Jason Brownlee

Clever Algorithms

Modern Artificial Intelligence Recipes

Clever Algorithms: Modern Artificial Intelligence Recipes

License Summary

Preface

About the book

The need for this project was born of frustration while working towards my Ph.D. I was investigating optimization algorithms and was implementing a large number of them for a software platform called the Optimization Algorithm Toolkit (OAT)[1]. Each algorithm required considerable effort to locate the relevant source material (from books, papers, articles, and existing implementations), decipher and interpret the technique, then to finally attempt to piece together a functional implementation.

Taking a broader perspective, I realized that the communication of algorithmic techniques in the field of Artificial Intelligence was clearly a difficult and outstanding open problem. Generally, algorithm descriptions are:

- *Incomplete*: many techniques are ambiguously described, partially described, or not described at all.
- *Inconsistent*: a given technique may be described using a variety of formal and semi-formal methods that vary across different techniques, limiting the transferability of skills an audience requires (such as mathematics, pseudo code, program code, and narratives). An inconsistent representation for techniques mean that the skills used to understand and internalize one technique may not be transferable to realizing different techniques or even extensions of the same technique.
- *Distributed*: the description of data structures, operations, and parameterization of a given technique may span a collection of papers, articles, books, and source code published over a number of years, the access to which may be restricted and difficult to obtain.

For the practitioner, a badly described algorithm may be simply frustrating, where the gaps in available information are filled with intuition and 'best guess'. At the other end of the spectrum, a badly described algorithm may an example of bad science and the failure of the scientific method, where the inability to understand and implement a technique may prevent the replication of results, the application, or the investigation and extension of a technique.

[1] OAT located at http://optalgtoolkit.sourceforge.net

The software I produced provided a first step solution to this problem: a set of working algorithms implemented in a (somewhat) consistent way and downloaded from a single location (features likely provided of any library of artificial intelligence techniques). The next logical step needed to address this problem is to develop a methodology that anybody can follow. The strategy to address the open problem of poor technique communication is to present complete algorithm descriptions (rather than implementations) in a consistent manner, and in a centralized location. This book is the outcome of developing such a strategy that not only provides a methodology for standardized algorithm descriptions, but provides a large corpus of complete and consistent algorithm descriptions in a single centralized location.

The algorithms described in this work are practical, interesting, and fun, and the goal of this project was to promote these features by making algorithms from the field more accessible, usable, and understandable. This project was developed over a number years though a lot of writing, discussion, and revision. The content was developed and released publicly under a permissive license on the website http://www.CleverAlgorithms.com, where forerunning technical reports and the content of this book are freely available. I hope that this project has succeeded in some small way and that you too can enjoy applying, learning, and playing with Clever Algorithms.

About the author

Jason Brownlee has a Bachelors degree in Applied Science, a Masters in Information Technology and a Ph.D. in Computer Science from Swinburne University of Technology in Melbourne, Australia. The subject of Jason's Masters research was Niching Genetic Algorithms. Jason's Ph.D. work was in the area of Artificial Immune Systems and involved research into extending the state of Clonal Selection inspired machine learning algorithms and devising new techniques inspired by the structure and function of the acquired immune system. Jason has earned a living as a Consultant on numerous enterprise-level information technology projects in retail, energy, and information services sectors. Jason has also worked as a Software Engineer investigating the use of intelligent agent technology in geospatial and information services domains in the defense sector. Jason has a long standing passion for both practical software engineering and basic research into machine learning and has developed and released many reports, software plug-ins, and software tools. Jason also enjoys writing and maintains a blog located at http://www.neverreadpassively.com and can be followed on twitter at http://twitter.com/jbrownlee.

Acknowledgments

Jason Brownlee would like to sincerely thank Daniel Angus for early discussions that lead to the inception of this book project. Jason would like to thank Ying Liu for her unrelenting support and patience throughout the development of the project.

Contents

Part I

Background

Chapter 1

Introduction

Welcome to Clever Algorithms! This is a handbook of recipes for computational problem solving techniques from the fields of Computational Intelligence, Biologically Inspired Computation, and Metaheuristics. Clever Algorithms are interesting, practical, and fun to learn about and implement. Research scientists may be interested in browsing algorithm inspirations in search of an interesting system or process analogues to investigate. Developers and software engineers may compare various problem solving algorithms and technique-specific guidelines. Practitioners, students, and interested amateurs may implement state-of-the-art algorithms to address business or scientific needs, or simply play with the fascinating systems they represent.

This introduction chapter provides relevant background information on Artificial Intelligence and Algorithms. The core of the book provides a large corpus of algorithm described in a complete and consistent manner. The final chapter covers some advanced topics to consider once a number of algorithms have been mastered. This book has been designed as a reference text rather then being read cover-to-cover, where specific techniques are looked up, or where the algorithms across whole fields of study are browsed. This book is an algorithm handbook and a technique guidebook, and I hope you find something useful.

1.1 What is AI

1.1.1 Artificial Intelligence

The field of classical *Artificial Intelligence* (AI) coalesced in the 1950s drawing on an understanding of the brain from neuroscience, the new mathematics of information theory, control theory referred to as cybernetics, and the dawn of the digital computer. AI is a cross-disciplinary field of research generally concerned with developing and investigating systems that operate or act intelligently. It is generally considered a discipline in the field of computer science given the strong focus on computation.

Russell and Norvig provide a perspective that defines Artificial Intelligence in four categories: (1) systems that think like humans, (2) systems that act like humans, (3)

systems that think rationally, (4) systems that act rationally [42]. In their definition, acting like a human suggests that a system can do some specific things humans can do, this includes fields such as the Turing test, natural language processing, automated reasoning, knowledge representation, machine learning, computer vision, and robotics. Thinking like a human suggests systems that model the cognitive information processing properties of humans, for example a general problem solver and systems that build internal models of their world. Thinking rationally suggests laws of rationalism and structured thought, such as syllogisms and formal logic. Finally, acting rationally suggests systems that do rational things such as expected utility maximization and rational agents.

Luger and Stubblefield suggest that AI is a sub-field of computer science concerned with the automation of intelligence, and like other sub-fields of computer science has both theoretical concerns (*how and why do the systems work?*) and application concerns (*where and when can the systems be used?*) [33]. They suggest a strong empirical focus to research, because although there may be a strong desire for mathematical analysis, the systems themselves defy analysis given their complexity. The machines and software investigated in AI are not black boxes, rather analysis proceeds by observing the systems interactions with their environment, followed by an internal assessment of the system to relate its structure back to their behavior.

Artificial Intelligence is therefore concerned with investigating mechanisms that underlie intelligence and intelligence behavior. The traditional approach toward designing and investigating AI (the so-called 'good old fashioned' AI) has been to employ a symbolic basis for these mechanisms. A newer approach historically referred to as scruffy artificial intelligence or soft computing does not necessarily use a symbolic basis, instead patterning these mechanisms after biological or natural processes. This represents a modern paradigm shift in interest from symbolic knowledge representations, to inference strategies for adaptation and learning, and has been referred to as neat versus scruffy approaches to AI. The neat philosophy is concerned with formal symbolic models of intelligence that can explain *why* they work, whereas the scruffy philosophy is concerned with intelligent strategies that explain *how* they work [44].

Neat AI

The traditional stream of AI concerns a top down perspective of problem solving, generally involving symbolic representations and logic processes that most importantly can explain why they work. The successes of this prescriptive stream include a multitude of specialist approaches such as rule-based expert systems, automatic theorem provers, and operations research techniques that underly modern planning and scheduling software. Although traditional approaches have resulted in significant success they have their limits, most notably scalability. Increases in problem size result in an unmanageable increase in the complexity of such problems meaning that although traditional techniques can guarantee an optimal, precise, or true solution, the computational execution time or computing memory required can be intractable.

Scruffy AI

There have been a number of thrusts in the field of AI toward less crisp techniques that are able to locate approximate, imprecise, or partially-true solutions to problems with a reasonable cost of resources. Such approaches are typically *descriptive* rather than *prescriptive*, describing a process for achieving a solution (how), but not explaining why they work (like the neater approaches).

Scruffy AI approaches are defined as relatively simple procedures that result in complex emergent and self-organizing behavior that can defy traditional reductionist analyses, the effects of which can be exploited for quickly locating approximate solutions to intractable problems. A common characteristic of such techniques is the incorporation of randomness in their processes resulting in robust probabilistic and stochastic decision making contrasted to the sometimes more fragile determinism of the crisp approaches. Another important common attribute is the adoption of an inductive rather than deductive approach to problem solving, generalizing solutions or decisions from sets of specific observations made by the system.

1.1.2 Natural Computation

An important perspective on scruffy Artificial Intelligence is the motivation and inspiration for the core information processing strategy of a given technique. Computers can only do what they are instructed, therefore a consideration is to distill information processing from other fields of study, such as the physical world and biology. The study of biologically motivated computation is called Biologically Inspired Computing [16], and is one of three related fields of Natural Computing [22, 23, 38]. Natural Computing is an interdisciplinary field concerned with the relationship of computation and biology, which in addition to Biologically Inspired Computing is also comprised of Computationally Motivated Biology and Computing with Biology [39, 35].

Biologically Inspired Computation

Biologically Inspired Computation is computation inspired by biological metaphor, also referred to as *Biomimicry*, and *Biomemetics* in other engineering disciplines [14, 6]. The intent of this field is to devise mathematical and engineering tools to generate solutions to computation problems. The field involves using procedures for finding solutions abstracted from the natural world for addressing computationally phrased problems.

Computationally Motivated Biology

Computationally Motivated Biology involves investigating biology using computers. The intent of this area is to use information sciences and simulation to model biological systems in digital computers with the aim to replicate and better understand behaviors in biological systems. The field facilitates the ability to better understand life-as-it-is and investigate life-as-it-could-be. Typically, work in this sub-field is not concerned with the construction of mathematical and engineering tools, rather it is focused on simulating

natural phenomena. Common examples include Artificial Life, Fractal Geometry (L-systems, Iterative Function Systems, Particle Systems, Brownian motion), and Cellular Automata. A related field is that of Computational Biology generally concerned with modeling biological systems and the application of statistical methods such as in the sub-field of Bioinformatics.

Computation with Biology

Computation with Biology is the investigation of substrates other than silicon in which to implement computation [1]. Common examples include molecular or DNA Computing and Quantum Computing.

