Runtime Analysis of Evolutionary Algorithms: Basic Introduction[1]

Per Kristian Lehre
University of Nottingham
Nottingham NG8 1BB, UK
PerKristian.Lehre@nottingham.ac.uk

Pietro S. Oliveto
University of Birmingham
Birmingham B15 2TT, UK
P.S.Oliveto@cs.bham.ac.uk

The University of Nottingham
UNITED KINGDOM · CHINA · MALAYSIA

UNIVERSITY OF BIRMINGHAM

GECCO 2013

GECCO'13 Companion, July 6–10, 2013, Amsterdam, The Netherlands.
ACM 978-1-4503-1964-5/13/07.

[1] For the latest version of these slides, see http://www.cs.nott.ac.uk/~pkl/gecco2013.

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
  1. understand theoretically the behaviour of EAs on different problems
  2. perform time complexity analysis of simple EAs on common toy problems
  3. read and understand research papers on the computational complexity of EAs
  4. have the basic skills to start independent research in the area
  5. follow the other theory tutorials later on today

Evolutionary Algorithms and Computer Science

Goals of design and analysis of algorithms

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

For Evolutionary Algorithms (General purpose)

1. convergence
   "Does the EA find the solution in finite time?"
2. 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 [Vose, 1999].
- *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];
- No Free Lunch [Wolpert and Macready, 1997]
  - Over all functions...
- 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!

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)*

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])

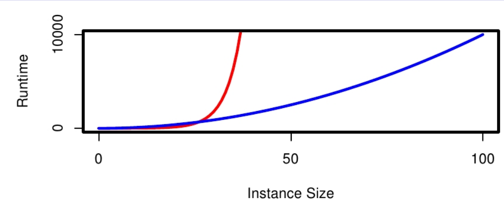
1. *There is a positive probability to reach any point in the search space from any other point*
2. *The best found solution is never removed from the population (elitism)*

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

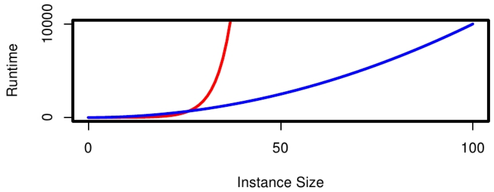
In practice, is it interesting that an algorithm converges to the optimum?

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

Computational Complexity of EAs



Computational Complexity of EAs

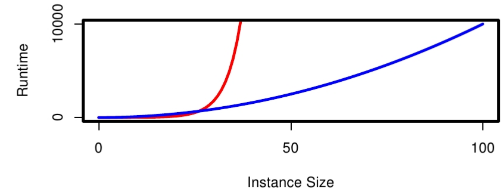


Generally means predicting the resources the algorithm requires:

- 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

Computational Complexity of EAs



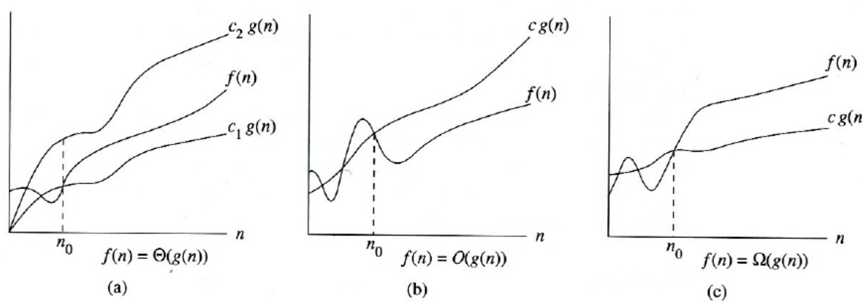
However (EAs):

1. In practice the time for a fitness function evaluation is much higher than the rest;
2. EAs are **randomised algorithms**
   - They do not perform the same operations even if the input is the same!
   - They do not output the same result if run twice!

Hence, the runtime of an EA is a random variable T_f.
We are interested in:

1. Estimating $E(T_f)$, the expected runtime of the EA for f;
2. Estimating $p(T_f \le t)$, the success probability of the EA in t steps for f.

Asymptotic notation



$$f(n) \in O(g(n)) \iff \exists \text{ constants } c, n_0 > 0 \text{ st. } 0 \le f(n) \le cg(n) \quad \forall n \ge n_0$$
$$f(n) \in \Omega(g(n)) \iff \exists \text{ constants } c, n_0 > 0 \text{ st. } 0 \le cg(n) \le f(n) \quad \forall n \ge n_0$$
$$f(n) \in \Theta(g(n)) \iff f(n) \in O(g(n)) \text{ and } f(n) \in \Omega(g(n))$$
$$f(n) \in o(g(n)) \iff \lim_{n\to\infty} \frac{f(n)}{g(n)} = 0$$

Goals

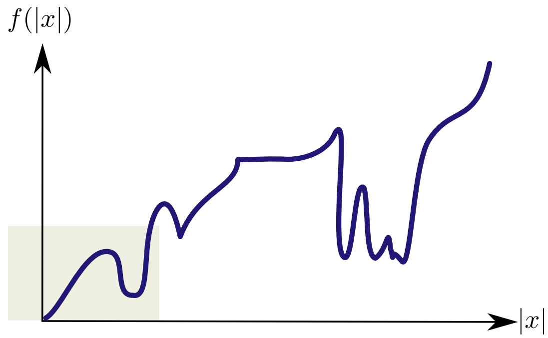
Understand how the runtime depends on:

- parameters of the problem
- parameters of the algorithm

In order to:

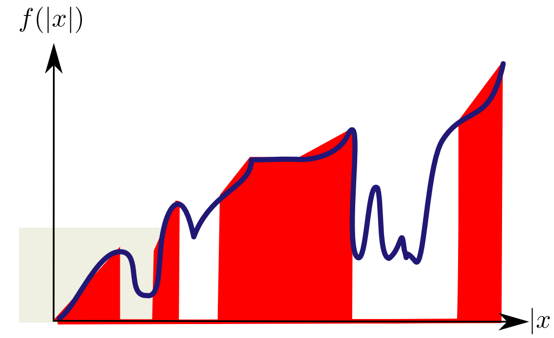
- explain the success or the failure of these methods in practical applications,
- understand which problems are optimized (or approximated) efficiently by a given algorithm and which are not
- guide the choice of the best algorithm for the problem at hand,
- determine the optimal parameter settings,
- aid the algorithm design.

Running Example - Functions of Unitation



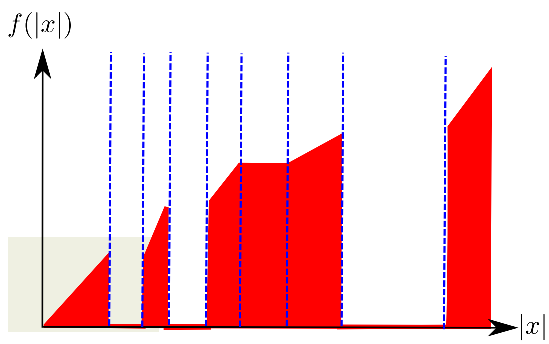
$$g(x) = f\left(\sum_{i=1}^{n} x_i\right) \quad \text{where} \quad f : \mathbb{R} \to \mathbb{R}$$

Running Example - Functions of Unitation



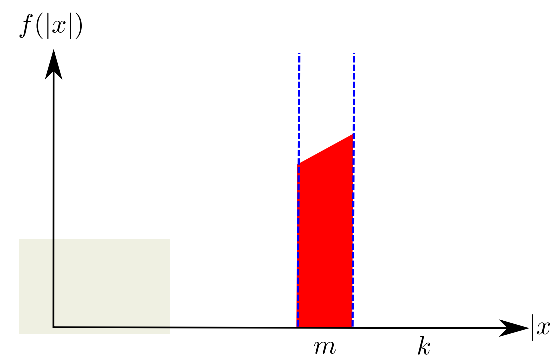
$$g(x) = f\left(\sum_{i=1}^{n} x_i\right) \quad \text{where} \quad f : \mathbb{R} \to \mathbb{R}$$

Running Example - Functions of Unitation



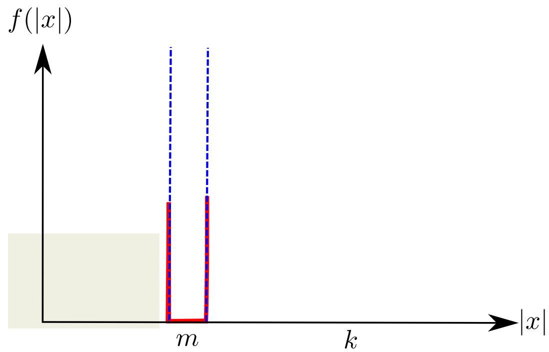
$$f(x) = \sum_{i=1}^{r} f_i(x)$$

Linear Unitation Block



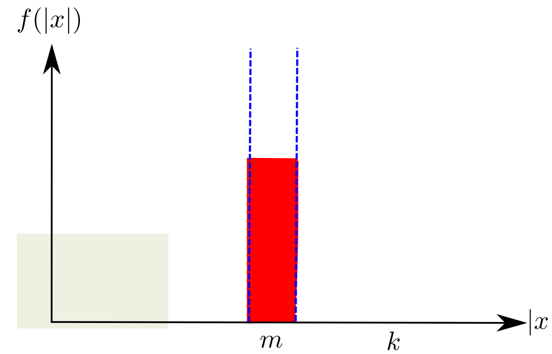
$$f(|x|) = \begin{cases} a|x| + b & \text{if } k < n - |x| \leq k + m \\ 0 & \text{otherwise.} \end{cases}$$

