The University Of Sheffield.

RIGOROUS RESEARCH

Theory of Evolutionary Computation: A Gentle Introduction to the Time Complexity Analysis of Evolutionary Algorithms

Pietro S. Oliveto

Department of Computer Science, University of Sheffield, UK

CEC 2017
Donostia, San Sebastian, Spain, 5 June 2017

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

---

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.
- *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!

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])

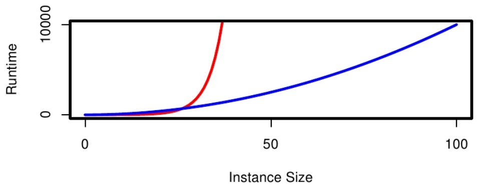
1. *There is a positive probability to reach any point in the search space from any other point*
2. *The best found solution is never removed from the population (elitism)*

- 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?

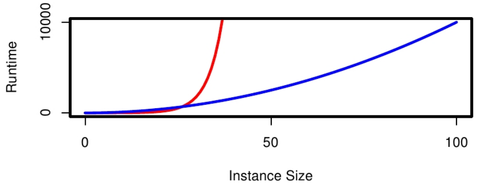
- Most EAs visit the global optimum in finite time (RLS does not!)
- How much time?

Computational Complexity of EAs



P. K. Lehre, 2011

Computational Complexity of EAs



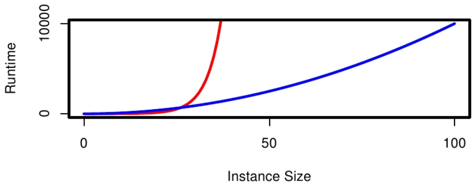
P. K. Lehre, 2011

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

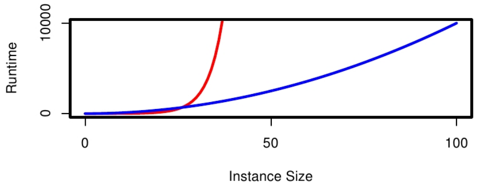


P. K. Lehre, 2011

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!

Computational Complexity of EAs



P. K. Lehre, 2011

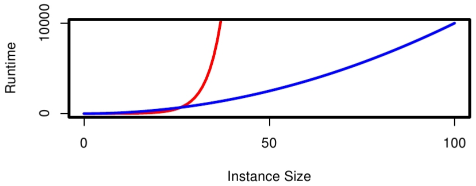
However (EAs):

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;

Computational Complexity of EAs



P. K. Lehre, 2011

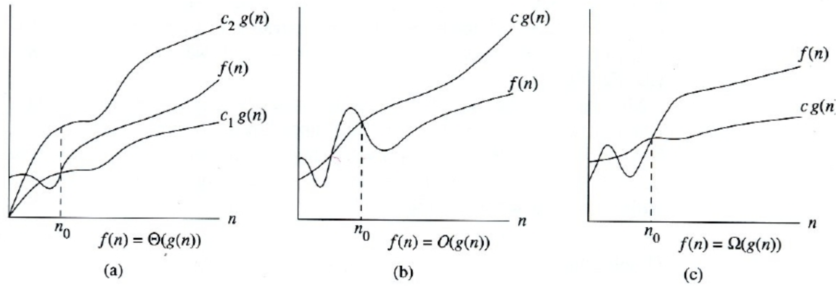
However (EAs):

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 $p(T_f \leq t)$, the success probability of the EA in t steps for f.

Asymptotic notation



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

Exercise 1: Asymptotic Notation

	$o(1)$	$O(1)$	$O(\log n)$	$O(n^2)$	$n^{\Theta(1)}$	$e^{\Omega(n)}$
$f_1(n) = \log(n^2)$			√	√		
$f_2(n) = \frac{n(n-1)}{2}$				√	√	
$f_3(n) = \sqrt{n}$				√	√	
$f_4(n) = n!$						√
$f_5(n) = \frac{1}{n}$	√	√	√	√		
$f_6(n) = 100$		√	√	√		
$f_7(n) = 2^n$						√
$f_8(n) = 2^{-n}n^n$						√

[Lehre, 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)

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

Evolutionary Algorithms

Algorithm ((μ+λ)-EA)

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

- if $\mu = \lambda = 1$ we get a (1+1)-EA;

Evolutionary Algorithms

Algorithm ((μ+λ)-EA)

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

- if $\mu = \lambda = 1$ we get a (1+1)-EA;
- $p = 1/n$ is generally considered as best choice [Bäck, 1993, Droste et al., 1998];

Evolutionary Algorithms

Algorithm ((μ+λ)-EA)

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

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

1+1-EA

Algorithm ((1+1)-EA)

```
• Initialise $P_0$ with $x \in \{1,0\}^n$ by flipping each bit with $p = 1/2$ ;
  Repeat
• Create $x'$ by flipping each bit in $x$ with $p = 1/n$;
• If $f(x') \geq f(x)$ Then $x' \in P_{t+1}$ Else $x \in P_{t+1}$;
• Let $t = t + 1$; Until stopping condition.
```

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

How does it work?

1+1-EA

Algorithm ((1+1)-EA)

- *Initialise* P_0 *with* $x \in \{1,0\}^n$ *by flipping each bit with* $p = 1/2$ *;* **Repeat**
- *Create* x' *by flipping each bit in* x *with* $p = 1/n$*;*
- **If** $f(x') \geq f(x)$ **Then** $x' \in P_{t+1}$ **Else** $x \in P_{t+1}$*;*
- *Let* $t = t + 1$*;* **Until** *stopping condition.*

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?

1+1-EA

Algorithm ((1+1)-EA)

- *Initialise* P_0 *with* $x \in \{1,0\}^n$ *by flipping each bit with* $p = 1/2$ *;* **Repeat**
- *Create* x' *by flipping each bit in* x *with* $p = 1/n$*;*
- **If** $f(x') \geq f(x)$ **Then** $x' \in P_{t+1}$ **Else** $x \in P_{t+1}$*;*
- *Let* $t = t + 1$*;* **Until** *stopping condition.*

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?

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

1+1-EA

Algorithm ((1+1)-EA)

- *Initialise* P_0 *with* $x \in \{1,0\}^n$ *by flipping each bit with* $p = 1/2$ *;* **Repeat**
- *Create* x' *by flipping each bit in* x *with* $p = 1/n$*;*
- **If** $f(x') \geq f(x)$ **Then** $x' \in P_{t+1}$ **Else** $x \in P_{t+1}$*;*
- *Let* $t = t + 1$*;* **Until** *stopping condition.*

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?

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

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

1+1-EA

Algorithm ((1+1)-EA)

- *Initialise* P_0 *with* $x \in \{1,0\}^n$ *by flipping each bit with* $p = 1/2$ *;* **Repeat**
- *Create* x' *by flipping each bit in* x *with* $p = 1/n$*;*
- **If** $f(x') \geq f(x)$ **Then** $x' \in P_{t+1}$ **Else** $x \in P_{t+1}$*;*
- *Let* $t = t + 1$*;* **Until** *stopping condition.*

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?

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

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

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

1+1-EA: 2

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

1+1-EA: 2

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

- What is the probability of exactly one bit flipping?

1+1-EA: 2

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

- What is the probability of exactly one bit flipping?

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

1+1-EA: 2

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

- What is the probability of exactly one bit flipping?

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

Is it more likely that 2 bits flip or none?

1+1-EA: 2

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

- What is the probability of exactly one bit flipping?

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

Is it more likely that 2 bits flip or none?

$$Pr(X=2)=\binom{n}{2}\cdot 1/n^2\cdot(1-1/n)^{n-2}=$$

$$=\frac{n\cdot(n-1)}{2}1/n^2\cdot(1-1/n)^{n-2}=$$
$$=1/2\cdot(1-1/n)^{n-1}\approx 1/(2e)$$

1+1-EA: 2

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

- What is the probability of exactly one bit flipping?

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

Is it more likely that 2 bits flip or none?

$$Pr(X=2)=\binom{n}{2}\cdot 1/n^2\cdot(1-1/n)^{n-2}=$$

$$=\frac{n\cdot(n-1)}{2}1/n^2\cdot(1-1/n)^{n-2}=$$
$$=1/2\cdot(1-1/n)^{n-1}\approx 1/(2e)$$

While

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

1+1-EA: General Upper bound

Theorem ([Droste et al., 2002])

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

1+1-EA: General Upper bound

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

1+1-EA: General Upper bound

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

1. Let i be the number of bit positions in which the current solution x and the global optimum x^* differ;

1+1-EA: General Upper bound

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

1. Let i be the number of bit positions in which the current solution x and the global optimum x^* differ;
2. Each bit flips with probability $1/n$, hence does not flip with probability $(1-1/n)$;

1+1-EA: General Upper bound

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

1. Let i be the number of bit positions in which the current solution x and the global optimum x^* differ;
2. Each bit flips with probability $1/n$, hence does not flip with probability $(1-1/n)$;
3. In order to reach the global optimum the algorithm has to mutate the i bits and leave the $n-i$ bits unchanged;

1+1-EA: General Upper bound

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

1. Let i be the number of bit positions in which the current solution x and the global optimum x^* differ;
2. Each bit flips with probability $1/n$, hence does not flip with probability $(1-1/n)$;
3. In order to reach the global optimum the algorithm has to mutate the i bits and leave the $n-i$ bits unchanged;
4. Then:

$$p(x^*|x) = \left(\frac{1}{n}\right)^i \left(1-\frac{1}{n}\right)^{n-i} \geq \left(\frac{1}{n}\right)^n = n^{-n} (p = n^{-n})$$

1+1-EA: General Upper bound

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

1. Let i be the number of bit positions in which the current solution x and the global optimum x^* differ;
2. Each bit flips with probability $1/n$, hence does not flip with probability $(1-1/n)$;
3. In order to reach the global optimum the algorithm has to mutate the i bits and leave the $n-i$ bits unchanged;
4. Then:

$$p(x^*|x) = \left(\frac{1}{n}\right)^i \left(1-\frac{1}{n}\right)^{n-i} \geq \left(\frac{1}{n}\right)^n = n^{-n} (p = n^{-n})$$

5. it implies an upper bound on the expected runtime of $O(n^n)$ $(E(X) = 1/p = n^n)$ (waiting time argument).

