A Gentle Introduction to the Time Complexity Analysis of Evolutionary Algorithms

Pietro S. Oliveto
Department of Computer Science, University of Sheffield, UK

SSCI 2015
Cape Town, South Africa, 7 December 2015

Aims and Goals of this Tutorial

- This tutorial will provide an overview of
  - the goals of time complexity analysis of Evolutionary Algorithms (EAs)
  - the most common and effective techniques
- You should attend if you wish to
  - theoretically understand the behaviour and performance of the search algorithms you design
  - familiarise with the techniques used in the time complexity analysis of EAs
  - pursue research in the area
- enable you or enhance your ability to
  - understand theoretically the behaviour of EAs on different problems
  - perform time complexity analysis of simple EAs on common toy problems
  - read and understand research papers on the computational complexity of EAs
  - have the basic skills to start independent research in the area

Introduction to the theory of EAs

Theoretical studies of Evolutionary Algorithms (EAs), albeit few, have always existed since the seventies [Goldberg, 1989];

- Early studies were concerned with explaining the behaviour rather than analysing their performance.
- Schema Theory was considered fundamental;
  - First proposed to understand the behaviour of the simple GA [Holland, 1992];
  - It cannot explain the performance or limit behaviour of EAs;
  - Building Block Hypothesis was controversial [Reeves and Rowe, 2002];
- Convergence results appeared in the nineties [Rudolph, 1998];
  - Related to the time limit behaviour of EAs.

Goals of design and analysis of algorithms

- correctness
  "does the algorithm always output the correct solution?"
- computational complexity
  "how many computational resources are required?"

For Evolutionary Algorithms (General purpose)

- convergence
  "Does the EA find the solution in finite time?"
- time complexity
  "how long does it take to find the optimum?"
  (time = n. of fitness function evaluations)

Brief history

Theoretical studies of Evolutionary Algorithms (EAs), albeit few, have always existed since the seventies [Goldberg, 1989];

- Early studies were concerned with explaining the behaviour rather than analysing their performance.
- Schema Theory was considered fundamental;
  - First proposed to understand the behaviour of the simple GA [Holland, 1992];
  - It cannot explain the performance or limit behaviour of EAs;
  - Building Block Hypothesis was controversial [Reeves and Rowe, 2002];
- Convergence results appeared in the nineties [Rudolph, 1998];
  - Related to the time limit behaviour of EAs.
Convergence

Definition

- Ideally the EA should find the solution in finite steps with probability 1 (visit the global optimum in finite time);
- If the solution is held forever after, then the algorithm converges to the optimum!

Conditions for Convergence ([Rudolph, 1998])

1. There is a positive probability to reach any point in the search space from any other point
2. The best found solution is never removed from the population (elitism)

- Canonical GAs using mutation, crossover and proportional selection Do Not converge!
- Elitist variants Do converge!

In practice, is it interesting that an algorithm converges to the optimum?

- Most EAs visit the global optimum in finite time (RLS does not!)
- How much time?

Computational Complexity of EAs

Generally means predicting the resources the algorithm requires:

- Usually the computational time: the number of primitive steps;
- Usually grows with size of the input;
- Usually expressed in asymptotic notation;

Exponential runtime: Inefficient algorithm
Polynomial runtime: "Efficient" algorithm

Asymptotic notation

\[

c(n) \in O(g(n)) \iff \exists \text{ constants } c, n_0 > 0 \text{ st. } 0 \leq c n \leq g(n) \forall n \geq n_0 \\

c(n) \in \Omega(g(n)) \iff f(n) \in O(g(n)) \text{ and } f(n) \in \Omega(g(n)) \\

c(n) \in \omega(g(n)) \iff \lim_{n \to \infty} \frac{c(n)}{g(n)} = 0
\]
Exercise 1: Asymptotic Notation

<table>
<thead>
<tr>
<th>( f_1(n) = \log(n^2) )</th>
<th>( O(1) )</th>
<th>( O(\log n) )</th>
<th>( O(n^2) )</th>
<th>( n^{\Theta(1)} )</th>
<th>( e^{\Omega(n)} )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( f_2(n) = \frac{n(n-1)}{2} )</td>
<td>( \sqrt{\sqrt{\cdot}} )</td>
<td>( \sqrt{\cdot} )</td>
<td>( \sqrt{\cdot} )</td>
<td>( \sqrt{\cdot} )</td>
<td>( \sqrt{\cdot} )</td>
</tr>
<tr>
<td>( f_3(n) = \sqrt{n} )</td>
<td>( \sqrt{\cdot} )</td>
<td>( \sqrt{\cdot} )</td>
<td>( \sqrt{\cdot} )</td>
<td>( \sqrt{\cdot} )</td>
<td>( \sqrt{\cdot} )</td>
</tr>
<tr>
<td>( f_4(n) = n! )</td>
<td>( \sqrt{\cdot} )</td>
<td>( \sqrt{\cdot} )</td>
<td>( \sqrt{\cdot} )</td>
<td>( \sqrt{\cdot} )</td>
<td>( \sqrt{\cdot} )</td>
</tr>
<tr>
<td>( f_5(n) = \frac{1}{n} )</td>
<td>( \sqrt{\cdot} )</td>
<td>( \sqrt{\cdot} )</td>
<td>( \sqrt{\cdot} )</td>
<td>( \sqrt{\cdot} )</td>
<td>( \sqrt{\cdot} )</td>
</tr>
<tr>
<td>( f_6(n) = 100 )</td>
<td>( \sqrt{\cdot} )</td>
<td>( \sqrt{\cdot} )</td>
<td>( \sqrt{\cdot} )</td>
<td>( \sqrt{\cdot} )</td>
<td>( \sqrt{\cdot} )</td>
</tr>
<tr>
<td>( f_7(n) = 2^n )</td>
<td>( \sqrt{\cdot} )</td>
<td>( \sqrt{\cdot} )</td>
<td>( \sqrt{\cdot} )</td>
<td>( \sqrt{\cdot} )</td>
<td>( \sqrt{\cdot} )</td>
</tr>
<tr>
<td>( f_8(n) = 2^{-n} n^n )</td>
<td>( \sqrt{\cdot} )</td>
<td>( \sqrt{\cdot} )</td>
<td>( \sqrt{\cdot} )</td>
<td>( \sqrt{\cdot} )</td>
<td>( \sqrt{\cdot} )</td>
</tr>
</tbody>
</table>

