending on one's perspective, EAs might be more general or more specific than computer intelligence.

Soft computing is another term that is related to EAs. Soft computing is a contrast to hard computing. Hard computing refers to exact, precise, numerically rigorous calculations. Soft computing refers to less exact calculations, such as those that humans perform during their daily routines. Soft computing algorithms calculate generally good (but inexact) solutions to problems that are difficult, noisy, multimodal, and multi-objective. Therefore, EAs are a subset of soft computing.

Other authors use terms like *nature-inspired computing* or *bio-inspired computing* to refer to EAs. However, some EAs, like differential evolution and estimation of distribution algorithms, might not be motivated by nature. Other EAs, like evolution strategies and opposition-based learning, have a very weak connection with natural processes. EAs are more general than nature-inspired algorithms because EAs include non-biologically motivated algorithms.

Another oft-used term for EAs is *machine learning*. Machine learning is the study of computer algorithms that learn from experience. However, this field often includes many algorithms other than EAs. Machine learning is generally considered to be more broad than EAs, and includes fields such as reinforcement learning, neural networks, clustering, support vector machines, and others.

Some authors like to use the term *heuristic algorithms* to refer to EAs. *Heuristic* comes from the Greek word $\eta\upsilon\rho\iota\sigma\kappa\omega$, which is transliterated as *eurisko* in English. The word means *find* or *discover*. It is also the source of the English exclamation *eureka*, which we use to express triumph when we discover something or solve a problem. Heuristic algorithms are methods that use rules of thumb or common sense approaches to solve a problem. Heuristic algorithms usually are not expected to find the best answer to a problem, but are only expected to find solutions that are "close enough" to the best. The term *metaheuristic* is used to describe a family of heuristic algorithms. Most, if not all, of the EAs that we discuss in this book can be implemented in many different ways and with many different options and parameters. Therefore, they can all be called metaheuristics. For example, the family of all ant colony optimization algorithms can be called the ant colony metaheuristic.

Most authors separate EAs from swarm intelligence. A swarm intelligence algorithm is one that is based on swarms that occur in nature (for example, swarms of ants or birds). Ant colony optimization (Chapter 10) and particle swarm optimization (Chapter 11) are two prominent swarm algorithms, and many researchers insist that they should not be classified as EAs. However, some authors consider swarm intelligence as a subset of EAs. For example, one of the inventors of particle swarm optimization refers to it as an EA [Shi and Eberhart, 1999]. Since swarm intelligence algorithms execute in the same general way as EAs, that is, by evolving a population of candidate problem solutions which improve with each iteration, we consider swarm intelligence to be an EA.

Terminology is imprecise and context-dependent, but in this book we settle on the term *evolutionary algorithm* to refer to an algorithm that evolves a problem solution over many iterations. Typically, one iteration of an EA is called a *generation* in keeping with its biological foundation. However, this simple definition of an EA is not perfect because, for example, it implies that gradient descent is an EA, and no one is prepared to admit that. So the terminology in the EA field is not uniform and can be confusing. We use the tongue-in-cheek definition that an algorithm is an EA if it is generally considered to be an EA. This circularity is bothersome at first, but those of us who work in the field get used to it after a while. After all, natural selection is defined as the survival of the fittest, and fitness is defined as those who are most likely to survive.

1.2 WHY ANOTHER BOOK ON EVOLUTIONARY ALGORITHMS?

There are many fine books on EAs, which raises the question: Why yet another textbook on the topic of EAs? The reason that this book has been written is to offer a pedagogical approach, perspective, and material, that is not available in any other single book. In particular, the hope is that this book will offer the following:

- A straightforward, bottom-up approach that assists the reader in obtaining a clear but theoretically rigorous understanding of EAs is given in the book. Many books discuss a variety of EAs as cookbook algorithms without any theoretical support. Other books read more like research monographs than textbooks, and are not entirely accessible to the average engineering student. This book tries to strike a balance by presenting easy-to-implement algorithms, along with some rigorous theory and discussion of trade-offs.

- Simple examples that provide the reader with an intuitive understanding of EA math, equations, and theory, are given in the book. Many books present EA theory, and then give examples or problems that are not amenable to an intuitive understanding. However, it is possible to present simple examples and problems that require only paper and pencil to solve. These simple problems allow the student to more directly see how the theory works itself out in practice.

