Introduction

### Theory... 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.

Bionics/Engineering
- evolution is a “natural” enhancing process.
- bionics: algorithmic simulation ⇒ “enhancing” algorithm.
- used for optimization.

Biology
- evolutionary algorithms.
- understanding model of natural evolution.

Computer Science
- evolutionary algorithms.
- successful applications.
- theoretical understanding.

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.

Basic EA
1. compute an initial population $P = \{X_1, \ldots, X_\mu\}$.
2. while (not termination condition)
   - produce an offspring population $P’ = \{Y_1, \ldots, Y_\lambda\}$ by crossover and/or mutation.
   - 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
\[ T = \# \text{fitness function evaluations until an optimal search point is sampled for the first time} \]

Method of Fitness-Based Partitions

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

\[ E(T) = \sum_v \frac{1}{p_v} \]
a bit more general group fitness values

Definition

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

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 \( f : \{0, 1\}^n \to \mathbb{R} \) and an \( f \)-based partition \( L_0, L_1, \ldots, L_k \).

Let \( s_i := \min \sum_{x \in L_i} \sum_{j=i+1}^{k} \frac{1}{n} H(x,y) \left( 1 - \frac{1}{n} \right)^n H(x,y) \)

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

**Definition**

\( f : \{0, 1\}^n \to \mathbb{R} \) is called **linear**

\[\iff \exists w_0, w_1, \ldots, w_n \in \mathbb{R} : \forall x \in \{0, 1\}^n : f(x) = w_0 + \sum_{i=1}^{n} w_i \cdot x[i] \]

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

For \((1+1)\)-EA, \( w.l.o.g. \) \( w_0 = 0, w_1 \geq w_2 \geq \cdots \geq w_n \geq 0 \)

**First Step** define \( f \)-based partition

\[
L_i := \left\{ x \in \{0, 1\}^n \mid \sum_{j=1}^{i} w_j \leq f(x) < \sum_{j=1}^{i+1} w_j \right\}, \ 0 \leq i \leq n
\]

**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** \( E(T_{(1+1)\text{-EA},f}) \leq \sum_{i=0}^{n-1} en = en^2 = O(n^2) \)

Very Simple Example

\((1+1)\)-EA on \textsc{OneMax} \( \left( \textsc{OneMax}(x) = \sum_{i=1}^{n} x[i] \right) \)

**First Step** define \( f \)-based partition

**trivial** for each fitness value one \( L_i \)

\[
L_i := \left\{ x \in \{0, 1\}^n \mid \textsc{OneMax}(x) = i \right\}, \ 0 \leq i \leq n
\]

**Second Step** find lower bounds for \( s_i \)

**Observation** It suffices to flip any 0-bit from the \( n-i \) 0-bits.

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

**Third Step** compute upper bound

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

Generalizing the Method

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

Consider \((1+\lambda)\)-EA on \textsc{LeadingOnes}.

\( \left( \textsc{LeadingOnes}(x) = \sum_{i=1}^{n} \prod_{j=1}^{i} x[j] \right) \)

**First Step** define \( f \)-based partition

**trivial** for each fitness value one \( L_i \)

\[
L_i := \left\{ x \in \{0, 1\}^n \mid \textsc{LeadingOnes}(x) = i \right\}, \ 0 \leq i \leq n
\]

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

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

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

*Source:* De Jong, Wagners (2008): On the choice of the offspring size in evolutionary algorithms. Evolutionary Computation 16(3) 413-440
(1 + \lambda)-ES on LEADINGONES

Second Step  
find lower bounds for \(s_i\)

Observation  
It suffices to flip exactly the leftmost 0-bit. 
\[ s_i \geq 1 - (1 - \frac{1}{en})^\lambda \geq 1 - e^{-\lambda/(en)} \]

Case Inspection

Case 1 \( \lambda \geq en \)
\[ s_i \geq 1 - \frac{1}{e} \]

Case 2 \( \lambda < en \)
\[ s_i \geq \frac{\lambda}{2en} \]

Third Step  
compute upper bound
\[
E(T(1+\lambda)\cdot EA, LEADINGONES) \leq \lambda \cdot \left( \sum_{i=0}^{n-1} \frac{1}{1-e^i} \right) + \left( \sum_{i=0}^{n-1} \frac{2en}{\lambda} \right)
\]

\[= O \left( \lambda \cdot \left( n + \frac{n^2}{4} \right) \right) = O \left( \lambda \cdot n + n^2 \right) \]