1.1.3 Computational Intelligence

Computational Intelligence is a modern name for the sub-field of AI concerned with sub-symbolic (also called messy, scruffy, and soft) techniques. Computational Intelligence describes techniques that focus on *strategy* and *outcome*. The field broadly covers sub-disciplines that focus on adaptive and intelligence systems, not limited to: Evolutionary Computation, Swarm Intelligence (Particle Swarm and Ant Colony Optimization), Fuzzy Systems, Artificial Immune Systems, and Artificial Neural Networks [20, 40]. This section provides a brief summary of the each of the five primary areas of study.

Evolutionary Computation

A paradigm that is concerned with the investigation of systems inspired by the neo-Darwinian theory of evolution by means of natural selection. Popular evolutionary algorithms include the Genetic Algorithm, Evolution Strategy, Genetic and Evolutionary Programming, and Differential Evolution [4, 5]. The evolutionary process is considered an adaptive strategy and is typically applied to search and optimization domains [26, 28].

Swarm Intelligence

A paradigm that considers collective intelligence as a behavior that emerges through the interaction and cooperation of large numbers of lesser intelligent agents. The paradigm consists of two dominant sub-fields (1) Ant Colony Optimization that investigates probabilistic algorithms inspired by the stigmergy and foraging behavior of ants [10, 18], and (2) Particle Swarm Optimization that investigates probabilistic algorithms inspired by the flocking and foraging behavior of birds and fish [43]. Like evolutionary computation, swarm intelligence-based techniques are considered adaptive strategies and are typically applied to search and optimization domains.

Artificial Neural Networks

Neural Networks are a paradigm that is concerned with the investigation of architectures and learning strategies inspired by the modeling of neurons in the brain [8]. Learning

strategies are typically divided into supervised and unsupervised which manage environmental feedback in different ways. Neural network learning processes are considered adaptive learning and are typically applied to function approximation and pattern recognition domains.

Fuzzy Intelligence

Fuzzy Intelligence is a paradigm that is concerned with the investigation of fuzzy logic, which is a form of logic that is not constrained to true and false like propositional logic, but rather functions which define approximate truth or degrees of truth [52]. Fuzzy logic and fuzzy systems are a logic system used as a reasoning strategy and are typically applied to expert system and control system domains.

Artificial Immune Systems

A collection of approaches inspired by the structure and function of the acquired immune system of vertebrates. Popular approaches include clonal selection, negative selection, dendritic cell algorithm, and immune network algorithms. The immune-inspired adaptive processes vary in strategy and show similarities to the fields of Evolutionary Computation and Artificial Neural Networks, and are typically used for optimization and pattern recognition domains [17].

1.1.4 Metaheuristics

Another popular name for the strategy-outcome perspective of scruffy AI is *Metaheuristics*. A heuristic is an algorithm that locates 'good enough' solutions to a problem without concern for whether the solution can be proven to be correct or optimal [36]. Heuristic methods trade-off concerns such as precision, quality, and accuracy in favor of computational effort (space and time efficiency). The Greedy search procedure that only takes cost-improving steps is an example of heuristic method.

Like heuristics, Metaheuristic may be considered a general algorithmic framework that can be applied to different optimization problems with relative few modifications to make them adapted to a specific problem [25, 46]. The difference is that Metaheuristics are intended to extend the capabilities of heuristics by combining one or more heuristic methods (referred to as procedures) using a higher-level strategy (hence 'meta'). A procedure in a metaheuristic is considered black-box in that little (if any) prior knowledge is known about it by the meta-heuristic, and as such it may be replaced with a different procedure. Procedures may be as simple as the manipulation of a representation, or as complex as another complete metaheuristic. Some examples of metaheuristics include iterated local search, tabu search, the genetic algorithm, ant colony optimization, and simulated annealing.

Blum and Roli outline nine properties of metaheuristics [9], as follows:

- Metaheuristics are strategies that "guide" the search process.

- The goal is to efficiently explore the search space in order to find (near-)optimal solutions.
- Techniques which constitute metaheuristic algorithms range from simple local search procedures to complex learning processes.
- Metaheuristic algorithms are approximate and usually non-deterministic.
- They may incorporate mechanisms to avoid getting trapped in confined areas of the search space.
- The basic concepts of metaheuristics permit an abstract level description.
- Metaheuristics are not problem-specific.
- Metaheuristics may make use of domain-specific knowledge in the form of heuristics that are controlled by the upper level strategy.
- Todays more advanced metaheuristics use search experience (embodied in some form of memory) to guide the search.

Hyperheuristics are yet another extension that focuses on heuristics that modify their parameters (online or offline) to improve the efficacy of solution, or the efficiency of the computation. Hyperheuristics provide high-level strategies that may employ machine learning and adapt their search behavior by modifying the application of the sub-procedures or even which procedures are used (operating on the space of heuristics which in turn operate within the problem domain) [12, 13].

1.1.5 Clever Algorithms

This book is concerned with Clever Algorithms which are algorithms drawn from many sub-fields of Artificial Intelligence not limited to the scruff fields of Biologically Inspired Computation, Computational Intelligence and Metaheuristics. The term *Clever Algorithms* is intended to unify a collection of interesting and useful computational tools under a consistent and accessible banner. An alternative name (*Inspired Algorithms*) was considered, although ultimately rejected given that not all of the algorithms to be described in the project have an inspiration (specifically a biological or physical inspiration) for their computational strategy. The set of algorithms described in this book may generally be referred to as 'unconventional optimization algorithms' (for example, see [15]), as optimization is the main form of computation provided by the listed approaches. A technically more appropriate name for these approaches is Stochastic Global Optimization (for example, see [49] and [34]).

1.2 Problems

Algorithms from the fields of Computational Intelligence, Biologically Inspired Computing, and Metaheuristics are applied to difficult problems, to which more traditional

approaches may not be suited. Michalewicz and Fogel propose five reasons why problems may be difficult [36] (page 11):

- The number of possible solutions in the search space is so large as to forbid an exhaustive search for the best answer.
- The problem is so complicated that just to facilitate any answer at all, we have to use such simplified models of the problem that any result is essentially useless.
- The evaluation function that describes the quality of any proposed solution is noisy or varies with time, thereby requiring not just a single solution but an entire series of solutions.
- The possible solutions are so heavily constrained that constructing even one feasible answer is difficult, let alone searching for an optimal solution.
- The person solving the problem is inadequately prepared or imagines some psychological barrier that prevents them from discovering a solution.

This section introduces two problem formalisms that embody many of the most difficult problems faced by Artificial and Computational Intelligence. They are: Function Optimization and Function Approximation. Each class of problem is described in terms of its general properties, a formalism, and a set of specialized sub-problems. These problem classes provide a tangible framing of the algorithmic techniques described throughout the work.

1.2.1 Function Optimization

Real-world optimization problems and generalizations thereof can be drawn from most fields of science, engineering, and information technology (for a sample see [2, 48]). Importantly, optimization problems have had a long tradition in the fields of Artificial Intelligence in motivating basic research into new problem solving techniques, and for investigating and verifying systemic behavior against benchmark problem instances.

Problem Description

Mathematically, optimization is defined as the search for a combination of parameters commonly referred to as decision variables ($x = \{x_1, x_2, x_3, \dots x_n\}$) which minimize or maximize some ordinal quantity (c) (typically a scalar called a score or cost) assigned by an objective function or cost function (f), under a set of constraints ($g = \{g_1, g_2, g_3, \dots g_n\}$). For example, a general minimization case would be as follows: $f(x\prime) \leq f(x), \forall x_i \in x$. Constraints may provide boundaries on decision variables (for example in a real-value hypercube $\Re^n$), or may generally define regions of feasibility and in-feasibility in the decision variable space. In applied mathematics the field may be referred to as Mathematical Programming. More generally the field may be referred to as Global or Function Optimization given the focus on the objective function (for more general information on optimization refer to [29]).

Sub-Fields of Study

The study of optimization is comprised of many specialized sub-fields, based on an overlapping taxonomy that focuses on the principle concerns in the general formalism. For example, with regard to the decision variables, one may consider univariate and multivariate optimization problems. The type of decision variables promotes specialities for continuous, discrete, and permutations of variables. Dependencies between decision variables under a cost function define the fields of Linear Programming, Quadratic Programming, and Nonlinear Programming. A large class of optimization problems can be reduced to discrete sets and are considered in the field of Combinatorial Optimization, to which many theoretical properties are known, most importantly that many interesting and relevant problems cannot be solved by an approach with polynomial time complexity (so-called NP-complete, for example see [37]).

The topography of the response surface for the decision variables under the cost function may be convex, which is a class of functions to which many important theoretical findings have been made, not limited to the fact that location of the local optimal configuration also means the global optimal configuration of decisional variables has been located [11]. Many interesting and real-world optimization problems produce cost surfaces that are non-convex or so called multi-modal[1] (rather than uni-modal) suggesting that there are multiple peaks and valleys. Further, many real-world optimization problems with continuous decision variables cannot be differentiated given their complexity or limited information availability meaning that derivative-based gradient decent methods that are well understood are not applicable, requiring the use of so-called 'direct search' (sample or pattern-based) methods [32]. Real-world objective function evaluation may be noisy, discontinuous, dynamic, and the constraints of real-world problem solving may require an approximate solution in limited time or using resources, motivating the need for heuristic approaches.

1.2.2 Function Approximation

The phrasing of real-world problems in the Function Approximation formalism are among the most computationally difficult considered in the broader field of Artificial Intelligence for reasons including: incomplete information, high-dimensionality, noise in the sample observations, and non-linearities in the target function. This section considers the Function Approximation Formalism and related specialization's as a general motivating problem to contrast and compare with Function Optimization.

Problem Description

Function Approximation is the problem of finding a function (f) that approximates a target function (g), where typically the approximated function is selected based on a

[1] Taken from statistics referring to the centers of mass in distributions, although in optimization it refers to 'regions of interest' in the search space, in particular valleys in minimization, and peaks in maximization cost surfaces.

sample of observations (x, also referred to as the training set) taken from the unknown target function. In machine learning, the function approximation formalism is used to describe general problem types commonly referred to as pattern recognition, such as classification, clustering, and curve fitting (called a decision or discrimination function). Such general problem types are described in terms of approximating an unknown Probability Density Function (PDF), which underlies the relationships in the problem space, and is represented in the sample data. This 'function approximation' perspective of such problems is commonly referred to as statistical machine learning and/or density estimation [24, 8].