- MATLAB®-based source code for all of the examples in the book is available at the author's web site.[2] A number of other texts supply source code, but it is often incomplete or outdated, which is frustrating for the reader. The author's email address is also available on the web site, and I enthusiastically welcome feedback, comments, suggestions for improvements, and corrections. Of course, web addresses are subject to obsolescence, but this book contains algorithmic, high-level pseudocode listings that are more permanent than any specific software listings. Note that the examples and the MATLAB code are not intended as efficient or competitive optimization algorithms; they are instead intended only to allow the reader to gain a basic understanding of the underlying concepts. Any serious research or application should rely on the sample code only as a preliminary starting point.

- This book includes theory and recently-developed EAs that are not available in most other textbooks. These topics include Markov theory models of EAs, dynamic system models of EAs, artificial bee colony algorithms, biogeography-based optimization, opposition-based learning, artificial fish swarm algorithms, shuffled frog leaping, bacterial foraging optimization, and many others. These topics are recent additions to the state of the art, and their coverage in this book is not matched in any other books. However, this book is not intended to survey the state-of-the-art in any particular area of EA research. This book is instead intended to provide a high-level overview of many areas of EA research so that the reader can gain a broad understanding of EAs, and so that the reader can be well-positioned to pursue additional studies in the state-of-the-art.

[2]See `http://academic.csuohio.edu/simond/EvolutionaryOptimization` – if the address changes, it should be easy to find with an internet search.

1.3 PREREQUISITES

In general, a student will not gain anything from a course like this without writing his own EA software. Therefore, competent programming skills could be listed as a prerequisite. At the university where I teach this course to electrical and computer engineering students, there are no specific course prerequisites; the prerequisite for undergraduates is senior standing, and there are no prerequisites for graduate students. However, I assume that undergraduates at the senior level, and graduate students, are good programmers.

The notation used in the book assumes that the reader is familiar with the standard mathematical notations that are used in algebra, geometry, set theory, and calculus. Therefore, another prerequisite for understanding this book is a level of mathematical maturity that is typical of an advanced senior undergraduate student. The mathematical notation is described in Section 1.5. If the reader can understand the notation described in that section, then there is a good chance that he will also be able to follow the discussion in the rest of the book.

The mathematics in the theoretical sections of this book (Chapter 4, Section 7.6, much of Chapter 13, and a few other scattered sections) require an understanding of probability and linear systems theory. It will be difficult for a student to follow that material unless he has had a graduate course in those two subjects. A course geared towards undergraduates should probably skip that material.

1.4 HOMEWORK PROBLEMS

The problems at the end of each chapter have been written to give flexibility to the instructor and student. The problems include written exercises and computer exercises. The written exercises are intended to strengthen the student's grasp of the theory, deepen the student's intuitive understanding of the concepts, and develop the student's analytical skills. The computer exercises are intended to help the student develop research skills, and learn how to apply the theory to the types of problems that are typically encountered in industry. Both types of problems are important for gaining proficiency with EAs. The distinction between written exercises and computer exercises is not strict but is more of a fuzzy division. That is, some of the written exercises might require some computer work, and the computer exercises require some analysis. The instructor might have EA-related assignments in mind based on his own interests. Semester-length, project-based assignments are often instructive for topics such as this. For example, students could be assigned to solve some practical optimization problem using the EAs discussed in this book, applying one EA per chapter, and then comparing the performance of the EAs and their variations at the end of the semester.

A solution manual to all of the problems in the text (both written exercises and computer exercises) is available from the publisher for instructors. Course instructors are encouraged to contact the publisher for further information about how to obtain the solution manual. In order to protect the integrity of the homework assignments, the solution manual will be provided only to course instructors.

1.5 NOTATION

Unfortunately, the English language does not have a gender-neutral, singular, third-person pronoun. Therefore, we use the term *he* or *him* to refer to a generic third person, whether male or female. This convention can feel awkward to both writers and readers, but it seems to be the most satisfactory resolution to a difficult solution.

The list below describes some of the mathematical notation in this book.

