Introduction

Why should you care?

- foundations — firm ground
- Proofs provide insights and understanding.
- generality — wide applicability
- knowledge vs. beliefs
- fundamental limitations — saves time
- much improved teaching
- “There is nothing more practical than a good theory.”

Aims and Goals of this Tutorial

- provide an overview of
  - goals and topics
  - methods and their applications
- enhance your ability to
  - read, understand, and appreciate such papers
  - make use of the results obtained this way
- enable you to
  - apply the methods to your problems
  - produce such results yourself
- explain
  - what is doable with the currently known methods
  - where there is need for more advanced methods
- entertain

Topics and Structure

- Introduction and Motivation
- (an extremely short) introduction to evolutionary algorithms
- overview of topics in theory (as presented here today)
- analytical tools and methods – and how to apply them
  - fitness-based partitions
  - expectations and deviations
  - simple general lower bounds
  - expected multiplicative decrease in distance
  - drift analysis
  - random walks and cover times
  - typical runs
  - instructive example functions
- general limitations
  - NFL
  - black box complexity
Evolution Strategies (Bienert, Rechenberg, Schwefel)
- developed in the '60s / '70s of the last century.
- continuous optimization problems, rely on mutation.

Genetic Algorithms (Holland)
- developed in the '60s / '70s.
- binary problems, rely on crossover.

Genetic Programming (Koza)
- developed in the '90s.
- try to build good “computer programs”.

Nowadays
- more general view ⇒ evolutionary algorithms.

Evolutionary Algorithms

Principle
- follow Darwin’s principle (survival of the fittest).
- work with a set of solutions called population.
- parent population produces offspring population by variation operators (mutation, crossover).
- select individuals from the parents and children to create new parent population.

Scheme of an evolutionary algorithm

Basic EA
1. compute an initial population \( P = \{ X_1, \ldots, X_\mu \} \).
2. while (not termination condition)
   a. produce an offspring population \( P' = \{ Y_1, \ldots, Y_\lambda \} \) by crossover and/or mutation.
   b. create new parent population \( P \) by selecting \( \mu \) individuals from \( P \) and \( P' \).
Design

Important issues
  - representation
  - crossover operator
  - mutation operator
  - selection method

Representation

Properties
  - representation has to fit to the considered problem.
  - small change in the representation $\implies$ small change in the solution (locality).
  - often direct representation works fine.

Mainly in this talk
  - search space $\{0, 1\}^n$.
  - individuals are bitstrings of length $n$.

Crossover operator

Aim
  - two individuals $x$ and $y$ should produce a new solution $z$.

1-point Crossover
  - choose a position $p \in \{1, \ldots, n\}$ uniformly at random
  - set $z_i = x_i$ for $1 \leq i \leq p$
  - set $z_i = y_i$ for $p < i \leq n$

Uniform Crossover
  - set $z_i$ equally likely to $x_i$ or $y_i$
  - if $x_i = y_i$ then $z_i = x_i = y_i$
  - if $x_i \neq y_i$ then $\text{Prob}(z_i = x_i) = \text{Prob}(z_i = y_i) = 1/2$

Mutation

Aim
  - produce from a current solution $x$ a new solution $z$.

Some Possibilities
  - flip one randomly chosen bit of $x$ to obtain $z$.
  - flip each bit of $x$ with probability $p$ to obtain $z$ (often $p = 1/n$).
Selection methods

**Fitness-proportional selection**
- choose new population from a set of \( r \) individuals \( \{ x_1, \ldots, x_r \} \).
- probability to choose \( x_i \) in the next selection step is \( f(x_i)/(\sum_{j=1}^r f(x_j)) \).
- \( \mu \) individuals are selected in this way.

**\((\mu, \lambda)\)-selection**
- \( \mu \) parents produce \( \lambda \) children.
- select \( \mu \) best individuals from the children.

**\((\mu + \lambda)\)-selection**
- \( \mu \) parents produce \( \lambda \) children.
- select \( \mu \) best individuals from the parents and children.

**Simple algorithms**

**(1+1) EA**
1. Choose \( s \in \{0,1\}^n \) randomly.
2. Produce \( s' \) by flipping each bit of \( s \) with probability \( 1/n \).
3. Replace \( s \) by \( s' \) if \( f(s') \geq f(s) \).
4. Repeat Steps 2 and 3 forever.

**RLS**
1. Choose \( s \in \{0,1\}^n \) randomly.
2. Produce \( s' \) from \( s \) by flipping one randomly chosen bit.
3. Replace \( s \) by \( s' \) if \( f(s') \geq f(s) \).
4. Repeat Steps 2 and 3 forever.

**Topics in Theory**

The most pressing open question depends very much on what you are interested in.

What you are interested in depends very much on who you are.

You may be
- **biologist** What is evolution and how does it work?
- **engineer** How do I solve my problem with an EA?
- **computer scientist** What can evolutionary algorithms do?

Evolutionary algorithms are
- a model of natural evolution
- a robust general purpose problem solver
- randomized algorithms

**here and today** computer scientist’s point of view
Two branches
1. design and analysis of algorithms
   "How long does it take to solve this problem?"
2. complexity theory
   "How much time is needed to solve this problem?"

For evolutionary algorithms
1. analysis (and design) or evolutionary algorithms
   "What’s the expected optimization time of this EA for this problem?"
2. general limitations — NFL and black box complexity
   "How much time is needed to solve this problem?"

At the end of the day, time is wall clock time.

In computer science more convenient: #computation steps
requires formal model of computation (Turing machine, . . .)
typical for evolutionary algorithms black box optimization
fitness function not known to algorithm
gathers knowledge only by means of function evaluations
often
   • evolutionary algorithm’s core rather simple and fast
   • evaluation of fitness function costly and slow
thus   ‘time’ = #fitness function evaluations often appropriate

Definition
Optimization Time $T$ = #fitness function evaluations until an
optimal search point is sampled for the first time

very simple, yet often powerful method for upper bounds
first for (1+1)-EA only

Observation due to plus-selection fitness is monotone increasing
Idea for each fitness value $v$, find probability $p_v$ to increase fitness
Observation $\mathbb{E}(\text{time to increase fitness from } v) = \frac{1}{p_v}$
Observation $\mathbb{E}(T) = \sum_{v} \frac{1}{p_v}$
a bit more general group fitness values

For $f : \{0, 1\}^n \rightarrow \mathbb{R}$, $L_0, L_1, \ldots, L_k \subseteq \{0, 1\}^n$ with
- $\forall i \neq j \in \{0, 1, \ldots, k\} : L_i \cap L_j = \emptyset$
- $\bigcup_{i=0}^{k} L_i = \{0, 1\}^n$
- $\forall i < j \in \{0, 1, \ldots, k\} : \forall x \in L_i : \forall y \in L_j : f(x) < f(y)$
- $L_k = \{ x \in \{0, 1\}^n \mid f(x) = \max \{ f(y) \mid y \in \{0, 1\}^n \} \}$
is called an $f$-based partition.