Sub-Fields of Study

The function approximation formalism can be used to phrase some of the hardest problems faced by Computer Science, and Artificial Intelligence in particular, such as natural language processing and computer vision. The general process focuses on (i) the collection and preparation of the observations from the target function, (ii) the selection and/or preparation of a model of the target function, and (ii) the application and ongoing refinement of the prepared model. Some important problem-based sub-fields include:

- *Feature Selection* where a feature is considered an aggregation of one-or-more attributes, where only those features that have meaning in the context of the target function are necessary to the modeling function [31, 27].

- *Classification* where observations are inherently organized into labelled groups (classes) and a supervised process models an underlying discrimination function to classify unobserved samples.

- *Clustering* where observations may be organized into groups based on underlying common features, although the groups are unlabeled requiring a process to model an underlying discrimination function without corrective feedback.

- *Curve or Surface Fitting* where a model is prepared that provides a 'best-fit' (called a regression) for a set of observations that may be used for interpolation over known observations and extrapolation for observations outside what has been modelled.

The field of Function Optimization is related to Function Approximation, as many-sub-problems of Function Approximation may be defined as optimization problems. Many of the technique paradigms used for function approximation are differentiated based on the representation and the optimization process used to minimize error or maximize effectiveness on a given approximation problem. The difficulty of Function Approximation problems centre around (i) the nature of the unknown relationships between attributes and features, (ii) the number (dimensionality) of of attributes and features, and (iii) general concerns of noise in such relationships and the dynamic availability of samples from the target function. Additional difficulties include the incorporation of

prior knowledge (such as imbalance in samples, incomplete information and the variable reliability of data), and problems of invariant features (such as transformation, translation, rotation, scaling and skewing of features).

1.3 Unconventional Optimization

Not all algorithms described in this book are for optimization, although, those that are may be referred to as 'unconventional' to differentiate them from the more traditional approaches. Examples of traditional approaches include (but are not not limited) to mathematical optimization algorithms (such as Newton's method and Gradient descent that uses derivatives to locate a local minimum) and direct search methods (such as the Simplex method and the Nelder-Mead method that use a search pattern to locate optima). Unconventional optimization algorithms are designed for the more difficult problem instances, the attributes of which were introduced in Section 1.2.1. This section introduces some common attributes of this class of algorithm.

1.3.1 Black Box Algorithms

Black Box optimization algorithms are those that exploit little, if any, information from a problem domain in order to devise a solution. They are generalized problem solving procedures that may be applied to a range of problems with very little modification [19]. Domain specific knowledge refers to making use of known relationships between solution representations and the objective cost function. Generally speaking, the less domain specific information incorporated into a technique, the more flexible the technique, although the less efficient it will be for a given problem. For example, 'random search' is the most general black box approach and is also the most flexible requiring only the generation of random solutions for a given problem. Random search also has a worst case behavior, that is worse than enumerating an entire search domain given the freedom it has to resample. In practice, the more prior knowledge available about a problem, the more information that should be exploited by a technique in order to efficiently locate a solution for the problem, heuristically or otherwise. Therefore, black box methods are those methods suitable for those problems where little information from the problem domain is available to be used by a problem solving approach.

1.3.2 No Free Lunch

The *No Free Lunch Theorem* of search and optimization by Wolpert and Macready proposes that all black box optimization algorithms are the same for searching for the extremum of a cost function when averaged over all possible functions [51, 50]. The theorem has caused a lot of pessimism and misunderstanding, particularly in relation to the evaluation and comparison of Metaheuristic and Computational Intelligence algorithms.

The implication of the theorem is that searching for the 'best' general-purpose black box optimization algorithm is irresponsible as no such procedure is theoretically possible. The theory applies to stochastic and deterministic optimization algorithms as well

as to algorithms that learn and adjust their search strategy over time. It is invariant to the performance measure used and the representation selected. The theorem is an important contribution to computer science, although its implications are theoretical. The original paper was produced at a time when grandiose generalizations were being made as to algorithm, representation, or configuration superiority. The practical impact of the theory is to encourage practitioners to bound claims of applicability for search and optimization algorithms. Wolpert and Macready encouraged effort be put into devising practical problem classes and into the matching of suitable algorithms to problem classes. Further, they compelled practitioners to exploit domain knowledge in optimization algorithm application, which is now an axiom in the field.

1.3.3 Stochastic Optimization

Stochastic optimization algorithms those that use randomness to elicit non-deterministic behaviors, contrasted to purely deterministic procedures. Most algorithms from the fields of Computational Intelligence, Biologically Inspired Computation, and Metaheuristics may be considered to belong the field of Stochastic Optimization. Algorithms that exploit randomness, are not random in behavior, rather they sample a problem space in a biased manner, focusing on areas of interest and neglecting less interesting areas [45]. A class of techniques that focus on the stochastic sampling of a domain are called Markov Chain Monte Carlo (MCMC) algorithms that provide good average performance, quickly, and generally offer a low chance of the worst case performance. Such approaches are suited to problems with many coupled degrees of freedom, for example large, high-dimensional spaces. MCMC approaches involve stochastically sampling from a target distribution function similar to Monte Carlo simulation methods using a process that resembles a biased Markov chain.

- *Monte Carlo* methods are used for selecting a statistical sample to approximate a given target probability density function and are traditionally used in statistical physics. Samples are drawn sequentially and the process may include criteria for rejecting samples and biasing the sampling locations within high-dimensional spaces.
- *Markov Chain* processes provide a probabilistic model for state transitions or moves within a discrete domain called a walk or a chain of steps. A Markov system is only dependent on the current position in the domain in order to probabilistically determine the next step in the walk.

MCMC techniques combine these two approaches to solve integration and optimization problems in large dimensional spaces by generating samples while exploring the space using a Markov chain process, rather than sequentially or independently [3]. The step generation is configured to bias sampling in more important regions of the domain. Three examples of MCMC techniques include the Metropolis-Hastings algorithm, Simulated annealing for global optimization, and the Gibbs sampler which are commonly employed in the fields of physics, chemistry, statistics, and economics.

1.3.4 Inductive Learning

Many unconventional optimization algorithms employ a process that includes the iterative improvement of candidate solutions against an objective cost function. This process of adaptation is generally a method by which the process obtains characteristics that improve the system's (candidate solution) relative performance in an environment (cost function). This adaptive behavior is commonly achieved through a 'selectionist process' of repetition of the steps: generation, test, and selection. The use of non-deterministic processes mean that the sampling of the domain (the generation step) is typically non-parametric, although guided by past experience.

The method of acquiring information is called inductive learning or learning from example, where the approach uses the implicit assumption that specific examples are representative of the broader information content of the environment, specifically with regard to anticipated need. Many unconventional optimization approaches maintain a single candidate solution, a population of samples, or a compression thereof that provides both an instantaneous representation of all of the information acquired by the process, and the basis for generating and making future decisions.

This method of simultaneously acquiring and improving information from the domain and the optimization of decision making (where to direct future effort) is called the k-armed bandit (two-armed and multi-armed bandit) problem from the field of statistical decision making known as game theory [41, 7]. This formalism considers the capability of a strategy to allocate available resources proportional to the future payoff the strategy is expected to receive. The classic example is the 2-armed bandit problem used by Goldberg to describe the behavior of the genetic algorithm [26]. The example involves an agent that learns which one of the two slot machines provides more return by pulling the handle of each (sampling the domain) and biasing future handle pulls proportional to the expected utility, based on the probabilistic experience with the past distribution of the payoff. The formalism may also be used to understand the properties of inductive learning demonstrated by the adaptive behavior of most unconventional optimization algorithms.

The stochastic iterative process of generate and test can be computationally wasteful, potentially re-searching areas of the problem space already searched, and requiring many trials or samples in order to achieve a 'good enough' solution. The limited use of prior knowledge from the domain (black box) coupled with the stochastic sampling process mean that the adapted solutions are created without top-down insight or instruction can sometimes be interesting, innovative, and even competitive with decades of human expertise [30].

1.4 Book Organization

The remainder of this book is organized into two parts: *Algorithms* that describes a large number of techniques in a complete and a consistent manner presented in a rough algorithm groups, and *Extensions* that reviews more advanced topics suitable for when

a number of algorithms have been mastered.

1.4.1 Algorithms

Algorithms are presented in six groups or kingdoms distilled from the broader fields of study each in their own chapter, as follows:

- *Stochastic Algorithms* that focus on the introduction of randomness into heuristic methods (Chapter 2).
- *Physical Algorithms* that focus on methods inspired by physical and social systems (Chapter 3).
- *Evolutionary Algorithms* that focus on methods in spired by evolution by means of natural selection (Chapter 4).
- *Probabilistic Algorithms* that focus on methods that build models and estimate distributions in search domains (Chapter 5).
- *Swarm Algorithms* that focus on methods that exploit the properties of collective intelligence (Chapter 6).
- *Immune Algorithms* that focus on methods inspired by the adaptive immune system of mammals (Chapter 7).

A given algorithm is more than just a procedure or code listing. Each approach is an island of research and the meta-information that define the context of a technique are just as important to understanding and application as abstract recipes and concrete implementations. A standardized algorithm description was adopted to provide a consistent presentation of algorithms with a mixture of softer narrative descriptions, programmatic descriptions both abstract and concrete, and most importantly useful sources for finding out more information about the technique.

The standardized algorithm description template covers the following subjects:

- *Name*: The algorithm name defines the canonical name used to refer to the technique, in addition to common aliases, abbreviations, and acronyms. The name is used as the heading of an algorithm descriptions.
- *Taxonomy*: The algorithm taxonomy defines where a techniques fits into the field, both the specific subfields of Computational Intelligence and Biologically Inspired Computation as well as the broader field of Artificial Intelligence. The taxonomy also provides a context for determining the relationships between algorithms.
- *Inspiration*: (optional) The inspiration describes the specific system or process that provoked the inception of the algorithm. The inspiring system may non-exclusively be natural, biological, physical, or social. The description of the inspiring system may include relevant domain specific theory, observation, nomenclature, and most

important must include those salient attributes of the system that are somehow abstractly or conceptually manifest in the technique.

- *Metaphor*: (optional) The metaphor is a description of the technique in the context of the inspiring system or a different suitable system. The features of the technique are made apparent through an analogous description of the features of the inspiring system. The explanation through analogy is not expected to be literal, rather the method is used as an allegorical communication tool. The inspiring system is not explicitly described, this is the role of the 'inspiration' topic, which represents a loose dependency for this topic.