Toy Problem Framework - Gap



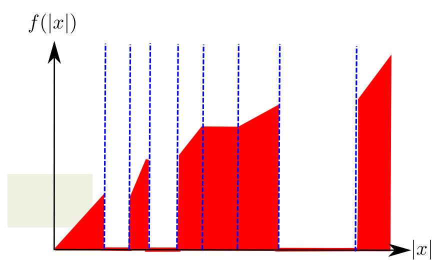
$$f(|x|) = \begin{cases} a & \text{if } n - |x| = k + m \\ 0 & \text{otherwise.} \end{cases}$$

Toy Problem Framework - Plateau



$$f(|x|) = \begin{cases} a & \text{if } k < n - |x| \le k + m \\ 0 & \text{otherwise.} \end{cases}$$

Upper bound on the total runtime



$$f(x) = \sum_{i=1}^{r} f_i(x)$$

Assumptions

r sub-functions $f_1, f_2, \dots, f_r$
T_i time to optimise sub-function f_i
the evolutionary algorithm is elitist

By linearity of expectation, an upper bound on the expected runtime is

$$\mathbb{E}[T] \le \mathbb{E}\left[\sum_{i=1}^{r} T_i\right] = \sum_{i=1}^{r} \mathbb{E}[T_i].$$

Evolutionary Algorithms

$(\mu+\lambda)$ EA

Initialise P_0 with μ individuals chosen uniformly a random from $\{0,1\}^n$
for $t = 0, 1, 2, \dots$ until stopping condition met **do**
Create λ new individuals by
- choosing $x \in P_t$ uniformly at random
- flipping each bit in x with probability p

Create the new population P_{t+1} by
choosing the best μ individuals out of $\mu + \lambda$.
end for

Evolutionary Algorithms

(μ+λ) EA

Initialise P_0 with μ individuals chosen uniformly a random from $\{0,1\}^n$
for $t = 0, 1, 2, \ldots$ until stopping condition met **do**
Create λ new individuals by
- choosing $x \in P_t$ uniformly at random
- flipping each bit in x with probability p

Create the new population P_{t+1} by
choosing the best μ individuals out of $\mu + \lambda$.
end for

- If $\mu = \lambda = 1$, then we get the (1+1) EA;

Introduction Motivation Evolutionary Algorithms Tail Inequalities Artificial Fitness Levels Drift Analysis Conclusions
General EAs

Evolutionary Algorithms

(μ+λ) EA

Initialise P_0 with μ individuals chosen uniformly a random from $\{0,1\}^n$
for $t = 0, 1, 2, \ldots$ until stopping condition met **do**
Create λ new individuals by
- choosing $x \in P_t$ uniformly at random
- flipping each bit in x with probability p

Create the new population P_{t+1} by
choosing the best μ individuals out of $\mu + \lambda$.
end for

- If $\mu = \lambda = 1$, then we get the (1+1) EA;
- $p = 1/n$ is generally considered a good parameter setting [Bäck, 1993, Droste et al., 1998];

Introduction Motivation Evolutionary Algorithms Tail Inequalities Artificial Fitness Levels Drift Analysis Conclusions
General EAs

Evolutionary Algorithms

(μ+λ) EA

Initialise P_0 with μ individuals chosen uniformly a random from $\{0,1\}^n$
for $t = 0, 1, 2, \ldots$ until stopping condition met **do**
Create λ new individuals by
- choosing $x \in P_t$ uniformly at random
- flipping each bit in x with probability p

Create the new population P_{t+1} by
choosing the best μ individuals out of $\mu + \lambda$.
end for

- If $\mu = \lambda = 1$, then we get the (1+1) EA;
- $p = 1/n$ is generally considered a good parameter setting [Bäck, 1993, Droste et al., 1998];
- By introducing stochastic selection and crossover we obtain a Genetic Algorithm (GA)

(1+1) Evolutionary Algorithm

(1+1) EA

Initialise x uniformly at random from $\{0,1\}^n$.
repeat
Create x' by flipping each bit in x with $p = 1/n$.
if $f(x') \geq f(x)$ **then**
$x \leftarrow x'$.
end if
until stopping condition met.

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

How does it work?

(1+1) Evolutionary Algorithm

(1+1) EA

Initialise x uniformly at random from $\{0,1\}^n$.
repeat
  Create x' by flipping each bit in x with $p = 1/n$.
  if $f(x') \geq f(x)$ **then**
    $x \leftarrow x'$.
  end if
until stopping condition met.

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?

Introduction Motivation Evolutionary Algorithms Tail Inequalities Artificial Fitness Levels Drift Analysis Conclusions
(1+1) EA and RLS

(1+1) Evolutionary Algorithm

(1+1) EA

Initialise x uniformly at random from $\{0,1\}^n$.
repeat
  Create x' by flipping each bit in x with $p = 1/n$.
  if $f(x') \geq f(x)$ **then**
    $x \leftarrow x'$.
  end if
until stopping condition met.

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?

$$E[X] = E[X_1 + X_2 + \cdots + X_n] = E[X_1] + E[X_2] + \cdots + E[X_n] =$$

Introduction Motivation Evolutionary Algorithms Tail Inequalities Artificial Fitness Levels Drift Analysis Conclusions
(1+1) EA and RLS

(1+1) Evolutionary Algorithm

(1+1) EA

Initialise x uniformly at random from $\{0,1\}^n$.
repeat
  Create x' by flipping each bit in x with $p = 1/n$.
  if $f(x') \geq f(x)$ **then**
    $x \leftarrow x'$.
  end if
until stopping condition met.

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?

$$E[X] = E[X_1 + X_2 + \cdots + X_n] = E[X_1] + E[X_2] + \cdots + E[X_n] =$$

$$(E[X_i] = 1 \cdot 1/n + 0 \cdot (1 - 1/n) = 1 \cdot 1/n = 1/n \quad E(X) = np)$$

Introduction Motivation Evolutionary Algorithms Tail Inequalities Artificial Fitness Levels Drift Analysis Conclusions
(1+1) EA and RLS

(1+1) Evolutionary Algorithm

(1+1) EA

Initialise x uniformly at random from $\{0,1\}^n$.
repeat
  Create x' by flipping each bit in x with $p = 1/n$.
  if $f(x') \geq f(x)$ **then**
    $x \leftarrow x'$.
  end if
until stopping condition met.

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?

$$E[X] = E[X_1 + X_2 + \cdots + X_n] = E[X_1] + E[X_2] + \cdots + E[X_n] =$$

$$(E[X_i] = 1 \cdot 1/n + 0 \cdot (1 - 1/n) = 1 \cdot 1/n = 1/n \quad E(X) = np)$$

$$= \sum_{i=1}^{n} 1 \cdot 1/n = n/n = 1$$

(1+1) EA: 2

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

(1+1) EA: 2

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 flipping exactly one bit?

(1+1) EA: 2

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 flipping exactly one bit?

$$\Pr(X = 1) = \binom{n}{1}\left(\frac{1}{n}\right)\left(1 - \frac{1}{n}\right)^{n-1} = \left(1 - \frac{1}{n}\right)^{n-1} \geq 1/e \approx 0.37$$

(1+1) EA: 2

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 flipping exactly one bit?

$$\Pr(X = 1) = \binom{n}{1}\left(\frac{1}{n}\right)\left(1 - \frac{1}{n}\right)^{n-1} = \left(1 - \frac{1}{n}\right)^{n-1} \geq 1/e \approx 0.37$$

Is flipping two bits more likely than flipping none?