Markov Inequality and Chernoff Bounds

**Theorem (Markov Inequality)**

\(X \geq 0\) random variable, \(s > 0\)
\[
\text{Prob}(X \geq s \cdot E(X)) \leq \frac{1}{s}
\]

**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 \(X := \sum_{i=1}^{n} X_i. \)

\[ \forall \delta > 0: \text{Prob}(X > (1 + \delta) \cdot E(X)) < \left( \frac{e^\delta}{(1+\delta)^{1+\delta}} \right) E(X) \]

\[ \forall 0 < \delta < 1: \text{Prob}(X < (1 - \delta) \cdot E(X)) < e^{-E(X)\delta^2/2} \]

A short detour into very basic probability theory

We already know, we care for \(E(T)\) — an expected value.

Often, we care for the probability to deviate from an expected value.

A lot is known about this, we should make use of this.

Consider \(x \in \{0,1\}^{100}\) selected uniformly at random.

**more formal**

for \(i \in \{1,2,\ldots, 100\}: B_i := \begin{cases} 1 & \text{if } i\text{-th bit is } 1 \\ 0 & \text{otherwise} \end{cases} \)

with \(\text{Prob}(B_i = 0) = \text{Prob}(B_i = 1) = \frac{1}{2}\)

\[ B := \sum_{i=1}^{100} B_i \quad \text{clearly } \quad E(B) = 50 \]

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

**Markov**

\[ \text{Prob}(B \geq 75) = \text{Prob}(M \geq \frac{3}{2} \cdot 50) \leq \frac{3}{4} \]

**Chernoff**

\[ \text{Prob}(B \geq 75) = \text{Prob}(B \geq \left( 1 + \frac{1}{2} \right) \cdot 50) \leq \left( \frac{e^{\frac{3}{2}}}{(3/2)^{3/2}} \right)^{50} < 0.0045 \]

**Truth**

\[ \text{Prob}(B \geq 75) = \sum_{i=75}^{100} \left( \binom{100}{i} \right) 2^{-100} \]

\[ \approx \frac{89,410,454,796,450,805,935,325}{316,912,650,057,955,350,374,176,801,344} < 0.000000282 \]
### 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**

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

**Proof.**

Define event $B$: (1+1)-EA initializes with $x = 0^n$

- Clearly $\text{Prob} B = 2^{-n}$

**Observation**

\[ E(T_{(1+1)}\text{-EA}, f \mid B) = n^n \]

since all bits have to flip simultaneously

**Law of Total Probability**

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

### Lower bound for OneMax

**Chernoff bounds**

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

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

**Generalization**

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

### Coupon Collector’s Theorem

**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

\[ E(X) = nH_n \]

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 $c \in \mathbb{R}$. 

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 $\frac{f(x_{opt}) - f(x)}{l}$.

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

**Figure: Distance Decrease**

$D = f(x_{opt}) - f(x)$

Example

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

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

A More Flexible Proof Method

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

How can we find a more flexible method?

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

Idea
consider a more general measure of progress

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

Caution
“Distance” need not be a metric!

Upper Bound Drift Theorem

Drift Theorem (Upper Bound)

Let $A$ be some evolutionary algorithm, $P_t$ its $t$-th population, $f$ some function, $Z$ the set of all possible populations, $d: Z \to \mathbb{R}_0^+$ some distance measure with $d(P) = 0 \iff P \text{ contains an optimum of } f$,
$M = \max \{d(P) \mid P \in Z\}$, $D_t := d(P_{t-1}) - d(P_t)$,
$\Delta := \min \{E(D_t \mid T \geq t) \mid t \in \mathbb{N}_0\}$.
$\Delta > 0 \iff E(T_{A,f}) \leq M/\Delta$

Proof
Observe
$M \geq E\left(\sum_{t=1}^{T} D_t\right)$

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

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

Consider
maximum distance $M := \max \{d(P) \mid P \in Z\}$,
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
$\Delta := \min \{E(D_t \mid T > t) \mid t \in \mathbb{N}_0\}$

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

Proof of the Drift Theorem (Upper Bound)