- *Strategy*: The strategy is an abstract description of the computational model. The strategy describes the information processing actions a technique shall take in order to achieve an objective. The strategy provides a logical separation between a computational realization (procedure) and a analogous system (metaphor). A given problem solving strategy may be realized as one of a number specific algorithms or problem solving systems.

- *Procedure*: The algorithmic procedure summarizes the specifics of realizing a strategy as a systemized and parameterized computation. It outlines how the algorithm is organized in terms of the computation, data structures, and representations.

- *Heuristics*: The heuristics section describes the commonsense, best practice, and demonstrated rules for applying and configuring a parameterized algorithm. The heuristics relate to the technical details of the techniques procedure and data structures for general classes of application (neither specific implementations nor specific problem instances).

- *Tutorial*: The tutorial description provides a guide to realizing the technique using a programming language. The result of completing the tutorial is a minimal yet complete implementation of the technique applied to a problem. The tutorial description provides explanations as to the design decisions and rationale for the way the technique is implemented.

- *References*: The references section includes a listing of both primary sources of information about the technique as well as useful introductory sources for novices to gain a deeper understanding of the theory and application of the technique. The description consists of hand-selected reference material including books, peer reviewed conference papers, journal articles, and potentially websites.

Source code examples are included in the algorithm descriptions, and the Ruby Programming Language was selected for use throughout the book. Ruby was selected because it supports the procedural programming paradigm that was adopted to ensure that examples can be easily ported to object-oriented and other paradigms. Additionally, Ruby is interpreted meaning the code can be directly executed without an introduced

compilation step, and it is free to download and use from the website[2]. Finally, Ruby is concise, expressive, and supports meta-programming features that improve the readability of code examples. All of the source code for the algorithms presented in this book is available from the books website at http://www.CleverAlgorithms.com.

1.4.2 Extensions

There are some some advanced topics that cannot be meaningfully considered until one has a firm grasp of a number of algorithms, and these are discussed at the back of the book. The Advanced Topics chapter addresses topics such as: the use of alternative programming paradigms when implementing clever algorithms, methodologies used when devising entirely new approaches, strategies to consider when testing clever algorithms, visualizing the behavior and results of algorithms, and finally comparing algorithms based on the results they produce using statistical methods. Like the background information provided in this chapter, the extensions provide a gentle introduction and starting point into some advanced topics and plenty of references for seeking a deeper understanding.

1.5 How to Read this Book

This book is a reference text that provides a large compendium of algorithm descriptions. It is a trusted handbook of practical computational recipes to consulted when one is confronted with difficult function optimization and approximation problems. It is also an encompassing guidebook of modern heuristic methods that may be browsed for inspiration, exploration, and general interest.

The audience for this work may be interested with the fields of Computational Intelligence, Biologically Inspired Computation, and Metaheuristics and may count themselves as belonging to one of the following broader groups:

- *Scientists*: Research scientists concerned with theoretically or empirically investigating algorithms, addressing questions such as: *What is the motivating system and strategy for a given technique? What are some algorithms that may be used in a comparison within a given subfield or across subfields?*

- *Engineers*: Programmers and developers concerned with implementing, applying, or maintaining algorithms, addressing questions such as: *What is the algorithm procedure for a given technique? What are the best practice heuristics for employing a given technique?*

- *Students*: Undergraduate and graduate students interested in learning about techniques, addressing questions such as: *What are some interesting algorithms to study? How to implement a given approach?*

[2]Ruby can be downloaded for free from http://www.ruby-lang.org

- *Amateurs*: Practitioners interested in knowing more about algorithms, addressing questions such as: *What classes of techniques exist and what algorithms do they provide? How to conceptualize the computation of a technique?*

1.6 Further Reading

This book is not an introduction to Artificial Intelligence or related sub-fields, nor is it a field guide for a specific class of algorithms. This section provides some pointers to selected books and articles for those readers seeking a deeper understanding of the fields of study to which the Clever Algorithms described in this book belong.

1.6.1 Artificial Intelligence

Artificial Intelligence is large field of study and many excellent texts have been written to introduce the subject. Russell and Novig's "*Artificial Intelligence: A Modern Approach*" is an excellent introductory text providing a broad and deep review of what the field has to offer and is useful for students and practitioners alike [42]. Luger and Stubblefield's "*Artificial Intelligence: Structures and Strategies for Complex Problem Solving*" is also an excellent reference text, providing a more empirical approach to the field.

1.6.2 Computational Intelligence

Introductory books for the field of Computational Intelligence generally focus on a handful of specific sub-fields and their techniques. Engelbrecht's "*Computational Intelligence: An Introduction*" provides a modern and detailed introduction to the field covering classic subjects such as Evolutionary Computation and Artificial Neural Networks, as well as more recent techniques such as Swarm Intelligence and Artificial Immune Systems [20]. Pedrycz's slightly more dated "*Computational Intelligence: An Introduction*" also provides a solid coverage of the core of the field with some deeper insights into fuzzy logic and fuzzy systems [40].

1.6.3 Biologically Inspired Computation

Computational methods inspired by natural and biologically systems represent a large fraction of the algorithms described in this book. The collection of articles published in de Castro and Von Zuben's "*Recent Developments in Biologically Inspired Computing*" provide a good overview of the state of the field, and the introductory chapter on need for such methods does an excellent job to motivate the field of study [14]. Forbes's "*Imitation of Life: How Biology Is Inspiring Computing*" set's the scene for Natural Computing and the interrelated disciplines, of which Biologically Inspired Computing is but one useful example [22]. Finally, Benyus's "*Biomimicry: Innovation Inspired by Nature*" provides a good introduction into the broader related field of a new frontier in science and technology that involves building systems inspired by an understanding of biological systems [6].

1.6.4 Metaheuristics

The field of Metaheuristics was initially constrained to heuristics for applying classical optimization procedures, although has expanded to encompass a broader and diverse set of techniques. Michalewicz and Fogel's "*How to Solve It: Modern Heuristics*" provides a practical tour of heuristic methods with a consistent set of worked examples. Glover and Kochenberger's "*Handbook of Metaheuristics*" provides a solid introduction into a broad collection of techniques and their capabilities.

1.6.5 The Ruby Programming Language

The Ruby Programming Language is a multi-paradigm dynamic language that appeared in approximately 1995. It's meta-programming capabilities coupled with concise and readable syntax have made it a popular language of choice for web development, scripting, and application development. The classic reference text for the language is Thomas, Fowler, and Hunt's "*Programming Ruby: The Pragmatic Programmers' Guide*" referred to as the 'pickaxe book' because of the picture of the pickaxe on the cover [47]. An updated edition is available that covers version 1.9 (compared to 1.8 in the cited version) that will work just as well for use as a reference for the examples in this book. Flanagan and Matsumoto's "*The Ruby Programming Language*" also provides a seminal reference text with contributions from Yukihiro Matsumoto, the author of the language [21].

Part II

Algorithms

Chapter 2

Stochastic Algorithms

2.1 Overview

todo

2.2 Random Search

The heading and alternate headings for the algorithm description.

2.2.1 Taxonomy

A small tree diagram showing related fields and algorithms.

2.2.2 Inspiration

A textual description of the inspiring system.

2.2.3 Metaphor

A textual description of the algorithm by analogy.

2.2.4 Strategy

A textual description of the information processing strategy.

2.2.5 Procedure

A pseudo code description of the algorithms procedure.

2.2.6 Heuristics

A bullet-point listing of best practice usage.

2.2.7 Tutorial

A textural narrative for realizing the algorithm with complete source code.

2.2.8 References

An bullet-point annotated reference list of primary sources and useful resources.

2.3 Adaptive Random Search

The heading and alternate headings for the algorithm description.

2.3.1 Taxonomy

A small tree diagram showing related fields and algorithms.

2.3.2 Inspiration

A textual description of the inspiring system.

2.3.3 Metaphor

A textual description of the algorithm by analogy.

2.3.4 Strategy

A textual description of the information processing strategy.

2.3.5 Procedure

A pseudo code description of the algorithms procedure.

2.3.6 Heuristics

A bullet-point listing of best practice usage.

2.3.7 Tutorial

A textural narrative for realizing the algorithm with complete source code.

2.3.8 References

An bullet-point annotated reference list of primary sources and useful resources.

2.4 Hill Climbing Search

The heading and alternate headings for the algorithm description.

2.4.1 Taxonomy

A small tree diagram showing related fields and algorithms.

2.4.2 Inspiration

A textual description of the inspiring system.

2.4.3 Metaphor

A textual description of the algorithm by analogy.

2.4.4 Strategy

A textual description of the information processing strategy.

2.4.5 Procedure

A pseudo code description of the algorithms procedure.

2.4.6 Heuristics

A bullet-point listing of best practice usage.

2.4.7 Tutorial

A textural narrative for realizing the algorithm with complete source code.

2.4.8 References

An bullet-point annotated reference list of primary sources and useful resources.

2.5 Guided Local Search

The heading and alternate headings for the algorithm description.

2.5.1 Taxonomy

A small tree diagram showing related fields and algorithms.

2.5.2 Inspiration

A textual description of the inspiring system.

2.5.3 Metaphor

A textual description of the algorithm by analogy.

2.5.4 Strategy

A textual description of the information processing strategy.

2.5.5 Procedure

A pseudo code description of the algorithms procedure.

2.5.6 Heuristics

A bullet-point listing of best practice usage.

2.5.7 Tutorial

A textural narrative for realizing the algorithm with complete source code.

2.5.8 References

An bullet-point annotated reference list of primary sources and useful resources.

2.6 Variable Neighborhood Search

The heading and alternate headings for the algorithm description.

2.6.1 Taxonomy

A small tree diagram showing related fields and algorithms.

2.6.2 Inspiration

A textual description of the inspiring system.

2.6.3 Metaphor

A textual description of the algorithm by analogy.

2.6.4 Strategy

A textual description of the information processing strategy.

2.6.5 Procedure

A pseudo code description of the algorithms procedure.

2.6.6 Heuristics

A bullet-point listing of best practice usage.

2.6.7 Tutorial

A textural narrative for realizing the algorithm with complete source code.

2.6.8 References

An bullet-point annotated reference list of primary sources and useful resources.

2.7 Reactive Search Optimization

The heading and alternate headings for the algorithm description.

2.7.1 Taxonomy

A small tree diagram showing related fields and algorithms.