(1+1) EA: 2

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 flipping exactly one bit?

$$\Pr(X = 1) = \binom{n}{1}\left(\frac{1}{n}\right)\left(1 - \frac{1}{n}\right)^{n-1} = \left(1 - \frac{1}{n}\right)^{n-1} \geq 1/e \approx 0.37$$

Is flipping two bits more likely than flipping none?

$$\begin{aligned}\Pr(X = 2) &= \binom{n}{2}\left(\frac{1}{n}\right)^2\left(1 - \frac{1}{n}\right)^{n-2} \\ &= \frac{n(n-1)}{2}\left(\frac{1}{n}\right)^2\left(1 - \frac{1}{n}\right)^{n-2} \\ &= \frac{1}{2}\left(1 - \frac{1}{n}\right)^{n-1} \approx 1/(2e)\end{aligned}$$

(1+1) EA: 2

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 flipping exactly one bit?

$$\Pr(X = 1) = \binom{n}{1}\left(\frac{1}{n}\right)\left(1 - \frac{1}{n}\right)^{n-1} = \left(1 - \frac{1}{n}\right)^{n-1} \geq 1/e \approx 0.37$$

Is flipping two bits more likely than flipping none?

$$\begin{aligned}\Pr(X = 2) &= \binom{n}{2}\left(\frac{1}{n}\right)^2\left(1 - \frac{1}{n}\right)^{n-2} \\ &= \frac{n(n-1)}{2}\left(\frac{1}{n}\right)^2\left(1 - \frac{1}{n}\right)^{n-2} \\ &= \frac{1}{2}\left(1 - \frac{1}{n}\right)^{n-1} \approx 1/(2e)\end{aligned}$$

While

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

(1+1) EA: 3

What is the probability of flipping j arbitrary bits?

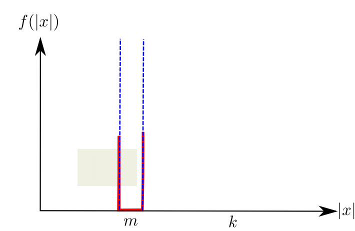
$$\binom{n}{j}\left(\frac{1}{n}\right)^j\left(1 - \frac{1}{n}\right)^{n-j} \leq \binom{n}{j}\left(\frac{1}{n}\right)^j \leq \left(\frac{n^j}{j!}\right)\left(\frac{1}{n}\right)^j = \frac{1}{j!} \leq \left(\frac{1}{2}\right)^{j-1}$$

What is the probability of flipping $j \geq 1$ bits, conditional on flipping at least j bits?

$$\frac{\binom{n}{j}\left(\frac{1}{n}\right)^j\left(1 - \frac{1}{n}\right)^{n-j}}{\binom{n}{j}\left(\frac{1}{n}\right)^j} \geq \left(1 - \frac{1}{n}\right)^{n-1} \geq 1/e$$

Gap block: upper and lower bounds

$$f(|x|) = \begin{cases} a & \text{if } n - |x| = k + m \\ 0 & \text{otherwise.} \end{cases}$$



The probability p of optimising a gap block of length m at position k is

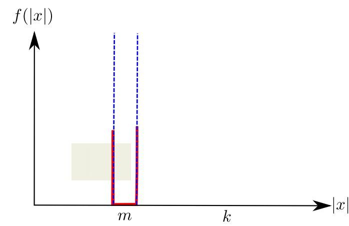
$$\binom{m+k}{m}\left(\frac{1}{n}\right)^m \frac{1}{e} \leq p \leq \binom{m+k}{m}\left(\frac{1}{n}\right)^m$$

The expected time to optimise the gap block is

$$\binom{m+k}{m}^{-1} n^m \leq \mathbb{E}[T] \leq e n^m \binom{m+k}{m}^{-1}$$

Gap block: upper and lower bounds

$$f(|x|) = \begin{cases} a & \text{if } n - |x| = k + m \\ 0 & \text{otherwise.} \end{cases}$$



The probability p of optimising a gap block of length m at position k is

$$\left(\frac{m+k}{nm}\right)^m \frac{1}{e} \le \binom{m+k}{m} \left(\frac{1}{n}\right)^m \frac{1}{e} \le p \le \binom{m+k}{m} \left(\frac{1}{n}\right)^m \le \left(\frac{(m+k)e}{nm}\right)^m$$

The expected time to optimise the gap block is

$$\left(\frac{nm}{(m+k)e}\right)^m \le \binom{m+k}{m}^{-1} n^m \le \mathbb{E}[T] \le en^m \binom{m+k}{m}^{-1} \le e\left(\frac{nm}{(m+k)e}\right)^m$$

Initialisation

How many one-bits in expectation after initialisation?

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?

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?

$$Pr(X = n/2) = \binom{n}{n/2} \frac{1}{n^{n/2}} \left(1 - \frac{1}{n}\right)^{n/2} \left(n = 100, Pr(X = 50) \approx 0.0796\right)$$

Tail Inequalities help us to deal with these kind of problems.

Tail Inequalities

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

Tail inequalities:

- Estimate the probability that X deviates from the expectation by a defined amount δ;
- For many intermediate results, expected values are useless;
- May turn expected runtimes into bounds that hold with *overwhelming* probability.

Markov Inequality

The fundamental inequality from which many others are derived.

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 $t \in R^+$,

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

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 $t \in R^+$,

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

- $E[X] = 1$; then: $Pr[X \geq 2] \leq \frac{E[X]}{2} \leq \frac{1}{2}$ **(Number of bits that flip)**

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 $t \in R^+$,

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

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

Introduction Motivation Evolutionary Algorithms Tail Inequalities Artificial Fitness Levels Drift Analysis Conclusions
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 $t \in R^+$,

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

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

Chernoff Bounds

Let $X_1, X_2, \dots X_n$ 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)

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

Introduction Motivation Evolutionary Algorithms Tail Inequalities Artificial Fitness Levels Drift Analysis Conclusions
Chernoff bounds

Chernoff Bounds

Let $X_1, X_2, \dots X_n$ 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)

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

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

Chernoff Bounds

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)

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

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

- $p_i = 1/2$, $E[X] = n \cdot 1/2 = n/2$,

Chernoff Bounds

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)

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

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:
- $Pr[X > (2/3)n] \leq \left(\frac{e^{1/3}}{(4/3)^{4/3}}\right)^{n/2} = c^{-n/2}$

Chernoff Bound Simple Application

Bitstring of length $n = 100$

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

Chernoff Bound Simple Application

Bitstring of length $n = 100$

$Pr(X_i) = 1/2$ and $E(X) = np = 100/2 = 50$.
What is the probability to have at least 75 1-bits?

Chernoff Bound Simple Application

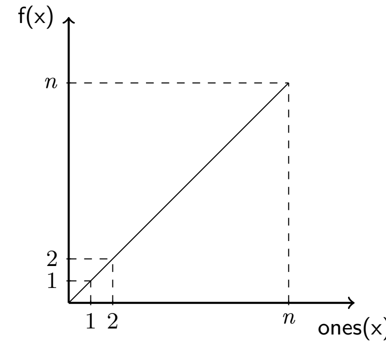
Bitstring of length $n = 100$

$Pr(X_i) = 1/2$ and $E(X) = np = 100/2 = 50$.
What is the probability to have at least 75 1-bits?

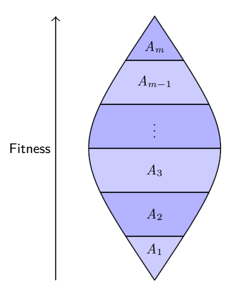
- Markov: $Pr(X \geq 75) \leq \frac{50}{75} = \frac{2}{3}$
- Chernoff: $Pr(X \geq (1+1/2)50) \leq \left(\frac{\sqrt{e}}{(3/2)^{3/2}}\right)^{50} < 0.0045$
- Truth: $Pr(X \geq 75) = \sum_{i=75}^{100} \binom{100}{i} 2^{-100} < 0.000000282$

OneMax

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



Fitness-based Partitions



Definition

A tuple $(A_1, A_2, \ldots, A_m)$ is an f-**based partition** of $f : \mathcal{X} \to \mathbb{R}$ if

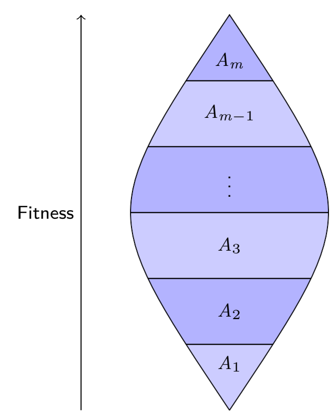
1. $A_1 \cup A_2 \cup \cdots \cup A_m = \mathcal{X}$
2. $A_i \cap A_j = \emptyset$ for $i \neq j$
3. $f(A_1) < f(A_2) < \cdots < f(A_m)$
4. $f(A_m) = \max_x f(x)$

Example

Partition of $\textsc{LeadingOnes}(x) = \sum_{i=1}^{n} \prod_{j=1}^{i} x_j$ into $n+1$ levels

$$A_j := \{x \in \{0,1\}^n \mid x_1 = x_2 = \cdots = x_{j-1} = 1 \wedge x_j = 0\}$$

Artificial Fitness Levels - Upper bounds



s_i : prob. of starting in A_i
u_i : prob. of jumping from A_i to any A_j, $i < j$.
T_i : Time to jump from A_i to any A_j, $i < j$.