Remember An $f$-based partition partitions the search space in accordance to fitness values
grouping fitness values arbitrarily.
Upper Bounds with $f$-Based Partitions

**Theorem**

Consider $(1+1)$-EA on $f: \{0,1\}^n \rightarrow \mathbb{R}$ and an $f$-based partition $L_0, L_1, \ldots, L_k$.
Let $s_i := \min_{x \in L_i} \sum_{j=k+1}^{n} \frac{1}{n} \sum_{y \in L_j} (\frac{1}{n})^{H(x,y)} (1 - \frac{1}{n})^{n-H(x,y)}$
for all $i \in \{0,1,\ldots,k-1\}$.

$$E(T_{(1+1)\text{-EA},f}) \leq \sum_{i=0}^{k-1} \frac{1}{s_i}$$

**Hint** most often, very simple lower bounds for $s_i$ suffice

---

**Example: Result for a Class of Functions**

Consider $(1+1)$-EA on linear function $f: \{0,1\}^n \rightarrow \mathbb{R}$.

**Definition**

$f: \{0,1\}^n \rightarrow \mathbb{R}$ is called linear
if $\exists w_0, w_1, \ldots, w_n \in \mathbb{R}: \forall x \in \{0,1\}^n: f(x) = w_0 + \sum_{i=1}^{n} w_i \cdot x[i]$.

**First Step** define $f$-based partition

$L_i := \{ x \in \{0,1\}^n | \sum_{j=1}^{i} w_j \leq f(x) < \sum_{j=1}^{i+1} w_j \}, 0 \leq i \leq n$

**Second Step** find lower bounds for $s_i$

**Observation** there is always at least 1-bit-mutation for leaving $L_i$.

$s_i \geq \frac{1}{n} (1 - \frac{1}{n})^{n-1} \geq \frac{1}{en}$

**Third Step** $E(T_{(1+1)\text{-EA},f}) \leq \sum_{i=0}^{n+1} \frac{1}{s_i}$

---

**Very Simple Example**

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

**First Step** define $f$-based partition

trivial for each fitness value one $L_i$

$L_i \ := \{ x \in \{0,1\}^n | \text{OneMax}(x) = i \}, 0 \leq i \leq n$

**Second Step** find lower bounds for $s_i$

**Observation** it suffices to flip any 0-bit from the $n-i$ 0-bits.

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

$$\left( (1 - \frac{1}{n})^{n-i} \geq \frac{1}{2} \geq (1 - \frac{1}{n})^{n} \right)$$

**Third Step** compute upper bound

$$E(T_{(1+1)\text{-EA},\text{OneMax}}) \leq \sum_{i=0}^{n-1} \frac{en}{n-i} = en \cdot \sum_{i=1}^{n} \frac{1}{i} = O(n \log n)$$

---

**Generalizing the Method**

Idea not restricted to $(1+1)$-EA, only.

Consider $(1+\lambda)$-EA on LeadingOnes $\text{LeadingOnes}(x) = \sum_{j=1}^{n} \prod_{j=1}^{i} x[j]$

**First Step** define $f$-based partition

trivial for each fitness value one $L_i$

$L_i \ := \{ x \in \{0,1\}^n | \text{LeadingOnes}(x) = i \}, 0 \leq i \leq n$

For the $(1+\lambda)$-EA, we re-define the $s_i$:

$s_i := \text{Prob}(\text{leave } L_i \text{ in one generation})$

**Observation** $E(T_{(1+\lambda)\text{-EA},f}) \leq \lambda \cdot \sum_{i=2}^{n} \frac{1}{s_i}$
(1 + λ)-ES on LeadingOnes

Second Step     find lower bounds for $s_i$
Observation      It suffices to flip exactly the leftmost 0-bit.
$s_i \geq 1 - \left(1 - \frac{1}{n^2}\right)^\lambda \geq 1 - e^{-\lambda/(en)}$

Case Inspection Case 1      $\lambda \geq en$
$s_i \geq 1 - \frac{1}{n^2}$

Case Inspection Case 2      $\lambda < en$
$s_i \geq \frac{\lambda}{2en}$

Third Step     compute upper bound
$E(\text{LeadingOnes}_\lambda) \leq \lambda \cdot \left(\sum_{i=0}^{n-1} \frac{1}{i!} \lambda^i \right) + \left(\sum_{i=1}^{n-1} \frac{\lambda^i}{i!}\right)$
$= O(\lambda \cdot n + n^2) = O(\lambda \cdot n + \sqrt{n})$

Markov Inequality and Chernoff Bounds

Theorem (Markov Inequality)
$X \geq 0$ random variable, $\delta > 0$
$\text{Prob}(X \geq \delta \cdot E(X)) \leq \frac{1}{\delta}$

Theorem (Chernoff Bounds)
Let $X_1, X_2, \ldots, X_n: \Omega \rightarrow \{0, 1\}$ independent random variables
with
$\forall i \in \{1, 2, \ldots, n\}: 0 < \text{Prob}(X_i = 1) < 1.$
Let $X := \sum_{i=1}^n X_i.$
$\forall \delta > 0: \text{Prob}(X > (1 + \delta) \cdot E(X)) < \left(\frac{e^\delta - 1}{\delta}\right)^{E(X)}$
$\forall 0 < \delta < 1: \text{Prob}(X < (1 - \delta) \cdot E(X)) < e^{-E(X)\delta^2/2}$

A Very Simple Application

Consider $x \in \{0, 1\}^{100}$ selected uniformly at random

more formal for $i \in \{1, 2, \ldots, 100\}$: $B_i := \begin{cases} 1 & i\text{-th bit is 1} \\ 0 & \text{otherwise} \end{cases}$
with $\text{Prob}(B_i = 0) = \text{Prob}(B_i = 1) = \frac{1}{2}$
$B := \sum_{i=1}^{100} B_i$ clearly $E(B) = 50$

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

Markov
$\text{Prob}(B \geq 75) = \text{Prob}(M \geq \frac{1}{2} \cdot 50) \leq \frac{3}{5}$

Chernoff
$\text{Prob}(B \geq 75) = \text{Prob}(B \geq (1 + \frac{1}{2}) \cdot 50)$
$\leq \left(\frac{e^\delta - 1}{\delta}\right)^{E(B)} < 0.0045$

Truth
$\text{Prob}(B \geq 75) = \sum_{i=1}^{100} \binom{100}{i} 2^{-100}$
$= \frac{89,310,453,795,650,905,935,325}{316,992,200,000,000,000,000,000,000} < 0.0000000282$
**The Law of Total Probability**

**Theorem (Law of Total Probability)**

Let $B_i$ with $i \in I$ be a partition of some probability space $\Omega$. 

$$\forall A \subseteq \Omega: \text{Prob}(A) = \sum_{i \in I} \text{Prob}(A | B_i) \cdot \text{Prob}(B_i)$$