[Lohm, Tutorial]

Motivation Overview

**Overview**
- **Goal**: Analyze the correctness and performance of EAs;
- **Difficulties**: General purpose, randomised;
- EAs find the solution in finite time; (convergence analysis)
- How much time? → Derive the expected runtime and the success probability;

**Next**
- Basic Probability Theory: probability space, random variables, expectations (expected runtime)
- Randomised Algorithm Tools: Tail inequalities (success probabilities)

Along the way
- Understand that the analysis cannot be done over all functions
- Understand why the success probability is important (expected runtime not always sufficient)

Evolutionary Algorithms

Algorithm ((\( \mu+\lambda \))-EA)

- Let \( t = 0 \);
  - Initialize \( P_0 \) with \( \mu \) individuals chosen uniformly at random;
  - Repeat
    - Create \( \lambda \) new individuals:
      - choose \( x \in P_0 \) uniformly at random;
      - flip each bit in \( x \) with probability \( p \);
    - Create the new population \( P_{t+1} \) by choosing the best \( \mu \) individuals out of \( \mu + \lambda \);
  - Let \( t = t + 1 \).
- Until a stopping condition is fulfilled.

If only one bit is flipped per iteration: Random Local Search (RLS).

How does it work?
- Given \( x \), how many bits will flip in expectation?

\[
E[X] = E[X_1 + X_2 + \cdots + X_n] = E[X_1] + E[X_2] + \cdots + E[X_n] = n p
\]

For \( p = 1/n \):

\[
E = \sum_{i=1}^{n} 1 \cdot 1/n = n/n = 1
\]
1+1-EA: General Upper bound

How likely is it that exactly one bit flips? \( Pr(X = j) = \binom{n}{j} p^j (1-p)^{n-j} \)

- What is the probability of exactly one bit flipping?
  \[ Pr(X = 1) = \binom{n}{1} \cdot \frac{1}{n} \cdot \frac{1}{1-n}^{n-1} = (1-1/n)^{n-1} \geq 1/e \approx 0.37 \]

Is it more likely that 2 bits flip or none?

\[ Pr(X = 2) = \frac{n}{2} \cdot \frac{1}{n^2} \cdot (1-1/n)^{n-2} = \]
\[ = \frac{n \cdot (n-1)}{2} \cdot \frac{1}{n^2} \cdot (1-1/n)^{n-2} = \]
\[ = 1/2 \cdot (1-1/n)^{n-1} \approx 1/(2e) \]

While

\[ Pr(X = 0) = \binom{n}{0} \frac{1}{n^0} \cdot (1-1/n)^n \approx 1/e \]

1+1-EA: Conclusions & Exercises

**Theorem** ([Droste et al., 2002])

The expected runtime of the (1+1)-EA for an arbitrary function defined in \( \{0,1\}^n \) is \( O(n^n) \)

**Proof**
- Let \( i \) be the number of bit positions in which the current solution \( x \) and the global optimum \( x^* \) differ;
- Each bit flips with probability \( 1/n \), hence does not flip with probability \( (1-1/n) \);
- In order to reach the global optimum the algorithm has to mutate the \( i \) bits and leave the \( n-i \) bits unchanged;
- Then:
  \[ p(x^*|x) = \left( \frac{1}{n} \right)^i \left( 1 - \frac{1}{n} \right)^{n-i} \geq \left( \frac{1}{n} \right)^n = n^{-n} (p = n^{-n}) \]
- it implies an upper bound on the expected runtime of \( O(n^n) \)
  \( E(X) = 1/p = n^n \) (waiting time argument).

