Introduction to Evolutionary Algorithms and Drift Theorems

Daniel Cermann

July 13, 2024

Abstract

This paper provides an introduction to evolutionary algorithms (EAs) and drift theorems, aimed at readers with little prior knowledge of these topics. Evolutionary algorithms are versatile tools for exploring large search spaces little is known about, and drift theory offers insights into their runtime behavior. The motivation and applications of EAs will be discussed, and basic concepts such as Random Local Search and the (1+1) Evolutionary Algorithm will be explained. Test functions and their role in evaluating EAs will also be covered. Finally, the paper will introduce drift theorems and their application in analyzing the efficiency and performance of evolutionary algorithms. Furthermore, a visualization tool for gaining intuitive insights into various testing functions, as well as their expected runtime, will be presented.

Contents

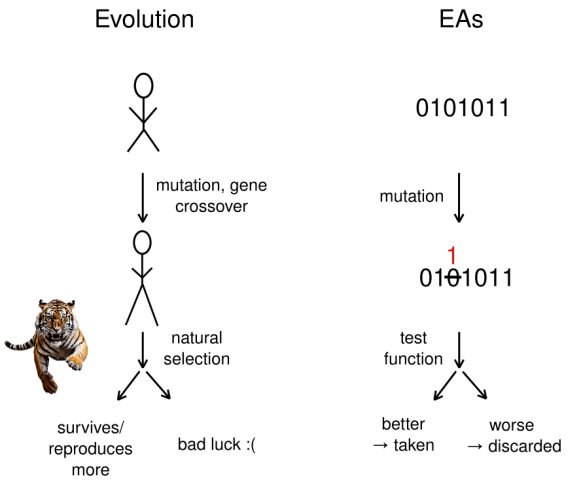
1 Introduction

Nowadays, artificial intelligence (AI) is on everyone's lips and has become one of the most prominent trending topics in public discussion. Many of the better-known models and techniques, such as neural networks, share a common trait: they operate within a well-specified problem space and are designed to identify patterns in given examples or training data.
Evolutionary algorithms (EAs), on the other hand, are a set of tools designed to explore vast and often poorly understood search spaces to find optimal or near-optimal solutions. These algorithms draw inspiration from the principles of natural evolution, employing mechanisms such as selection, mutation, and crossover to iteratively improve potential solutions. Given their versatility and robustness, EAs have been successfully applied to a wide range of problems across various fields.
To analyze the runtime behavior of these algorithms, it is essential to understand drift theorems. Drift theorems provide a framework for examining how certain stochastic processes, like those underlying EAs, evolve over time. By understanding the drift in an evolutionary algorithm, we can gain insights into its efficiency and performance.
This paper aims to introduce the fundamentals of evolutionary algorithms and drift theorems to readers with little prior knowledge. Additionally, a visualization tool will be presented to provide an intuitive understanding of various test functions and their expected runtime. Finally, further reading materials will be suggested for those interested in delving deeper into the topics discussed.

2 Fundamentals of Evolutionary Algorithms

Random local search and the (1+1) evolutionary algorithm

In this section, we will introduce the basic concepts of evolutionary algorithms, starting with random local search and the (1+1) evolutionary algorithm. Those are two fundamental algorithms that are used as building blocks for more complex evolutionary algorithms and offer deep insights into the challenges in the field.
Figure 2 sketches the fundamental ideas of evolutionary algorithms and contrasts it to its inspiration: evolution in nature.



Darwin's theory of natural selection considers the long-term adaption of a species to its environment. It suggests that individuals with advantageous traits are more likely to survive and reproduce, thereby passing those traits to future generations.
Evolutionary algorithms fundamentally work in the same way. Only, instead of animal or plant species bitstrings or other representations of potential solutions are evolved. Here, we will focus on binary strings, but the principles can be transferred to other representations and real-life problems as well.
For an evolutionary algorithm you always need some function that takes a bitstring, in this context often called individual, and changes any number of its bit. This step is called mutation and often is non-deterministic. In a next step the EA determines if the new individual is better than the old one. For this, a so called fitness or test function is used. This function takes an individual and returns a value that represents how well the individual adapted to the given function. The goal of the EA is to maximize this value. Finding a suited fitness function for a real-life problem can be very challenging and is crucial for the success of an evolutionary algorithm.
The following pseudocode shows how random local search and (1+1) EA work.