General Upper bound Exercises

Theorem

The expected runtime of the (1+1)-EA with mutation probability $p = 1/2$ *for an arbitrary function defined in* $\{0,1\}^n$ *is* $O(2^n)$

General Upper bound Exercises

Theorem

The expected runtime of the (1+1)-EA with mutation probability $p = 1/2$ *for an arbitrary function defined in* $\{0,1\}^n$ *is* $O(2^n)$

Proof Left as Exercise.

Theorem

The expected runtime of the (1+1)-EA with mutation probability $p = \chi/n$ *for an arbitrary function defined in* $\{0,1\}^n$ *is* $O((n/\chi)^n)$

General Upper bound Exercises

Theorem

The expected runtime of the (1+1)-EA with mutation probability $p = 1/2$ *for an arbitrary function defined in* $\{0,1\}^n$ *is* $O(2^n)$

Proof Left as Exercise.

Theorem

The expected runtime of the (1+1)-EA with mutation probability $p = \chi/n$ *for an arbitrary function defined in* $\{0,1\}^n$ *is* $O((n/\chi)^n)$

Proof Left as Exercise.

Theorem

The expected runtime of RLS for an arbitrary function defined in $\{0,1\}^n$ *is infinite.*

General Upper bound Exercises

Theorem

The expected runtime of the (1+1)-EA with mutation probability $p = 1/2$ *for an arbitrary function defined in* $\{0,1\}^n$ *is* $O(2^n)$

Proof Left as Exercise.

Theorem

The expected runtime of the (1+1)-EA with mutation probability $p = \chi/n$ *for an arbitrary function defined in* $\{0,1\}^n$ *is* $O((n/\chi)^n)$

Proof Left as Exercise.

Theorem

The expected runtime of RLS for an arbitrary function defined in $\{0,1\}^n$ *is infinite.*

Proof Left as Exercise.

1+1-EA: Conclusions & Exercises

In general:

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

1+1-EA: Conclusions & Exercises

In general:

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

What about RLS?

1+1-EA: Conclusions & Exercises

In general:

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

What about RLS?

- Expectation: E[X] = 1

1+1-EA: Conclusions & Exercises

In general:

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

What about RLS?

- Expectation: E[X] = 1
- P(1-bitflip) = 1

1+1-EA: Conclusions & Exercises

In general:

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

What about RLS?

- Expectation: E[X] = 1
- P(1-bitflip) = 1

What about initialisation?

1+1-EA: Conclusions & Exercises

In general:

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

What about RLS?

- Expectation: E[X] = 1
- P(1-bitflip) = 1

What about initialisation?

- How many one-bits in expectation after initialisation?

1+1-EA: Conclusions & Exercises

In general:

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

What about RLS?

- Expectation: E[X] = 1
- P(1-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?

1+1-EA: Conclusions & Exercises

In general:

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

What about RLS?

- Expectation: E[X] = 1
- P(1-bitflip) = 1

What about initialisation?

- How many one-bits in expectation after initialisation?

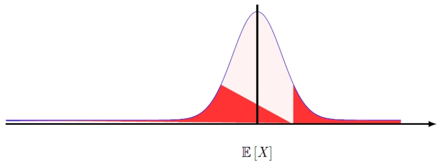
$$E[X] = n \cdot 1/2 = n/2$$

How likely is it that we get exactly $n/2$ one-bits?

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

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

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

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

Chernoff Bounds

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

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 $X_1, X_2, \dots X_n$ 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?

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

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?

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

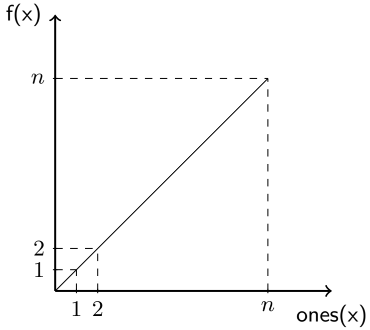
Bitstring of length $n = 100$

$Pr(X_i) = 1/2$ 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

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



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

$\boxed{0}$	$\boxed{0}$	$\boxed{0}$	$\boxed{0}$	$\boxed{0}$	$\boxed{0}$
0	1	2	3	4	5

$p_0 = \frac{6}{6} \quad E(T_0) = \frac{6}{6}$

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

0	0	0	0	0	0
0	1	2	3	4	5

$p_0 = \frac{6}{6} \quad E(T_0) = \frac{6}{6}$

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

0	0	0	0	0	1
0	1	2	3	4	5

$p_0 = \frac{6}{6} \quad E(T_0) = \frac{6}{6}$

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

0	0	0	0	0	1
0	1	2	3	4	5

$p_0 = \frac{6}{6} \quad E(T_0) = \frac{6}{6}$

0	0	0	0	0	1
0	1	2	3	4	5

$p_0 = \frac{6}{6} \quad E(T_0) = \frac{6}{6}$

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

0	0	0	0	0	1
0	1	2	3	4	5

$p_0 = \frac{6}{6} \quad E(T_0) = \frac{6}{6}$

0	0	0	0	0	1
0	1	2	3	4	5

$p_1 = \frac{5}{6} \quad E(T_1) = \frac{6}{5}$

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

0	0	0	0	0	1
0	1	2	3	4	5

$p_0 = \frac{6}{6} \quad E(T_0) = \frac{6}{6}$

0	0	1	0	0	1
0	1	2	3	4	5

$p_1 = \frac{5}{6} \quad E(T_1) = \frac{6}{5}$

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

0	0	0	0	0	1
0	1	2	3	4	5

$p_0 = \frac{6}{6} \quad E(T_0) = \frac{6}{6}$

0	0	1	0	0	1
0	1	2	3	4	5

$p_1 = \frac{5}{6} \quad E(T_1) = \frac{6}{5}$

0	0	1	0	0	1
0	1	2	3	4	5

$p_1 = \frac{5}{6} \quad E(T_1) = \frac{6}{5}$

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

0	0	0	0	0	1
0	1	2	3	4	5

$p_0 = \frac{6}{6} \quad E(T_0) = \frac{6}{6}$

0	0	1	0	0	1
0	1	2	3	4	5

$p_1 = \frac{5}{6} \quad E(T_1) = \frac{6}{5}$

0	0	1	0	0	1
0	1	2	3	4	5

$p_2 = \frac{4}{6} \quad E(T_2) = \frac{6}{4}$

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

0	0	0	0	0	1
0	1	2	3	4	5

$p_0 = \frac{6}{6} \quad E(T_0) = \frac{6}{6}$

0	0	1	0	0	1
0	1	2	3	4	5

$p_1 = \frac{5}{6} \quad E(T_1) = \frac{6}{5}$

1	0	1	0	0	1
0	1	2	3	4	5

$p_2 = \frac{4}{6} \quad E(T_2) = \frac{6}{4}$

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

$$\begin{array}{cccccc} \boxed{0}&\boxed{0}&\boxed{0}&\boxed{0}&\boxed{0}&\boxed{1}\\ 0&1&2&3&4&5\end{array} \qquad p_0=\frac{6}{6} \quad E(T_0)=\frac{6}{6}$$

$$\begin{array}{cccccc} \boxed{0}&\boxed{0}&\boxed{1}&\boxed{0}&\boxed{0}&\boxed{1}\\ 0&1&2&3&4&5\end{array} \qquad p_1=\frac{5}{6} \quad E(T_1)=\frac{6}{5}$$

$$\begin{array}{cccccc} \boxed{1}&\boxed{0}&\boxed{1}&\boxed{0}&\boxed{0}&\boxed{1}\\ 0&1&2&3&4&5\end{array} \qquad p_2=\frac{4}{6} \quad E(T_2)=\frac{6}{4}$$

$$\begin{array}{cccccc} \boxed{1}&\boxed{0}&\boxed{1}&\boxed{0}&\boxed{0}&\boxed{1}\\ 0&1&2&3&4&5\end{array} \qquad p_2=\frac{4}{6} \quad E(T_2)=\frac{6}{4}$$

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

$$\begin{array}{cccccc} \boxed{0}&\boxed{0}&\boxed{0}&\boxed{0}&\boxed{0}&\boxed{1}\\ 0&1&2&3&4&5\end{array} \qquad p_0=\frac{6}{6} \quad E(T_0)=\frac{6}{6}$$

$$\begin{array}{cccccc} \boxed{0}&\boxed{0}&\boxed{1}&\boxed{0}&\boxed{0}&\boxed{1}\\ 0&1&2&3&4&5\end{array} \qquad p_1=\frac{5}{6} \quad E(T_1)=\frac{6}{5}$$

$$\begin{array}{cccccc} \boxed{1}&\boxed{0}&\boxed{1}&\boxed{0}&\boxed{0}&\boxed{1}\\ 0&1&2&3&4&5\end{array} \qquad p_2=\frac{4}{6} \quad E(T_2)=\frac{6}{4}$$

$$\begin{array}{cccccc} \boxed{1}&\boxed{0}&\boxed{1}&\boxed{0}&\boxed{0}&\boxed{1}\\ 0&1&2&3&4&5\end{array} \qquad p_3=\frac{3}{6} \quad E(T_0)=\frac{6}{3}$$