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

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

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

$= \sum_{i=1}^{\infty} \sum_{t=i}^{\infty} \text{Prob}(T = t) \cdot E(D_i \mid T = t)$
Proof of the Drift Theorem (Upper Bound) (cont.)

\[
\sum_{i=1}^{\infty} \sum_{t=1}^{\infty} \text{Prob}(T = t) \cdot E(D_i \mid T = t) \\
= \sum_{i=1}^{\infty} \sum_{t=1}^{\infty} \text{Prob}(T \geq i) \cdot \text{Prob}(T = t \mid T \geq i) \cdot E(D_i \mid T = t) \\
= \sum_{i=1}^{\infty} \text{Prob}(T \geq i) \sum_{t=1}^{\infty} \text{Prob}(T = t \mid T \geq i) \cdot E(D_i \mid T = t \land T \geq i) \\
= \sum_{i=1}^{\infty} \text{Prob}(T \geq i) \cdot E(D_i \mid T \geq i) \geq \Delta \cdot \sum_{i=1}^{\infty} \text{Prob}(T \geq i) = \Delta \cdot E(T) \\
\text{thus } E(T) \leq \frac{M}{\Delta}
\]

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

\[
\sum_{i=1}^{\infty} \sum_{t=1}^{\infty} \text{Prob}(T \geq i) \cdot \text{Prob}(T = t \mid T \geq i) \cdot E(D_i \mid T = t) \\
= \sum_{i=1}^{\infty} \text{Prob}(T \geq i) \sum_{t=1}^{\infty} \text{Prob}(T = t \mid T \geq i) \cdot E(D_i \mid T = t \land T \geq i) \\
= \sum_{i=1}^{\infty} \text{Prob}(T \geq i) \cdot E(D_i \mid T \geq i) \geq \Delta \cdot \sum_{i=1}^{\infty} \text{Prob}(T \geq i) = \Delta \cdot E(T) \\
\text{thus } E(T) \leq \frac{M}{\Delta}
\]

A Simple Application

Consider \((1, n)-\text{EA on LeadingOnes}\)

**Theorem**

\[E(T_{(1, n)\text{-EA, LeadingOnes}}) = O(n^2)\]

**Proof.**

\[d(x) := n - \text{LeadingOnes}(x) \sim M = n\]

\[
E(d(x_{t-1}) - d(x_t) \mid T > t) \\
\geq 1 \cdot \left(1 - \left(1 - \frac{1}{en}\right)^n\right) - n \cdot \left(1 - \left(1 - \frac{1}{n}\right)^n\right)^n \\
= \Omega(1) \\
\text{thus } E(T) = O(n) \\
\text{thus } E(T_{(1, n)\text{-EA, LeadingOnes}}) = n \cdot E(T) = O(n^2)
\]

Another Example

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

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

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

### Calculation for 011 ⋯ 1

\[
\mathbb{E}(d(x_{t-1}) - d(x_t) \mid T \geq t) = \frac{1}{n} \left(1 - \frac{1}{n}\right)^{n-1} (\ln(3) - \ln(1)) \\
+ \binom{n/2}{1} \left(\frac{1}{n}\right)^2 \left(1 - \frac{1}{n}\right)^{n-2} (\ln(3) - \ln(1 + 1)) \\
- \sum_{b_r=1}^{n/2} \binom{n/2}{b_r} \left(\frac{1}{n}\right)^{1+b_r} \left(1 - \frac{1}{n}\right)^{n-b_r-1} (\ln(1 + b_r) - \ln(3)) \\
- \sum_{b_l=1}^{(n/2)-1} \sum_{b_r=0}^{n/2} \binom{n/2}{b_l} \binom{n/2}{b_r} \left(\frac{1}{n}\right)^{1+b_l+b_r} \left(1 - \frac{1}{n}\right)^{n-b_l-b_r-1} \\
(\ln(1 + 2b_l + b_r) - \ln(3))
= \Omega \left(\frac{1}{n}\right)
\]

### Calculation for 1^{n/2} 01^{(n/2)-1}

\[
\mathbb{E}(d(x_{t-1}) - d(x_t) \mid T \geq t) = \frac{1}{n} \left(1 - \frac{1}{n}\right)^{n-1} (\ln(2) - \ln(1)) \\
- \binom{n/2}{1} \left(\frac{1}{n}\right)^2 \left(1 - \frac{1}{n}\right)^{n-2} (\ln(1 + 2) - \ln(2)) \\
- \sum_{b_r=2}^{(n/2)-1} \binom{n/2}{b_r} \left(\frac{1}{n}\right)^{1+b_r} \left(1 - \frac{1}{n}\right)^{n-b_r-1} (\ln(1 + b_r) - \ln(2)) \\
= \Omega \left(\frac{1}{n}\right)
\]