**Immediate consequence**

$$\text{Prob}(A) \geq \text{Prob}(A | B) \cdot \text{Prob}(B)$$

Useful for lower bounds when some event “determines” expected optimization time

**A Very Simple Example**

Consider $(1+1)$-EA on $f: \{0, 1\}^n \rightarrow \mathbb{R}$ with $f(x) := \begin{cases} n - \frac{1}{2} & \text{if } x = 0^n, \\ \text{OneMax}(x) & \text{otherwise} \end{cases}$

**Theorem**

$$\mathbb{E}(T_{(1+1)}-\text{EA}, f) = \Omega\left(\left(\frac{n}{2}\right)^n\right)$$

**Proof.**

Define event $B$: $(1+1)$-EA initializes with $x = 0^n$ clearly $\text{Prob}(B) = 2^{-n}$

**Observation**

$$\mathbb{E}(T_{(1+1)}-\text{EA}, f | B) = n^n$$

since all bits have to flip simultaneously

Law of Total Probability

$$\mathbb{E}(T_{(1+1)}-\text{EA}, f) \geq n^n \cdot 2^{-n} = \left(\frac{n}{2}\right)^n$$

---

**Proposition**

Given $n$ different coupons. Choose at each trial a coupon uniformly at random. Let $X$ be a random variable describing the number of trials required to choose each coupon at least once. Then

$$\mathbb{E}(X) = nH_n$$

holds, where $H_n$ denotes the $n$th Harmonic number, and

$$\lim_{n \to \infty} \text{Prob}(X \leq n(\ln n - c)) = e^{-e^c}$$

holds for each constant $c \in \mathbb{R}$. 

---

**Lower bound for OneMax**

- Expected number of 1-bits in initial solution is $n/2$.
- At least $n/3$ 0-bits with probability $1 - e^{-\Omega(n)}$ (Chernoff).

**Lower Bound**

- Probability that at least one 0-bit has not been flipped during $t = (n - 1) \ln n$ steps is

$$1 - (1 - (1 - 1/n)^{(n-1)\ln n})^{n/3} \geq 1 - e^{-1/3} = \Omega(1).$$

- Expected optimization time for OneMax is $\Omega(n \log n)$

**Generalization**

$\Omega(n \log n)$ for each function with poly. number of optima.

---


---

**Expected multiplicative distance decrease**

**Basic idea**
- Assumption: Function values are integers.
- Define a set $O$ of $l$ operations to obtain an optimal solution.
- Average gain of these $l$ operations is $f(x_{opt}) - f(x)$.

**Upper bound**
- Let $d_{\text{max}} = \max_{x \in \{0,1\}^n} f(x_{opt}) - f(x)$.
- 1 operation: expected distance at most $(1 - 1/l) \cdot d_{\text{max}}$.
- $t$ operations: expected distance at most $(1 - 1/l)^t \cdot d_{\text{max}}$.
- Expected number of $O(l \cdot d_{\text{max}})$ operations to reach optimum.
- Assume: expected time for each operation is at most $r$.
- Upper bound $O(r \cdot l \cdot d_{\text{max}})$ to obtain an optimal solution.

---

**Example**

**Linear Functions**
- $f(x) = w_1x_1 + w_2x_2 + \cdots + w_nx_n$.
- $w_i \in \mathbb{Z}$.
- $w_{\text{max}} = \max_i w_i$.

**Upper bound**
- Consider all operations that flip a single bit.
- Each necessary operation is accepted.
- $d_{\text{max}} = n \cdot w_{\text{max}}$.
- Expected number of operations $O(n \log d_{\text{max}})$.
- Waiting time for a single bit flip $O(1)$.
- Upper bound $O(n \log n + \log w_{\text{max}})$.
- If $w_{\text{max}} = \text{poly}(n)$, upper bound $O(n \log n)$.

---

**A More Flexibel Proof Method**

**Sad Facts**
- $f$-based partitions restricted to "well behaving" functions
- direct lower bound often too difficult

**How can we find a more flexibel method?**

**Observation**
- $f$-based partition measure progress by $f(x_{t+1}) - f(x_t)$

**Idea**
- Consider a more general measure of progress

**Define**
- distance $d: \mathbb{Z} \rightarrow \mathbb{N}_0$, ($\mathbb{Z}$ set of all populations) with $d(P) = 0 \Leftrightarrow P$ contains optimal solution

**Caution**
- “Distance” need not be a metric!

---

**Drift**

**Define**
- distance $d: \mathbb{Z} \rightarrow \mathbb{R}_0^+$, ($\mathbb{Z}$ set of all populations) with $d(P) = 0 \Leftrightarrow P$ contains optimal solution

**Observation**
- $T = \min \{t \mid d(P_t) = 0\}$

**Consider**
- maximum distance $M := \max \{d(P) \mid P \in \mathbb{Z}\}$
- decrease in distance $D_t := d(P_{t+1}) - d(P_t)$

**Definition**
- $E(D_t \mid T \geq t)$ is called drift.

**Pessimistic point of view**
- $\Delta := \min \{E(D_t \mid T \geq t) \mid t \in \mathbb{N}_0\}$

**Drift Theorem (Upper Bound)**
- $\Delta > 0 \Rightarrow E(T) \leq M/\Delta$

Upper Bound Drift Theorem

**Proof**

Observe $M \geq E\left(\sum_{i=1}^{T} D_i\right)$

$M \geq E\left(\sum_{i=1}^{T} D_i\right) = \sum_{t=1}^{\infty} \text{Prob}(T = t) \cdot E\left(\sum_{i=1}^{T} D_i \mid T = t\right)$

$= \sum_{i=1}^{\infty} \text{Prob}(T = t) \cdot \sum_{i=1}^{t} E(D_i \mid T = t)$

$= \sum_{i=1}^{\infty} \text{Prob}(T = t) \cdot E(D_i \mid T = t)$

$= \sum_{i=1}^{\infty} \sum_{t=i}^{\infty} \text{Prob}(T = t) \cdot E(D_i \mid T = t)$

Proof of the Drift Theorem (Upper Bound) (cont.)

$\geq \sum_{i=1}^{\infty} \sum_{t=i}^{\infty} \text{Prob}(T = t) \cdot E(D_i \mid T = t)$

$= \sum_{i=1}^{\infty} \text{Prob}(T \geq i) \cdot \sum_{t=i}^{\infty} \text{Prob}(T = t \mid T \geq i) \cdot E(D_i \mid T = t)$

$= \sum_{i=1}^{\infty} \text{Prob}(T \geq i) \sum_{t=i}^{\infty} \text{Prob}(T = t \mid T \geq i) \cdot E(D_i \mid T = t \wedge T \geq i)$

$= \sum_{i=1}^{\infty} \text{Prob}(T \geq i) E(D_i \mid T \geq i) \geq \Delta \sum_{i=1}^{\infty} \text{Prob}(T \geq i) = \Delta \cdot E(T)$

thus $E(T) \leq M \Delta$
Consider $(1+1)$-EA on linear function $f : \{0, 1\}^n \to \mathbb{R}$ now with drift analysis.