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

$$\begin{array}{cccccc} \boxed{0}&\boxed{0}&\boxed{0}&\boxed{0}&\boxed{0}&\boxed{1}\\ 0&1&2&3&4&5\end{array} \qquad p_0=\frac{6}{6} \quad E(T_0)=\frac{6}{6}$$

$$\begin{array}{cccccc} \boxed{0}&\boxed{0}&\boxed{1}&\boxed{0}&\boxed{0}&\boxed{1}\\ 0&1&2&3&4&5\end{array} \qquad p_1=\frac{5}{6} \quad E(T_1)=\frac{6}{5}$$

$$\begin{array}{cccccc} \boxed{1}&\boxed{0}&\boxed{1}&\boxed{0}&\boxed{0}&\boxed{1}\\ 0&1&2&3&4&5\end{array} \qquad p_2=\frac{4}{6} \quad E(T_2)=\frac{6}{4}$$

$$\begin{array}{cccccc} \boxed{1}&\boxed{0}&\boxed{1}&\boxed{0}&\boxed{0}&\boxed{0}\\ 0&1&2&3&4&5\end{array} \qquad p_3=\frac{3}{6} \quad E(T_3)=\frac{6}{3}$$

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

$$\begin{array}{cccccc} \boxed{0}&\boxed{0}&\boxed{0}&\boxed{0}&\boxed{0}&\boxed{1}\\ 0&1&2&3&4&5\end{array} \qquad p_0=\frac{6}{6} \quad E(T_0)=\frac{6}{6}$$

$$\begin{array}{cccccc} \boxed{0}&\boxed{0}&\boxed{1}&\boxed{0}&\boxed{0}&\boxed{1}\\ 0&1&2&3&4&5\end{array} \qquad p_1=\frac{5}{6} \quad E(T_1)=\frac{6}{5}$$

$$\begin{array}{cccccc} \boxed{1}&\boxed{0}&\boxed{1}&\boxed{0}&\boxed{0}&\boxed{1}\\ 0&1&2&3&4&5\end{array} \qquad p_2=\frac{4}{6} \quad E(T_2)=\frac{6}{4}$$

$$\begin{array}{cccccc} \boxed{1}&\boxed{0}&\boxed{1}&\boxed{0}&\boxed{0}&\boxed{1}\\ 0&1&2&3&4&5\end{array} \qquad p_2=\frac{4}{6} \quad E(T_2)=\frac{6}{4}$$

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

0	0	0	0	0	1
0	1	2	3	4	5

$p_0 = \frac{6}{6} \quad E(T_0) = \frac{6}{6}$

0	0	1	0	0	1
0	1	2	3	4	5

$p_1 = \frac{5}{6} \quad E(T_1) = \frac{6}{5}$

1	0	1	0	0	1
0	1	2	3	4	5

$p_2 = \frac{4}{6} \quad E(T_2) = \frac{6}{4}$

1	0	1	0	0	1
0	1	2	3	4	5

$p_3 = \frac{3}{6} \quad E(T_3) = \frac{6}{3}$

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

0	0	0	0	0	1
0	1	2	3	4	5

$p_0 = \frac{6}{6} \quad E(T_0) = \frac{6}{6}$

0	0	1	0	0	1
0	1	2	3	4	5

$p_1 = \frac{5}{6} \quad E(T_1) = \frac{6}{5}$

1	0	1	0	0	1
0	1	2	3	4	5

$p_2 = \frac{4}{6} \quad E(T_2) = \frac{6}{4}$

1	0	1	0	1	1
0	1	2	3	4	5

$p_3 = \frac{3}{6} \quad E(T_3) = \frac{6}{3}$

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

0	0	0	0	0	1
0	1	2	3	4	5

$p_0 = \frac{6}{6} \quad E(T_0) = \frac{6}{6}$

0	0	1	0	0	1
0	1	2	3	4	5

$p_1 = \frac{5}{6} \quad E(T_1) = \frac{6}{5}$

1	0	1	0	0	1
0	1	2	3	4	5

$p_2 = \frac{4}{6} \quad E(T_2) = \frac{6}{4}$

1	0	1	0	1	1
0	1	2	3	4	5

$p_3 = \frac{3}{6} \quad E(T_3) = \frac{6}{3}$

1	0	1	0	1	1
0	1	2	3	4	5

$p_3 = \frac{3}{6} \quad E(T_3) = \frac{6}{3}$

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

0	0	0	0	0	1
0	1	2	3	4	5

$p_0 = \frac{6}{6} \quad E(T_0) = \frac{6}{6}$

0	0	1	0	0	1
0	1	2	3	4	5

$p_1 = \frac{5}{6} \quad E(T_1) = \frac{6}{5}$

1	0	1	0	0	1
0	1	2	3	4	5

$p_2 = \frac{4}{6} \quad E(T_2) = \frac{6}{4}$

1	0	1	0	1	1
0	1	2	3	4	5

$p_3 = \frac{3}{6} \quad E(T_3) = \frac{6}{3}$

1	0	1	0	1	1
0	1	2	3	4	5

$p_4 = \frac{2}{6} \quad E(T_4) = \frac{6}{2}$

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

0	0	0	0	0	1	$p_0 = \frac{6}{6}$	$E(T_0) = \frac{6}{6}$
0	1	2	3	4	5		
0	0	1	0	0	1	$p_1 = \frac{5}{6}$	$E(T_1) = \frac{6}{5}$
0	1	2	3	4	5		
1	0	1	0	0	1	$p_2 = \frac{4}{6}$	$E(T_2) = \frac{6}{4}$
0	1	2	3	4	5		
1	0	1	0	1	1	$p_3 = \frac{3}{6}$	$E(T_3) = \frac{6}{3}$
0	1	2	3	4	5		
1	1	1	0	1	1	$p_4 = \frac{2}{6}$	$E(T_4) = \frac{6}{2}$
0	1	2	3	4	5		

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

0	0	0	0	0	1	$p_0 = \frac{6}{6}$	$E(T_0) = \frac{6}{6}$
0	1	2	3	4	5		
0	0	1	0	0	1	$p_1 = \frac{5}{6}$	$E(T_1) = \frac{6}{5}$
0	1	2	3	4	5		
1	0	1	0	0	1	$p_2 = \frac{4}{6}$	$E(T_2) = \frac{6}{4}$
0	1	2	3	4	5		
1	0	1	0	1	1	$p_3 = \frac{3}{6}$	$E(T_3) = \frac{6}{3}$
0	1	2	3	4	5		
1	1	1	0	1	1	$p_4 = \frac{2}{6}$	$E(T_4) = \frac{6}{2}$
0	1	2	3	4	5		
1	1	1	0	1	1	$p_4 = \frac{2}{6}$	$E(T_4) = \frac{6}{2}$
0	1	2	3	4	5		

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

0	0	0	0	0	1	$p_0 = \frac{6}{6}$	$E(T_0) = \frac{6}{6}$
0	1	2	3	4	5		
0	0	1	0	0	1	$p_1 = \frac{5}{6}$	$E(T_1) = \frac{6}{5}$
0	1	2	3	4	5		
1	0	1	0	0	1	$p_2 = \frac{4}{6}$	$E(T_2) = \frac{6}{4}$
0	1	2	3	4	5		
1	0	1	0	1	1	$p_3 = \frac{3}{6}$	$E(T_3) = \frac{6}{3}$
0	1	2	3	4	5		
1	1	1	0	1	1	$p_4 = \frac{2}{6}$	$E(T_4) = \frac{6}{2}$
0	1	2	3	4	5		
1	1	1	0	1	1	$p_5 = \frac{1}{6}$	$E(T_5) = \frac{6}{1}$
0	1	2	3	4	5		

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

0	0	0	0	0	1	$p_0 = \frac{6}{6}$	$E(T_0) = \frac{6}{6}$
0	1	2	3	4	5		
0	0	1	0	0	1	$p_1 = \frac{5}{6}$	$E(T_1) = \frac{6}{5}$
0	1	2	3	4	5		
1	0	1	0	0	1	$p_2 = \frac{4}{6}$	$E(T_2) = \frac{6}{4}$
0	1	2	3	4	5		
1	0	1	0	1	1	$p_3 = \frac{3}{6}$	$E(T_3) = \frac{6}{3}$
0	1	2	3	4	5		
1	1	1	0	1	1	$p_4 = \frac{2}{6}$	$E(T_4) = \frac{6}{2}$
0	1	2	3	4	5		
1	1	1	1	1	1	$p_5 = \frac{1}{6}$	$E(T_5) = \frac{6}{1}$
0	1	2	3	4	5		

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

0	0	0	0	0	1
0	1	2	3	4	5

$p_0 = \frac{6}{6}$ $E(T_0) = \frac{6}{6}$

0	0	1	0	0	1
0	1	2	3	4	5

$p_1 = \frac{5}{6}$ $E(T_1) = \frac{6}{5}$

1	0	1	0	0	1
0	1	2	3	4	5

$p_2 = \frac{4}{6}$ $E(T_2) = \frac{6}{4}$

1	0	1	0	1	1
0	1	2	3	4	5

$p_3 = \frac{3}{6}$ $E(T_3) = \frac{6}{3}$

1	1	1	0	1	1
0	1	2	3	4	5

$p_4 = \frac{2}{6}$ $E(T_4) = \frac{6}{2}$

1	1	1	1	1	1
0	1	2	3	4	5

$p_5 = \frac{1}{6}$ $E(T_5) = \frac{6}{1}$

$$E(T) = E(T_0) + E(T_1) + \cdots + E(T_5) = 1/p_0 + 1/p_1 + \cdots + 1/p_5 =$$

$$= \sum_{i=0}^{5} \frac{1}{p_i} = \sum_{i=0}^{5} \frac{6}{i} = 6 \sum_{i=1}^{6} \frac{1}{i} = 6 \cdot 2.45 = 14.7$$