2.7.2 Inspiration

A textual description of the inspiring system.

2.7.3 Metaphor

A textual description of the algorithm by analogy.

2.7.4 Strategy

A textual description of the information processing strategy.

2.7.5 Procedure

A pseudo code description of the algorithms procedure.

2.7.6 Heuristics

A bullet-point listing of best practice usage.

2.7.7 Tutorial

A textural narrative for realizing the algorithm with complete source code.

2.7.8 References

An bullet-point annotated reference list of primary sources and useful resources.

2.8 Greedy Randomized Adaptive Search

The heading and alternate headings for the algorithm description.

2.8.1 Taxonomy

A small tree diagram showing related fields and algorithms.

2.8.2 Inspiration

A textual description of the inspiring system.

2.8.3 Metaphor

A textual description of the algorithm by analogy.

2.8.4 Strategy

A textual description of the information processing strategy.

2.8.5 Procedure

A pseudo code description of the algorithms procedure.

2.8.6 Heuristics

A bullet-point listing of best practice usage.

2.8.7 Tutorial

A textural narrative for realizing the algorithm with complete source code.

2.8.8 References

An bullet-point annotated reference list of primary sources and useful resources.

2.9 Scatter Search

The heading and alternate headings for the algorithm description.

2.9.1 Taxonomy

A small tree diagram showing related fields and algorithms.

2.9.2 Inspiration

A textual description of the inspiring system.

2.9.3 Metaphor

A textual description of the algorithm by analogy.

2.9.4 Strategy

A textual description of the information processing strategy.

2.9.5 Procedure

A pseudo code description of the algorithms procedure.

2.9.6 Heuristics

A bullet-point listing of best practice usage.

2.9.7 Tutorial

A textural narrative for realizing the algorithm with complete source code.

2.9.8 References

An bullet-point annotated reference list of primary sources and useful resources.

2.10 Tabu Search

The heading and alternate headings for the algorithm description.

2.10.1 Taxonomy

A small tree diagram showing related fields and algorithms.

2.10.2 Inspiration

A textual description of the inspiring system.

2.10.3 Metaphor

A textual description of the algorithm by analogy.

2.10.4 Strategy

A textual description of the information processing strategy.

2.10.5 Procedure

A pseudo code description of the algorithms procedure.

2.10.6 Heuristics

A bullet-point listing of best practice usage.

2.10.7 Tutorial

A textural narrative for realizing the algorithm with complete source code.

2.10.8 References

An bullet-point annotated reference list of primary sources and useful resources.

2.11 Reactive Tabu Search

The heading and alternate headings for the algorithm description.

2.11.1 Taxonomy

A small tree diagram showing related fields and algorithms.

2.11.2 Inspiration

A textual description of the inspiring system.

2.11.3 Metaphor

A textual description of the algorithm by analogy.

2.11.4 Strategy

A textual description of the information processing strategy.

2.11.5 Procedure

A pseudo code description of the algorithms procedure.

2.11.6 Heuristics

A bullet-point listing of best practice usage.

2.11.7 Tutorial

A textural narrative for realizing the algorithm with complete source code.

2.11.8 References

An bullet-point annotated reference list of primary sources and useful resources.

2.12 Summary

todo

Chapter 3

Physical Algorithms

3.1 Overview

todo

3.2 Simulated Annealing

The heading and alternate headings for the algorithm description.

3.2.1 Taxonomy

A small tree diagram showing related fields and algorithms.

3.2.2 Inspiration

A textual description of the inspiring system.

3.2.3 Metaphor

A textual description of the algorithm by analogy.

3.2.4 Strategy

A textual description of the information processing strategy.

3.2.5 Procedure

A pseudo code description of the algorithms procedure.

3.2.6 Heuristics

A bullet-point listing of best practice usage.

3.2.7 Tutorial

A textural narrative for realizing the algorithm with complete source code.

3.2.8 References

An bullet-point annotated reference list of primary sources and useful resources.

3.3 Adaptive Simulated Annealing

The heading and alternate headings for the algorithm description.

3.3.1 Taxonomy

A small tree diagram showing related fields and algorithms.

3.3.2 Inspiration

A textual description of the inspiring system.

3.3.3 Metaphor

A textual description of the algorithm by analogy.

3.3.4 Strategy

A textual description of the information processing strategy.

3.3.5 Procedure

A pseudo code description of the algorithms procedure.

3.3.6 Heuristics

A bullet-point listing of best practice usage.

3.3.7 Tutorial

A textural narrative for realizing the algorithm with complete source code.

3.3.8 References

An bullet-point annotated reference list of primary sources and useful resources.

3.4 Memetic Algorithm

The heading and alternate headings for the algorithm description.

3.4.1 Taxonomy

A small tree diagram showing related fields and algorithms.

3.4.2 Inspiration

A textual description of the inspiring system.

3.4.3 Metaphor

A textual description of the algorithm by analogy.

3.4.4 Strategy

A textual description of the information processing strategy.

3.4.5 Procedure

A pseudo code description of the algorithms procedure.

3.4.6 Heuristics

A bullet-point listing of best practice usage.

3.4.7 Tutorial

A textural narrative for realizing the algorithm with complete source code.

3.4.8 References

An bullet-point annotated reference list of primary sources and useful resources.

3.5 Extremal Optimization

The heading and alternate headings for the algorithm description.

3.5.1 Taxonomy

A small tree diagram showing related fields and algorithms.

3.5.2 Inspiration

A textual description of the inspiring system.

3.5.3 Metaphor

A textual description of the algorithm by analogy.

3.5.4 Strategy

A textual description of the information processing strategy.

3.5.5 Procedure

A pseudo code description of the algorithms procedure.

3.5.6 Heuristics

A bullet-point listing of best practice usage.

3.5.7 Tutorial

A textural narrative for realizing the algorithm with complete source code.

3.5.8 References

An bullet-point annotated reference list of primary sources and useful resources.

3.6 Cultural Algorithm

The heading and alternate headings for the algorithm description.

3.6.1 Taxonomy

A small tree diagram showing related fields and algorithms.

3.6.2 Inspiration

A textual description of the inspiring system.

3.6.3 Metaphor

A textual description of the algorithm by analogy.

3.6.4 Strategy

A textual description of the information processing strategy.

3.6.5 Procedure

A pseudo code description of the algorithms procedure.

3.6.6 Heuristics

A bullet-point listing of best practice usage.

3.6.7 Tutorial

A textural narrative for realizing the algorithm with complete source code.

3.6.8 References

An bullet-point annotated reference list of primary sources and useful resources.

3.7 Summary

todo

Chapter 4

Evolutionary Algorithms

4.1 Overview

todo

4.2 Genetic Algorithm

The heading and alternate headings for the algorithm description.

4.2.1 Taxonomy

A small tree diagram showing related fields and algorithms.

4.2.2 Inspiration

A textual description of the inspiring system.

4.2.3 Metaphor

A textual description of the algorithm by analogy.

4.2.4 Strategy

A textual description of the information processing strategy.

4.2.5 Procedure

A pseudo code description of the algorithms procedure.

4.2.6 Heuristics

A bullet-point listing of best practice usage.

4.2.7 Tutorial

A textural narrative for realizing the algorithm with complete source code.

4.2.8 References

An bullet-point annotated reference list of primary sources and useful resources.

4.3 Genetic Programming

The heading and alternate headings for the algorithm description.

4.3.1 Taxonomy

A small tree diagram showing related fields and algorithms.

4.3.2 Inspiration

A textual description of the inspiring system.

4.3.3 Metaphor

A textual description of the algorithm by analogy.

4.3.4 Strategy

A textual description of the information processing strategy.

4.3.5 Procedure

A pseudo code description of the algorithms procedure.

4.3.6 Heuristics

A bullet-point listing of best practice usage.

4.3.7 Tutorial

A textural narrative for realizing the algorithm with complete source code.

4.3.8 References

An bullet-point annotated reference list of primary sources and useful resources.

4.4 Evolutionary Programming

The heading and alternate headings for the algorithm description.

4.4.1 Taxonomy

A small tree diagram showing related fields and algorithms.

4.4.2 Inspiration

A textual description of the inspiring system.

4.4.3 Metaphor

A textual description of the algorithm by analogy.

4.4.4 Strategy

A textual description of the information processing strategy.

4.4.5 Procedure

A pseudo code description of the algorithms procedure.

4.4.6 Heuristics

A bullet-point listing of best practice usage.

4.4.7 Tutorial

A textural narrative for realizing the algorithm with complete source code.

4.4.8 References

An bullet-point annotated reference list of primary sources and useful resources.

4.5 Evolution Strategies

The heading and alternate headings for the algorithm description.

4.5.1 Taxonomy

A small tree diagram showing related fields and algorithms.

4.5.2 Inspiration

A textual description of the inspiring system.

4.5.3 Metaphor

A textual description of the algorithm by analogy.

4.5.4 Strategy

A textual description of the information processing strategy.

4.5.5 Procedure

A pseudo code description of the algorithms procedure.

4.5.6 Heuristics

A bullet-point listing of best practice usage.

4.5.7 Tutorial

A textural narrative for realizing the algorithm with complete source code.

4.5.8 References

An bullet-point annotated reference list of primary sources and useful resources.

4.6 Learning Classifier System

The heading and alternate headings for the algorithm description.

4.6.1 Taxonomy

A small tree diagram showing related fields and algorithms.

4.6.2 Inspiration

A textual description of the inspiring system.

4.6.3 Metaphor

A textual description of the algorithm by analogy.

4.6.4 Strategy

A textual description of the information processing strategy.

4.6.5 Procedure

A pseudo code description of the algorithms procedure.

4.6.6 Heuristics

A bullet-point listing of best practice usage.

4.6.7 Tutorial

A textural narrative for realizing the algorithm with complete source code.

4.6.8 References

An bullet-point annotated reference list of primary sources and useful resources.

4.7 Differential Evolution

The heading and alternate headings for the algorithm description.

4.7.1 Taxonomy

A small tree diagram showing related fields and algorithms.

4.7.2 Inspiration

A textual description of the inspiring system.

4.7.3 Metaphor

A textual description of the algorithm by analogy.

4.7.4 Strategy

A textual description of the information processing strategy.

4.7.5 Procedure

A pseudo code description of the algorithms procedure.

4.7.6 Heuristics

A bullet-point listing of best practice usage.

4.7.7 Tutorial

A textural narrative for realizing the algorithm with complete source code.