In general:

\[ P(i \text{ - bitflip}) = \binom{n}{i} \frac{1}{n^i} \left( \frac{1-1}{n} \right)^{n-i} \leq \frac{1}{i!} \left( \frac{1-1}{n} \right)^{n-i} \approx \frac{1}{i! e} \]

What about RLS?
- Expectation: \( E[X] = 1 \)
- \( P(1\text{-bitflip}) = 1 \)

What about initialisation?
- How many one-bits in expectation after initialisation?
  \( E[X] = n \cdot 1/2 = n/2 \)

How likely is it that we get exactly \( n/2 \) one-bits?

\[ Pr(X = n/2) = \binom{n}{n/2} \frac{1}{n^{n/2}} \left( 1 - \frac{1}{n} \right)^{n/2} \]

\( n = 100, Pr(X = 50) \approx 0.0796 \)

Tail Inequalities help us!
Tail Inequalities

Given a random variable $X$ it may assume values that are considerably larger or lower than its expectation;

**Tail inequalities:**
- The expectation can often be estimate easily;
- We would like to know the probability of deviating far from the expectation i.e., the “tails” of the distribution
- Tail inequalities give bounds on the tails given the expectation.

Markov’s inequality

**Markov Inequality**

The fundamental inequality from which many others are derived.

**Definition (Markov’s Inequality)**

Let $X$ be a random variable assuming only non-negative values, and $E[X]$ its expectation. Then for all $t \in \mathbb{R}^+$,

$$Pr[X \geq t] \leq \frac{E[X]}{t}.$$  

- $E[X] = 1$; then: $Pr[X \geq 2] \leq \frac{E[X]}{2}$ (Number of bits that flip)
- $E[X] = n/2$; then $Pr[X \geq (2/3)n] \leq \frac{E[X]}{(2/3)n} = \frac{n/2}{(2/3)n} = \frac{3}{4}$ (Number of one-bits after initialisation)

Markov’s inequality is often used iteratively in repeated phases to obtain stronger bounds!

Chernoff bounds

**Chernoff Bound Simple Application**

Let $X_1, X_2, \ldots, X_n$ be independent Poisson trials each with probability $p_i$; For $X = \sum_{i=1}^{n} X_i$, the expectation is $E(X) = \sum_{i=1}^{n} p_i$.

**Definition (Chernoff Bounds)**

- for $0 \leq \delta \leq 1$, $Pr(X \leq (1-\delta)E[X]) \leq e^{-\frac{\delta^2 E[X]}{2}}$.
- for $\delta > 0$, $Pr(X > (1+\delta)E[X]) \leq e^{-\frac{\delta^2 E[X]}{2(1+\delta)^2}}$.

What is the probability that we have more than $(2/3)n$ one-bits at initialisation?

- $p_i = 1/2$, $E[X] = n \cdot 1/2 = n/2$, (we fix $\delta = 1/3 \rightarrow (1+\delta)E[X] = (2/3)n$); then:
  - $Pr[X \geq (2/3)n] \leq \left(\frac{e^{1/3}}{(4/3)^{1/3}}\right)^{n/2} = c^{-n/2}$

Bitstring of length $n = 100$

$Pr(X_i) = 1/2$ and $E(X) = np = 100 \cdot 1/2 = 50$.

What is the probability to have at least 75 1-bits?

- **Markov**: $Pr(X \geq 75) \leq \frac{50}{75} = \frac{2}{3}$

- **Chernoff**: $Pr(X \geq (1+1/2)50) \leq \left(\frac{25}{(1+2)^{25/2}}\right)^{50} \approx 0.0045$

- **Truth**: $Pr(X \geq 75) = \sum_{i=75}^{100} \binom{100}{i} 2^{-100} < 0.000000282$
**OneMax**

\[
\text{OneMax} (x) = \sum_{i=1}^{n} x[i]
\]

**RLS for OneMax (OneMax \(x) = \sum_{i=1}^{n} x[i]\))**

\[
\begin{array}{ccc}
0 & 0 & 0 \\
0 & 1 & 2 \\
0 & 1 & 2 \\
0 & 1 & 2 \\
0 & 1 & 2 \\
\end{array}
\]

\[
p_0 = \frac{n}{n} \quad E(T_0) = \frac{n}{n}
\]

\[
\begin{array}{ccc}
0 & 0 & 0 \\
0 & 1 & 0 \\
0 & 1 & 0 \\
0 & 1 & 0 \\
0 & 1 & 0 \\
\end{array}
\]

\[
p_1 = \frac{n-1}{n} \quad E(T_1) = \frac{n-1}{n}
\]

\[
\begin{array}{ccc}
0 & 0 & 0 \\
0 & 1 & 0 \\
0 & 1 & 0 \\
0 & 1 & 0 \\
0 & 1 & 0 \\
\end{array}
\]

\[
p_2 = \frac{n-2}{n} \quad E(T_2) = \frac{n-2}{n}
\]

\[
\begin{array}{ccc}
1 & 1 & 1 \\
0 & 1 & 1 \\
0 & 1 & 1 \\
0 & 1 & 1 \\
0 & 1 & 1 \\
\end{array}
\]

\[
p_{n-1} = \frac{1}{n} \quad E(T_{n-1}) = \frac{1}{n}
\]

\[
E(T) = E(T_0) + E(T_1) + \cdots + E(T_{n-1}) = 1/p_0 + 1/p_1 + \cdots + 1/p_{n-1} = \sum_{i=0}^{n-1} \frac{1}{p_i} = \sum_{i=1}^{n} \frac{n}{i} = n \log n + O(n) = O(n \log n)
\]

---

**Coupon collector’s problem**

There are \(n\) types of coupons and at each trial one coupon is chosen at random. Each coupon has the same probability of being extracted. The goal is to find the exact number of trials before the collector has obtained all the \(n\) coupons.