RLS for ONEMAX(ONEMAX$(x)=\sum_{i=1}^{n} x[i]$) : Generalisation

0	0	0	0	0	0	0	0	0	0
0	1	2	3						n

$p_0 = \frac{n}{n}$ $E(T_0) = \frac{n}{n}$

RLS for ONEMAX(ONEMAX$(x)=\sum_{i=1}^{n} x[i]$) : Generalisation

0	0	0	0	0	0	0	0	0	0
0	1	2	3		5				n

$p_0 = \frac{n}{n}$ $E(T_0) = \frac{n}{n}$

RLS for ONEMAX(ONEMAX$(x)=\sum_{i=1}^{n} x[i]$) : Generalisation

0	0	0	0	0	1	0	0	0	0
0	1	2	3						n

$p_0 = \frac{n}{n}$ $E(T_0) = \frac{n}{n}$

RLS for ONEMAX(ONEMAX$(x)=\sum_{i=1}^{n} x[i]$) : Generalisation

0	0	0	0	0	1	0	0	0	0	$p_0 = \frac{n}{n}$	$E(T_0) = \frac{n}{n}$
0	1	2	3						n		
0	0	0	0	0	1	0	0	0	0	$p_0 = \frac{n}{n}$	$E(T_0) = \frac{n}{n}$
0	1	2	3						n		

RLS for ONEMAX(ONEMAX$(x)=\sum_{i=1}^{n} x[i]$) : Generalisation

0	0	0	0	0	1	0	0	0	0	$p_0 = \frac{n}{n}$	$E(T_0) = \frac{n}{n}$
0	1	2	3						n		
0	0	0	0	0	1	0	0	0	0	$p_1 = \frac{n-1}{n}$	$E(T_1) = \frac{n}{n-1}$
0	1	2	3						n		

RLS for ONEMAX(ONEMAX$(x)=\sum_{i=1}^{n} x[i]$) : Generalisation

0	0	0	0	0	1	0	0	0	0	$p_0 = \frac{n}{n}$	$E(T_0) = \frac{n}{n}$
0	1	2	3						n		
0	0	1	0	0	1	0	0	0	0	$p_1 = \frac{n-1}{n}$	$E(T_1) = \frac{n}{n-1}$
0	1	2	3						n		

RLS for ONEMAX(ONEMAX$(x)=\sum_{i=1}^{n} x[i]$) : Generalisation

0	0	0	0	0	1	0	0	0	0	$p_0 = \frac{n}{n}$	$E(T_0) = \frac{n}{n}$
0	1	2	3						n		
0	0	1	0	0	1	0	0	0	0	$p_1 = \frac{n-1}{n}$	$E(T_1) = \frac{n}{n-1}$
0	1	2	3						n		
0	0	1	0	0	1	0	0	0	0	$p_1 = \frac{n-1}{n}$	$E(T_1) = \frac{n}{n-1}$
0	1	2	3						n		

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

0	0	0	0	0	1	0	0	0	0
0	1	2	3						n

$p_0 = \frac{n}{n}$ $E(T_0) = \frac{n}{n}$

0	0	1	0	0	1	0	0	0	0
0	1	2	3						n

$p_1 = \frac{n-1}{n}$ $E(T_1) = \frac{n}{n-1}$

0	0	1	0	0	1	0	0	0	0
0	1	2	3						n

$p_2 = \frac{n-2}{n}$ $E(T_2) = \frac{n}{n-2}$

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

0	0	0	0	0	1	0	0	0	0
0	1	2	3						n

$p_0 = \frac{n}{n}$ $E(T_0) = \frac{n}{n}$

0	0	1	0	0	1	0	0	0	0
0	1	2	3						n

$p_1 = \frac{n-1}{n}$ $E(T_1) = \frac{n}{n-1}$

0	0	1	0	0	1	0	0	0	1
0	1	2	3						n

$p_2 = \frac{n-2}{n}$ $E(T_2) = \frac{n}{n-2}$

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

0	0	0	0	0	1	0	0	0	0
0	1	2	3						n

$p_0 = \frac{n}{n}$ $E(T_0) = \frac{n}{n}$

0	0	1	0	0	1	0	0	0	0
0	1	2	3						n

$p_1 = \frac{n-1}{n}$ $E(T_1) = \frac{n}{n-1}$

0	0	1	0	0	1	0	0	0	1
0	1	2	3						n

$p_2 = \frac{n-2}{n}$ $E(T_2) = \frac{n}{n-2}$

1	1	1	1	0	1	1	1	1	1
0	1	2	3						n

$p_{n-1} = \frac{1}{n}$ $E(T_{n-1}) = \frac{n}{1}$

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

0	0	0	0	0	1	0	0	0	0
0	1	2	3						n

$p_0 = \frac{n}{n}$ $E(T_0) = \frac{n}{n}$

0	0	1	0	0	1	0	0	0	0
0	1	2	3						n

$p_1 = \frac{n-1}{n}$ $E(T_1) = \frac{n}{n-1}$

0	0	1	0	0	1	0	0	0	1
0	1	2	3						n

$p_2 = \frac{n-2}{n}$ $E(T_2) = \frac{n}{n-2}$

1	1	1	1	0	1	1	1	1	1
0	1	2	3	4					n

$p_{n-1} = \frac{1}{n}$ $E(T_{n-1}) = \frac{n}{1}$

RLS for ONEMAX(ONEMAX$(x)=\sum_{i=1}^{n} x[i]$) : Generalisation

$$\begin{array}{cccccccccc} \boxed{0} & \boxed{0} & \boxed{0} & \boxed{0} & \boxed{0} & \boxed{1} & \boxed{0} & \boxed{0} & \boxed{0} & \boxed{0} \\ 0 & 1 & 2 & 3 & & & & & & n \end{array} \qquad p_0 = \tfrac{n}{n} \qquad E(T_0) = \tfrac{n}{n}$$

$$\begin{array}{cccccccccc} \boxed{0} & \boxed{0} & \boxed{1} & \boxed{0} & \boxed{0} & \boxed{1} & \boxed{0} & \boxed{0} & \boxed{0} & \boxed{0} \\ 0 & 1 & 2 & 3 & & & & & & n \end{array} \qquad p_1 = \tfrac{n-1}{n} \qquad E(T_1) = \tfrac{n}{n-1}$$

$$\begin{array}{cccccccccc} \boxed{0} & \boxed{0} & \boxed{1} & \boxed{0} & \boxed{0} & \boxed{1} & \boxed{0} & \boxed{0} & \boxed{0} & \boxed{1} \\ 0 & 1 & 2 & 3 & & & & & & n \end{array} \qquad p_2 = \tfrac{n-2}{n} \qquad E(T_2) = \tfrac{n}{n-2}$$

$$\begin{array}{cccccccccc} \boxed{1} & \boxed{1} & \boxed{1} & \boxed{1} & \boxed{1} & \boxed{1} & \boxed{1} & \boxed{1} & \boxed{1} & \boxed{1} \\ 0 & 1 & 2 & 3 & & & & & & n \end{array} \qquad p_{n-1} = \tfrac{1}{n} \qquad E(T_{n-1}) = \tfrac{n}{1}$$

RLS for ONEMAX(ONEMAX$(x)=\sum_{i=1}^{n} x[i]$) : Generalisation

$$\begin{array}{cccccccccc} \boxed{0} & \boxed{0} & \boxed{0} & \boxed{0} & \boxed{0} & \boxed{1} & \boxed{0} & \boxed{0} & \boxed{0} & \boxed{0} \\ 0 & 1 & 2 & 3 & & & & & & n \end{array} \qquad p_0 = \tfrac{n}{n} \qquad E(T_0) = \tfrac{n}{n}$$

$$\begin{array}{cccccccccc} \boxed{0} & \boxed{0} & \boxed{1} & \boxed{0} & \boxed{0} & \boxed{1} & \boxed{0} & \boxed{0} & \boxed{0} & \boxed{0} \\ 0 & 1 & 2 & 3 & & & & & & n \end{array} \qquad p_1 = \tfrac{n-1}{n} \qquad E(T_1) = \tfrac{n}{n-1}$$

$$\begin{array}{cccccccccc} \boxed{0} & \boxed{0} & \boxed{1} & \boxed{0} & \boxed{0} & \boxed{1} & \boxed{0} & \boxed{0} & \boxed{0} & \boxed{1} \\ 0 & 1 & 2 & 3 & & & & & & n \end{array} \qquad p_2 = \tfrac{n-2}{n} \qquad E(T_2) = \tfrac{n}{n-2}$$

$$\begin{array}{cccccccccc} \boxed{1} & \boxed{1} & \boxed{1} & \boxed{1} & \boxed{1} & \boxed{1} & \boxed{1} & \boxed{1} & \boxed{1} & \boxed{1} \\ 0 & 1 & 2 & 3 & & & & & & n \end{array} \qquad p_{n-1} = \tfrac{1}{n} \qquad E(T_{n-1}) = \tfrac{n}{1}$$

$$E(T) = E(T_0) + E(T_1) + \cdots + E(T_{n-1}) = 1/p_1 + 1/p_2 + \cdots + 1/p_{n-1} =$$

$$= \sum_{i=0}^{n-1} \frac{1}{p_i} = \sum_{i=1}^{n} \frac{n}{i} = n \sum_{i=1}^{n} \frac{1}{i} = n \cdot H(n) = n \log n + \Theta(n) = O(n \log n)$$