4.7.8 References

An bullet-point annotated reference list of primary sources and useful resources.

4.8 Grammatical Evolution

The heading and alternate headings for the algorithm description.

4.8.1 Taxonomy

A small tree diagram showing related fields and algorithms.

4.8.2 Inspiration

A textual description of the inspiring system.

4.8.3 Metaphor

A textual description of the algorithm by analogy.

4.8.4 Strategy

A textual description of the information processing strategy.

4.8.5 Procedure

A pseudo code description of the algorithms procedure.

4.8.6 Heuristics

A bullet-point listing of best practice usage.

4.8.7 Tutorial

A textural narrative for realizing the algorithm with complete source code.

4.8.8 References

An bullet-point annotated reference list of primary sources and useful resources.

4.9 Non-dominated Sorting Genetic Algorithm

The heading and alternate headings for the algorithm description.

4.9.1 Taxonomy

A small tree diagram showing related fields and algorithms.

4.9.2 Inspiration

A textual description of the inspiring system.

4.9.3 Metaphor

A textual description of the algorithm by analogy.

4.9.4 Strategy

A textual description of the information processing strategy.

4.9.5 Procedure

A pseudo code description of the algorithms procedure.

4.9.6 Heuristics

A bullet-point listing of best practice usage.

4.9.7 Tutorial

A textural narrative for realizing the algorithm with complete source code.

4.9.8 References

An bullet-point annotated reference list of primary sources and useful resources.

4.10 Strength Pareto Evolutionary Algorithm

The heading and alternate headings for the algorithm description.

4.10.1 Taxonomy

A small tree diagram showing related fields and algorithms.

4.10.2 Inspiration

A textual description of the inspiring system.

4.10.3 Metaphor

A textual description of the algorithm by analogy.

4.10.4 Strategy

A textual description of the information processing strategy.

4.10.5 Procedure

A pseudo code description of the algorithms procedure.

4.10.6 Heuristics

A bullet-point listing of best practice usage.

4.10.7 Tutorial

A textural narrative for realizing the algorithm with complete source code.

4.10.8 References

An bullet-point annotated reference list of primary sources and useful resources.

4.11 Island Population Genetic Algorithm

The heading and alternate headings for the algorithm description.

4.11.1 Taxonomy

A small tree diagram showing related fields and algorithms.

4.11.2 Inspiration

A textual description of the inspiring system.

4.11.3 Metaphor

A textual description of the algorithm by analogy.

4.11.4 Strategy

A textual description of the information processing strategy.

4.11.5 Procedure

A pseudo code description of the algorithms procedure.

4.11.6 Heuristics

A bullet-point listing of best practice usage.

4.11.7 Tutorial

A textural narrative for realizing the algorithm with complete source code.

4.11.8 References

An bullet-point annotated reference list of primary sources and useful resources.

4.12 Summary

todo

Chapter 5

Probabilistic Algorithms

5.1 Overview

todo

5.2 Cross-Entropy Method

The heading and alternate headings for the algorithm description.

5.2.1 Taxonomy

A small tree diagram showing related fields and algorithms.

5.2.2 Inspiration

A textual description of the inspiring system.

5.2.3 Metaphor

A textual description of the algorithm by analogy.

5.2.4 Strategy

A textual description of the information processing strategy.

5.2.5 Procedure

A pseudo code description of the algorithms procedure.

5.2.6 Heuristics

A bullet-point listing of best practice usage.

5.2.7 Tutorial

A textural narrative for realizing the algorithm with complete source code.

5.2.8 References

An bullet-point annotated reference list of primary sources and useful resources.

5.3 Population-Based Incremental Learning

The heading and alternate headings for the algorithm description.

5.3.1 Taxonomy

A small tree diagram showing related fields and algorithms.

5.3.2 Inspiration

A textual description of the inspiring system.

5.3.3 Metaphor

A textual description of the algorithm by analogy.

5.3.4 Strategy

A textual description of the information processing strategy.

5.3.5 Procedure

A pseudo code description of the algorithms procedure.

5.3.6 Heuristics

A bullet-point listing of best practice usage.

5.3.7 Tutorial

A textural narrative for realizing the algorithm with complete source code.

5.3.8 References

An bullet-point annotated reference list of primary sources and useful resources.

5.4 Probabilistic Incremental Program Evolution

The heading and alternate headings for the algorithm description.

5.4.1 Taxonomy

A small tree diagram showing related fields and algorithms.

5.4.2 Inspiration

A textual description of the inspiring system.

5.4.3 Metaphor

A textual description of the algorithm by analogy.

5.4.4 Strategy

A textual description of the information processing strategy.

5.4.5 Procedure

A pseudo code description of the algorithms procedure.

5.4.6 Heuristics

A bullet-point listing of best practice usage.

5.4.7 Tutorial

A textural narrative for realizing the algorithm with complete source code.

5.4.8 References

An bullet-point annotated reference list of primary sources and useful resources.

5.5 Compact Genetic Algorithm

The heading and alternate headings for the algorithm description.

5.5.1 Taxonomy

A small tree diagram showing related fields and algorithms.

5.5.2 Inspiration

A textual description of the inspiring system.

5.5.3 Metaphor

A textual description of the algorithm by analogy.

5.5.4 Strategy

A textual description of the information processing strategy.

5.5.5 Procedure

A pseudo code description of the algorithms procedure.

5.5.6 Heuristics

A bullet-point listing of best practice usage.

5.5.7 Tutorial

A textural narrative for realizing the algorithm with complete source code.

5.5.8 References

An bullet-point annotated reference list of primary sources and useful resources.

5.6 Extended Compact Genetic Algorithm

The heading and alternate headings for the algorithm description.

5.6.1 Taxonomy

A small tree diagram showing related fields and algorithms.

5.6.2 Inspiration

A textual description of the inspiring system.

5.6.3 Metaphor

A textual description of the algorithm by analogy.

5.6.4 Strategy

A textual description of the information processing strategy.

5.6.5 Procedure

A pseudo code description of the algorithms procedure.

5.6.6 Heuristics

A bullet-point listing of best practice usage.

5.6.7 Tutorial

A textural narrative for realizing the algorithm with complete source code.

5.6.8 References

An bullet-point annotated reference list of primary sources and useful resources.

5.7 Bayesian Optimization Algorithm

The heading and alternate headings for the algorithm description.

5.7.1 Taxonomy

A small tree diagram showing related fields and algorithms.

5.7.2 Inspiration

A textual description of the inspiring system.

5.7.3 Metaphor

A textual description of the algorithm by analogy.

5.7.4 Strategy

A textual description of the information processing strategy.

5.7.5 Procedure

A pseudo code description of the algorithms procedure.

5.7.6 Heuristics

A bullet-point listing of best practice usage.

5.7.7 Tutorial

A textural narrative for realizing the algorithm with complete source code.

5.7.8 References

An bullet-point annotated reference list of primary sources and useful resources.

5.8 Hierarchical Bayesian Optimization Algorithm

The heading and alternate headings for the algorithm description.

5.8.1 Taxonomy

A small tree diagram showing related fields and algorithms.

5.8.2 Inspiration

A textual description of the inspiring system.

5.8.3 Metaphor

A textual description of the algorithm by analogy.

5.8.4 Strategy

A textual description of the information processing strategy.

5.8.5 Procedure

A pseudo code description of the algorithms procedure.

5.8.6 Heuristics

A bullet-point listing of best practice usage.

5.8.7 Tutorial

A textural narrative for realizing the algorithm with complete source code.

5.8.8 References

An bullet-point annotated reference list of primary sources and useful resources.

5.9 Univariate Marginal Distribution Algorithm

The heading and alternate headings for the algorithm description.

5.9.1 Taxonomy

A small tree diagram showing related fields and algorithms.

5.9.2 Inspiration

A textual description of the inspiring system.

5.9.3 Metaphor

A textual description of the algorithm by analogy.

5.9.4 Strategy

A textual description of the information processing strategy.

5.9.5 Procedure

A pseudo code description of the algorithms procedure.

5.9.6 Heuristics

A bullet-point listing of best practice usage.

5.9.7 Tutorial

A textural narrative for realizing the algorithm with complete source code.

5.9.8 References

An bullet-point annotated reference list of primary sources and useful resources.

5.10 Bivariate Marginal Distribution Algorithm

The heading and alternate headings for the algorithm description.

5.10.1 Taxonomy

A small tree diagram showing related fields and algorithms.

5.10.2 Inspiration

A textual description of the inspiring system.

5.10.3 Metaphor

A textual description of the algorithm by analogy.

5.10.4 Strategy

A textual description of the information processing strategy.

5.10.5 Procedure

A pseudo code description of the algorithms procedure.

5.10.6 Heuristics

A bullet-point listing of best practice usage.

5.10.7 Tutorial

A textural narrative for realizing the algorithm with complete source code.

5.10.8 References

An bullet-point annotated reference list of primary sources and useful resources.

5.11 Gaussian Adaptation

The heading and alternate headings for the algorithm description.

5.11.1 Taxonomy

A small tree diagram showing related fields and algorithms.

5.11.2 Inspiration

A textual description of the inspiring system.

5.11.3 Metaphor

A textual description of the algorithm by analogy.

5.11.4 Strategy

A textual description of the information processing strategy.

5.11.5 Procedure

A pseudo code description of the algorithms procedure.

5.11.6 Heuristics

A bullet-point listing of best practice usage.

5.11.7 Tutorial

A textural narrative for realizing the algorithm with complete source code.

5.11.8 References

An bullet-point annotated reference list of primary sources and useful resources.

5.12 Summary

todo

Chapter 6

Swarm Algorithms

6.1 Overview

todo

6.2 Particle Swarm Optimization

The heading and alternate headings for the algorithm description.

6.2.1 Taxonomy

A small tree diagram showing related fields and algorithms.

6.2.2 Inspiration

A textual description of the inspiring system.

6.2.3 Metaphor

A textual description of the algorithm by analogy.

6.2.4 Strategy

A textual description of the information processing strategy.

6.2.5 Procedure

A pseudo code description of the algorithms procedure.

6.2.6 Heuristics

A bullet-point listing of best practice usage.

6.2.7 Tutorial

A textural narrative for realizing the algorithm with complete source code.

6.2.8 References

An bullet-point annotated reference list of primary sources and useful resources.

6.3 AntNet

The heading and alternate headings for the algorithm description.