Remember \( f(x) = \sum_{i=1}^{n} w_i \cdot x[i] \)
with \( w_1 \geq w_2 \geq \cdots \geq w_n > 0 \)

Define \( d(x) := \ln \left( 1 + 2 \sum_{i=1}^{n/2} (1 - x[i]) + \sum_{i=(n/2)+1}^{n} (1 - x[i]) \right) \)

Observe \( M = \max \{ d(x) \mid x \in \{0, 1\}^n \} = \ln \left( 1 + \frac{3}{2} n \right) = \Theta(\ln n) \)


---

### Drift Analysis for $(1+1)$-EA on general linear functions

\( d(x) := \ln \left( 1 + 2 \sum_{i=1}^{n/2} (1 - x[i]) + \sum_{i=(n/2)+1}^{n} (1 - x[i]) \right) \)

Need lower bound for \( \mathbb{E}(d(x_{t-1}) - d(x_t) \mid T \geq t) \)

Observe minimal for \( x_{t-1} = 011 \cdots 1 \) or \( 11 \cdots 1 \) \( 01 \cdots 1 \) \( \text{left} \) \( \text{right} \)

Consider separately and do tedious calculations...
Result for (1+1)-EA on General Linear Functions

We have

- \( d(x) := \ln (1 + 2 \sum_{i=1}^{n/2} (1 - x[i]) + \sum_{i=n/2+1}^{n} (1 - x[i]) ) \)
- \( d(x) \leq \ln (1 + (3/2)n) = O(\log n) \)
- \( E(d(x_{i-1}) - d(x_i) | T \geq t) = \Omega(1/n) \)

Together \( E(T_{(1+1)\, EA}) = O(\alpha \log n) \) for any linear \( f \)

Drift Analysis of Lower Bounds

We have drift analysis for upper bounds

How can we obtain lower bounds when analyzing drift?

Idea Check proof of drift theorem (upper bound).
Can inequalities be reversed?

Remember \( M \geq E \left( \sum_{i=1}^{T} D_i \right) = \cdots = \sum_{i=1}^{\infty} Prob(T \geq i) \cdot E(D_i | T \geq i) \)

\[ \geq \Delta \cdot \sum_{i=1}^{\infty} Prob(T \geq i) = \Delta \cdot E(T) \]

with

- \( M = \max\{d(P) | P \in Z\} \)
- \( \Delta = \min\{E(d(P_{i-1}) - d(P_i) | T \geq t)\} \)

Modification for a Lower Bound Technique

- Observation only two inequalities need to be reversed
- \( M \geq \sum_{i=1}^{\infty} \Delta_i \cdot \sum_{i=1}^{\infty} Prob(d(P_{i-1}) - d(P_i) | T \geq t) \)

Clearly for lower bound \( \Delta = \max\{E(d(P_{i-1}) - d(P_i) | T \geq t)\} \)

Sensible and sufficient for "\( \leq \)"

Clear for lower bound instead of \( M \min\{d(P) | P \in Z\} \)

Possible and sufficient for "\( \leq \)"

but pointless, since \( \min\{d(P) | P \in Z\} = 0 \)

Closing the Distance

- Clearly \( E \left( \sum_{i=1}^{T} D_i \right) \) fixed, if initial population is known

Thus lower bound on \( d(P_0) \) yields lower bound on \( E(T) \)

Making this concrete

- \( E(T | d(P_0) \geq M_a) \geq M_a / \Delta \)
- \( E(T) \geq \max\{E(d(P_0) \geq M_a) \cdot E(T | d(P_0) \geq M_a) \} \)
- \( E(T) \geq \max\{E(d(P_0) \geq d) \cdot d / \Delta \geq E(d(P_0)) / \Delta \} \)

Thus drift analysis suitable as method for upper and lower bounds
**Lower Bound for (1+1) EA on LeadingOnes**

**Define** trivial distance
\[ d(x) := n - \text{LeadingOnes}(x) \]

**Observation** necessary for decreasement of distance
left-most 0-bit flips

thus \( \text{Prob} \) (decrease distance) \( \leq \frac{1}{n} \)

How can we bound the decrease in distance?

**Observation** trivially, by \( n \) — not useful

better question How can we bound the expected decrease in distance?

---

**Expected Decrease in Distance on LeadingOnes**

**Note** decrease in distance \( \cong \) increase in fitness

**Observation** two sources for increase in fitness
- the left-most 0-bit
- bits to the right of this bits that happen to be 1-bits

**Observation** bits to the right of the left-most 0-bit have no influence on selection and never had influence on selection

**Claim** These bits are uniformly distributed.

obvious holds after random initialization

Claim standard bit mutations do not change this

---

**Standard Bit Mutations on Uniformly Distributed Bits**

**Claim** \( \forall t \in \mathbb{N}_0 : \forall x \in \{0,1\}^n \): \( \text{Prob} (x_t = x) = 2^{-n} \)

clearly holds for \( t = 0 \)

\[
\text{Prob} (x_t = x) = \sum_{x' \in \{0,1\}^n} \text{Prob} \left( (x_{t-1} = x') \land (\text{mut}(x') = x) \right)
\]

\[
= \sum_{x' \in \{0,1\}^n} \text{Prob} (x_{t-1} = x') \cdot \text{Prob} (\text{mut}(x') = x)
\]

\[
= 2^{-n} \cdot \sum_{x' \in \{0,1\}^n} \text{Prob} (\text{mut}(x') = x)
\]

\[
= 2^{-n} \cdot \sum_{x' \in \{0,1\}^n} \text{Prob} (\text{mut}(x') = x')
\]

\[
= 2^{-n} \quad \Box
\]

---

**Expected Increase in Fitness and Expected Initial Distance**

\[
E \left( \text{increase in fitness} \right) = \sum_{i=1}^{n} i \cdot \text{Prob} (\text{fitness increase} = i)
\]

\[
= \sum_{i=1}^{n} i \cdot \frac{1}{n} \cdot 2^{-i} \leq \frac{1}{n} \sum_{i=1}^{\infty} i \cdot \frac{1}{2} = \frac{2}{n}
\]

\[
E \left( d(x_0) \right) = n - \sum_{i=1}^{n} i \cdot \text{Prob} (\text{LeadingOnes}(x_0) = i)
\]

\[
= n - \sum_{i=1}^{n} \frac{i}{2^{i+1}} \geq n - \frac{1}{2} \sum_{i=1}^{\infty} \frac{i}{2} = n - 1
\]

thus \( E \left( T_{(1+1) \text{EA} \text{LeadingOnes}} \right) \geq \frac{(n-1)n}{2} = \Omega(n^2) \)

thus \( E \left( T_{(1+1) \text{EA} \text{LeadingOnes}} \right) = \Theta(n^2) \)
Random Walks

Random Walks on Graphs
Given: An undirected connected graph.
- A random walk starts at a vertex \( v \).
- Whenever it reaches a vertex \( w \), it chooses in the next step a random neighbor of \( w \).