Coupon collector's problem

The 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

$$E(T) = \sum_{i=0}^{n-1} \frac{1}{p_{i+1}} = \sum_{i=0}^{n-1} \frac{n}{n-i} = n \sum_{i=0}^{n-1} \frac{1}{i} =$$

$$= n(\log n + \Theta(1)) = n \log n + O(n).$$

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

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

$\frac{1}{n}$	Probability of choosing a given coupon
$1 - \frac{1}{n}$	Probability of not choosing a given coupon
$\left(1 - \frac{1}{n}\right)^t$	Probability of not choosing a given coupon for t rounds

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

$n \cdot \left(1 - \frac{1}{n}\right)^t$ (Union Bound)

Hence, for $t = cn \ln n$

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

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 $\epsilon > 0$

$$Pr(T < (1-\epsilon)(n-1) \ln n) \leq exp(-n^{\epsilon})$$

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 $\textsc{OneMax}$*is* $\Theta(n \ln n)$. *Furthermore,*

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

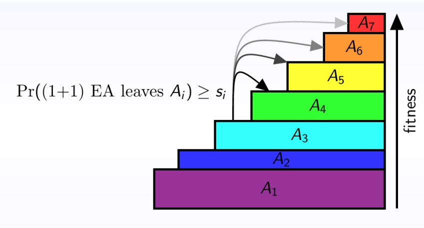
and

$$Pr(T < (1-\epsilon)(n-1) \ln n) \leq exp(-n^{\epsilon})$$

What about the (1+1)-EA?

Artificial Fitness Levels

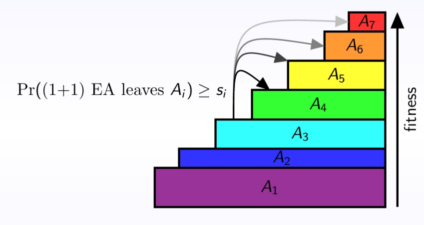
Observation Due to elitism, fitness is monotone increasing



D. Sudholt, Tutorial 2011

Artificial Fitness Levels

Observation Due to elitism, fitness is monotone increasing



D. Sudholt, Tutorial 2011

Idea Divide the search space $|S| = 2^n$ into $m < 2^n$ sets $A_1, \dots A_m$ such that:

1. $\forall i \neq j: \qquad A_i \cap A_j = \emptyset$
2. $\bigcup_{i=0}^{m} A_i = \{0,1\}^n$
3. for all points $a \in A_i$ and $b \in A_j$ it happens that $f(a) < f(b)$ if $i < j$.

requirement A_m contains only optimal search points.

Artificial Fitness Levels [Droste et al., 2002]

Idea Divide the search space $|S| = 2^n$ into $m < 2^n$ sets $A_1, \dots A_m$ such that:

1. $\forall i \neq j: \qquad A_i \cap A_j = \emptyset$
2. $\bigcup_{i=0}^{m} A_i = \{0,1\}^n$
3. for all points $a \in A_i$ and $b \in A_j$ it happens that $f(a) < f(b)$ if $i < j$.

requirement A_m contains only optimal search points.

Artificial Fitness Levels [Droste et al., 2002]

Idea Divide the search space $|S| = 2^n$ into $m < 2^n$ sets $A_1, \dots A_m$ such that:

1. $\forall i \neq j: \qquad A_i \cap A_j = \emptyset$
2. $\bigcup_{i=0}^{m} A_i = \{0,1\}^n$
3. for all points $a \in A_i$ and $b \in A_j$ it happens that $f(a) < f(b)$ if $i < j$.

requirement A_m contains only optimal search points.

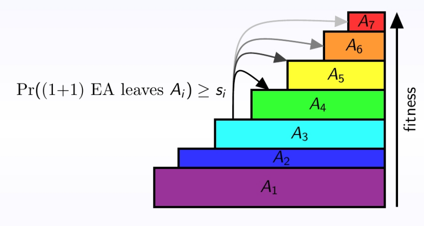
Then:

s_i probability that point in A_i is mutated to a point in A_j with $j > i$.

Expected time: $E(T) \leq \sum_i \frac{1}{s_i}$

Very simple, yet often powerful method for upper bounds

Artificial Fitness Levels



D. Sudholt, Tutorial 2011

Let:

- $p(A_i)$ be the probability that a random initial point belongs to level A_i
- s_i be the probability to leave level A_i for A_j with $j > i$
- **Then:**

$$E(T) \leq \sum_{1 \leq i \leq m-1} p(A_i) \cdot \left(\frac{1}{s_i} + \dots + \frac{1}{s_{m-1}} \right) \leq \left(\frac{1}{s_1} + \dots + \frac{1}{s_{m-1}} \right) = \sum_{i=1}^{m-1} \frac{1}{s_i}$$

- Inequality 1: Law of total probability $(E(T) = \sum_i Pr(F) \cdot E(T|F))$
- Inequality 2: Trivial!

(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, which occurs with probability

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

(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, which occurs with probability

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

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 $n \ln n$?

(1+1)-EA lower bound for OneMax

Theorem (Droste,Jansen,Wegener, 2002)

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

Proof Idea

(1+1)-EA lower bound for OneMax

Theorem (Droste,Jansen,Wegener, 2002)

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

Proof Idea

1. At most $n/2$ one-bits are created during initialisation with probability at least $1/2$ **(By symmetry of the binomial distribution)**.

(1+1)-EA lower bound for OneMax

Theorem (Droste,Jansen,Wegener, 2002)

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

Proof Idea

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

Lower bound for OneMax

Theorem (Droste,Jansen,Wegener, 2002)

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

Proof of 2.

$1 - 1/n$	a given bit does not flip

Lower bound for OneMax

Theorem (Droste,Jansen,Wegener, 2002)

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

Proof of 2.

$1 - 1/n$	a given bit does not flip
$(1 - 1/n)^t$	a given bit does not flip in t steps

Lower bound for OneMax

Theorem (Droste,Jansen,Wegener, 2002)

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

Proof of 2.

$1 - 1/n$	a given bit does not flip
$(1 - 1/n)^t$	a given bit does not flip in t steps
$1 - (1 - 1/n)^t$	it flips at least once in t steps

Lower bound for OneMax

Theorem (Droste,Jansen,Wegener, 2002)

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

Proof of 2.

$1 - 1/n$	a given bit does not flip
$(1 - 1/n)^t$	a given bit does not flip in t steps
$1 - (1 - 1/n)^t$	it flips at least once in t steps
$(1 - (1 - 1/n)^t)^{n/2}$	$n/2$ bits flip at least once in t steps

---

Lower bound for OneMax

Theorem (Droste,Jansen,Wegener, 2002)

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

Proof of 2.

$1 - 1/n$	a given bit does not flip
$(1 - 1/n)^t$	a given bit does not flip in t steps
$1 - (1 - 1/n)^t$	it flips at least once in t steps
$(1 - (1 - 1/n)^t)^{n/2}$	$n/2$ bits flip at least once in t steps
$1 - [1 - (1 - 1/n)^t]^{n/2}$	at least one of the $n/2$ bits does not flip in t steps

Lower bound for OneMax

Theorem (Droste,Jansen,Wegener, 2002)

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

Proof of 2.

$1 - 1/n$	a given bit does not flip
$(1-1/n)^t$	a given bit does not flip in t steps
$1-(1-1/n)^t$	it flips at least once in t steps
$(1-(1-1/n)^t)^{n/2}$	$n/2$ bits flip at least once in t steps
$1-[1-(1-1/n)^t]^{n/2}$	at least one of the $n/2$ bits does not flip in t steps

Set $t = (n-1)\log n$. Then:

$$1 - [1-(1-1/n)^t]^{n/2} = 1 - [1 - (1-1/n)^{(n-1)\log n}]^{n/2} \geq$$
$$\geq 1 - [1 - (1/e)^{\log n}]^{n/2} = 1 - [1-1/n]^{n/2} =$$
$$= 1 - [1-1/n]^{n \cdot 1/2} \geq 1 - (2e)^{-1/2} = c$$

Lower bound for OneMax(2)

Theorem (Droste, Jansen, Wegener, 2002)

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

Proof

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

Lower bound for OneMax(2)

Theorem (Droste, Jansen, Wegener, 2002)

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

Proof

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

The Expected runtime is:

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

First inequality: law of total probability

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

Artificial Fitness Levels Exercises: $\left(\textsc{LeadingOnes}(x) = \sum_{i=1}^{n} \prod_{j=1}^{i} x[j]\right)$

Theorem

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

Artificial Fitness Levels Exercises: $\left(\textsc{LeadingOnes}(x) = \sum_{i=1}^{n} \prod_{j=1}^{i} x[j]\right)$

Theorem

The expected runtime of RLS for $\textsc{LeadingOnes}$ *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 $s_i^{-1} = n$
- $E(T) \leq \sum_{i=1}^{n-1} s_i^{-1} = \sum_{i=1}^{n} n = O(n^2)$

Artificial Fitness Levels Exercises: $\left(\textsc{LeadingOnes}(x) = \sum_{i=1}^{n} \prod_{j=1}^{i} x[j]\right)$

Theorem

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

Artificial Fitness Levels Exercises: $\left(\textsc{LeadingOnes}(x) = \sum_{i=1}^{n} \prod_{j=1}^{i} x[j]\right)$

Theorem

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

Proof Left as Exercise.