**Theorem (The coupon collector’s Theorem)**

Let \(T\) be the time for all the \(n\) coupons to be collected. Then

\[
E(T) = \sum_{i=0}^{n-1} \frac{1}{p_{i+1}} = \sum_{i=0}^{n-1} \frac{n}{n-i} = n \sum_{i=0}^{n-1} \frac{1}{i} = n \log n + O(1) = n \log n + O(n).
\]
Drift Analysis
Conclusions
Conclusions
Evolutionary Algorithms
Artificial Fitness Levels
Evolutionary Algorithms
Tail Inequalities
Artificial Fitness Levels
Tail Inequalities
Coupon collector’s problem: Upper bound on time

What is the probability that the time to collect \( n \) coupons is greater than \( n \ln n + O(n) \)?

**Theorem (Coupon collector upper bound on time)**

Let \( T \) be the time for all the \( n \) coupons to be collected. Then

\[
\Pr(T \geq (1 + \epsilon)n \ln n) \leq n^{-\epsilon}
\]

**Proof**

\[
\Pr(T \geq (1 + \epsilon)n \ln n) \leq n^{-\epsilon}
\]

\[
\frac{1}{1 - \frac{1}{n}} \quad \text{Probability of choosing a given coupon}
\]

\[
(1 - \frac{1}{n})^t \quad \text{Probability of not choosing a given coupon for } t \text{ rounds}
\]

The probability that one of the \( n \) coupons is not chosen in \( t \) rounds is less than

\[
n \cdot (1 - \frac{1}{n})^t \quad \text{(Union Bound)}
\]

Hence, for \( t = cn \ln n \)

\[
\Pr(T \geq cn \ln n) \leq n(1 - 1/n)^{cn \ln n} \leq n \cdot e^{-c \ln n} = n(n^{-c}) = n^{-c+1}
\]

**Observation**

Due to elitism, fitness is monotone increasing

**Idea**

Divide the search space \( |S| = 2^n \) into \( m < 2^n \) sets \( A_1, \ldots, A_m \) such that:

\[
\bullet \ i \neq j : \quad A_i \cap A_j = \emptyset
\]

\[
\bigcup_{i=0}^{m} A_i = \{0,1\}^n
\]

\[
\text{for all points } a \in A_i \text{ and } b \in A_j \text{ it happens that } f(a) < f(b) \text{ if } i < j.
\]

**Requirement**

\( A_m \) contains only optimal search points.

Then:

\( p_i \) probability that point in \( A_i \) is mutated to a point in \( A_j \) with \( j > i \).

**Expected time:** \( E(T) \leq \sum_i \frac{1}{p_i} \)

Very simple, yet often powerful method for upper bounds

---

**Coupon collector’s problem: lower bound on time**

What is the probability that the time to collect \( n \) coupons is less than \( n \ln n + O(n) \)?

**Theorem (Coupon collector lower bound on time (Doerr, 2011))**

Let \( T \) be the time for all the \( n \) coupons to be collected. Then for all \( \epsilon > 0 \)

\[
\Pr(T < (1 - \epsilon)(n - 1) \ln n) \leq \exp(-n^\epsilon)
\]

**Corollary**

The expected time for RLS to optimise \( \text{OneMax} \) is \( \Theta(n \ln n) \). Furthermore,

\[
\Pr(T \geq (1 + \epsilon)n \ln n) \leq n^{-\epsilon}
\]

and

\[
\Pr(T < (1 - \epsilon)(n - 1) \ln n) \leq \exp(-n^\epsilon)
\]

What about the \((1+1)\text{-EA}\)?

---

**Artificial Fitness Levels [Droste et al., 2002]**

Let:

\[
p(A_i) \text{ be the probability that a random initial point belongs to level } A_i
\]

\[
a_i \text{ be the probability to leave level } A_i \text{ for } A_j \text{ with } j > i
\]

**Then:**

**Expected time:** \( E(T) \leq \sum_{1 \leq i \leq m} \frac{1}{s_i} \)

\[
E(T) \leq \sum_{1 \leq i \leq m-1} p(A_i) \cdot \left( \frac{1}{s_i} + \cdots + \frac{1}{s_{m-1}} \right) \leq \sum_{1 \leq i \leq m-1} \frac{1}{s_i}
\]

**Inequality 1:** Law of total probability \( E(T) = \sum_i Pr(F_i) \cdot E(T|F_i) \)

**Inequality 2:** Trivial!
The expected runtime of the (1+1)-EA for OneMax is $O(n \ln n)$.

Proof
- The current solution is in level $A_i$ if it has $i$ zeroes (hence $n-i$ ones).
- To reach a higher fitness level it is sufficient to flip a zero into a one and leave the other bits unchanged, which occurs with probability

$$s_i = \frac{i}{n} \left(1 - \frac{1}{n}\right)^{n-1} \geq \frac{i}{en}$$

Then (Artificial Fitness Levels):

$$E(T) \leq \sum_{i=1}^{m-1} s_i^{-1} \leq \sum_{i=1}^{n} \frac{en}{i} \leq e \cdot n \sum_{i=1}^{m-1} \frac{1}{i} \leq e \cdot n \cdot (\ln n + 1) = O(n \ln n)$$

Is the (1+1)-EA quicker than $n \ln n$?