Theorem (Upper bound for Cover Time)
Given an undirected connected graph with \( n \) vertices and \( m \) edges, the expected number of steps until a random walk has visited all vertices is at most \( 2m(n-1) \).


Example: Plateaus

Definition
Plateau(\( x \)) :=
\[
\begin{align*}
&n - OneMax(x) & x \notin \{10^{n-i}, 0 \leq i \leq n\} \\
n + 1 & x \in \{10^{n-i}, 0 \leq i < n\} \\
n + 2 & x = 1^n.
\end{align*}
\]

Upper bound (RLS)
- Solution with fitness \( \geq n + 1 \) in expected time \( O(n \log n) \).
- Random walk on the plateau of fitness \( n + 1 \).
- Probability 1/2 to increase (reduce) the number of ones.
- Expected waiting time for an accepted step \( \Theta(n) \).
- Optimum reached within \( O(n^2) \) expected accepted steps.
- Upper bound \( O(n^3) \) (same holds for (1+1)-EA).

Method of Typical Runs

Phase 1: Given EA starts with random initialization, with probability at least \( 1 - p_1 \), it reaches a population satisfying condition \( C_1 \) in at most \( T_1 \) steps.

Phase 2: Given EA starts with a population satisfying condition \( C_1 \), with probability at least \( 1 - p_2 \), it reaches a population satisfying condition \( C_2 \) in at most \( T_2 \) steps.

... Phase \( k \): Given EA starts with a population satisfying condition \( C_{k-1} \), with probability at least \( 1 - p_k \), it reaches a population containing a global optimum in at most \( T_k \) steps.

This yields: \( \text{Prob} \left( T_{EA,f} \leq \sum_{i=1}^{k} T_i \right) \geq 1 - \sum_{i=1}^{k} p_i \)

From Success Probability to Expected Optimization Time

Sometimes
“Phase 1: Given EA starts with random initialization”
can be replaced by
“Phase 1: EA may start with an arbitrary population”

In this case, a failure in any phase can be described as a \text{restart}.

This yields: \( \mathbb{E} (T_{EA,f}) \leq \frac{\sum_{i=1}^{k} T_i}{1 - \sum_{i=1}^{k} p_i} \)
A Concrete Example

\[ \text{JUMP}_k(x) : \{0, 1\}^n \rightarrow \mathbb{R} \text{ with } k \in \{1, 2, \ldots, n\} \]

\[ \text{JUMP}_k(x) := \begin{cases} 
  n - \text{OneMax}(x) & \text{if } n - k < \text{OneMax}(x) < n \\
  k + \text{OneMax}(x) & \text{otherwise}
\end{cases} \]

GA on \textit{JUMP}_k

**Theorem**

Let \( k = O(\log n) \), \( c \in \mathbb{R}^+ \) a sufficiently large constant, \( \mu = n^{O(1)} \), \( \mu \geq k \log^2 n \), \( 0 < p_c \leq 1/(ckn) \).

\[ E(T_{\text{GA}}(\mu, p_c)) = O(\mu n^2 k + 2^k/p_c) \]

**Method of Proof: Typical Run**

GECCO 2007 Tutorial / Computational Complexity and Evolutionary Computation

A Steady State GA

\((\mu+1)\text{-EA with prob. } p_c \text{ for uniform crossover}\)

1. **Initialization**
   Choose \( x_1, \ldots, x_\mu \in \{0, 1\}^n \) uniformly at random.

2. **Selection and Variation**
   With probability \( p_c \):
   - Select \( z_1 \) and \( z_2 \) independently from \( x_1, \ldots, x_\mu \).
   - \( z := \text{uniform crossover}(z_1, z_2) \)
   - \( y := \text{standard } 1/n \text{ bit mutation}(z) \)
   Otherwise:
   - Select \( z \) from \( x_1, \ldots, x_\mu \).
   - \( y := \text{standard } 1/n \text{ bit mutation}(z) \)

3. **Selection for Replacement**
   If \( f(y) \geq \min\{f(x_1), \ldots, f(x_\mu)\} \)
   Then Replace some \( x_i \) with min. \( f \)-value by \( y \).

4. **“Stopping Criterion”**
   Continue at 2.
Zero-Bits at the First Position

Consider one generation.

Let \( z \) be the current number of zero-bits in first position.

The value of \( z \) can change by at most 1.

- event \( A_z^+ \): \( z \) changes to \( z + 1 \)
- event \( A_z^- \): \( z \) changes to \( z - 1 \)

**Goal:** Estimate \( \text{Prob}(A_z^+) \) and \( \text{Prob}(A_z^-) \).

A Closer Look at \( A_z^+ \)

**“Smaller/Simpler” Events:**

<table>
<thead>
<tr>
<th>Event</th>
<th>Description</th>
<th>Probability</th>
</tr>
</thead>
<tbody>
<tr>
<td>( B_z )</td>
<td>do crossover</td>
<td>( p_c )</td>
</tr>
<tr>
<td>( C_z )</td>
<td>at selection for replacement, select ( x ) with 1 at first position</td>
<td>( (\mu - z) / \mu )</td>
</tr>
<tr>
<td>( D_z )</td>
<td>at selection for reproduction, select parent with 0 at first position</td>
<td>( z / \mu )</td>
</tr>
<tr>
<td>( E_z )</td>
<td>no mutation at first position</td>
<td>( 1 - 1 / n )</td>
</tr>
</tbody>
</table>
| \( F_{z,i}^+ \) | out of \( k - 1 \) 0-bits \( i \) mutate and                              | \( (k-1)(n-k)(\frac{1}{2})^i(1 - \frac{1}{2})^{n-2i} \)
| \( G_{z,i}^+ \) | out of \( k \) 0-bits \( i \) mutate and                                  | \( (k)(n-k-1)(\frac{1}{2})^{i-1}(1 - \frac{1}{2})^{n-2} \)

**Observe:**

\[
A_z^+ \subseteq B_z \cup \left( C_z \cap \left( D_z \cap E_z \cup \left( F_{z,i}^+ \cup G_{z,i}^+ \right) \right) \right)
\]
Using
\[ A^+_z \subseteq B_z \cup \left( B_z \cap C_z \cap \left[ \left( D_z \cap E_z \cap \bigcup_{i=0}^{k-1} F^+_x i \right) \cup \left( D_z \cap E_z \cap \bigcup_{i=1}^{k} G^+_x i \right) \right] \right) \]
together with
\[
\begin{align*}
\text{Prob} (B_z) &= p_c \\
\text{Prob} (C_z) &= \frac{\mu - z}{\mu} \\
\text{Prob} (D_z) &= \frac{n}{n} \\
\text{Prob} (E_z) &= 1 - \frac{1}{n} \\
\text{Prob} (F^+_x i) &= \binom{k-i}{i} \left( \frac{1}{n} \right)^{2i} (1 - \frac{1}{n})^{n-2i} \\
\text{Prob} (G^+_x i) &= \binom{n-k-1}{i} \left( \frac{1}{n} \right)^{2i-1} (1 - \frac{1}{n})^{n-2i}
\end{align*}
\]
yields some bound on \( \text{Prob} (A^+_z) \).