Expected runtime

$$\begin{aligned}\mathbb{E}[T] &\leq \sum_{i=1}^{m-1} s_i \mathbb{E}\left[\sum_{j=i}^{m-1} T_j\right] \\ &= \sum_{i=1}^{m-1} s_i \sum_{j=i}^{m-1} \mathbb{E}[T_j] \\ &= \sum_{i=1}^{m-1} s_i \sum_{j=i}^{m-1} 1/u_j \leq \sum_{j=i}^{m-1} 1/u_j.\end{aligned}$$

(1+1)-EA for OneMax

Theorem

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

Proof

(1+1)-EA for OneMax

Theorem

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)

(1+1)-EA for OneMax

Theorem

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

(1+1)-EA for OneMax

Theorem

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
- The probability is $s_i \geq i \cdot \frac{1}{n}\left(1 - \frac{1}{n}\right)^{n-1} \geq \frac{i}{en}$

(1+1)-EA for OneMax

Theorem

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
- The probability is $s_i \geq i \cdot \frac{1}{n}\left(1-\frac{1}{n}\right)^{n-1} \geq \frac{i}{en}$
- Hence, $\frac{1}{s_i} \leq \frac{en}{i}$

(1+1)-EA for OneMax

Theorem

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
- The probability is $s_i \geq i \cdot \frac{1}{n}\left(1-\frac{1}{n}\right)^{n-1} \geq \frac{i}{en}$
- Hence, $\frac{1}{s_i} \leq \frac{en}{i}$
- Then (Artificial Fitness Levels):

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

Linear Unitation Block: Upper bound

Theorem

The expected runtime of the (1+1)-EA for OneMax *is* $O(n \ln((m+k)/k))$.

Proof

Linear Unitation Block: Upper bound

Theorem

The expected runtime of the (1+1)-EA for OneMax *is* $O(n \ln((m+k)/k))$.

Proof

- The probability is $s_i \geq i \cdot \frac{1}{n}\left(1-\frac{1}{n}\right)^{n-1} \geq \left(\frac{i}{en}\right)$

Linear Unitation Block: Upper bound

Theorem

The expected runtime of the (1+1)-EA for OneMax *is* $O(n \ln((m+k)/k))$.

Proof

- The probability is $s_i \geq i \cdot \frac{1}{n}\left(1-\frac{1}{n}\right)^{n-1} \geq \left(\frac{i}{en}\right)$
- Hence, $\left(\frac{1}{s_i}\right) \leq \left(\frac{en}{i}\right)$

Linear Unitation Block: Upper bound

Theorem

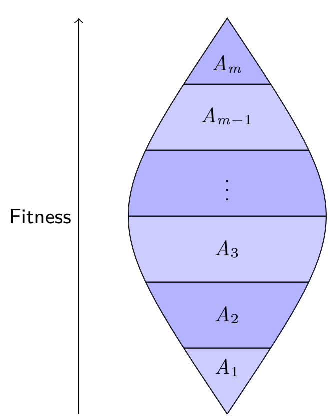
The expected runtime of the (1+1)-EA for OneMax *is* $O(n \ln((m+k)/k))$.

Proof

- The probability is $s_i \geq i \cdot \frac{1}{n}\left(1-\frac{1}{n}\right)^{n-1} \geq \left(\frac{i}{en}\right)$
- Hence, $\left(\frac{1}{s_i}\right) \leq \left(\frac{en}{i}\right)$
- Then (Artificial Fitness Levels):

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

Artificial Fitness Levels - Lower bounds[2]



Theorem ([Sudholt, 2010])

Let

s_i : *prob. of starting in* A_i

u_i : *prob. of leaving* A_i, *and*

p_{ij} : *prob. of jumping from* A_i *to* A_j.

If there exists a $\chi \in [0,1)$ *st. for* $i < j$

$$p_{ij} \geq \chi \sum_{k=j}^{m-1} p_{ik},$$

then

$$\mathbb{E}[T] \geq \chi \sum_{i=1}^{m-1} s_i \sum_{j=i}^{m-1} \frac{1}{u_j}.$$

[2] A different version of the theorem is presented.

(1+1) EA lower bound for OneMax

Fitness level $A_i := \{x : |x| = i\}$

$$x = \overbrace{1111111111111111111111111}^{i}\overbrace{000000000000000000000}^{n-i} \in A_i$$

Probability p_{ij} of jumping to level $j > i$ and beyond

$$p_{ij} = \binom{n-i}{j-i}\left(\frac{1}{n}\right)^{j-i}\left(1-\frac{1}{n}\right)^{n-(j-i)}$$

$$\sum_{k=j}^{n-1} p_{ik} \leq \binom{n-i}{j-i}\left(\frac{1}{n}\right)^{j-i}$$

Hence, for $\chi = 1/e$

$$p_{ij} \geq \left(1-\frac{1}{n}\right)^{n-(j-i)} \sum_{k=j}^{n-1} p_{ik} \geq \chi \sum_{k=j}^{n-1} p_{ik}$$

(1+1) EA lower bound for OneMax

Theorem

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

Probability u_i of any improvement

$$u_i \leq \frac{n-i}{n}$$

Assuming that $s_0 = 1$, we get

$$\mathbb{E}[T] \geq \left(\frac{1}{e}\right) \sum_{i=0}^{n-1} \frac{1}{u_i}$$

$$\geq \left(\frac{1}{e}\right) \sum_{i=0}^{n-1} \frac{n}{n-i} = \left(\frac{n}{e}\right) \sum_{i=1}^{n} \frac{1}{i}$$

Onemax Block:Lower Bound

For $0 \leq i \leq m$, define $A_i := \{x : n - |x| = k + m - i\}$. Note that

$$p_{ij} = \binom{k+m-i}{j-i} \left(\frac{1}{n}\right)^{j-i} \left(1 - \frac{1}{n}\right)^{n-(j-i)}$$

$$\sum_{k=j}^{m-1} p_{ik} \leq \binom{k+m-i}{j-i} \left(\frac{1}{n}\right)^{j-i}$$

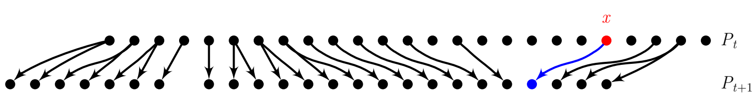
Therefore,

$$p_{ij} \geq \left(1 - \frac{1}{n}\right)^{n-(j-i)} \sum_{k=j}^{m-1} p_{ik} \geq \left(\frac{1}{e}\right) \sum_{k=j}^{m-1} p_{ik}$$

and assuming that $s_0 = 1$, we get

$$\mathbb{E}[T] \geq \left(\frac{1}{e}\right) \sum_{i=0}^{m-1} \frac{1}{u_i} \geq \left(\frac{1}{e}\right) \sum_{i=0}^{m-1} \frac{n}{m+k-i} = \left(\frac{n}{e}\right) \left(\sum_{i=1}^{m+k} \frac{1}{i} - \sum_{i=1}^{k} \frac{1}{i} \right)$$

Advanced: Fitness levels for non-elitist populations

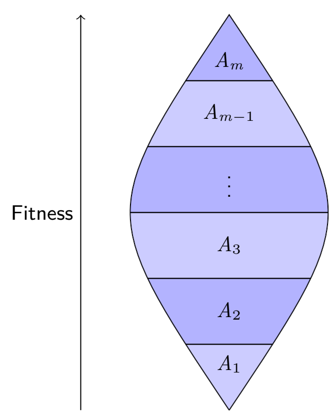


for $t = 0, 1, 2, \ldots$ until termination condition **do**
  for $i = 1$ to λ **do**
    Sample i-th parent x according to $p_{\mathrm{sel}}(P_t, f)$
    Sample i-th offspring $P_{t+1}(i)$ according to $p_{\mathrm{var}}(x)$
  end for
end for

A general algorithmic scheme for non-elitistic EAs

- $f : \mathcal{X} \to \mathbb{R}$ fitness function over arbitrary finite search space $\mathcal{X}$
- p_{sel} selection mechanism (e.g. (μ, λ)-selection)
- p_{var} variation operator (e.g. mutation)

Advanced: Fitness Levels for non-Elitist Populations



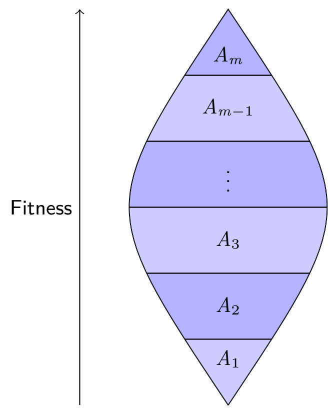
Theorem ([Lehre, 2011a])

If exists $\delta, \gamma_*, s_1, \ldots, s_m, s_*, p_0 \in (0,1)$ *st.*

(C1) $p_{\mathrm{var}}\left(y \in A_j^+ \mid x \in A_j\right) \geq s_j \geq s_*$
upgrade probability s_j