Theorem (Droste, Jansen, Wegener, 2002)

The expected runtime of the (1+1)-EA for OneMax is $\Omega(n \log n)$.

Proof Idea
- At most $n/2$ one-bits are created during initialisation with probability at least $1/2$ (By symmetry of the binomial distribution).
- There is a constant probability that in $en \ln n$ steps one of the $n/2$ remaining zero-bits does not flip.

The Expected runtime is:

$$E[T] = \sum_{i=1}^{n} t \cdot p(t) \geq [(n-1) \log n] \cdot p(t = (n-1) \log n) \geq [(n-1) \log n] \cdot [1/2] \cdot (1 - (2e)^{-1/2}) = \Omega(n \log n)$$

First inequality: law of total probability

The upper bound given by artificial fitness levels is indeed tight!
**Theorem**

The expected runtime of RLS for **LeadingOnes** is $O(n^2)$.

**Proof**
- Let partition $A_i$ contain search points with exactly $i$ leading ones
- To leave level $A_i$, it suffices to flip the zero at position $i + 1$
- $s_i = \frac{1}{n}$ and $s_i^{-1} = n$
- $E(T) \leq \sum_{i=1}^{n-1} s_i^{-1} = \sum_{i=1}^{n} n = O(n^2)$

---

**Theorem**

The expected runtime of the (1+1)-EA for **LeadingOnes** is $O(n^2)$.

**Proof** Left as Exercise.

---

**Theorem**

The expected runtime of the $(\mu+1)$-EA for **LeadingOnes** is $O(\mu \cdot n^2)$.

**Proof** Left as Exercise.

---

**Theorem**

The expected runtime of the $(\mu+1)$-EA for **OneMax** is $O(\mu \cdot n \log n)$.

**Proof** Left as Exercise.

---

D. Sudholt, Tutorial 2011

Let:
- $T_i$ be the expected time for a fraction $\chi(i)$ of the population to be in level $A_i$
- $s_i$ be the probability to leave level $A_i$ for $A_j$ with $j > i$ given $\chi(i)$ in level $A_i$
- Then:

$$E(T) \leq \sum_{i=1}^{m-1} \left( \frac{1}{s_i} + T_i \right)$$
Theorem

The expected runtime of the \((\mu+1)\)-EA for **LeadingOnes** is \(O(\mu n \log n + n^2)\) [Witt, 2006].

Proof

- Let partition \(A_i\) contain search points with exactly \(i\) leading ones.
- To leave level \(A_i\), it suffices to flip the zero at position \(i+1\) of the best individual.
- We set \(\chi(i) = \frac{n}{\ln n}\).
- Given \(j\) copies of the best individual another replica is created with probability
  \(\frac{j}{\mu} \left(1 - \frac{1}{n}\right)^n \geq \frac{1}{2e\mu}\).
- \(T_0 \leq \sum_{j=1}^{\frac{n}{\ln n}} \frac{2\mu}{n} = 2e\mu \ln n\)
  \(s_i \geq \frac{n}{\ln n} \cdot \frac{1}{\mu} = \frac{1}{e\mu \ln n}\) Case 1: \(\mu > \frac{n}{\ln n}\)
  \(s_i \geq \frac{n}{\ln n} \cdot \frac{1}{\mu} \geq \frac{1}{e\mu}\) Case 2: \(\mu \leq \frac{n}{\ln n}\)
- \(E(T) \leq \sum_{i=1}^{n-1} (T_i + s_i) \leq \sum_{i=1}^{n} \left(2e\mu \ln n + (en + e\mu \ln n)\right) = n \cdot \left(2e\mu \ln n + (en + e\mu \ln n)\right) = O(n\mu \ln n + n^2)\)

New population by sampling and mutating \(\lambda\) parents independently:

**Theorem** ([Lehre, GECCO 2011])

If

- \(C_1\): for one offspring \(\text{Prob}(A_i \rightarrow A_{i+1} \cup \cdots \cup A_m) \geq s_i\)
- \(C_2\): for one offspring \(\text{Prob}(A_i \rightarrow A_{i} \cup \cdots \cup A_m) \geq \gamma_0\)
- \(C_3\): selection is sufficiently strong: \(\beta(\gamma, P)/\gamma \geq (1 + \delta)/\gamma_0\)
- \(C_4\): population size sufficiently large: \(\lambda \geq \frac{2(1+\delta)}{2e\beta} \cdot \ln \left(\frac{m}{\min_j \gamma_j}\right)\)

then the expected number of function evaluations is at most

\[O\left(\lambda^2 + \sum_{i=1}^{m-1} \frac{1}{s_i}\right)\].

Lower bounds with fitness levels [Sudholt, 2010]

Let \(u_i \cdot \gamma_{i,j}\) be an upper bound for \(\text{Prob}(A_i \rightarrow A_j)\) and \(\sum_{i=1}^{m} \gamma_{i,j} = 1\).
Assume for all \(j > i\) and \(0 < \chi \leq 1\) that \(\gamma_{i,j} \geq \frac{\chi}{\sum_{k=j}^{m} \gamma_{i,k}}\). Then

\[E(\text{optimization time}) \geq \sum_{i=1}^{m-1} \text{Prob}(A \text{ starts in } A_i) \cdot \frac{1}{u_i} \cdot \frac{\chi}{\sum_{j=i}^{m-1} \frac{1}{u_j}}\]

\(u_i\) := probability to leave level \(A_i\);
\(\gamma_{i,j}\) := probability of jumping from \(A_i\) to \(A_j\).
Artificial Fitness Levels: Conclusions

- It’s a powerful general method to obtain (often) tight upper bounds on the runtime of simple EAs;
- For offspring populations tight bounds can often be achieved with the general method;
- For parent populations takeover times have to be introduced;
- Recent methods have been presented to deal with non-elitism and for lower bounds.