### 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**
\[
M \geq \mathbb{E} \left(\sum_{i=1}^{T} D_i\right) = \cdots = \sum_{i=1}^{\infty} \mathbb{P}(T \geq i) \mathbb{E}(D_i \mid T \geq i)
\geq \Delta \cdot \sum_{i=1}^{\infty} \mathbb{P}(T \geq i) = \Delta \cdot \mathbb{E}(T)
\]

with

- \( M = \max\{d(P) \mid P \in Z\} \)
- \( \Delta = \min\{\mathbb{E}(d(P_{t-1}) - d(P_t) \mid T \geq t)\} \)

**Result for (1+1)-EA on General Linear Functions**

We have

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

**together**
\( \mathbb{E}(T_{(1+1) \text{ EA}, f}) = O(n \log n) \) for any linear \( f \)
Modification for a Lower Bound Technique

observation only two inequalities need to be reversed
1 \( M \geq \sum \cdots \) with \( M = \max \{d(P) \mid P \in Z\} \)
2 \( \sum \cdots \geq \Delta l \cdot \sum \cdots \) with \( \Delta l = \min \{E(d(P_{l-1}) - d(P)) \mid T \geq t\} \)

Clearly for lower bound \( \Delta u = \max \{E(d(P_{l-1}) - d(P)) \mid T \geq t\} \)
sensible and sufficient for “\( \leq \)"

Clearly for lower bound instead of \( M \min \{d(P) \mid P \in Z\} \)
possible and sufficient for “\( \leq \)”,
but pointless, since \( \min \{d(P) \mid P \in Z\} = 0 \)

Closing the Distance

clearly \( E \left( \sum_{t=1}^{T} D_t \right) \) fixed, if initial population is known

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

Making this concrete
1. \( E(T \mid d(P_0) \geq M_u) \geq M_u / \Delta u \)
2. \( E(T) \geq \text{Prob}(d(P_0) \geq M_u) \cdot E(T \mid d(P_0) \geq M_u) \geq \text{Prob}(d(P_0) \geq M_u) \cdot M_u / \Delta u \)
3. \( E(T) \geq \sum \text{Prob}(d(P_0) \geq d) \cdot d / \Delta u \geq E(d(P_0)) / \Delta u \)

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 decrease of distance
left-most 0-bit flips

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

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 \( \equiv \) increase in fitness

Observation two sources for increase in fitness
1. the left-most 0-bit
2. 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 \( t = 0 \)

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

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

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

\[
= 2^{-n} \square
\]

Expected Increase in Fitness and Expected Initial Distance

\[
\text{E (increase in fitness)} = \sum_{i=1}^{n} \frac{i \cdot \text{Prob (fitness increase = i)}}{n} \cdot 2^{-i} \leq \frac{1}{n} \sum_{i=1}^{\infty} \frac{i}{2^n} = \frac{2}{n}
\]

\[
\text{E (d(x_0))} = n - \sum_{i=1}^{n} \text{Prob (LEADINGONES(x_0) = i)}
\]

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

thus \( \text{E (T(1+1) EA,LEADINGONES)} \geq \frac{(n-1)n}{2} = \Omega(n^2) \)

thus \( \text{E (T(1+1) EA,LEADINGONES)} = \Theta(n^2) \)

Result Cover Time

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 \( 2m(n-1) \).

R. Abliese et al.: Random walks, universal traversal sequences, and the complexity of maze problems, FOCS 1979
Example: Plateaus

Definition

\[ \text{PLATEAU}(x) := \begin{cases} 
|x|_0 & : x \not\in \{1^i0^{n-i}, 0 \leq i \leq n\} \\
n + 1 & : x \in \{1^i0^{n-i}, 0 \leq i < n\} \\
n + 2 & : x = 1^n.
\]

Result: PLATEAU

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).

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 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

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

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

A Steady State GA

\((\mu+1)\)-EA with prob. \(p_c\) 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.