Random local search (RLS)

Sample $x \in \{0,1\}^n$ uniformly at random
for $i = 1$ to ∞ **do**
  $y \leftarrow flipOne(x)$
  if $f(y) \geq f(x)$ **then** $x \leftarrow y$

(1 + 1) Evolutionary algorithm (EA)

Sample $x \in \{0,1\}^n$ uniformly at random
for $i = 1$ to ∞ **do**
  $y \leftarrow mutate(x)$
  if $f(y) \geq f(x)$ **then** $x \leftarrow y$

The main difference between the two algorithms is the mutation function. **Random local search** mutates indivuals by randomly choosing one of the n bits and inverting it. **(1 + 1) EA** on the other hand uses a mutation function that flips each bit with a probability of $\frac{1}{n}$. Consequently, in each iteration of the (1 + 1) EA, between 0 and n bits of the indivual can be flipped, while **RLS** always flips exactly one.

2.1 Test Functions

As already mentioned, the chosen fitness function is of utmost importance for the success of an evolutionary algorithm. While fitness functions of real-life problems can be very complex and hard to define, we will focus on simple test functions in these analyses.
As most of the discussed functions are not very complex only a brief definition, an explanation and a short example will be given.

2.1.1 OneMax

The fitness function *OneMax* assigns to each bit string italicx the number of one-bits in x [**?**]. It can be defined as follows:

$$\text{OneMax}(x) = \sum_{i=1}^{n} x_i$$

where x_i is the i-th bit of x.
Some things to note about this fitness function is, that *oneMax* is has some attributes,

that simplify evolutionary algorithms significantly when run with these. For example, this function possesses exactly one optimum that being the string consisting only of ones. This optimum being the only one consequent also is the global optimum. TODO: INSERT STUFF HERE LIKE MONOTICALLY INCREASING; HAS EXACTLY ONE OPTIMUM WHICH IS THE GLOBAL OPTIMUM
An extension of *oneMax* are *linearFunctions*. These test functions assign every bit x_i a real weight w_i. The definition of *linearFunctions* is as follows:

$$\text{linearFunctions}(x) = \sum_{i=1}^{n} w_i \cdot x_i$$

where x_i is the i-th bit of x and w_i is the weight of the i-th bit.

2.1.2 LeadingOnes

LeadingOnes is a function that assigns to each bit string x the length of the longest prefix of x that consists of one-bits [**?**]. It can be defined as follows:

$$\text{LeadingOnes}(x) = \sum_{i=1}^{n} \prod_{j=0}^{i} x_i$$

where x_i is the i-th bit of x.

2.1.3 Needle

Evolutions algorithms often rely on the fitness function to incrementally lead to the optimal or at least close-to optimal solution by iteratively improving the current solution. The *needle* function is a test function that is designed to make this very difficult. Only the bit string consisting only of ones has a fitness bigger than zero while any other bit string has the fitness zero. This means, an evolutionary algorithm has no way of approaching the optimal solution, but instead has to encounter it randomly. It is looking for the figurative needle in a haystack. It is defined as follows:

$$\text{needle}(x) = \begin{cases} n & \text{if } x = 1^n \\ 0 & \text{otherwise} \end{cases}$$

where $x = 1^n$ is the bitstring of length n that consists only of ones.

3 Drift Theorems

3.1 Introduction to Drift Theorems

3.2 Application of Drift Theorems

Explain how drift theorems can be applied to analyze the runtime behavior of evolutionary algorithms.

3.2.1 Expected Runtime of OneMax

3.2.2 Expected Runtime of LeadingOnes

4 Conclusion and Further Reading

Summarize the key points discussed in the paper and suggest further reading materials for readers interested in deepening their understanding of evolutionary algorithms and drift theorems.

References