Drift Analysis: Example 1

Friday night dinner at the restaurant.
Peter walks back from the restaurant to the hotel.

- The restaurant is \( n \) meters away from the hotel;
- Peter moves towards the hotel of 1 meter in each step

**Question**
*How many steps does Peter need to reach his hotel?*
\( n \) steps

Drift Analysis: Example 2

Friday night dinner at the restaurant.
Peter walks back from the restaurant to the hotel but had a few drinks.

- The restaurant is \( n \) meters away from the hotel;
- Peter moves towards the hotel of 1 meter in each step with probability 0.6.
- Peter moves away from the hotel of 1 meter in each step with probability 0.4.

**Question**
*How many steps does Peter need to reach his hotel?*
\( 5n \) steps

Let us calculate this through drift analysis.
Drift Analysis (2): Formalisation

- Define the same distance function \( d(x) \) as before to measure the distance from the hotel;
  \[ d(x) = x, \quad x \in \{0, \ldots, n\} \]
  (simply the number of metres from the hotel).
- Estimate the expected "speed" (drift), the expected decrease in distance in one step from the goal;
  \[ d(X_i) - d(X_{i+1}) = \begin{cases} 
  0, & \text{if } X_i = 0, \\
  1, & \text{if } X_i \in \{1, \ldots, n\} \text{ with probability 0.6} \\
  -1, & \text{if } X_i \in \{1, \ldots, n\} \text{ with probability 0.4} 
\end{cases} \]
- The expected decrease in distance (drift) is:
  \[ E[d(X_i) - d(X_{i+1})] = 0.6 \cdot 1 + 0.4 \cdot (-1) = 0.6 - 0.4 = 0.2 \]

**Time**

Then the expected time to reach the hotel (goal) is:
\[ E(T) = \frac{\text{maximum distance}}{\text{drift}} = \frac{n}{0.2} = 5n \]

Drift Analysis for Leading Ones

**Theorem**

The expected time for the (1+1)-EA to optimise LeadingOnes is \( O(n^2) \)

**Proof**

- Let \( d(X_i) = i \) where \( i \) is the number of missing leading ones;
- The negative drift is 0 since if a leading one is removed from the current solution the new point will not be accepted;
- A positive drift (i.e. of length 1) is achieved as long as the first 0 is flipped and the leading ones are remained unchanged:
  \[ E(\Delta^+(t)) = \sum_{k=1}^{n-1} k \cdot (p(\Delta^+(t)) = k) \geq 1 \cdot \frac{1}{n} \cdot (1 - 1/n)^{n-1} \geq 1/(en) \]
- Hence, \( E[\Delta(t) \mid d(X_i)] \geq 1/(en) = \delta \)
- The expected runtime is (i.e. Eq. (6)):
  \[ E(T \mid d(X_0) > 0) \leq \frac{d(X_0)}{\delta} \leq \frac{n}{1/(en)} = e \cdot n^2 = O(n^2) \]

Exercises

**Theorem**

The expected time for RLS to optimise LeadingOnes is \( O(n^2) \)

**Proof** Left as exercise.

**Theorem**

Let \( \lambda \geq en \). Then the expected time for the (1+\lambda)-EA to optimise LeadingOnes is \( O(\lambda n) \)

**Proof** Left as exercise.

**Theorem**

Let \( \lambda < en \). Then the expected time for the (1+\lambda)-EA to optimise LeadingOnes is \( O(n^2) \)

**Proof** Left as exercise.
The distance function is defined as the number of missing leading ones, i.e., let $d(x) = n - i$ where $i$ is the number of leading ones.

**Theorem**

Let $\lambda = n$. Then the expected time for the $(1, \lambda)$-EA to optimise LeadingOnes is $O(n^2)$.

**Proof**

- **Distance**: let $d(x) = n - i$ where $i$ is the number of leading ones;
- **Drift**:

$E[d(X_t) - d(X_{t+1})|d(X_t) = n - i]$

$\geq 1 \cdot \left(1 - \left(1 - \frac{1}{en}\right)^n\right) - n \cdot \left(1 - \left(1 - \frac{1}{n}\right)^n\right)^n$

$c_1 - n \cdot c_2^\delta = \Omega(1)$

Hence,

$E(\text{generations}) \leq \frac{\max \text{ distance}}{\text{drift}} = \frac{n}{\Omega(1)} = O(n)$

and,

$E(T) \leq n \cdot E(\text{generations}) = O(n^2)$
Drift Analysis for OneMax

Let \( g(X_i) = \ln(i + 1) \) where \( i \) is the number of zeroes in the bitstring:
- For \( x \geq 1 \), it holds that \( \ln(1 + 1/x) \geq 1/x - 1/(2x^2) \geq 1/(2x) \);
- The distance decreases as long as a 0 is flipped and the ones remain unchanged:
  \[
  E(\Delta(t)) = E[|d(X_i) - d(X_{i+1})| | d(X_i) = i] \geq 1 \frac{i}{en} \left(1 - \frac{1}{n}\right)^{n-1} \geq \frac{1}{en} \geq \frac{1}{en} = \delta
  \]
- The initial distance is \( d(X_0) \leq \ln(n + 1) \)

The expected runtime is (i.e. Eq. (6)):
\[
E(T | d(X_0) > 0) \leq E[|d(X_0)|] \leq \frac{n/2}{1/\ln n} = e/2 \cdot n^2 = O(n^2)
\]
We need a different distance function!