Definition of the Phases

**Notation:**

\(x_i[j]\) is the \(j\)-th bit of \(x_i\)

\(\text{OPT} : n + k \in \{\text{\textsc{Jump}}_k(x_1), \ldots, \text{\textsc{Jump}}_k(x_\mu)\}\)

\[
\begin{array}{c|c|c|c}
   i & C_{i-1} & C_i & T_i \\
   \hline
   1 & \emptyset & \min\{\text{\textsc{Jump}}_k(x_1), \ldots, \text{\textsc{Jump}}_k(x_\mu)\} \geq n & O(\mu n \log n) \\
   2 & C_1 & \left( \forall j \in \{1, \ldots, n\} : \sum_{h=1}^{\mu} (1 - x_h[j]) \leq \frac{n}{4k} \right) \lor \text{OPT} & O(\mu n^2 k) \\
   3 & C_2 & \text{OPT} & O\left(2^{2k}/p_c\right) \\
\end{array}
\]
### Phase 1: Towards the Gap

Reaching some point \( x \) with \( \text{JUMP}_k(x) \geq n \) is not more difficult than optimizing \( \text{OneMax} \).

For \( \mu = 1 \), \( O(n \log n) \) follows.

For larger \( \mu \), observe:

With probability at least \((1 - p_c) \cdot (1 - 1/n)^n = \Omega(1)\)
a copy of a parent is produced.

Making a copy of some \( x_j \) with \( \text{JUMP}_k(x_j) \geq \text{JUMP}_k(x_i) \) is not worse than choosing \( x_i \).

This implies \( O(\mu n \log n) \) as expected length.

Markov’s inequality: failure probability \( p_1 \leq \varepsilon \) for any constant \( \varepsilon > 0 \).

### Phase 2: At the Gap

We are going to prove:

After \( c' \mu n^2 k \) generations (\( c' \) const. suff. large) with probability at most \( p'_2 \)
there are at most \( \mu/(4k) \) zero-bits at the first position.

This implies:

After \( c' \mu n^2 k \) generations (\( c' \) const. suff. large) there are at most \( \mu/(4k) \) zero-bits at any position
with probability at most \( p_2 := n \cdot p'_2 \).

### 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 \( \text{Prob} (A^-_z) \).

### “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 - \frac{1}{n} )</td>
</tr>
<tr>
<td>( F^+_{z,i} )</td>
<td>out of ( k - 1 ) 0-bits ( i ) mutate</td>
<td>( \binom{k-1}{i} \binom{n-k}{i} \left( \frac{1}{n} \right)^2 i \left( 1 - \frac{1}{n} \right)^{n-2i} )</td>
</tr>
<tr>
<td>( G^+_{z,i} )</td>
<td>out of ( k ) 0-bits ( i ) mutate</td>
<td>( \binom{k}{i} \binom{n-k-1}{i-1} \left( \frac{1}{n} \right)^{2i-1} \left( 1 - \frac{1}{n} \right)^{n-2} )</td>
</tr>
</tbody>
</table>

Observe:

\( A^+_z \subseteq B_z \cup \left( B_z \cap C_z \cap \left[ \left( D_z \cap E_z \cap \bigcup_{i=0}^{k-1} F^+_{z,i} \right) \cup \left( D_z \cap E_z \cap \bigcup_{i=1}^{k} G^+_{z,i} \right) \right] \)
A Still Closer Look at $A_z^+$

Using

$$A_z^+ \subseteq B_z \cup \left( B_z \cap C_z \cap \left[ (D_z \cap E_z \cap \bigcup_{i=0}^{k-1} F_{z,i}^+) \cup (D_z \cap E_z \cap \bigcup_{i=1}^{k} G_{z,i}^-) \right] \right)$$

together with

- $\Pr(B_z) = p_c$
- $\Pr(C_z) = \frac{\mu - z}{\mu}$
- $\Pr(D_z) = \frac{e}{\mu n}$
- $\Pr(E_z) = 1 - \frac{1}{n}$
- $\Pr(F_{z,i}^+) = \binom{k-1}{i} \binom{n-k}{i} \left( \frac{1}{n} \right)^{2i} \left( 1 - \frac{1}{n} \right)^{n-2i}$
- $\Pr(G_{z,i}^-) = \binom{k}{i} \binom{n-k-1}{i-1} \left( \frac{1}{n} \right)^{2i-1} \left( 1 - \frac{1}{n} \right)^{n-2i}$

yields some bound on $\Pr(A_z^+)$. 