- $x \leftarrow y$ is a computational notation that indicates that y is assigned to the variable x. For example, consider the following algorithm:

```
$a$ = coefficient of $x^2$
$b$ = coefficient of $x^1$
$c$ = coefficient of $x^0$
$x^* \leftarrow (-b + \sqrt{b^2 - 4ac})/(2a)$
```

  The first three lines are not assignment statements in the algorithm; they simply describe or define the values of a, b, and c. These three parameters could have been set by the user, or by some other algorithm or process. The last line, however, is an assignment statement that indicates the value on the right side of the arrow is written to x^*.

- $df(\cdot)/dx$ is the total derivative of $f(\cdot)$ with respect to x. For example, suppose that $y = 2x$ and $f(x, y) = 2x + 3y$. Then $f(x, y) = 8x$ and $df(\cdot)/dx = 8$.

- $f_x(\cdot)$, also denoted as $\partial f(\cdot)/\partial x$, is the partial derivative of $f(\cdot)$ with respect to x. For example, suppose again that $y = 2x$ and $f(x, y) = 2x + 3y$. Then $f_x(x, y) = 2$.

- $\{x : x \in S\}$ is the set of all x such that x belongs to the set S. A similar notation is used to denote those values of x that satisfy any other particular condition. For example, $\{x : x^2 = 4\}$ is the same as $\{x : x \in \{-2, +2\}\}$, which is the same as $\{-2, +2\}$.

- $[a, b]$ is the closed interval between a and b, which means $\{x : a \leq x \leq b\}$. This might be a set of integers or a set of real numbers, depending on the context.

- (a, b) is the open interval between a and b, which is $\{x : a < x < b\}$. This might be a set of integers or a set of real numbers, depending on the context.

- If is it understood from the context that $i \in S$, the $\{x_i\}$ is shorthand for $\{x_i : i \in S\}$. For example, if $i \in [1, N]$, then $\{x_i\} = \{x_1, x_2, \cdots, x_N\}$.

- $S_1 \cup S_2$ is the set of all x such that x belongs to either set S_1 or set S_2. For example, if $S_1 = \{1, 2, 3\}$ and $S_2 = \{7, 8\}$, then $S_1 \cup S_2 = \{1, 2, 3, 7, 8\}$.

- $|S|$ is the number of elements in the set S. For example, if $S = \{i : i \in [4, 8]\}$, then $|S| = 5$. If $S = \{3, 19, \pi, \sqrt{2}\}$, then $|S| = 4$. If $S = \{\alpha : 1 < \alpha < 3\}$, then $|S| = \infty$.

- $\emptyset$ is the empty set. $|\emptyset| = 0$.

- $x \bmod y$ is the remainder after x is divided by y. For example, $8 \bmod 3 = 2$.
- $\lceil x \rceil$ is the ceiling of x; that is, the smallest integer that is greater than or equal to x. For example, $\lceil 3.9 \rceil = 4$, and $\lceil 5 \rceil = 5$.
- $\lfloor x \rfloor$ is the floor of x; that is, the largest integer that is less than or equal to x For example, $\lfloor 3.9 \rfloor = 3$, and $\lfloor 5 \rfloor = 5$.
- $\min_x f(x)$ indicates the problem of finding the value of x that gives the smallest value of $f(x)$. Also, it can indicate the smallest value of $f(x)$. For example, suppose that $f(x) = (x-1)^2$. Then we can solve the problem $\min_x f(x)$ using calculus or by graphing the function $f(x)$ and visually noting the smallest value of $f(x)$. We find for this example that $\min_x f(x) = 0$. A similar definition holds for $\max_x f(x)$.
- $\arg\min_x f(x)$ is the value of x that results in the smallest value of $f(x)$. For example, suppose again that $f(x) = (x-1)^2$. The smallest value of $f(x)$ is 0, which occurs when $x = 1$, so for this example $\arg\min_x f(x) = 1$. A similar definition holds for $\arg\max_x f(x)$.
- R^s is the set of all real s-element vectors. It may indicate either column vectors or row vectors, depending on the context.
- $R^{s \times p}$ is the set of all real $s \times p$ matrices.
- $\{y_k\}_{k=L}^{U}$ is the set of all y_k, where the integer k ranges from L to U. For example, $\{y_k\}_{k=2}^{5} = \{y_2, y_3, y_4, y_5\}$.
- $\{y_k\}$ is the set of all y_k, where the integer k ranges from a context-dependent lower limit to a context-dependent upper limit. For example, suppose the context indicates that there are three values: y_1, y_2, and y_3. Then $\{y_k\} = \{y_1, y_2, y_3\}$.
- $\exists$ means "there exists," and $\nexists$ means "there does not exist." For example, if $Y = \{6, 1, 9\}$, then $\exists\, y < 2 : y \in Y$. However, $\nexists\, y > 10 : y \in Y$.
- $A \Longrightarrow B$ means that A implies B. For example, $(x > 10) \Longrightarrow (x > 5)$.
- I is the identity matrix. Its dimensions depend on the context.