Multiplicative Drift Theorem

Theorem (Multiplicative Drift, [Doerr et al., 2010])

Let \( \{X_t\}_{t \in \mathbb{N}_0} \) be random variables describing a Markov process over a finite state space \( S \subseteq \mathbb{R} \). Let \( T \) be the random variable that denotes the earliest point in time \( t \in \mathbb{N}_0 \) such that \( X_t = 0 \).
- If there exist \( \delta, c_{\min}, c_{\max} > 0 \) such that
  - \( E[X_t - X_{t+1} | X_t] \geq \delta X_t \) and
  - \( c_{\min} \leq X_t \leq c_{\max} \)
  for all \( t < T \), then
  \[
  E[T] \leq \frac{2}{\delta} \cdot \ln \left(1 + \frac{c_{\max}}{c_{\min}}\right)
  \]

Drift Analysis for OneMax

The expected time for the (1+1)-EA to optimise OneMax is \( O(n \ln n) \)

Proof
- Distance: let \( X_t \) be the number of zeroes at time step \( t \):
  - \( E[X_{t+1} | X_t] = X_t - 1 \cdot \frac{X_t}{en} = X_t \cdot \left(1 - \frac{1}{en}\right) \)
  - \( E[X_t - X_{t+1} | X_t] \leq X_t - X_t \cdot \left(1 - \frac{1}{en}\right) = \frac{X_t}{en} \cdot \delta = \frac{1}{en} \)
- \( \delta = c_{\min} \leq X_t \leq c_{\max} = n \)

Hence,
\[
E[T] \leq \frac{2}{\delta} \cdot \ln \left(1 + \frac{c_{\max}}{c_{\min}}\right) = 2en \ln(1 + n) = O(n \ln n)
\]
The expected time for RLS to optimise OneMax is $O(n \log n)$.

Proof Left as exercise.

Let $\lambda \geq e_n$. Then the expected time for the $(1+\lambda)$-EA to optimise OneMax is $O(\lambda n)$.

Proof Left as exercise.

Let $\lambda < e_n$. Then the expected time for the $(1+\lambda)$-EA to optimise OneMax is $O(n \log n)$.

Proof Left as exercise.

The expected time for the RLS to optimise OneMax is $O(n \log n)$.

Proof Left as exercise.

Friday night dinner at the restaurant.
Peter walks back from the restaurant to the hotel but had too many drinks.

- The restaurant is $n$ meters away from the hotel;
- Peter moves towards the hotel of 1 meter in each step with probability 0.4.
- Peter moves away from the hotel of 1 meter in each step with probability 0.6.

Question
How many steps does Peter need to reach his hotel?

at least $2^{cn}$ steps with overwhelming probability (exponential time)
We need Negative-Drift Analysis.

Define the same distance function $d(x) = x, x \in \{0, \ldots, n\}$ (metres from the hotel) ($b=n-1$, $a=1$).

Estimate the increase in distance from the goal (negative drift);

$d(X_t) - d(X_{t+1}) = \begin{cases} 0, & \text{if } X_t = 0, \\ 1, & \text{if } X_t \in \{1, \ldots, n\} \text{ with probability 0.6} \\ -1, & \text{if } X_t \in \{1, \ldots, n\} \text{ with probability 0.4} \end{cases}$

The expected increase in distance (negative drift) is: (Condition 1)

$E[d(X_t) - d(X_{t+1})] = 0.6 \cdot 1 + 0.4 \cdot (-1) = 0.6 - 0.4 = 0.2$

Probability of jumps (i.e. $\text{Prob}(\Delta_t(i) = -j) \leq \frac{1}{(1+r)^j}$) (set $\delta = r = 1$) (Condition 2):

$\text{Prob}(\Delta_t(i) = -j) = \begin{cases} 0 < (1/2)^{j-1}, & \text{if } j > 1, \\ 0.6 < (1/2)^0 = 1, & \text{if } j = 1 \end{cases}$

Then the expected time to reach the hotel (goal) is:

$Pr(T \leq 2^{(b-a)}) = Pr(T \leq 2^{(n-2)}) = 2^{-\Omega(n)}$
**Theorem (Oliveto,Witt, Algorithmica 2011)**

Let \( \eta > 0 \) be constant. Then there is a constant \( c > 0 \) such that with probability \( 1 - 2^{-\Omega(n)} \) the (1+1)-EA on Needle creates only search points with at most \( n/2 + \eta n \) ones in \( 2^c n \) steps.