"Smaller/Simpler" Events:

<table>
<thead>
<tr>
<th>event</th>
<th>description</th>
<th>probability</th>
</tr>
</thead>
<tbody>
<tr>
<td>( B_z )</td>
<td>do crossover</td>
<td>( p_c )</td>
</tr>
<tr>
<td>( C_z )</td>
<td>at selection for replacement, select ( x ) with 1 at first position</td>
<td>( (\mu - z)/\mu )</td>
</tr>
<tr>
<td>( D_z )</td>
<td>at selection for reproduction, select parent with 0 at first position</td>
<td>( z/\mu )</td>
</tr>
<tr>
<td>( E_z )</td>
<td>no mutation at first position</td>
<td>( 1 - 1/n )</td>
</tr>
<tr>
<td>( F^+_x i )</td>
<td>out of ( k - 1 ) 0-bits ( i ) - 1 mutate and out of ( n - k ) 1-bits ( i ) mutate</td>
<td>( \binom{k-i}{i} \binom{n-k-1}{i} \left( \frac{1}{n} \right)^{2i} (1 - \frac{1}{n})^{n-2i} )</td>
</tr>
<tr>
<td>( G^+_x i )</td>
<td>out of ( k ) 0-bits ( i ) mutate and out of ( n - k - 1 ) 1-bits ( i ) mutate</td>
<td>( \binom{k}{i} \binom{n-k-1}{i} \left( \frac{1}{n} \right)^{2i-1} (1 - \frac{1}{n})^{n-2i} )</td>
</tr>
</tbody>
</table>

Observe:
\[ A^-_z \supseteq D_z \cap C_z \cap \left[ \left( D_z \cap E_z \cap \bigcup_{i=1}^{k} F^-_x i \right) \cup \left( D_z \cap E_z \cap \bigcup_{i=0}^{k} G^-_x i \right) \right] \]

Using
\[ A^-_z \supseteq D_z \cap C_z \cap \left[ \left( D_z \cap E_z \cap \bigcup_{i=1}^{k} F^-_x i \right) \cup \left( D_z \cap E_z \cap \bigcup_{i=0}^{k} G^-_x i \right) \right] \]
together with the known probabilities yields again some bound.

Instead of considering the two bounds directly, we consider their difference:

If \( z \) is large, say \( z \geq \frac{k}{8\mu} \),
\[ \text{Prob} (A^-_z) - \text{Prob} (A^+_z) = \Omega \left( \frac{1}{n^2} \right) \]

Bias Towards 1-Bits

We know: \( z \geq \frac{\mu}{n^2} \Rightarrow \text{Prob} (A^-_z) - \text{Prob} (A^+_z) = \Omega \left( \frac{1}{n^2} \right) \)

Consider \( c^* \mu n^2 k \) generations; \( c^* \) sufficiently large constant

\[ E \text{ (difference in 0-bits)} = \Omega \left( \frac{a z^2}{n} \right) = \Omega (n k) \]

Having \( c^* \) sufficiently large implies \( < \mu/(4k) \) 0-bits at the end of the phase.

Really?

Only if \( z \geq \mu/(8k) \) holds all the time!
Coping with Our Assumption

As long as \( z \geq \frac{\mu}{8k} \) holds, things work out nicely.

Consider last point of time, when \( z < \frac{\mu}{8k} \) holds in the \( c^*n^2k \) generations.

Case 1: at most \( \frac{\mu}{8k} \) generations left

number of 0-bits \( < \frac{\mu}{8k} + \frac{\mu}{4k} = \frac{\mu}{4k} \)
no problem

Case 2: more than \( \frac{\mu}{8k} \) generations left

Observation: \( \mu/(8k) = \Omega(\log^2 n) \)
For \( \Omega(\log^2 n) \) generations, our assumption holds.

Apply Chernoff’s bound for these generations.

Yields \( p^* = e^{-\Omega(\log^2 n)} \).

Together: \( p_2 = n \cdot p^* = e^{-\Omega(\log^2 n) + \ln n} = e^{-\Omega(\log^2 n)} \)

Phase 3: Finding the Optimum

In the beginning, we have at most \( \frac{\mu}{4k} \) 0-bits at each position.

In the same way as for Phase 2, we make sure that we always have at most \( \frac{\mu}{2k} \) 0-bits at each position.

\[
\text{Prob (find optimum in current generation)} \geq \text{Prob (crossover and select two parents without common 0-bit and create 1^n with uniform crossover and no mutation)}
\]

\[
\text{Prob (crossover)} = p_c
\]

\[
\text{Prob (create 1^n with uniform crossover)} = (1/2)^{2k}
\]

\[
\text{Prob (no mutation)} = (1 - 1/n)^n
\]

\[
\text{Prob (select two parent without common 0-bit)} \leq k \cdot \frac{\mu/(2k)}{\mu} = \frac{1}{2}
\]

Together:

\[
\text{Prob (find optimum in current generation)} = \Omega(p_c \cdot 2^{-2k})
\]

Concluding Phase 3

We have

\[
\text{Prob (find optimum in current generation)} = \Omega(p_c \cdot 2^{-2k})
\]

\[
\text{Prob (find optimum in } c_3^*2^{2k}/p_c \text{ generations)} \geq 1 - \varepsilon(c_3)
\]

failure probability \( p_3 \leq \varepsilon' \) for any constant \( \varepsilon' > 0 \)

Concluding the Proof

Length of the three phases:

\[
O(\mu n \log n) + O(\mu n^2k) + O(2^{2k}/p_c) = O(\mu n^2k + 2^{2k}/p_c)
\]

Sum of Failure Probabilities:

\[
\varepsilon + e^{-\Omega(\log^2 n)} + \varepsilon' \leq \varepsilon^* < 1
\]

\[
E(T_{GA}(\mu, p_c)) = O(\mu n^2k + 2^{2k}/p_c)
\]
Black Box Optimization

Setting
- Given two finite spaces $S$ and $R$.
- Find for a given function $f : S \rightarrow R$ an optimal solution.
- Count number of fitness evaluations.
- No search point is evaluated more than once.

Definition (Black Box Algorithm)
An algorithm $A$ is called black box algorithm if its finds for each $f : S \rightarrow R$ an optimal solution after a finite number of fitness evaluations.

NFL

Theorem (NFL)
Given two finite spaces $R$ and $S$ and two arbitrary black box algorithms $A$ and $A'$. The average number of fitness evaluations among all functions $f : S \rightarrow R$ is the same for $A$ and $A'$.

What Follows from NFL?

Implications
- Considering all functions, each black box algorithm has the same performance.
- Considering all functions, each algorithm is as good as random search.
- Hill climbing is as good as Hill descending.

Questions
- Is the result surprising? Perhaps
- Is it interesting? No!!

What Does Not Follow from NFL?