Fitness Levels Advanced Exercises (Populations)

Theorem

The expected runtime of (1+λ)-EA for $\textsc{LeadingOnes}$ *is* $O(\lambda n + n^2)$ *[Jansen et al., 2005].*

Fitness Levels Advanced Exercises (Populations)

Theorem

The expected runtime of (1+λ)-EA for $\textsc{LeadingOnes}$ *is* $O(\lambda n + n^2)$ *[Jansen et al., 2005].*

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 $i+1$
- $s_i = 1 - \left(1 - \frac{1}{en}\right)^{\lambda} \geq 1 - e^{-\lambda/(en)}$
  1. $s_i \geq 1 - \frac{1}{e}$ Case 1: $\lambda \geq en$
  2. $s_i \geq \frac{\lambda}{2en}$ Case 2: $\lambda < en$
- $E(T) \leq \lambda \cdot \sum_{i=1}^{n-1} s_i^{-1} \leq \lambda\left(\left(\sum_{i=1}^{n} \frac{1}{c}\right) + \left(\sum_{i=1}^{n} \frac{2en}{\lambda}\right)\right) =$
  $O\left(\lambda \cdot \left(n + \frac{n^2}{\lambda}\right)\right) = O(\lambda \cdot n + n^2)$

Fitness Levels Advanced Exercises (Populations)

Theorem

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

Fitness Levels Advanced Exercises (Populations)

Theorem

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

Proof Left as Exercise.

Theorem

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

Fitness Levels Advanced Exercises (Populations)

Theorem

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

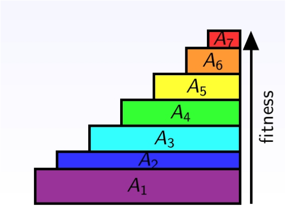
Proof Left as Exercise.

Theorem

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

Proof Left as Exercise.

Artificial Fitness Levels for Populations



D. Sudholt, Tutorial 2011

Let:

- T_o 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:**

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

Applications to $(\mu{+}1)$-EA

Theorem

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

Applications to $(\mu{+}1)$-EA

Theorem

The expected runtime of $(\mu{+}1)$-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

---

Applications to $(\mu{+}1)$-EA

Theorem

The expected runtime of $(\mu{+}1)$-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

Applications to (μ+1)-EA

Theorem

The expected runtime of (μ+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 $\chi(i) = n/\ln n$

Applications to (μ+1)-EA

Theorem

The expected runtime of (μ+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 $\chi(i) = n/\ln n$
- Given j copies of the best individual another replica is created with probability $\frac{j}{\mu}\left(1-\frac{1}{n}\right)^n \geq \frac{j}{2e\mu}$

Applications to (μ+1)-EA

Theorem

The expected runtime of (μ+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 $\chi(i) = n/\ln n$
- Given j copies of the best individual another replica is created with probability $\frac{j}{\mu}\left(1-\frac{1}{n}\right)^n \geq \frac{j}{2e\mu}$
- $T_o \leq \sum_{j=1}^{n/\ln n} \frac{2e\mu}{j} \leq 2e\mu \ln n$

Applications to (μ+1)-EA

Theorem

The expected runtime of (μ+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 $\chi(i) = n/\ln n$
- Given j copies of the best individual another replica is created with probability $\frac{j}{\mu}\left(1-\frac{1}{n}\right)^n \geq \frac{j}{2e\mu}$
- $T_o \leq \sum_{j=1}^{n/\ln n} \frac{2e\mu}{j} \leq 2e\mu \ln n$
  1. $s_i \geq \frac{n/\ln n}{\mu} \cdot \frac{1}{en} = \frac{1}{e\mu \ln n}$ Case 1: $\mu > \frac{n}{\ln n}$
  2. $s_i \geq \frac{n/\ln n}{\mu} \cdot \frac{1}{en} \geq \frac{1}{en}$ Case 2: $\mu \leq \frac{n}{\ln n}$

Applications to (μ+1)-EA

Theorem

The expected runtime of (μ+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 $\chi(i) = n/\ln n$
- Given j copies of the best individual another replica is created with probability $\frac{j}{\mu}\left(1 - \frac{1}{n}\right)^n \geq \frac{j}{2e\mu}$
- $T_o \leq \sum_{j=1}^{n/\ln n} \frac{2e\mu}{j} \leq 2e\mu \ln n$
  1. $s_i \geq \frac{n/\ln n}{\mu} \cdot \frac{1}{en} = \frac{1}{e\mu \ln n}$ Case 1: $\mu > \frac{n}{\ln n}$
  2. $s_i \geq \frac{n/\ln n}{\mu} \cdot \frac{1}{en} \geq \frac{1}{en}$ Case 2: $\mu \leq \frac{n}{\ln n}$
- $E(T) \leq \sum_{i=1}^{n-1}(T_o + s_i^{-1}) \leq \sum_{i=1}^{n}\left(2e\mu \ln n + (en + e\mu \ln n)\right) = n \cdot \left(2e\mu \ln n + (en + e\mu \ln n)\right) = O(n\mu \ln n + n^2)$

Populations Fitness Levels: Exercise

Theorem

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

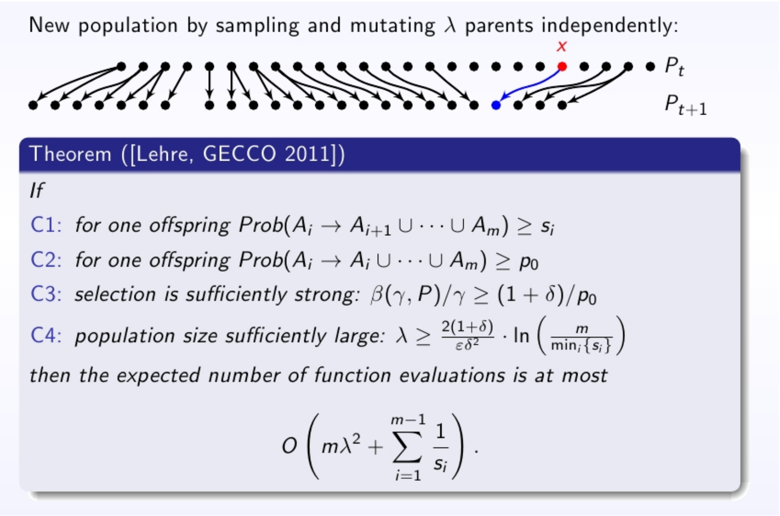
Populations Fitness Levels: Exercise

Theorem

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

Proof Left as Exercise.

Advanced: Fitness Levels for non-Elitist Populations [Lehre, 2011]



New population by sampling and mutating λ parents independently:

Theorem ([Lehre, GECCO 2011])

If

C1: *for one offspring* $Prob(A_i \rightarrow A_{i+1} \cup \cdots \cup A_m) \geq s_i$

C2: *for one offspring* $Prob(A_i \rightarrow A_i \cup \cdots \cup A_m) \geq p_0$

C3: *selection is sufficiently strong:* $\beta(\gamma, P)/\gamma \geq (1 + \delta)/p_0$

C4: *population size sufficiently large:* $\lambda \geq \frac{2(1+\delta)}{\varepsilon\delta^2} \cdot \ln\left(\frac{m}{\min_i\{s_i\}}\right)$

then the expected number of function evaluations is at most

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

Advanced: Fitness Levels for Lower Bounds [Sudholt, 2010]

Lower bounds with fitness levels [Sudholt, 2010]

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

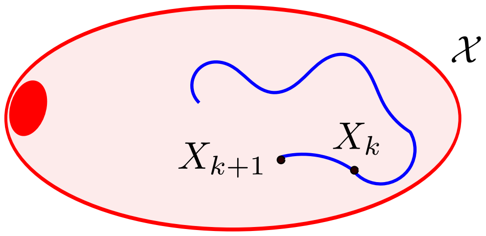
$$\mathrm{E}(\text{optimization time}) \geq \sum_{i=1}^{m-1} \mathrm{Prob}(\mathcal{A} \text{ starts in } A_i) \cdot \chi \sum_{j=i}^{m-1} \frac{1}{u_i}.$$

$u_i :=$ 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.

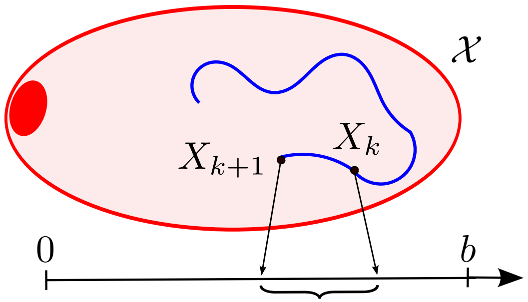
What is Drift[1] Analysis?



---

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

What is Drift[1] Analysis?



$$\Delta_k := d(X_{k+1}) - d(X_k)$$

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

from properties of Δ.

---

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

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?

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: Formalisation

- Define a distance function $d(x)$ to measure the distance from the hotel;

  $$d(x) = x, \qquad x \in \{0, \dots, n\}$$

  (In our case the distance is simply the number of metres from the hotel).
- Estimate the expected "speed" (drift), the expected decrease in distance in one step from the goal;

  $$d(X_t) - d(X_{t+1}) = \begin{cases} 0, \text{if } X_t = 0, \\ 1, \text{if } X_t \in \{1, \dots, n\} \end{cases}$$

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

$$E(T) = \frac{maximum \quad distance}{drift} = \frac{n}{1} = n$$

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?