**Proof Idea**
- By Chernoff bounds the probability that the initial bit string has less than \( n/2 - \gamma n \) zeroes is \( e^{-\Omega(n)} \).
- we set \( b := n/2 - \gamma n \) and \( a := n/2 - 2\gamma n \) where \( \gamma := \eta/2 \);

**Proof of Condition 1**
\[
E(\Delta(i)) = \frac{n-i}{n} - i \frac{i}{n} = \frac{n-2i}{n} \geq 2\gamma = \epsilon
\]

**Proof of Condition 2**
\[
\text{Prob}(|\Delta(i)| \leq -j) \leq \left( \begin{array}{c} n \\ j \end{array} \right) \left( \frac{1}{n} \right)^j \leq \frac{n^j}{j!} \left( \frac{1}{n} \right)^j \frac{1}{j!} \leq \left( \frac{1}{2} \right)^{j-1}
\]

This proves Condition 2 by setting \( \delta = \epsilon = 1 \).

**Exercise: Trap Functions**

\[
\text{TRAP}(x) = \begin{cases} 
  n + 1 & \text{if } x = 0^n \\
  \text{ONE MAX}(x) & \text{otherwise.}
\end{cases}
\]

**Theorem**

With overwhelming probability at least \( 1 - 2^{-\Omega(n)} \) the (1+1)-EA requires \( 2^{\Omega(n)} \) steps to optimise TRAP.

**Proof** Left as exercise.

---

**Overview**
- Additive Drift Analysis (upper and lower bounds);
- Multiplicative Drift Analysis;
- Simplified Negative-Drift Theorem;

**Advanced Lower bound Drift Techniques**
- Drift Analysis for Stochastic Populations (mutation) [Lehre, 2010];
- Simplified Drift Theorem combined with bandwidth analysis (mutation + crossover stochastic populations = GAs) [Oliveto and Witt, 2012];

---

**Other Techniques (Not covered)**
- Family Trees [Witt, 2006]
- Gambler’s Ruin & Martingales [Jansen and Wegener, 2001]
State of the Art in Computational Complexity of RSHs

<table>
<thead>
<tr>
<th>Problem</th>
<th>(1+1) EA</th>
<th>(1+1) EA</th>
<th>(1+1) EA</th>
<th>(1+1) EA</th>
<th>(1+1) EA</th>
</tr>
</thead>
<tbody>
<tr>
<td>OneMax</td>
<td>O(n log n)</td>
<td>O(n log n)</td>
<td>O(n log n)</td>
<td>O(n log n)</td>
<td>O(n log n)</td>
</tr>
<tr>
<td>Linear Functions</td>
<td>O(n log n)</td>
<td>O(n log n)</td>
<td>O(n log n)</td>
<td>O(n log n)</td>
<td>O(n log n)</td>
</tr>
<tr>
<td>Max. Matching</td>
<td>2Ω(√n)</td>
<td>PRAS</td>
<td>2Ω(√n)</td>
<td>PRAS</td>
<td>2Ω(√n)</td>
</tr>
<tr>
<td>Satisfiability</td>
<td>3Ω(√n)</td>
<td>PRAS</td>
<td>3Ω(√n)</td>
<td>PRAS</td>
<td>3Ω(√n)</td>
</tr>
<tr>
<td>3Sum Shortest Path</td>
<td>O(n log n)</td>
<td>O(n log n)</td>
<td>O(n log n)</td>
<td>O(n log n)</td>
<td>O(n log n)</td>
</tr>
<tr>
<td>MST</td>
<td>O(n^2 log n)</td>
<td>O(n^2 log n)</td>
<td>O(n^2 log n)</td>
<td>O(n^2 log n)</td>
<td>O(n^2 log n)</td>
</tr>
<tr>
<td>Max. Clique (rand. planar)</td>
<td>Ω(n^5/2)</td>
<td>O(n log n)</td>
<td>Ω(n^2 log n)</td>
<td>O(n^2 log n)</td>
<td>Ω(n^2 log n)</td>
</tr>
<tr>
<td>Partition</td>
<td>O(n^2 log n)</td>
<td>O(n^2 log n)</td>
<td>O(n^2 log n)</td>
<td>O(n^2 log n)</td>
<td>O(n^2 log n)</td>
</tr>
<tr>
<td>Vertex Cover</td>
<td>O(n log n)</td>
<td>O(n log n)</td>
<td>O(n log n)</td>
<td>O(n log n)</td>
<td>O(n log n)</td>
</tr>
<tr>
<td>Set Cover</td>
<td>O(√n)</td>
<td>PRAS</td>
<td>O(√n)</td>
<td>PRAS</td>
<td>O(√n)</td>
</tr>
<tr>
<td>Intersection of</td>
<td>O(√n)</td>
<td>PRAS</td>
<td>O(√n)</td>
<td>PRAS</td>
<td>O(√n)</td>
</tr>
<tr>
<td>p ≥ 3 matchings</td>
<td>O(√n)</td>
<td>PRAS</td>
<td>O(√n)</td>
<td>PRAS</td>
<td>O(√n)</td>
</tr>
<tr>
<td>ISOFIMA conf.</td>
<td>O(√n)</td>
<td>PRAS</td>
<td>O(√n)</td>
<td>PRAS</td>
<td>O(√n)</td>
</tr>
</tbody>
</table>

See [Oliveto et al., 2007] for an overview.

P. K. Lehre, 2008

Further Reading