A Still Closer Look at $A_z^-$

Using

$$A_z^- \supseteq B_z \cap C_z \cap \left[ (D_z \cap E_z \cap \bigcup_{i=1}^{k} F_{z,i}^-) \cup (D_z \cap E_z \cap \bigcup_{i=0}^{k} G_{z,i}^-) \right]$$

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 $z \geq \frac{\mu}{4k}$:

$$\Pr(A_z^-) - \Pr(A_z^+) = \Omega \left( \frac{1}{nk} \right)$$

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>$1 - \frac{1}{n}$</td>
</tr>
<tr>
<td>$F_{z,i}^-$</td>
<td>out of $k - 1$ 0-bits $i - 1$ mutate and out of $n - k$ 1-bits $i$ mutate</td>
<td>$(k-1) \binom{n-k}{i} \left( \frac{1}{n} \right)^{2i-1} \left( 1 - \frac{1}{n} \right)^{n-2}$</td>
</tr>
<tr>
<td>$G_{z,i}^-$</td>
<td>out of $k$ 0-bits $i$ mutate and out of $n - k - 1$ 1-bits $i$ mutate</td>
<td>$\binom{k}{i} \binom{n-k-1}{i-1} \left( \frac{1}{n} \right)^{2i-1} \left( 1 - \frac{1}{n} \right)^{n-2}$</td>
</tr>
</tbody>
</table>

Observe:

$$A_z^- \supseteq B_z \cap C_z \cap \left[ (D_z \cap E_z \cap \bigcup_{i=1}^{k} F_{z,i}^-) \cup (D_z \cap E_z \cap \bigcup_{i=0}^{k} G_{z,i}^-) \right]$$

Bias Towards 1-Bits

We know: $z \geq \frac{\mu}{8k} \Rightarrow \Pr(A_z^-) - \Pr(A_z^+) = \Omega \left( \frac{1}{nk} \right)$

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

$$E(\text{difference in 0-bits}) = \Omega \left( \frac{n^2 k}{nk} \right) = \Omega(nk)$$

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 \( z \geq \mu/(8k) \) holds, things work out nicely.

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

Case 1: at most \( \mu/(8k) \) generations left

number of 0-bits < \( \mu/(8k) + \mu/(8k) = \mu/(4k) \)

no problem

Case 2: more than \( \mu/(8k) \) 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 \( p'_2 = e^{-\Omega(\log^2 n)} \).

Together: \( p_2 = n \cdot p'_2 = e^{-\Omega(\log^2 n) + \ln n} = e^{-\Omega(\log^2 n)} \)

Phase 3: Finding the Optimum

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

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

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

\[
\text{Prob (crossover)} = p_c \\
\text{Prob (create } 1^n \text{ with uniform crossover)} = (1/2)^{2k} \\
\text{Prob (no mutation)} = (1 - 1/n)^n \\
\text{Prob (select two parent without common 0-bit)} \leq k \cdot \frac{\mu/(2k)}{\mu} = \frac{1}{2}
\]

Together:

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

Concluding the Proof

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 \( \varepsilon' > 0 \)

Length of the three phases:

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

Sum of Failure Probabilities:

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

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

Comparison With Computational Complexity

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

with \( w_i, w_{i,j} \in \mathbb{R} \)

known: Optimization of \( \mathcal{F} \) is NP-hard since MAX-2-SAT is contained in \( \mathcal{F} \).

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

Proof

\[ w_0 = f(0^n) \quad (1 \text{ search point}) \]
\[ w_i = f(0^{i-1}10^{n-i}) - w_0 \quad (n \text{ search points}) \]
\[ w_{i,j} = f(0^{i-1}10^{j-i-1}10^{n-j}) - w_i - w_j - w_0 \quad (\binom{n}{2} \text{ search points}) \]

Compute optimal solution \( x^* \) without access to the oracle.

\[ f(x^*) \quad (1 \text{ search point}) \]

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

**Observation:** \( \forall F: B_F \leq |F| \)

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

**Proof**

Consider pure random search without re-sampling of search points. For each step \( t \), \( \text{Prob( find global optimum )} \geq 2^{-n} \).