(C2) $p_{\mathrm{var}}\left(y \in A_j \cup A_j^+ \mid x \in A_j\right) \geq p_0$
resting probability p_0

Advanced: Fitness Levels for non-Elitist Populations



Theorem ([Lehre, 2011a])

If exists $\delta, \gamma_*, s_1, \dots, s_m, s_*, p_0 \in (0,1)$ *st.*

(C1) $p_{\text{var}}\left(y \in A_j^+ \mid x \in A_j\right) \geq s_j \geq s_*$
upgrade probability s_j

(C2) $p_{\text{var}}\left(y \in A_j \cup A_j^+ \mid x \in A_j\right) \geq p_0$
resting probability p_0

(C3) $\beta(\gamma) > \gamma(1+\delta)/p_0$ *for all* $\gamma < \gamma_*$
"high" selective pressure

(C4) $\lambda > c' \ln(m/s_*)$ *for some const.* c'
"large" population size

then for a constant $c > 0$

$$\mathbb{E}[T] \leq c\left(m\lambda^2 + \sum_{j=1}^{m-1} \frac{1}{s_j}\right)$$

Measuring Selective Pressure

Definition

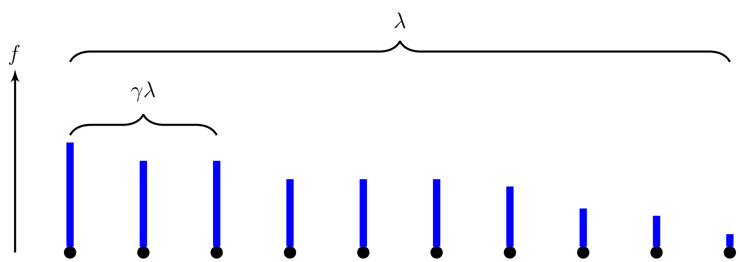
Let $x^{(1)}, x^{(2)}, \dots, x^{(\lambda)}$ be the individuals in a population $P \in \mathcal{X}^\lambda$, sorted according to a fitness function $f : \mathcal{X} \to \mathbb{R}$, i.e.

$$f\left(x^{(1)}\right) \geq f\left(x^{(2)}\right) \geq \cdots \geq f\left(x^{(\lambda)}\right).$$

For any $\gamma \in (0,1)$, the **cumulative selection probability** of p_{sel} is

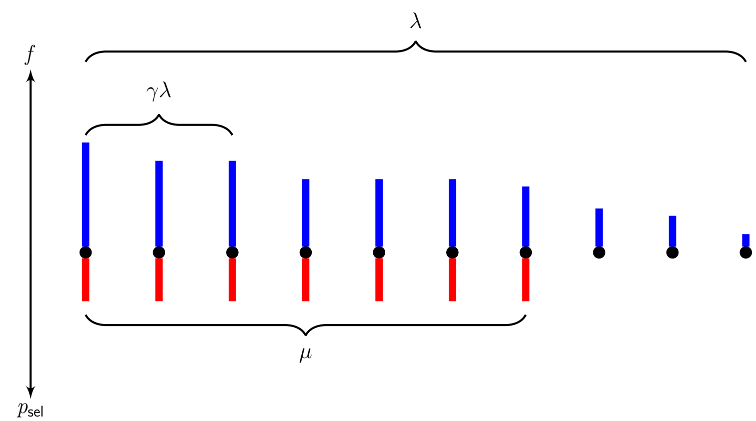
$$\beta(\gamma) := \Pr\left(\, f(y) \geq f\left(x^{(\gamma\lambda)}\right) \;\middle|\; y \text{ is sampled from } p_{\text{sel}}(P,f)\,\right)$$

Cumulative Selection Prob. - Example



$$\beta(\gamma) = \Pr\left(\, f(y) \geq f\left(x^{(\gamma\lambda)}\right) \;\middle|\; y \text{ is sampled from } p_{\text{sel}}(P,f)\,\right)$$

Cumulative Selection Prob. - Example (μ, λ)-selection



$$\beta(\gamma) = \Pr\left(\, f(y) \geq f\left(x^{(\gamma\lambda)}\right) \;\middle|\; y \text{ is sampled from } p_{\text{sel}}(P,f)\,\right)$$
$$\geq \frac{\gamma\lambda}{\mu} \quad \text{if } \gamma\lambda \leq \mu$$

Example Application[3]

(μ,λ) EA with bit-wise mutation rate χ/n on LeadingOnes

Partition of fitness function into $m := n + 1$ levels

$$A_j := \{x \in \{0,1\}^n \mid x_1 = x_2 = \cdots = x_{j-1} = 1 \wedge x_j = 0\}$$

If $\lambda/\mu > e^{\chi}$ and $\lambda > c'' \ln(n)$ then

(C1) $p_{\text{var}}\left(y \in A_j^+ \mid x \in A_j\right) = \Omega(1/n)$

(C2) $p_{\text{var}}\left(y \in A_j \cup A_j^+ \mid x \in A_j\right) \approx e^{-\chi}$

(C3) $\beta(\gamma) \geq \gamma\lambda/\mu > \gamma e^{\chi}$

(C4) $\lambda > c'' \ln(n)$

[3]Calculations on this slide are approximate. See [Lehre, 2011a] for exact calculations.

Example Application[3]

(μ,λ) EA with bit-wise mutation rate χ/n on LeadingOnes

Partition of fitness function into $m := n + 1$ levels

$$A_j := \{x \in \{0,1\}^n \mid x_1 = x_2 = \cdots = x_{j-1} = 1 \wedge x_j = 0\}$$

If $\lambda/\mu > e^{\chi}$ and $\lambda > c'' \ln(n)$ then

(C1) $p_{\text{var}}\left(y \in A_j^+ \mid x \in A_j\right) = \Omega(1/n) \quad =: \quad s_j =: s_*$

(C2) $p_{\text{var}}\left(y \in A_j \cup A_j^+ \mid x \in A_j\right) \approx e^{-\chi} \quad =: \quad p_0$

(C3) $\beta(\gamma) \geq \gamma\lambda/\mu > \gamma e^{\chi} \quad = \quad \gamma/p_0$

(C4) $\lambda > c'' \ln(n) \quad > \quad c\ln(m/s^*)$

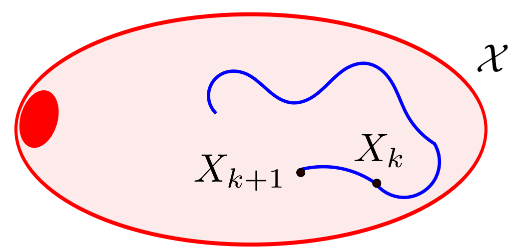
then $\mathbb{E}[T] = O(m\lambda^2 + \sum_{j=1}^{m} s_j^{-1}) = O(n\lambda^2 + n^2)$

[3]Calculations on this slide are approximate. See [Lehre, 2011a] for exact calculations.

Artificial Fitness Levels: Conclusions

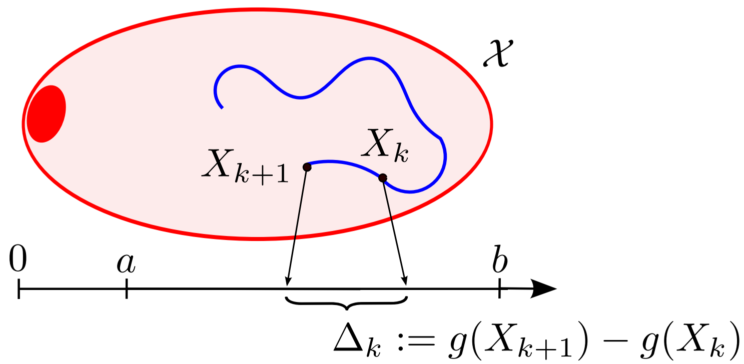
- It's a powerful general method to obtain (often) tight upper bounds on the runtime of simple EAs;
- For offspring populations tight bounds can often be achieved with the general method;
- For parent populations takeover times have to be introduced [Witt, 2006];
- A recent method has been presented to deal with population and non-elitism [Lehre, 2011a].

What is Drift[4] Analysis?



[4]NB! (Stochastic) drift is a different concept than *genetic drift* in population genetics.

What is Drift[4] Analysis?

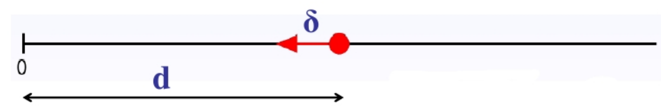


- Prediction of the long term behaviour of a process X
  - hitting time, stability, occupancy time etc.

  from properties of Δ.

[4] NB! (Stochastic) drift is a different concept than *genetic drift* in population genetics.