Drawbacks
- No one wants to consider all functions!!
- More realistic is to consider a class of functions or problems.
- NFL Theorem does not hold in this case.
- NFL Theorem useless for understanding realistic scenarios.

Implication
- Restrict considerations to class of functions/problems.
- Are there general results for such cases where NFL does not hold?
- $\Rightarrow$ black box complexity.
Motivation for Complexity Theory

If our evolutionary algorithm performs poorly is it our fault or is the problem intrinsically hard?

Example $\text{NEEDLE}(x) := \prod_{i=1}^{n} x[i]$

Such questions are answered by complexity theory.

Typically one concentrates on computational complexity with respect to run time.

Is this really fair when looking at evolutionary algorithms?

Black Box Optimization

When talking about NFL we have realized classical algorithms and black box algorithms work in different scenarios.

<table>
<thead>
<tr>
<th>Classical Algorithms</th>
<th>Black Box Algorithms</th>
</tr>
</thead>
<tbody>
<tr>
<td>Problem class known</td>
<td>Problem class unknown</td>
</tr>
<tr>
<td>Problem instance known</td>
<td>Problem instance unknown</td>
</tr>
</tbody>
</table>

This different optimization scenario requires a different complexity theory.

We consider Black Box Complexity.

We hope for general lower bounds for all black box algorithms.

Notation

Let $F \subseteq \{ f : S \rightarrow W \}$ be a class of functions, $A$ a black box algorithm for $F$, $x_t$ the $t$-th search point sampled by $A$.

optimization time of $A$ on $f \in F$:

$T_{A,f} = \min \{ t \mid f(x_t) = \max \{ f(x) \in S \} \}$

worst case expected optimization time of $A$ on $F$:

$T_{A,F} = \max \{ E(T_{A,f}) \mid f \in F \}$

black box complexity of $F$:

$B_F = \min \{ T_{A,F} \mid A \text{ is black box algorithm for } F \}$


Comparison With Computational Complexity

$$
F := \left\{ f : \{0,1\}^n \rightarrow \mathbb{R} \mid f(x) = w_0 + \sum_{i=1}^{n} w_i x_i + \sum_{1 \leq i < j \leq n} w_{i,j} x_i x_j \right\}
$$

with $w_i, w_{i,j} \in \mathbb{R}$ known: Optimization of $F$ is NP-hard since MAX-2-SAT is contained in $F$.

Theorem: $B_F = O(n^2)$

Proof

$w_0 = f(0^n)$ (1 search point)

$w_i = f(0^{i-1}10^{n-i}) - w_0$ ($n$ search points)

$w_{i,j} = f(0^i10^{i-1}10^{n-i}) - w_i - w_j - w_0$ ($\binom{n}{2}$ search points)

Compute optimal solution $x^*$ without access to the oracle.

$f(x^*)$ (1 search point)

together: $\binom{n}{2} + n + 2 = O(n^2)$ search points
From Functions to Classes of Functions

**Observation:** ∀F: B_F ≤ |F|

**Consequence:** B_f = 1 for any f — pointless

Can we still have meaningful results for our example functions?

Evolutionary algorithms are often symmetric with respect to 0s and 1s.

**Definition:** For f: \{0, 1\}^n → \mathbb{R}, we define \( f^* := \{ f_a | a \in \{0, 1\}^n \} \) where \( f_a(x) := f(a \oplus x) \).

Clearly, such EAs perform equal on all \( f' \in f^* \).

### A General Upper Bound

**Theorem**

For any \( F \subseteq \{ f: \{0, 1\}^n \rightarrow \mathbb{R} \} \), \( B_F \leq 2^{n-1} + 1/2 \) holds.

**Proof**

Consider pure random search without re-sampling of search points.

For each step \( t \), \( \text{Prob}(\text{find global optimum}) \geq 2^{-n} \).

\[
B_F \leq \sum_{i=1}^{2^n} i \cdot 2^n = 2^n(2^n + 1) = 2^{n-1} + 1/2
\]

Remark

We already knew this from NFL.

---

**An Important Tool**

very powerful general tool for lower bounds known

**Theorem (Yao’s Minimax Principle)**

For all distributions \( p \) over \( I \) and all distributions \( q \) over \( A \):

\[
\min_A \mathbb{E}(T_{A,I} p) \leq \max_I \mathbb{E}(T_{A,I} q)
\]

in words:

We get a lower bound for the worst-case performance of a randomized algorithm by proving a lower bound on the worst-case performance of an optimal deterministic algorithm for an arbitrary probability distribution over the inputs.

**Remark**

We can use this to estimate the performance of deterministic algorithms.

---

**B_{\text{Needle}}**

**Theorem**

\( B_{\text{Needle}}^* = 2^{n-1} + 1/2 \)

**Proof by application of Yao’s Minimax Principle**

The upper bound coincides with the general upper bound.

We consider each \( \text{Needle}_{\text{a}} \) as possible input.

We choose the uniform distribution.

Deterministic algorithms sample the search space in a pre-defined order without re-sampling.

Since the position of the unique global optimum is chosen uniformly at random, we have \( \text{Prob}(T = t) = 2^{-n} \) for all \( t \in \{1, \ldots, 2^n\} \).

This implies \( \mathbb{E}(T) = \sum_{t=1}^{2^n} t \cdot 2^n = 2^n(2^n + 1) = 2^{n-1} + 1/2 \).

Remark

We already knew this from NFL.
**OneMax Theorem**

\[ B_{\text{OneMax}^*} = \Omega \left( \frac{n}{\log n} \right) \]

**Proof by application of Yao’s Minimax Principle:**

We choose the uniform distribution.

A deterministic algorithm is a tree with at least \( 2^n \) nodes: otherwise at least one \( f \in \text{OneMax}^* \) cannot be optimized.

The degree of the nodes is bounded by \( n + 1 \): this is the number of different function values.

Therefore, the average depth of the tree is bounded below by

\[
\left( \log_{n+1} 2^n \right) - 1 \geq \Omega \left( \frac{n}{\log n} \right).
\]

**Remark:** \( B_{\text{OneMax}^*} = O(n) \) is easy to see.

---

**Unimodal Functions**

Consider \( f : \{0, 1\}^n \to \mathbb{R} \).

We call \( x \in \{0, 1\}^n \) a local maximum of \( f \), iff for all \( x' \in \{0, 1\}^n \) with \( H(x, x') = 1 \)

\( f(x) \geq f(x') \) holds.

We call \( f \) unimodal, iff \( f \) has exactly one local optimum.

We call \( f \) weakly unimodal, iff all local optima are global optima, too.