See the list of acronyms on page xxiii for more notation.

1.6 OUTLINE OF THE BOOK

This book is divided into six parts.

1. Part I consists of this introduction, and one more chapter that covers introductory material related to optimization. It introduces different types of optimization problems, the simple-but-effective hill climbing algorithm, and concludes with a discussion about what makes an algorithm intelligent.

2. Part II discusses the four EAs that are commonly considered to be the classics:

    - Genetic algorithms;
    - Evolutionary programming;
    - Evolution strategies;
    - Genetic programming.

    Part II also includes a chapter that discusses approaches for the mathematical analysis of GAs. Part II concludes with a chapter that discusses some of the many algorithmic variations that can be used in these classic algorithms. These same variations can also be used in the more recent EAs, which are covered in the next part.

3. Part III discusses some of the more recent EAs. Some of these are not really that recent, dating back to the 1980s, but others date back only to the first decade of the 21st century.

4. Part IV discusses special types of optimization problems, and shows how the EAs of the earlier chapters can be modified to solve them. These special types of problems include:

    - Combinatorial problems, whose domain consists of integers;
    - Constrained problems, whose domain is restricted to a known set;
    - Multi-objective problems, in which it is desired to minimize more than one objective simultaneously; and
    - Problems with noisy or expensive fitness functions for which it is difficult to precisely obtain the performance of a candidate solution, or for which it is computationally expensive to evaluate the performance of a candidate solution.

5. Part V includes several appendices that discuss topics that are important or interesting.

    - Appendix A offers some miscellaneous, practical advice for the EA student and researcher.
    - Appendix B discusses the no-free-lunch theorem, which tells us that, on average, all optimization algorithms perform the same. It also discusses how statistics should be used to evaluate the differences between EAs.
    - Appendix C gives some standard benchmark functions that can be used to compare the performance of different EAs.

1.7 A COURSE BASED ON THIS BOOK

Any course based on this book should start with Chapters 1 and 2, which give an overview of optimization problems. From that point on, the remaining chapters can be studied in almost any order, depending on the preference and interests of the instructor. The obvious exceptions are that the study of genetic algorithms (Chapter 3) should precede the study of their mathematical models (Chapter 4).

Also, at least one chapter in Parts II or III (that is, at least one specific EA) needs to be covered in detail before any of the chapters in Part IV.

Most courses will, at a minimum, cover Chapters 3 and 5–7 to give the student a background in the classic EAs. If the students have sufficient mathematical sophistication, and if there is time, then the course should also include Chapter 4 somewhere along the line. Chapter 4 is important for graduate students because it helps them see that EAs are not only a qualitative subject, but there can and should be some theoretical basis for them also. Too much EA research today is based on minor algorithmic adjustments without any mathematical support. Many EA practitioners only care about getting results, which is fine, but academic researchers need to be involved in theory as well as practice.

The chapters in Parts III and IV can be covered on the basis of the instructor's or the students' specific interests.

The appendices are not included in the main part of the book because they are not about EAs per se, but the importance of the appendices should not be underestimated. In particular, the material in Appendices B and C are of critical importance and should be included in every EA course. I recommend that these two appendices be discussed in some detail immediately after the first chapter in Parts II or III.

Putting the above advice together, here is a proposed outline for a one-semester graduate course.

- Chapters 1 and 2.
- Chapter 3.
- Appendices B and C.
- Chapters 4–8. I recommend skipping Chapter 4 for most undergraduate students and for short courses.
- A few chapters in Part III, based on the instructor's preference. At the risk of starting an "EA war" with my readers, I will go out on a limb and claim that ACO, PSO, and DE are among the most important "other" EAs, and so the instructor should cover Chapters 10–12 at a minimum.
- A few chapters in Part IV, based on the instructor's preference and the available time.