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;

$$d(x) = x, \qquad x \in \{0, \dots, n\}$$

  (simply the number of metres from the hotel).
- Estimate the expected "speed" (drift), the expected decrease in distance in one step from the goal;

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

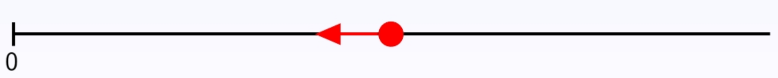
- The expected dicrease in distance (drift) is:

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

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

$$E(T) = \frac{maximum \quad distance}{drift} = \frac{n}{0.2} = 5n$$

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

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

and

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

Drift Analysis for Leading Ones

Theorem

The expected time for the (1+1)-EA to optimise LEADINGONES *is $O(n^2)$*

Proof

Drift Analysis for Leading Ones

Theorem

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

Proof

1. Let $d(X_t) = i$ where i is the number of missing leading ones;

Drift Analysis for Leading Ones

Theorem

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

Proof

1. Let $d(X_t) = i$ where i is the number of missing leading ones;
2. The negative drift is 0 since if a leading one is removed from the current solution the new point will not be accepted;

Drift Analysis for Leading Ones

Theorem

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

Proof

1. Let $d(X_t) = i$ where i is the number of missing leading ones;
2. The negative drift is 0 since if a leading one is removed from the current solution the new point will not be accepted;
3. 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:

$$E(\Delta^+(t)) = \sum_{k=1}^{n-i} k \cdot (p(\Delta^+(t)) = k) \geq 1 \cdot 1/n \cdot \big(1 - 1/n\big)^{n-1} \geq 1/(en)$$

Drift Analysis for Leading Ones

Theorem

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

Proof

1. Let $d(X_t) = i$ where i is the number of missing leading ones;
2. The negative drift is 0 since if a leading one is removed from the current solution the new point will not be accepted;
3. 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:

$$E(\Delta^+(t)) = \sum_{k=1}^{n-i} k \cdot (p(\Delta^+(t)) = k) \geq 1 \cdot 1/n \cdot \big(1 - 1/n\big)^{n-1} \geq 1/(en)$$

4. Hence, $E[\Delta(t) | d(X_t)] \geq 1/(en) = \delta$

Drift Analysis for Leading Ones

Theorem

The expected time for the (1+1)-EA to optimise LEADINGONES *is* $O(n^2)$

Proof

1. Let $d(X_t) = i$ where i is the number of missing leading ones;
2. The negative drift is 0 since if a leading one is removed from the current solution the new point will not be accepted;
3. 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:

$$E(\Delta^+(t)) = \sum_{k=1}^{n-i} k \cdot (p(\Delta^+(t)) = k) \geq 1 \cdot 1/n \cdot (1-1/n)^{n-1} \geq 1/(en)$$

4. Hence, $E[\Delta(t) | d(X_t)] \geq 1/(en) = \delta$
5. 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

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+λ)-EA to optimise* LEADINGONES *is* $O(\lambda n)$

Proof

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+λ)-EA to optimise* LEADINGONES *is* $O(\lambda n)$

Proof Left as exercise.

Theorem

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

Proof

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+λ)-EA to optimise* LEADINGONES *is* $O(\lambda n)$

Proof Left as exercise.

Theorem

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

Proof Left as exercise.

(1,λ)-EA Analysis for LEADINGONES

Theorem

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

Proof

(1,λ)-EA Analysis for LEADINGONES

Theorem

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

Proof

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

$$\begin{aligned} &E[d(X_t) - d(X_{t+1}) | d(X_t) = n - i] \\ &\geq 1 \cdot \left(1 - \left(1 - \frac{1}{en}\right)^n\right) - n \cdot \left(1 - \left(1 - \frac{1}{n}\right)^n\right)^n \\ &= c_1 - n \cdot c_2^n = \Omega(1) \end{aligned}$$

Hence,

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

and,

$$E(T) \leq n \cdot E(generations) = O(n^2)$$

Additive Drift Theorem

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:*

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

then

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

and

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

Theorem

The expected time for the (1+1)-EA to optimise LeadingOnes *is* $\Omega(n^2)$.

Theorem

The expected time for the (1+1)-EA to optimise LeadingOnes *is* $\Omega(n^2)$.

Sources of progress

1. Flipping the leftmost zero-bit;
2. Bits to right of the leftmost zero-bit that are one-bits (free riders).

Proof

Theorem

The expected time for the (1+1)-EA to optimise LeadingOnes *is* $\Omega(n^2)$.

Sources of progress

1. Flipping the leftmost zero-bit;
2. Bits to right of the leftmost zero-bit that are one-bits (free riders).

Proof

1. Let the current solution have $n-i$ leading ones (i.e. $1^{n-i}0*$).

Theorem

The expected time for the (1+1)-EA to optimise LeadingOnes *is* $\Omega(n^2)$.

Sources of progress

1. Flipping the leftmost zero-bit;
2. Bits to right of the leftmost zero-bit that are one-bits (free riders).

Proof

1. Let the current solution have $n-i$ leading ones (i.e. $1^{n-i}0*$).
2. We define the distance function as the number of missing leading ones, i.e. $d(X)=i$.

Theorem

The expected time for the (1+1)-EA to optimise LeadingOnes *is* $\Omega(n^2)$.

Sources of progress

1. Flipping the leftmost zero-bit;
2. Bits to right of the leftmost zero-bit that are one-bits (free riders).

Proof

1. Let the current solution have $n-i$ leading ones (i.e. $1^{n-i}0*$).
2. We define the distance function as the number of missing leading ones, i.e. $d(X) = i$.
3. The $n-i+1$ bit is a zero;

Theorem

The expected time for the (1+1)-EA to optimise LeadingOnes *is* $\Omega(n^2)$.

Sources of progress

1. Flipping the leftmost zero-bit;
2. Bits to right of the leftmost zero-bit that are one-bits (free riders).

Proof

1. Let the current solution have $n-i$ leading ones (i.e. $1^{n-i}0*$).
2. We define the distance function as the number of missing leading ones, i.e. $d(X) = i$.
3. The $n-i+1$ bit is a zero;
4. let $E[Y]$ be the expected number of one-bits after the first zero (i.e. the *free riders*).

Theorem

The expected time for the (1+1)-EA to optimise LeadingOnes *is* $\Omega(n^2)$.

Sources of progress

1. Flipping the leftmost zero-bit;
2. Bits to right of the leftmost zero-bit that are one-bits (free riders).

Proof

1. Let the current solution have $n-i$ leading ones (i.e. $1^{n-i}0*$).
2. We define the distance function as the number of missing leading ones, i.e. $d(X) = i$.
3. The $n-i+1$ bit is a zero;
4. let $E[Y]$ be the expected number of one-bits after the first zero (i.e. the *free riders*).
5. Such $i-1$ bits are uniformely distributed at initialisation and still are!

Theorem

The expected time for the (1+1)-EA to optimise LeadingOnes *is* $\Omega(n^2)$.

Sources of progress

1. Flipping the leftmost zero-bit;
2. Bits to right of the leftmost zero-bit that are one-bits (free riders).

Proof

1. Let the current solution have $n-i$ leading ones (i.e. $1^{n-i}0*$).
2. We define the distance function as the number of missing leading ones, i.e. $d(X) = i$.
3. The $n-i+1$ bit is a zero;
4. let $E[Y]$ be the expected number of one-bits after the first zero (i.e. the *free riders*).
5. Such $i-1$ bits are uniformely distributed at initialisation and still are!

Drift Theorem for LeadingOnes (lower bound)

Theorem

The expected time for the (1+1)-EA to optimise LeadingOnes *is* $\Omega(n^2)$.

The expected number of free riders is:

$$E[Y] = \sum_{k=1}^{i-1} k \cdot Pr(Y = k) = \sum_{k=1}^{i-1} Pr(Y \geq k) = \sum_{k=1}^{i-1} (1/2)^k \leq 1$$

Drift Theorem for LeadingOnes (lower bound)

Theorem

The expected time for the (1+1)-EA to optimise LeadingOnes *is* $\Omega(n^2)$.

The expected number of free riders is:

$$E[Y] = \sum_{k=1}^{i-1} k \cdot Pr(Y = k) = \sum_{k=1}^{i-1} Pr(Y \geq k) = \sum_{k=1}^{i-1} (1/2)^k \leq 1$$

- The negative drift is 0;

Drift Theorem for LeadingOnes (lower bound)

Theorem

The expected time for the (1+1)-EA to optimise LeadingOnes *is* $\Omega(n^2)$.

The expected number of free riders is:

$$E[Y] = \sum_{k=1}^{i-1} k \cdot Pr(Y = k) = \sum_{k=1}^{i-1} Pr(Y \geq k) = \sum_{k=1}^{i-1} (1/2)^k \leq 1$$

- The negative drift is 0;
- Let $p(A)$ be the probability that the first zero-bit flips into a one-bit.

Drift Theorem for LeadingOnes (lower bound)

Theorem

The expected time for the (1+1)-EA to optimise LeadingOnes *is* $\Omega(n^2)$.

The expected number of free riders is:

$$E[Y] = \sum_{k=1}^{i-1} k \cdot Pr(Y = k) = \sum_{k=1}^{i-1} Pr(Y \geq k) = \sum_{k=1}^{i-1} (1/2)^k \leq 1$$

- The negative drift is 0;
- Let $p(A)$ be the probability that the first zero-bit flips into a one-bit.
- The positive drift (i.e. the decrease in distance) is bounded as follows:

$$E(\Delta^+(t)) \leq p(A) \cdot E[\Delta^+(t)|A] = 1/n \cdot (1 + E[Y]) \leq 2/n = \delta$$

Drift Theorem for LeadingOnes (lower bound)

Theorem

The expected time for the (1+1)-EA to optimise LeadingOnes *is* $\Omega(n^2)$.