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

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 \( A \):

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

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.

**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 \( \text{Needle}_a \) 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 \( E(T) = \sum_{i=1}^{2^n} i \cdot 2^n = \frac{2^n(2^{n+1})}{2n+1} = 2^{n-1} + \frac{1}{2} \).

Remark We already knew this from NFL.
### 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

\[
\frac{n}{\log_2 (n+1)} = \Omega \left( \frac{n}{\log n} \right).
\]

**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.

Does this mean unimodal functions are easy to optimize?

### Path Functions

Consider the following functions:

\[
P := (p_1, p_2, \ldots, p_{l(n)}) \text{ with } p_1 = 1^n \text{ is a path — not necessarily a simple path.}
\]

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

**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. $p_1 := 1^n$; $i := 2$
2. While $i \leq l(n)$ do
3. Choose $p_i \in \{x \mid H(x, p_{i-1}) = 1\}$ 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.

Our Proof Strategy

We need to prove that
an optimal deterministic algorithm
needs on average more than $2^{n^\delta}$ steps
to find a global optimum.

We strengthen the position of the deterministic algorithm by
1. letting it know which functions have probability 0.
2. giving away for free the knowledge about any $p_i$ with $f(p_i) \leq f(p_j)$ once $p_j$ is sampled,
3. giving away for free the knowledge about $p_{j+1}, \ldots, p_{j+n}$ if $p_j$ is the current known best path point and some point not on the path is sampled,
4. giving away for free the knowledge about $p_{l(n)}$ (the global optimum) once $p_{j+n}$ is sampled while $p_j$ is the current known best path point.

Deterministic Algorithm Too Strong?

Omit all circles from $P$.
The remaining length $l'(n)$ is called the true length of $P$.

What lower bound can be proven this way?

at best: $(l'(n) - n + 1)/n$

Observation: We need a good lower bound on $l'(n)$.

How likely is it to return to old path points?

alternatively: What is the probability distribution for the Hamming distance points on the path?
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: $(\gamma \leq j/n) \Rightarrow (\gamma n \leq j)$

**Case 1:** $H_t \geq 2\gamma n$

Clearly, $H_j \geq \frac{2\gamma n}{\gamma n}$\(\text{number of steps}\)\(\text{in the beginning}\)\(= \gamma n > \alpha(\beta)n\).

**Case 2:** $H_t < 2\gamma n$

Clearly, $H_i < 3\gamma n$ for all $i \in \{t, \ldots, j\}$.

Therefore, $\text{Prob}(H_i = H_{i-1} + 1) \geq 1 - 3\gamma \geq 7/10$,

$\text{Prob}(H_i = H_{i-1} - 1) \leq 3/10$.

Define independent random variable $S_i, S_{i+1}, \ldots, S_j \in \{0, 1\}$ with

$\text{Prob}(S_k = 1) = 7/10$.

Define $S := \sum_{k=t}^{j} S_k$.

Observation: $\text{Prob}(S \geq (3/5)\gamma n) \leq \text{Prob}(H_j \geq (1/5)\gamma n)$

Since
1. $H_t \geq 0$
2. $\text{Prob}(H_t = H_{t-1} + 1) \geq \text{Prob}(S_t = 1)$
3. $\geq (3/5)\gamma n$ increasing steps $\Rightarrow \leq (2/5)\gamma n$ decreasing steps
4. $H_j \geq (3/5)\gamma n - (2/5)\gamma n$

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

Apply Chernoff Bounds:

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

$\text{Prob}(S < \frac{4}{5}\gamma n)$

$= \text{Prob}(S < (1 - \frac{1}{7}) \frac{7\gamma n}{10})$

$< e^{-(7/10)\gamma n(1/7)^2/2} = e^{-(1/140)\gamma n} = 2^{-\Omega(n)}$
**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**, $N = \emptyset$, $i \geq 1$.

Algorithm decides to sample $x$ as next point.

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

$\Prob(x = p_j \text{ with } j \geq n) = 2^{-\Omega(n)}$.

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

Consider random path construction starting in $p_i$.

**Similar to Lemma:**

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

---

**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 $\Prob(A \mid E)$

$\Prob(A \cap E) = \Prob(A) \Prob(E)$.

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

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

---

**Later Steps With Close Known Points**

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

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

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

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

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