Additive Drift Theorem



Theorem (Additive Drift Theorem for Upper Bounds [He and Yao, 2001])

Let $\{X_t\}_{t\geq 0}$ be a Markov process over a set of states S, and $d : S \to \mathbb{R}_0^+$ a function that assigns a non-negative real number to every state. Let the time to reach the optimum be $T := \min\{t \geq 0 : d(X_t) = 0\}$. If there exists $\delta > 0$ such that at any time step $t \geq 0$ and at any state $X_t > 0$ the following condition holds:

$$E(\Delta(t)|d(X_t) > 0) = E(d(X_t) - d(X_{t+1}) \mid d(X_t) > 0) \geq \delta \tag{1}$$

then

$$E(T \mid X_0 > 0) \leq \frac{d(X_0)}{\delta} \tag{2}$$

and

$$E(T) \leq \frac{E(d(X_0))}{\delta}. \tag{3}$$

Plateau Block Function: Upper Bound

Let $k + m > n/2 + \epsilon n$.

$$PlateauBlock_\ell(|x|) = \begin{cases} a & \text{if } k \leq n - |x| \leq k + m \\ 0 & \text{otherwise.} \end{cases}$$

Theorem

The expected time for the (1+1)-EA to optimise the Plateau function is $O(m)$.

Proof

Let X_t be the number of 0-bits at time t. Then the drift is

$$E(\Delta(t) \geq \frac{X_t}{n} - \frac{n - X_t}{n} = \frac{2X_t}{n} - 1 \geq \frac{2k}{n} - 1$$

Hence, by drift analysis

$$E[T] \leq \frac{m}{(2k)/n - 1} = \frac{mn}{2k - n} = O(m)$$

where the last equality holds as long as $k > n/2 + \epsilon n$

Additive Drift Theorem: Lower Bounds

Theorem (Additive Drift Theorem for Lower Bounds [He and Yao, 2004])

Let $\{X_t\}_{t\geq 0}$ be a Markov process over a set of states S, and $d : S \to \mathbb{R}_0^+$ a function that assigns a non-negative real number to every state. Let the time to reach the optimum be $T := \min\{t \geq 0 : d(X_t) = 0\}$. If there exists $\delta > 0$ such that at any time step $t \geq 0$ and at any state $X_t > 0$ the following condition holds:

$$E(\Delta(t)|d(X_t) > 0) = E(d(X_t) - d(X_{t+1}) \mid d(X_t) > 0) \leq \delta \tag{4}$$

then

$$E(T \mid X_0 > 0) \geq \frac{d(X_0)}{\delta} \tag{5}$$

and

$$E(T) \geq \frac{E(d(X_0))}{\delta}. \tag{6}$$

Plateau Block Function: Lower Bound

Let $k + m > n/2 + \epsilon n$.

$$PlateauBlock_\ell(|x|) = \begin{cases} a & \text{if } k \leq n - |x| \leq k + m \\ 0 & \text{otherwise.} \end{cases}$$

Theorem

The expected time for the (1+1)-EA to optimise the Plateau function is $\Omega(m)$.

Proof

Let X_t be the number of 0-bits at time t. Then the drift is

$$E(\Delta(t)) = \frac{X_t}{n} - \frac{n - X_t}{n} = \frac{2X_t}{n} - 1 \leq \frac{2(m+k)}{n} - 1$$

Hence, by drift analysis

$$E[T] \geq \frac{m}{2(m+k)/n - 1} = \frac{mn}{2(m+k) - n} = \Theta(m)$$

where the last equality holds as long as $(k + m) > n/2 + \epsilon n$

Drift Analysis for OneMax

Lets calculate the runtime of the (1+1)-EA using the additive Drift Theorem.

1. Let $g(X_t) = i$ where i is the number of zeroes in the bitstring;

Drift Analysis for OneMax

Lets calculate the runtime of the (1+1)-EA using the additive Drift Theorem.

1. Let $g(X_t) = i$ where i is the number of zeroes in the bitstring;
2. The negative drift is 0 since solution with less one-bits will not be accepted;

Drift Analysis for OneMax

Lets calculate the runtime of the (1+1)-EA using the additive Drift Theorem.

1. Let $g(X_t) = i$ where i is the number of zeroes in the bitstring;
2. The negative drift is 0 since solution with less one-bits will not be accepted;
3. A positive drift is achieved as long as a 0 is flipped and the ones remain unchanged:

$$E(\Delta^+(t)) = E[d(X_t) - d(X_{t+1})] \geq 1 \cdot \frac{i}{n}\left(1 - \frac{1}{n}\right)^{n-1} \geq \frac{i}{en} \geq \frac{1}{en}$$

Drift Analysis for OneMax

Lets calculate the runtime of the (1+1)-EA using the additive Drift Theorem.

1. Let $g(X_t) = i$ where i is the number of zeroes in the bitstring;
2. The negative drift is 0 since solution with less one-bits will not be accepted;
3. A positive drift is achieved as long as a 0 is flipped and the ones remain unchanged:

$$E(\Delta^+(t)) = E[d(X_t) - d(X_{t+1})] \geq 1 \cdot \frac{i}{n}\left(1 - \frac{1}{n}\right)^{n-1} \geq \frac{i}{en} \geq \frac{1}{en}$$

4. Hence, $E[\Delta(t)] = E(\Delta^+(t)) - E(\Delta^-(t)) \geq 1/(en) = \delta$

Drift Analysis for OneMax

Lets calculate the runtime of the (1+1)-EA using the additive Drift Theorem.

1. Let $g(X_t) = i$ where i is the number of zeroes in the bitstring;
2. The negative drift is 0 since solution with less one-bits will not be accepted;
3. A positive drift is achieved as long as a 0 is flipped and the ones remain unchanged:

$$E(\Delta^+(t)) = E[d(X_t) - d(X_{t+1})] \geq 1 \cdot \frac{i}{n}\left(1 - \frac{1}{n}\right)^{n-1} \geq \frac{i}{en} \geq \frac{1}{en}$$

4. Hence, $E[\Delta(t)] = E(\Delta^+(t)) - E(\Delta^-(t)) \geq 1/(en) = \delta$
5. The expected initial distance is $E(d(X_0)) = n/2$

The expected runtime is (i.e. Eq. (6)):

$$E(T \mid d(X_0) > 0) \leq \frac{E[(d(X_0)]}{\delta} \leq \frac{n/2}{1/(en)} = e/2 \cdot n^2 = O(n^2)$$

We need a different distance function!

Drift Analysis for OneMax

1. Let $g(X_t) = \ln(i+1)$ where i is the number of zeroes in the bitstring;

Drift Analysis for OneMax

1. Let $g(X_t) = \ln(i+1)$ where i is the number of zeroes in the bitstring;
2. The negative drift is 0 since solution with less one-bits will not be accepted;

Drift Analysis for OneMax

1. Let $g(X_t) = \ln(i+1)$ where i is the number of zeroes in the bitstring;
2. The negative drift is 0 since solution with less one-bits will not be accepted;
3. A positive drift is achieved as long as a 0 is flipped and the ones remain unchanged:

$$E(\Delta^+(t)) = E[d(X_t)-d(X_{t+1})] \geq \ln(i+1)\cdot\frac{i}{n}\left(1-\frac{1}{n}\right)^{n-1} \geq \frac{\ln(i+1)}{en} \geq \frac{\ln(2)}{en}$$

Drift Analysis for OneMax

1. Let $g(X_t) = \ln(i+1)$ where i is the number of zeroes in the bitstring;
2. The negative drift is 0 since solution with less one-bits will not be accepted;
3. A positive drift is achieved as long as a 0 is flipped and the ones remain unchanged:

$$E(\Delta^+(t)) = E[d(X_t)-d(X_{t+1})] \geq \ln(i+1)\cdot\frac{i}{n}\left(1-\frac{1}{n}\right)^{n-1} \geq \frac{\ln(i+1)}{en} \geq \frac{\ln(2)}{en}$$

4. Hence, $\Delta(t) = E(\Delta^+(t)) - E(\Delta^-(t)) \geq \ln(2)/(en) = \delta$

Drift Analysis for OneMax

1. Let $g(X_t) = \ln(i+1)$ where i is the number of zeroes in the bitstring;
2. The negative drift is 0 since solution with less one-bits will not be accepted;
3. A positive drift is achieved as long as a 0 is flipped and the ones remain unchanged:

$$E(\Delta^+(t)) = E[d(X_t)-d(X_{t+1})] \geq \ln(i+1)\cdot\frac{i}{n}\left(1-\frac{1}{n}\right)^{n-1} \geq \frac{\ln(i+1)}{en} \geq \frac{\ln(2)}{en}$$

4. Hence, $\Delta(t) = E(\Delta^+(t)) - E(\Delta^-(t)) \geq \ln(2)/(en) = \delta$
5. The distance is $d(X_0) \leq \ln(n+1)$

The expected runtime is (i.e. Eq. (6)):

$$E(T \mid d(X_0) > 0) \leq \frac{d(X_0)}{\delta} \leq \frac{\ln(n+1)}{\ln(2)/(en)} = O(n \ln n)$$

If the amount of progress is proportional to the distance from the optimum we need to use a logarithmic distance!