The expected number of free riders is:

$$E[Y] = \sum_{k=1}^{i-1} k \cdot Pr(Y = k) = \sum_{k=1}^{i-1} Pr(Y \geq k) = \sum_{k=1}^{i-1} (1/2)^k \leq 1$$

- The negative drift is 0;
- Let $p(A)$ be the probability that the first zero-bit flips into a one-bit.
- The positive drift (i.e. the decrease in distance) is bounded as follows:

$$E(\Delta^+(t)) \leq p(A) \cdot E[\Delta^+(t)|A] = 1/n \cdot (1 + E[Y]) \leq 2/n = \delta$$

- Since, also at initialisation the expected number of free riders is less than 1, it follows that $E[d(X_0)] \geq n - 1$,

By applying the Drift Theorem we get

$$E(T) \geq \frac{E(d(X_0))}{\delta} = \frac{n-1}{2/n} = \Omega(n^2)$$

Drift Analysis for OneMax

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

1. Let $d(X_t) = i$ 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 $d(X_t) = i$ 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 $d(X_t) = i$ 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:

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

Drift Analysis for OneMax

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

1. Let $d(X_t) = i$ 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:

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

4. 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. For $x \geq 1$, it holds that $\ln(1 + 1/x) \geq 1/x - 1/(2x^2) \geq 1/(2x)$;

Drift Analysis for OneMax

1. Let $g(X_t) = \ln(i+1)$ where i is the number of zeroes in the bitstring;
2. For $x \geq 1$, it holds that $\ln(1 + 1/x) \geq 1/x - 1/(2x^2) \geq 1/(2x)$;
3. The distance decreases as long as a 0 is flipped and the ones remain unchanged:

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

$$\geq \frac{i}{en}\left(\ln(i+1) - \ln(i)\right) = \frac{i}{en}\ln\left(1 + \frac{1}{i}\right) \geq \frac{i}{en}\frac{1}{2i} = \frac{1}{2en} := \delta$$

Drift Analysis for OneMax

1. Let $g(X_t) = \ln(i+1)$ where i is the number of zeroes in the bitstring;
2. For $x \geq 1$, it holds that $\ln(1+1/x) \geq 1/x - 1/(2x^2) \geq 1/(2x)$;
3. The distance decreases as long as a 0 is flipped and the ones remain unchanged:

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

$$\geq \frac{i}{en}(\ln(i+1) - \ln(i)) = \frac{i}{en} \ln\left(1+\frac{1}{i}\right) \geq \frac{i}{en}\frac{1}{2i} = \frac{1}{2en} := \delta$$

4. The initial distance is $d(X_0) \leq \ln(n+1)$

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

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

If the amount of progress depends on the distance from the optimum we need to use a logarithmic distance!

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

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

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

Hence,

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

Exercises

Theorem

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

Proof

Exercises

Theorem

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

Proof Left as exercise.

Theorem

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

Proof

Exercises

Theorem

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

Proof Left as exercise.

Theorem

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

Proof Left as exercise.

Theorem

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

Proof

Exercises

Theorem

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

Proof Left as exercise.

Theorem

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

Proof Left as exercise.

Theorem

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

Proof Left as exercise.

Drift Analysis: Example 3

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?

Drift Analysis: Example 3

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

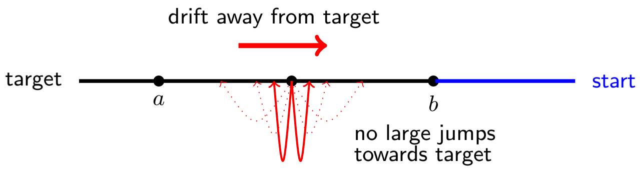
- The restaurant is n meters away from the hotel;
- Peter moves towards the hotel of 1 meter in each step with probability 0.4.
- Peter moves away from the hotel of 1 meter in each step with probability 0.6.

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.

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. $\mathrm{Prob}(|\Delta_t(i)| = -j) \ \leq \ \frac{1}{(1+\delta)^{j-r}}$ *for* $i > a$ *and* $j \geq 1$.

Then

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

.

Negative-Drift Analysis: Example (3)

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

Negative-Drift Analysis: Example (3)

- Define the same distance function $d(x) = x, x \in \{0, \dots, n\}$ (metres from the hotel) (b=n-1, a=1).
- Estimate the increase in distance from the goal (negative drift);

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

Negative-Drift Analysis: Example (3)

- Define the same distance function $d(x) = x, x \in \{0, \dots, n\}$ (metres from the hotel) (b=n-1, a=1).
- Estimate the increase in distance from the goal (negative drift);

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

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

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

Negative-Drift Analysis: Example (3)

- Define the same distance function $d(x) = x, x \in \{0, \dots, n\}$ (metres from the hotel) (b=n-1, a=1).
- Estimate the increase in distance from the goal (negative drift);

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

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

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

Negative-Drift Analysis: Example (3)

- Define the same distance function $d(x) = x, x \in \{0, \dots, n\}$ (metres from the hotel) (b=n-1, a=1).
- Estimate the increase in distance from the goal (negative drift);

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

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

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

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

$$Pr(T \le 2^{c(b-a)}) = Pr(T \le 2^{c(n-2)}) = 2^{-\Omega(n)}$$

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

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

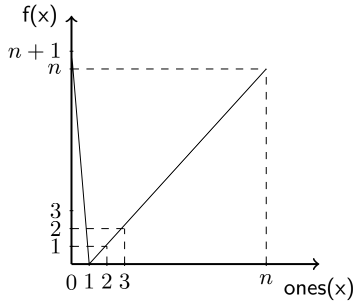
Proof of Condition 2

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

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

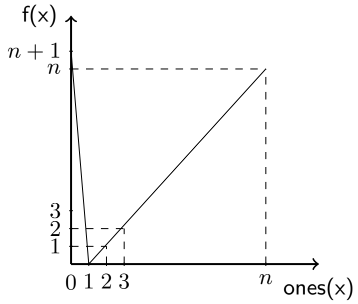
Exercise: Trap Functions

$$\text{TRAP}(x) = \begin{cases} n+1 & \text{if } x = 0^n \\ \text{ONEMAX}(x) & \text{otherwise.} \end{cases}$$



Exercise: Trap Functions

$$\text{TRAP}(x) = \begin{cases} n+1 & \text{if } x = 0^n \\ \text{ONEMAX}(x) & \text{otherwise.} \end{cases}$$



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.

Drift Analysis Conclusion

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];

Final Overview

Overview

- Basic Probability Theory
- Tail Inequalities
- Artificial Fitness Levels
- Drift Analysis

Other Techniques (Not covered)

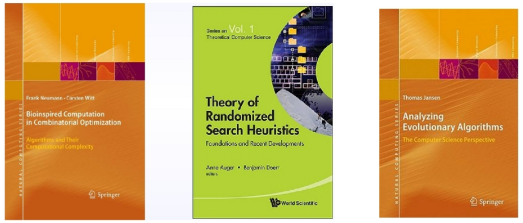
- Family Trees [Witt, 2006]
- Gambler's Ruin & Martingales [Jansen and Wegener, 2001]

Not only toy problems...

MST	(1+1) EA	$\Theta(m^2 \log(nw_{max}))$
	(1+λ) EA	$O(n \log(nw_{max}))$
	1-ANT	$O(mn \log(nw_{max}))$
Max. Clique	(1+1) EA	$\Theta(n^5)$
(rand. planar)	($16n$+1) RLS	$\Theta(n^{5/3})$
Eulerian Cycle	(1+1) EA	$\Theta(m^2 \log m)$
Partition	(1+1) EA	4/3 approx., competitive avg.
Vertex Cover	(1+1) EA	$e^{\Omega(n)}$, arb. bad approx.
Set Cover	(1+1) EA	$e^{\Omega(n)}$, arb. bad approx.
	SEMO	Pol. $O(\log n)$-approx.
Intersection of	(1+1) EA	$1/p$-approximation in
$p \geq 3$ matroids		$O(\|E\|^{p+2} \log(\|E\| w_{\max}))$
UIO/FSM conf.	(1+1) EA	$e^{\Omega(n)}$

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

Further Reading



[Neumann and Witt, 2010, Auger and Doerr, 2011, Jansen, 2013]

References I

Auger, A. and Doerr, B. (2011).
Theory of Randomized Search Heuristics: Foundations and Recent Developments.
World Scientific Publishing Co., Inc., River Edge, NJ, USA.

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. (2010).
Multiplicative drift analysis.
In *Proceedings of the 12th annual conference on Genetic and evolutionary computation*, GECCO '10, pages 1449–1456. 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.

Droste, S., Jansen, T., and Wegener, I. (2002).
On the analysis of the (1+1) evolutionary algorithm.
Theoretical Computer Science, 276(1-2):51–81.

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

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

References II

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. (2013).
Analyzing Evolutionary Algorithms.
Springer-Verlag New York, Inc., New York, NY, USA, 1st edition.

Jansen, T., Jong, K. A. D., and Wegener, I. A. (2005).
On the choice of the offspring population size in evolutionary algorithms.
Evolutionary Computation, 13(4):413–440.

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.

Lehre, P. K. (2010).
Negative drift in populations.
In *PPSN (1)*, pages 244–253.

References III

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

Neumann, F. and Witt, C. (2010).
Bioinspired Computation in Combinatorial Optimization: Algorithms and Their Computational Complexity.
Springer-Verlag New York, Inc., New York, NY, USA, 1st edition.

Oliveto, P. and Witt, C. (2012).
On the analysis of the simple genetic algorithm (to appear).
In *Proceedings of the 12th annual conference on Genetic and evolutionary computation*, GECCO '12, pages –. ACM.

Oliveto, P. S., He, J., and Yao, X. (2007).
Time complexity of evolutionary algorithms for combinatorial optimization: A decade of results.
International Journal of Automation and Computing, 4(3):281–293.

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.

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

References IV

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

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.