6.3.1 Taxonomy

A small tree diagram showing related fields and algorithms.

6.3.2 Inspiration

A textual description of the inspiring system.

6.3.3 Metaphor

A textual description of the algorithm by analogy.

6.3.4 Strategy

A textual description of the information processing strategy.

6.3.5 Procedure

A pseudo code description of the algorithms procedure.

6.3.6 Heuristics

A bullet-point listing of best practice usage.

6.3.7 Tutorial

A textural narrative for realizing the algorithm with complete source code.

6.3.8 References

An bullet-point annotated reference list of primary sources and useful resources.

6.4 Ant System

The heading and alternate headings for the algorithm description.

6.4.1 Taxonomy

A small tree diagram showing related fields and algorithms.

6.4.2 Inspiration

A textual description of the inspiring system.

6.4.3 Metaphor

A textual description of the algorithm by analogy.

6.4.4 Strategy

A textual description of the information processing strategy.

6.4.5 Procedure

A pseudo code description of the algorithms procedure.

6.4.6 Heuristics

A bullet-point listing of best practice usage.

6.4.7 Tutorial

A textural narrative for realizing the algorithm with complete source code.

6.4.8 References

An bullet-point annotated reference list of primary sources and useful resources.

6.5 MAX-MIN Ant System

The heading and alternate headings for the algorithm description.

6.5.1 Taxonomy

A small tree diagram showing related fields and algorithms.

6.5.2 Inspiration

A textual description of the inspiring system.

6.5.3 Metaphor

A textual description of the algorithm by analogy.

6.5.4 Strategy

A textual description of the information processing strategy.

6.5.5 Procedure

A pseudo code description of the algorithms procedure.

6.5.6 Heuristics

A bullet-point listing of best practice usage.

6.5.7 Tutorial

A textural narrative for realizing the algorithm with complete source code.

6.5.8 References

An bullet-point annotated reference list of primary sources and useful resources.

6.6 Rank-Based Ant System

The heading and alternate headings for the algorithm description.

6.6.1 Taxonomy

A small tree diagram showing related fields and algorithms.

6.6.2 Inspiration

A textual description of the inspiring system.

6.6.3 Metaphor

A textual description of the algorithm by analogy.

6.6.4 Strategy

A textual description of the information processing strategy.

6.6.5 Procedure

A pseudo code description of the algorithms procedure.

6.6.6 Heuristics

A bullet-point listing of best practice usage.

6.6.7 Tutorial

A textural narrative for realizing the algorithm with complete source code.

6.6.8 References

An bullet-point annotated reference list of primary sources and useful resources.

6.7 Ant Colony System

The heading and alternate headings for the algorithm description.

6.7.1 Taxonomy

A small tree diagram showing related fields and algorithms.

6.7.2 Inspiration

A textual description of the inspiring system.

6.7.3 Metaphor

A textual description of the algorithm by analogy.

6.7.4 Strategy

A textual description of the information processing strategy.

6.7.5 Procedure

A pseudo code description of the algorithms procedure.

6.7.6 Heuristics

A bullet-point listing of best practice usage.

6.7.7 Tutorial

A textural narrative for realizing the algorithm with complete source code.

6.7.8 References

An bullet-point annotated reference list of primary sources and useful resources.

6.8 Multiple Ant Colony System

The heading and alternate headings for the algorithm description.

6.8.1 Taxonomy

A small tree diagram showing related fields and algorithms.

6.8.2 Inspiration

A textual description of the inspiring system.

6.8.3 Metaphor

A textual description of the algorithm by analogy.

6.8.4 Strategy

A textual description of the information processing strategy.

6.8.5 Procedure

A pseudo code description of the algorithms procedure.

6.8.6 Heuristics

A bullet-point listing of best practice usage.

6.8.7 Tutorial

A textural narrative for realizing the algorithm with complete source code.

6.8.8 References

An bullet-point annotated reference list of primary sources and useful resources.

6.9 Population-based Ant Colony Optimization

The heading and alternate headings for the algorithm description.

6.9.1 Taxonomy

A small tree diagram showing related fields and algorithms.

6.9.2 Inspiration

A textual description of the inspiring system.

6.9.3 Metaphor

A textual description of the algorithm by analogy.

6.9.4 Strategy

A textual description of the information processing strategy.

6.9.5 Procedure

A pseudo code description of the algorithms procedure.

6.9.6 Heuristics

A bullet-point listing of best practice usage.

6.9.7 Tutorial

A textural narrative for realizing the algorithm with complete source code.

6.9.8 References

An bullet-point annotated reference list of primary sources and useful resources.

6.10 Bees Algorithm

The heading and alternate headings for the algorithm description.

6.10.1 Taxonomy

A small tree diagram showing related fields and algorithms.

6.10.2 Inspiration

A textual description of the inspiring system.

6.10.3 Metaphor

A textual description of the algorithm by analogy.

6.10.4 Strategy

A textual description of the information processing strategy.

6.10.5 Procedure

A pseudo code description of the algorithms procedure.

6.10.6 Heuristics

A bullet-point listing of best practice usage.

6.10.7 Tutorial

A textural narrative for realizing the algorithm with complete source code.

6.10.8 References

An bullet-point annotated reference list of primary sources and useful resources.

6.11 Bacterial Foraging Optimization Algorithm

The heading and alternate headings for the algorithm description.

6.11.1 Taxonomy

A small tree diagram showing related fields and algorithms.

6.11.2 Inspiration

A textual description of the inspiring system.

6.11.3 Metaphor

A textual description of the algorithm by analogy.

6.11.4 Strategy

A textual description of the information processing strategy.

6.11.5 Procedure

A pseudo code description of the algorithms procedure.

6.11.6 Heuristics

A bullet-point listing of best practice usage.

6.11.7 Tutorial

A textural narrative for realizing the algorithm with complete source code.

6.11.8 References

An bullet-point annotated reference list of primary sources and useful resources.

6.12 Summary

todo

Chapter 7

Immune Algorithms

7.1 Overview

todo

7.2 Clonal Selection Algorithm

The heading and alternate headings for the algorithm description.

7.2.1 Taxonomy

A small tree diagram showing related fields and algorithms.

7.2.2 Inspiration

A textual description of the inspiring system.

7.2.3 Metaphor

A textual description of the algorithm by analogy.

7.2.4 Strategy

A textual description of the information processing strategy.

7.2.5 Procedure

A pseudo code description of the algorithms procedure.

7.2.6 Heuristics

A bullet-point listing of best practice usage.

7.2.7 Tutorial

A textural narrative for realizing the algorithm with complete source code.

7.2.8 References

An bullet-point annotated reference list of primary sources and useful resources.

7.3 Negative Selection Algorithm

The heading and alternate headings for the algorithm description.

7.3.1 Taxonomy

A small tree diagram showing related fields and algorithms.

7.3.2 Inspiration

A textual description of the inspiring system.

7.3.3 Metaphor

A textual description of the algorithm by analogy.

7.3.4 Strategy

A textual description of the information processing strategy.

7.3.5 Procedure

A pseudo code description of the algorithms procedure.

7.3.6 Heuristics

A bullet-point listing of best practice usage.

7.3.7 Tutorial

A textural narrative for realizing the algorithm with complete source code.

7.3.8 References

An bullet-point annotated reference list of primary sources and useful resources.

7.4 Artificial Immune Recognition System

The heading and alternate headings for the algorithm description.

7.4.1 Taxonomy

A small tree diagram showing related fields and algorithms.

7.4.2 Inspiration

A textual description of the inspiring system.

7.4.3 Metaphor

A textual description of the algorithm by analogy.

7.4.4 Strategy

A textual description of the information processing strategy.

7.4.5 Procedure

A pseudo code description of the algorithms procedure.

7.4.6 Heuristics

A bullet-point listing of best practice usage.

7.4.7 Tutorial

A textural narrative for realizing the algorithm with complete source code.

7.4.8 References

An bullet-point annotated reference list of primary sources and useful resources.

7.5 Immune Network Algorithm

The heading and alternate headings for the algorithm description.

7.5.1 Taxonomy

A small tree diagram showing related fields and algorithms.

7.5.2 Inspiration

A textual description of the inspiring system.

7.5.3 Metaphor

A textual description of the algorithm by analogy.

7.5.4 Strategy

A textual description of the information processing strategy.

7.5.5 Procedure

A pseudo code description of the algorithms procedure.

7.5.6 Heuristics

A bullet-point listing of best practice usage.

7.5.7 Tutorial

A textural narrative for realizing the algorithm with complete source code.

7.5.8 References

An bullet-point annotated reference list of primary sources and useful resources.

7.6 Dendritic Cell Algorithm

The heading and alternate headings for the algorithm description.

7.6.1 Taxonomy

A small tree diagram showing related fields and algorithms.

7.6.2 Inspiration

A textual description of the inspiring system.

7.6.3 Metaphor

A textual description of the algorithm by analogy.

7.6.4 Strategy

A textual description of the information processing strategy.

7.6.5 Procedure

A pseudo code description of the algorithms procedure.

7.6.6 Heuristics

A bullet-point listing of best practice usage.

7.6.7 Tutorial

A textural narrative for realizing the algorithm with complete source code.

7.6.8 References

An bullet-point annotated reference list of primary sources and useful resources.

7.7 Summary

todo

Part III

Extensions

Chapter 8

Advanced Topics

A chapter focused on applying, testing, visualizing, saving results, and comparing algorithms. The meta concerns once an algorithm is selected for a given practical problem solving scenario.

8.1 Programming Paradigms

Algorithms can be implements on many different programming paradigms. Take the GA for example and realize it using a bunch of different paradigms.

8.1.1 Procedural Programming

The GA under a procedural paradigm

8.1.2 Object-Oriented Programming

The GA under a object oriented paradigm. Strategy pattern. modular operators, etc.

8.1.3 Agent Oriented Programming

A GA under an agent oriented programming paradigm. not really suited. algorithm as an agent with goals?

8.1.4 Functional Programming

The GA under a functional paradigm. closure etc

8.1.5 Meta-Programming

A GA under meta programming. A DSL i guess.

8.1.6 Flow Programming

A GA under a data flow or pipeline model.

8.1.7 Map Reduce

A GA under a map reduce paradigm.

8.2 Devising New Algorithms

A methodology for devising new unconventional optimization algorithms...