Multiplicative Drift Theorem

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

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

1. $E[X_t - X_{t+1} \mid X_t] \geq \delta X_t$ *and*
2. $c_{\min} \leq X_t \leq c_{\max}$,

for all $t < T$, then

$$E[T] \leq \frac{2}{\delta} \cdot \ln\left(1 + \frac{c_{\max}}{c_{\min}}\right)$$

(1+1)-EA Analysis for OneMax

Theorem

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

Proof

Introduction Motivation Evolutionary Algorithms Tail Inequalities Artificial Fitness Levels Drift Analysis Conclusions
Multiplicative Drift Theorem

(1+1)-EA Analysis for OneMax

Theorem

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

Proof

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

Hence,

$$E[T] \leq \frac{2}{\delta} \cdot \ln\left(1 + \frac{c_{\max}}{c_{\min}}\right) = 2en\ln(1+n) = O(n \ln n)$$

Introduction Motivation Evolutionary Algorithms Tail Inequalities Artificial Fitness Levels Drift Analysis Conclusions
Multiplicative Drift Theorem

Linear Unitation Block: Upper Bound

Theorem

The expected time for the (1+1)-EA to optimise the Linear Unitation Block is $O(n\ln((m+k)/k))$

Proof

Introduction Motivation Evolutionary Algorithms Tail Inequalities Artificial Fitness Levels Drift Analysis Conclusions
Multiplicative Drift Theorem

Linear Unitation Block: Upper Bound

Theorem

The expected time for the (1+1)-EA to optimise the Linear Unitation Block is $O(n\ln((m+k)/k))$

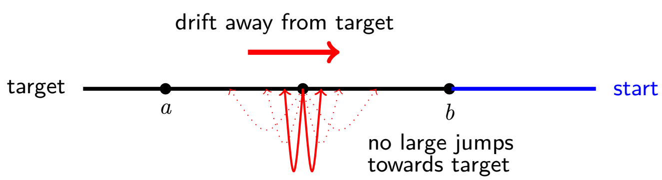
Proof

- Distance: let i be the number of zeroes;
- $E[X_{t+1}|X_t = i] \geq i - 1 \cdot \frac{i}{en} = i \cdot \left(1 - \frac{1}{en}\right) = X_t \cdot \left(1 - \frac{1}{en}\right)$
- $E[X_t - X_{t+1}|X_t = i] \leq X_t - X_t \cdot \left(1 - \frac{1}{en}\right) = \frac{1}{en} X_t \; (\delta = \frac{1}{en})$
- $k = c_{\min} \leq X_t \leq c_{\max} = m + k$

Hence,

$$E[T] \leq \frac{2}{\delta} \cdot \ln\left(1 + \frac{c_{\max}}{c_{\min}}\right) = 2en\ln(1 + (m+k)/k) = O(n\ln((m+k)/k))$$

Simplified Drift Theorem



Theorem (Simplified Negative-Drift Theorem, [Oliveto and Witt, 2011])

Suppose there exist three constants δ,ϵ,r such that for all $t \geq 0$:

1. $E(\Delta_t(i)) \geq \epsilon$ *for* $a < i < b$,
2. $\text{Prob}(|\Delta_t(i)| = -j) \leq \frac{1}{(1+\delta)^{j-r}}$ *for* $i > a$ *and* $j \geq 1$.

Then

$$\text{Prob}(T^* \leq 2^{c^*(b-a)}) = 2^{-\Omega(b-a)}$$

.

Needle in a Haystack

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^{cn} steps.*

Needle in a Haystack

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^{cn} 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

$$E(\Delta(i)) = \frac{n-i}{n} - \frac{i}{n} = \frac{n-2i}{n} \geq 2\gamma = \epsilon$$

Proof of Condition 2

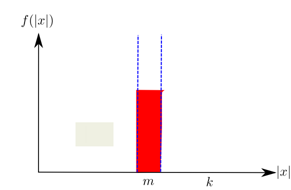
$$Prob(\Delta(i) \leq -j) \leq \binom{n}{j}\left(\frac{1}{n}\right)^j \leq \frac{1}{j!} \leq \left(\frac{1}{2}\right)^{j-1}$$

This proves Condition 2 by setting $\delta = r = 1$.

Plateau Block Function: Lower Bound

Let $k + m < (1/2 - \epsilon)n$.

$$PlateauBlock_r(|x|) = \begin{cases} a & \text{if } k \leq n - |x| \leq k+m \\ 0 & \text{otherwise.} \end{cases}$$



Theorem

The time for the (1+1)-EA to optimise $PlateauBlock_r$ is at least $2^{\Omega(m)}$ with probability at least $1 - 2^{-\Omega(m)}$.

Proof

Let X_t be the number of 0-bits at time t.

$$E(\Delta(t) = \frac{n - X_t}{n} - \frac{X_t}{n} = 1 - \frac{2X_t}{n} \geq \frac{n}{n} - \frac{2(k+m)}{n} = \frac{n - 2(k+m)}{n}$$

If $2(k+m) < n(1-\epsilon)$ by the simplified drift theorem

$$P(T < 2^{cm}) = 2^{-\Omega(m)}$$

Plateau Block Function: Upper Bound

Theorem

The expected time for the (1+1)-EA to optimise $PlateauBlock_r$ is at most $e^{O(m)}$.

Proof

We calculate the probability p of m consecutive steps across the plateau

$$\prod_{i=m+1}^{k+m} p_i \geq \prod_{i=1}^{m} \frac{k+i}{en} \geq \left(\frac{1}{en}\right)^m \frac{(k+m)!}{k!} \geq \left(\frac{1}{en}\right)^m \left(\frac{k+m}{e}\right)^m = \left(\frac{k+m}{e^2 n}\right)^m$$

where

$$\frac{(k+m)!}{k!} = m! \cdot \frac{(k+m)!}{m!k!} = m! \binom{k+m}{m} \geq \left(\frac{m}{e}\right)^m \left(\frac{k+m}{m}\right)^m = \left(\frac{k+m}{e}\right)^m$$

Hence,

$$\mathbb{E}[T] \leq m \cdot 1/p = m \left(\frac{e^2 n}{k+m}\right)^m$$

Some history[5]

Origins

- Stability of equilibria in ODEs (Lyapunov, 1892)
- Stability of Markov Chains (see eg [Meyn and Tweedie, 1993])
- 1982 paper by Hajek [Hajek, 1982]
  - Simulated annealing (1988) [Sasaki and Hajek, 1988]

Drift Analysis of Evolutionary Algorithms

- Introduced to EC in 2001 by He and Yao [He and Yao, 2001, He and Yao, 2004] (additive drift)
  - (1+1) EA on linear functions: $O(n \ln n)$ [He and Yao, 2001]
  - (1+1) EA on maximum matching by Giel and Wegener [Giel and Wegener, 2003]
- Simplified drift in 2008 by Oliveto and Witt [Oliveto and Witt, 2011]
- Multiplicative drift by Doerr et al [Doerr et al., 2010b]
  - (1+1) EA on linear functions: $en \ln(n) + O(n)$ [Witt, 2012]
- Variable drift by Johannsen [Johannsen, 2010] and Mitavskiy et al. [Mitavskiy et al., 2009]
- Population drift by Lehre [Lehre, 2011b]

[5] More on drift in GECCO 2012 tutorial by Lehre http://www.cs.nott.ac.uk/~pkl/drift

Final Overview

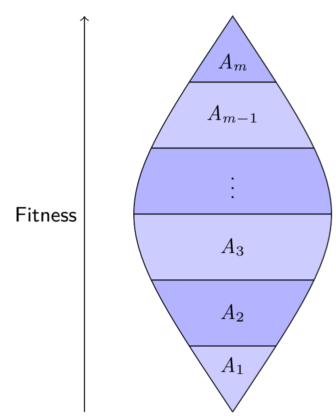
Overview

- Tail Inequalities
- Artificial Fitness Levels
- Drift Analysis

Other Techniques (Not covered)

- Family Trees [Witt, 2006]
- Gambler's Ruin & Martingales [Jansen and Wegener, 2001]

General lower bound on block example



Theorem

Let

s_i : *prob. of starting in* A_i

T_i: *time spent in level* A_i

γ_{ij} : *prob. of jumping to* A_j *when leaving* A_i

If there exists a $\chi \in [0,1)$ *st. for* $i < j$

$$\gamma_{ij} \geq \chi \sum_{k=j}^{m-1} \gamma_{ik},$$

$$\sum_{k=i+1}^{m-1} \gamma_{ik} = 1$$

then

$$\mathbb{E}[T] \geq \chi \sum_{i=1}^{m-1} s_i \sum_{j=i}^{m-1} \mathbb{E}[T_j]$$

General lower bound - Proof by induction[6]