**Observation:** (Weakly) Unimodal functions can be optimized by hill-climbers.

**Path Functions**

Consider the following functions:

\( P := (p_1, p_2, \ldots, p_l) \) with \( p_1 = 1^n \) is a path \( \text{--- not necessarily a simple path.} \)

\( f_P(x) := \begin{cases} 
  n + i & \text{if } x = p_i \text{ and } x \neq p_j \text{ for all } j > i, \\
  \text{OneMax}(x) & \text{if } x \notin P
\end{cases} \)

**Observation:** \( f_P \) is unimodal.

\( P_l(n) := \{ f_P \mid P \text{ has length } l(n) \} \)
### Random Paths

Construct $P$ with length $l(n)$ randomly:

1. $p_{1} := \frac{1}{n}$; $i := 2$
2. While $i \leq l(n)$ do
3. \[ \text{Choose } p_{i} \in \{ x \mid H(x, p_{i-1}) = 1 \} \text{ uniformly at random.} \]
4. $i := i + 1$

For each path $P$ with length $l(n)$, we can calculate the probability to construct $P$ randomly this way.

**Remark:** Paths $P$ constructed this way are likely to contain circles.

### A lower bound on $B_d$

**Theorem:** $\forall \delta$ with $0 < \delta < 1$ constant: $B_d > 2^{n^\delta}$.

For a proof, we want to apply Yao’s Minimax Principle.

We define a probability distribution in the following way:

$\delta < \varepsilon < 1$ constant; $l(n) := 2^{n^\varepsilon}$

For all $f \in \mathcal{U}$ we define

\[ \text{Prob}(f) := \begin{cases} p & \text{if } f \in \mathcal{P}_{l(n)} \text{ and } P \text{ is constructed with prob. } p, \\ 0 & \text{otherwise.} \end{cases} \]
Distance Between Points on the Path

**Lemma**

∀β > 0 constant: \( \exists \alpha(\beta) > 0 \) constant: ∀\( i \leq l(n) - \beta n \):

∀\( j \geq \beta n \): Prob (\( H(p_i, p_{i+j}) \) ≤ \( \alpha(\beta)n \)) = \( 2^{-\Omega(n)} \)

**Proof**: Due to symmetry:

Considering \( i = 1 \) and some \( j \geq \beta n \) suffices.

\( H_t \) := \( H(p_1, p_t) \)

We want to prove: Prob (\( H_j \leq \alpha(\beta)n \)) = \( 2^{-\Omega(n)} \)

We choose \( \alpha(\beta) := \min\{1/50, \beta/5\} \).

Due to the random path construction:

1. \( H_{i+1} \in \{H_i - 1, H_i + 1\} \)
2. Prob (\( H_{i+1} = H_i + 1 \)) = 1 - \( H_i/n \)
3. Prob (\( H_{i+1} = H_i - 1 \)) = \( H_i/n \)

Proof of Lemma Continued

Define \( \gamma := \min\{1/10, j/n\} \).

Observations:

1. \( \gamma \leq 1/10 \)
2. \( \gamma \geq 5\alpha(\beta) \)
3. \( \gamma \) bounded below and above by positive constants

Consider the last \( \gamma n \) steps towards \( p_j \).

Let \( t \) be the first of these steps.

**Note**: \( \gamma \leq j/n \) ⇒ \( \gamma n \leq j \)

**Case 1**: \( H_t \geq 2\gamma n \)

Clearly, \( H_j \geq 2\gamma n - \gamma n \) = \( \gamma n > \alpha(\beta)n \).

Proof of Lemma Continued

We have \( \gamma n \) independent random variable \( S_t, S_{t+1}, \ldots, S_j \in \{0, 1\} \) with Prob (\( S_k = 1 \)) = \( 7/10 \) and \( S := \sum_{k=t}^{j} S_k \).

Apply Chernoff Bounds:

\( E(S) = (7/10)\gamma n \)

\( \text{Prob} \ (S < \text{\( \frac{2}{3} \)}\gamma n) = \text{Prob} \ (S < (1 - \frac{1}{3})\gamma n) \)

\( < e^{-\gamma n(1/7)^2/2} = e^{-\gamma n/140) = 2^{-\Omega(n)}} \)
True Path Length

Lemma with $\beta = 1$ yields:

$\text{Prob} \text{ (return to path after } n \text{ steps)} = 2^{-\Omega(n)}$

$\text{Prob} \text{ (return to path after } \geq n \text{ steps happens anywhere)} = 2^{\alpha} \cdot 2^{-\Omega(n)} = 2^{-\Omega(n)}$

$\text{Prob} \text{ (hit } x) = 1 - 2^{-\Omega(n)}$

We can prove at best lower bound of

$\frac{l'(n) - n + 1}{n^2} > \frac{l(n)}{n} - 1 > 2^{\alpha}$.

An Optimal Deterministic Algorithm

Let $N$ denote the points known not to belong to $P$.
Let $p_i$ denote the best currently known point on the path.

Initially, $N = 0$, $i \geq 1$.

Algorithm decides to sample $x$ as next point.

Case 1: $H(p_i, x) \leq \alpha(1)n$

$\text{Prob} \text{ (hit } x) = 2^{-\Omega(n)}$

Case 2: $H(p_i, x) > \alpha(1)n$

Consider random path construction starting in $p_i$.

Similar to Lemma:

$\text{Prob} \text{ (hit } x) = 2^{-\Omega(n)}$

Later steps

$N \neq \emptyset$

Partition $N$:

$N_{\text{far}} := \{ y \in N \mid H(y, p_i) \geq \alpha(1/2)n \}$

$N_{\text{near}} := N \setminus N_{\text{far}}$

Case 1: $N_{\text{near}} = \emptyset$

Consider random path construction starting in $p_i$.

$A$: path hits $x$

$E$: path hits no point in $N_{\text{far}}$

Clearly, optimal deterministic algorithm avoid $N_{\text{far}}$.

Thus, we are interested in $\text{Prob} (A \mid E)$

$= \frac{\text{Prob}(A \cap E)}{\text{Prob}(E)} \leq \frac{\text{Prob}(A)}{\text{Prob}(E)}$

Clearly, $\text{Prob} (E) = 1 - 2^{-\Omega(n)}$.

Thus, $\text{Prob} (A \mid E) \leq (1 + 2^{-\Omega(n)}) \text{Prob} (A) = 2^{-\Omega(n)}$.

Later steps with close known points

Case 2: $N_{\text{near}} \neq \emptyset$

Knowing points near by can increase $\text{Prob} (A)$.

Ignore the first $n/2$ steps of path construction; consider $p_{i+n/2}$.

$\text{Prob} (N_{\text{near}} = \emptyset \text{ now}) = 1 - 2^{-\Omega(n)}$

Repeat Case 1.
... and that was it for today.

There is more, but you have a good idea of what can be done.

**Reminder** — What we have just seen:
- analysis of the expected optimization time of some evolutionary algorithms by means of
  - fitness-based partitions
  - Markov’s inequality and Chernoff bounds
  - coupon collector’s theorem
  - expected multiplicative distance decrease
  - drift analysis
  - random walks and cover times
  - typical runs
  - example functions
- general limitations for evolutionary algorithms by means of
  - NFL
  - black box complexity

---

**References for Overview of Known Results**

- D. Sudholt (2005): Crossover is provably essential for the ring Model on Trees. In GECCO, 1161–1167.