8.2.1 Conceptual Framework for Bio-Inspired Algorithms

A generic methodology for devising new biologically inspired algorithms

8.2.2 Information Processing Methodology

An info processing centric approach to devising new algorithms

8.2.3 Investigation

small models, rigor

8.2.4 Communication

you need to effectively describe them, like as in this book! goal is to be known and used, make it open and usable by anyone. like open source, documented, common languages, benchmark problems, a website, lots of papers

8.3 Testing Algorithms

This section will focus on the problem that 'adaptive systems work even when they are not implemented correctly' (they work in-spite of the developer). Topics will include unit testing algorithms, system testing software, specific concerns when testing inspired algorithms, examples of testing algorithms with the ruby unit testing framework, examples of testing algorithms with rspec.

8.3.1 Types of Testing

unit, TDD, system, user acceptance, black box, white box

8.3.2 Algorithm Testing Methodology

testing is hard these systems 'work' even with bugs, hard to test present a methodology for testing - discrete unit tests, behavior testing

8.3.3 Example

develop and show tests for the GA

8.4 Visualizing Algorithms

This section will focus on the use of visualization as a low-fidelity form of system testing. Topics will include free visualization packages such as R, GNUPlot and Processing. Examples visualizing a decision surface, a functions response surface, and candidate solutions.

8.4.1 Visualizing

we can do it as a form of testing. research aid - view on a complex process, can observe, take notes, formulate hypothesis think of all the measures you can, than measure them

Offline Plots

examples?

Online Plots

examples?

8.4.2 Visualization Tools

can use lots of things, can use lots of things

8.4.3 Example

Visuzlize genes through time for a ga run, with fitness graphs, and plots of domain

8.5 Saving Algorithm Results

This section will focus on algorithms and techniques as a fallible means to an end and the need to maintain save results. Topics will include check-pointing, storage in a database, storage on the filesystem, and algorithm restarting. Examples will be given for database, filesystem checkpointing and algorithm restarting.

8.5.1 Check-pointing

algorithms crash and it sucks, need to be able to pickup where you left off

8.5.2 Share Results

make them public with papers and source code

8.5.3 Example

show an example of check pointing

8.6 Comparing Algorithms

This section will focus on comparing algorithm's based on the solutions they provide. Topics will include the use statistical hypothesis testing and free software such as R, algorithm parameter selection, distribution testing, distribution comparisons. Examples will be given for algorithm parameter selection, result distribution classification, and pair-wise result distribution comparison.

8.6.1 No Free Lunch

all same over all problems with no prior info

8.6.2 Benchmarking

standard problem instances what problems? what algorithms? what configurations what are you measuring? what are you comparing?

8.6.3 Statistical Hypothesis Testing

you need stats or you will be killed by Zed Shaw need stats to compare results

8.6.4 Example

genetic algorithm vs something, use R to compare

8.7 Summary

We learned lots of advanced topics, there are more.

Bibliography

[1] S. Aaronson. NP-complete problems and physical reality. *ACM SIGACT News (COLUMN: Complexity theory)*, 36(1):30–52, 2005.

[2] M. M. Ali, C. Storey, and A. Trn. Application of stochastic global optimization algorithms to practical problems. *Journal of Optimization Theory and Applications*, 95(3):545–563, 1997.

[3] C. Andrieu, N. de Freitas, A. Doucet, and M. I. Jordan. An introduction to mcmc for machine learning. *Machine Learning*, 50:5–43, 2003.

[4] Thomas Bäck, D.B Fogel, and Z Michalewicz, editors. *Evolutionary Computation 1: Basic Algorithms and Operators*. IoP, 2000.

[5] Thomas Bäck, D.B Fogel, and Z Michalewicz, editors. *Evolutionary Computation 2: Advanced Algorithms and Operations*. IoP, 2000.

[6] Janine M. Benyus. *Biomimicry: innovation inspired by nature*. Quill, 1998.

[7] D. Bergemann and J. Valimaki. Bandit problems. Cowles Foundation Discussion Papers 1551, Cowles Foundation, Yale University, January 2006.

[8] C. M. Bishop. *Neural Networks for Pattern Recognition*. Oxford University Press, 1995.

[9] Christian Blum and Andrea Roli. Metaheuristics in combinatorial optimization: Overview and conceptual comparison. *ACM Computing Surveys (CSUR)*, 35(3):268–308, 2003.

[10] Eric Bonabeau, Marco Dorigo, and Guy Theraulaz. *Swarm intelligence: from natural to artificial systems*. Oxford University Press US, 1999.

[11] S. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge University Press, 2004.

[12] E. K. Burke, E. Hart, G. Kendall, J. Newall, P. Ross, and S. Schulenburg. *Handbook of Metaheuristics*, chapter Hyper-heuristics: An emerging direction in modern search technology, pages 457–474. Kluwer, 2003.

[13] E. K. Burke, G. Kendall, and E. Soubeiga. A tabu-search hyper-heuristic for timetabling and rostering. *Journal of Heuristics*, 9(6):451–470, 2003.

[14] Leandro N. De Castro and Fernando J. Von Zuben. *Recent developments in biologically inspired computing.* Idea Group Inc, 2005.

[15] David Corne, Marco Dorigo, and Fred Glover. *New ideas in optimization.* McGraw-Hill, 1999.

[16] L. N. de Castro and F. J. Von Zuben. *Recent developments in biologically inspired computing*, chapter From biologically inspired computing to natural computing. Idea Group, 2005.

[17] Leandro N. de Castro and Jonathan Timmis. *Artificial Immune Systems: A New Computational Intelligence Approach.* Springer, 2002.

[18] Marco Dorigo and Thomas Stützle. *Ant colony optimization.* MIT Press, 2004.

[19] Stefan Droste, Thomas Jansen, and Ingo Wegener. Upper and lower bounds for randomized search heuristics in black-box optimization. *Theory of Computing Systems*, 39(4):525–544, 2006.

[20] Andries P. Engelbrecht. *Computational intelligence: an introduction.* John Wiley and Sons, second edition, 2007.

[21] David Flanagan and Yukihiro Matsumoto. *The Ruby Programming Language.* O'Reilly Media, 2008.

[22] N. Forbes. Biologically inspired computing. *Computing in Science and Engineering*, 2(6):83–87, 2000.

[23] Nancy Forbes. *Imitation of Life: How Biology Is Inspiring Computing.* The MIT Press, 2005.

[24] K. Fukunaga. *Introduction to Statistical Pattern Recognition.* Academic Press, 1990.

[25] Fred Glover and Gary A. Kochenberger. *Handbook of metaheuristics.* Springer, 2003.

[26] David Edward Goldberg. *Genetic algorithms in search, optimization, and machine learning.* Addison-Wesley, 1989.

[27] I Guyon and A Elisseeff. An introduction to variable and feature selection. *Journal of Machine Learning Research*, 3:1157–1182, 2003.

[28] John Henry Holland. *Adaptation in natural and artificial systems: an introductory analysis with applications to biology, control, and artificial intelligence.* University of Michigan Press, 1975.

[29] R. Horst, P. M. Pardalos, and N. V. Thoai. *Introduction to Global Optimization.* Kluwer Academic Publishers, 2nd edition, 2000.

[30] John R. Koza. *Genetic programming IV: routine human-competitive machine intelligence.* Springer, 2003.

[31] M. Kudo and J. Sklansky. Comparison of algorithms that select features for pattern classifiers. *Pattern Recognition*, 33:25–41, 2000.

[32] R. M. Lewis, V. T., and M. W. Trosset. Direct search methods: then and now. *Journal of Computational and Applied Mathematics*, 124:191–207, 2000.

[33] George F. Luger and William A. Stubblefield. *Artificial Intelligence: Structures and Strategies for Complex Problem Solving.* Benjamin/Cummings Pub. Co., second edition, 1993.

[34] Sean Luke. *Essentials of Metaheuristics.* (self-published) http://cs.gmu.edu/∼sean/book/metaheuristics, 2009.

[35] P. Marrow. Nature-inspired computing technology and applications. *BT Technology Journal*, 18(4):13–23, 2000.

[36] Zbigniew Michalewicz and David B. Fogel. *How to solve it: modern heuristics.* Springer, 2004.

[37] C. H. Papadimitriou and K. Steiglitz. *Combinatorial Optimization: Algorithms and Complexity.* Courier Dover Publications, 1998.

[38] R. Paton. *Computing With Biological Metaphors*, chapter Introduction to computing with biological metaphors, pages 1–8. Chapman & Hall, 1994.

[39] G. Paun. Bio-inspired computing paradigms (natural computing). *Unconventional Programming Paradigms*, 3566:155–160, 2005.

[40] W. Pedrycz. *Computational Intelligence: An Introduction.* CRC Press, 1997.

[41] H. Robbins. Some aspects of the sequential design of experiments. *Bull. Amer. Math. Soc.*, 58:527–535, 1952.

[42] Stuart Russell and Peter Norvig. *Artificial Intelligence: A Modern Approach.* Prentice Hall, third edition, 2009.

[43] Yuhui Shi, editor. *Swarm intelligence.* Morgan Kaufmann, 2001.

[44] A. Sloman. *Evolving Knowledge in Natural Science and Artificial Intelligence*, chapter Must intelligent systems be scruffy? Pitman, 1990.

[45] James C. Spall. *Introduction to stochastic search and optimization: estimation, simulation, and control.* John Wiley and Sons, 2003.

[46] El-Ghazali Talbi. *Metaheuristics: From Design to Implementation.* John Wiley and Sons, 2009.

[47] Dave Thomas, Chad Fowler, and Andy Hunt. *Programming Ruby: The Pragmatic Programmers' Guide.* Pragmatic Bookshelf, second edition, 2004.

[48] A. Trn, M.M. Ali, and S. Viitanen. Stochastic global optimization: Problem classes and solution techniques. *Journal of Global Optimization*, 14:437–447, 1999.

[49] Thomas Weise. *Global Optimization Algorithms - Theory and Application.* Thomas Weise, 2009-06-26 edition, 2007.

[50] D. H. Wolpert and W. G. Macready. No free lunch theorems for search. Technical report, Santa Fe Institute, Sante Fe, NM, USA, 1995.

[51] D. H. Wolpert and W. G. Macready. No free lunch theorems for optimization. *IEEE Transactions on Evolutionary Computation*, 1(67):67–82, 1997.

[52] Lotfi Asker Zadeh, George J. Klir, and Bo Yuan. *Fuzzy sets, fuzzy logic, and fuzzy systems: selected papers.* World Scientific, 1996.