Induction hypothesis

$$E_j \geq \mathbb{E}\left[T_j\right] + \chi \sum_{k=j+1}^{m-1} \mathbb{E}\left[T_k\right]$$

Assumptions

$$\gamma_{ij} \geq \chi \sum_{k=j}^{m-1} \gamma_{ik}$$

$$\sum_{k=i+1}^{m-1} \gamma_{ik} = 1$$

$$\begin{aligned} E_i &\geq \mathbb{E}\left[T_i\right] + \sum_{j=i+1}^{m-1} \gamma_{ij} E_j \\ &\geq \mathbb{E}\left[T_i\right] + \sum_{j=i+1}^{m-1} \gamma_{ij} \left(\mathbb{E}\left[T_j\right] + \chi \sum_{k=j+1}^{m-1} \mathbb{E}\left[T_k\right] \right) \\ &= \mathbb{E}\left[T_i\right] + \sum_{j=i+1}^{m-1} \mathbb{E}\left[T_j\right] \left(\gamma_{ij} + \chi \sum_{k=i+1}^{j-1} \gamma_{ik} \right) \\ &\geq \mathbb{E}\left[T_i\right] + \sum_{j=i+1}^{m-1} \mathbb{E}\left[T_j\right] \left(\chi \sum_{k=j}^{m-1} \gamma_{ik} + \chi \sum_{k=i+1}^{j-1} \gamma_{ik} \right) \\ &= \mathbb{E}\left[T_i\right] + \chi \sum_{j=i+1}^{m-1} \mathbb{E}\left[T_j\right]. \end{aligned}$$

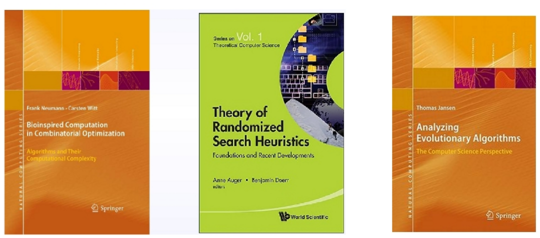
[6] Simple generalisation of [Sudholt, 2010].

State of the Art in Computational Complexity of RSHs

Papers at this Gecco:

- The Generalized Minimum Spanning Tree Problem: A Parameterized Complexity Analysis of Bi-level Optimisation, (Çörüş, Lehre, Neumann);
- Parameterized Average-Case Complexity of the Hypervolume Indicator, (Bringmann, Friedrich);
- Analysis of Diversity Mechanisms for Robust Optimisation in Dynamic Environments with Low Frequencies of Change, (Oliveto, Zarges);
- Lessons From the Black-Box: A Fast Crossover-Based Genetic Algorithm, (B. Doerr, C. Doerr, Ebel);
- Runtime Analysis of Mutation-Based Geometric Semantic Genetic Programming for Basis Functions Regression, (Moraglio, Mambrini);
- A Theoretical Runtime and Empirical Analysis of Different Alternating Variable Searches for Search-Based Testing, (Kempka, McMinn, Sudholt);
- Plenty more in the Theory Track!

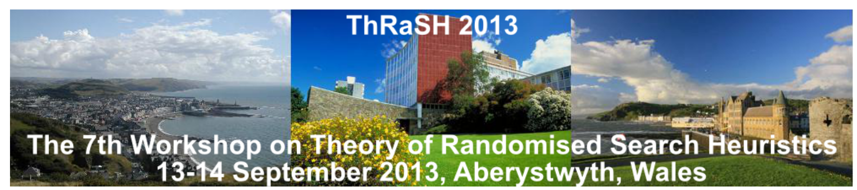
Further Reading



Next Tutorials

- Bioinspired Computation in Combinatorial Optimization - Algorithms and Their Computational Complexity, (Neumann, Witt)
- Black-Box Complexity: From Complexity Theory to Playing Mastermind, (B. Doerr, C. Doerr)
- Artificial Immune Systems for Optimization, (Jansen, Zarges)
- Elementary Landscapes: Theory and Applications (Sutton, Whitley)

Coming up Workshop



Goals

- To contribute to the theoretical understanding of randomised search heuristics;
- to stimulate interactions within the research field and between people from different disciplines working on randomised algorithms;
- discuss recent ideas and detecting challenging topics for future work.

References I

Bäck, T. (1993).
Optimal mutation rates in genetic search.
In *In Proceedings of the Fifth International Conference on Genetic Algorithms (ICGA)*, pages 2–8.

Doerr, B., Johannsen, D., and Winzen, C. (2010a).
Multiplicative drift analysis.
In *Proceedings of the 12th annual conference on Genetic and evolutionary computation*, GECCO '10, pages 1449–1456. ACM.

Doerr, B., Johannsen, D., and Winzen, C. (2010b).
Multiplicative drift analysis.
In *GECCO '10: Proceedings of the 12th annual conference on Genetic and evolutionary computation*, pages 1449–1456, New York, NY, USA. ACM.

Droste, S., Jansen, T., and Wegener, I. (1998).
A rigorous complexity analysis of the $(1+1)$ evolutionary algorithm for separable functions with boolean inputs.
Evolutionary Computation, 6(2):185–196.

Giel, O. and Wegener, I. (2003).
Evolutionary algorithms and the maximum matching problem.
In *Proceedings of the 20th Annual Symposium on Theoretical Aspects of Computer Science (STACS 2003)*, pages 415–426.

Goldberg, D. E. (1989).
Genetic Algorithms for Search, Optimization, and Machine Learning.
Addison-Wesley.

References II

Hajek, B. (1982).
Hitting-time and occupation-time bounds implied by drift analysis with applications.
Advances in Applied Probability, 13(3):502–525.

He, J. and Yao, X. (2001).
Drift analysis and average time complexity of evolutionary algorithms.
Artificial Intelligence, 127(1):57–85.

He, J. and Yao, X. (2004).
A study of drift analysis for estimating computation time of evolutionary algorithms.
Natural Computing: an international journal, 3(1):21–35.

Holland, J. H. (1992).
Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control, and Artificial Intelligence.
The MIT Press.

Jansen, T. and Wegener, I. (2001).
Evolutionary algorithms - how to cope with plateaus of constant fitness and when to reject strings of the same fitness.
IEEE Trans. Evolutionary Computation, 5(6):589–599.

Johannsen, D. (2010).
Random combinatorial structures and randomized search heuristics.
PhD thesis, Universität des Saarlandes.

References III

Lehre, P. K. (2011a).
Fitness-levels for non-elitist populations.
In *Proceedings of the 13th annual conference on Genetic and evolutionary computation*, GECCO '11, pages 2075–2082. ACM.

Lehre, P. K. (2011b).
Negative drift in populations.
In *Proceedings of Parallel Problem Solving from Nature - (PPSN XI)*, volume 6238 of *LNCS*, pages 244–253. Springer Berlin / Heidelberg.

Meyn, S. P. and Tweedie, R. L. (1993).
Markov Chains and Stochastic Stability.
Springer-Verlag.

Mitavskiy, B., Rowe, J. E., and Cannings, C. (2009).
Theoretical analysis of local search strategies to optimize network communication subject to preserving the total number of links.
International Journal of Intelligent Computing and Cybernetics, 2(2):243–284.

Oliveto, P. S. and Witt, C. (2011).
Simplified drift analysis for proving lower bounds inevolutionary computation.
Algorithmica, 59(3):369–386.

Reeves, C. R. and Rowe, J. E. (2002).
Genetic Algorithms: Principles and Perspectives: A Guide to GA Theory.
Kluwer Academic Publishers, Norwell, MA, USA.

References IV

Rudolph, G. (1998).
Finite Markov chain results in evolutionary computation: A tour d'horizon.
Fundamenta Informaticae, 35(1–4):67–89.

Sasaki, G. H. and Hajek, B. (1988).
The time complexity of maximum matching by simulated annealing.
Journal of the Association for Computing Machinery, 35(2):387–403.

Sudholt, D. (2010).
General lower bounds for the running time of evolutionary algorithms.
In *PPSN (1)*, pages 124–133.

Vose, M. D. (1999).
The simple genetic algorithm: foundations and theory.
MIT Press, Cambridge, MA.

Witt, C. (2006).
Runtime analysis of the $(\mu{+}1)$ ea on simple pseudo-boolean functions evolutionary computation.
In *GECCO '06: Proceedings of the 8th annual conference on Genetic and evolutionary computation*, pages 651–658, New York, NY, USA. ACM Press.

Witt, C. (2012).
Optimizing linear functions with randomized search heuristics - the robustness of mutation.
In Dürr, C. and Wilke, T., editors, *29th International Symposium on Theoretical Aspects of Computer Science (STACS 2012)*, volume 14 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 420–431, Dagstuhl, Germany. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik.

References V

Wolpert, D. and Macready, W. G. (1997).
No free lunch theorems for optimization.
IEEE Trans. Evolutionary Computation, 1(1):67–82.