Computational Complexity and Evolutionary Computation

Thomas Jansen
Universität Dortmund
Germany
Thomas.Jansen@udo.edu

Frank Neumann
MPI Saarbrücken
Germany
fne@mpi-inf.mpg.de

GECCO 2007
London, 8th of July, 2007

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

GECCO'07, July 7-11, 2007, London, England, United Kingdom.
ACM 978-1-59593-698-1/07/0007.

History

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 $\Rightarrow$ evolutionary algorithms.

Points of Views

Bionics/Engineering

- evolution is a "natural" enhancing process.
- bionics: algorithmic simulation $\Longrightarrow$ "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.

Scheme of an evolutionary algorithm

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 μ 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 $\Longrightarrow$ 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 solutuion z.

1-point Crossover

- choose a position $p \in \{1, \dots, n\}$ 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 $Prob(z_i = x_i) = Prob(z_i = y_i) = 1/2$

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 $f(x_i)/(\sum_{j=1}^{r} f(x_j))$.
- μ individuals are selected in this way.

(μ, λ)-selection

- μ parents produce λ children.
- select μ best individuals from the children.

$(\mu + \lambda)$-selection

- μ parents produce λ children.
- select μ 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.

$(\mu + \lambda)$-EA

$(\mu + \lambda)$-EA

1. Choose μ individuals uniformly at random from $\{0,1\}^n$.
2. Produce λ children by mutation.
3. Apply $(\mu + \lambda)$-selection to parents and children.
4. Go to 2.)

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

Algorithms in Computer Science

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?"*

'Time' and Evolutionary Algorithms

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

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 $\text{E}\left(\text{time to increase fitness from } v\right) = \frac{1}{p_v}$

Observation $\text{E}\left(T\right) = \sum\limits_{v} \frac{1}{p_v}$

a bit more general group fitness values

Method of Fitness-Based Partitions

Definition

For $f\colon \{0,1\}^n \to \mathbb{R}$, $L_0, L_1, \dots, L_k \subseteq \{0,1\}^n$ with

1. $\forall i \neq j \in \{0,1,\dots,k\}\colon L_i \cap L_j = \emptyset$
2. $\bigcup\limits_{i=0}^{k} L_i = \{0,1\}^n$
3. $\forall i < j \in \{0,1,\dots,k\}\colon \forall x \in L_i\colon \forall y \in L_j\colon f(x) < f(y)$
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 parition.

Remember An f-based partition
partitions the search space in accordance to fitness values
grouping fitness values arbitrarily.

Droste/Jansen/Wegener: On the analysis of the (1+1) EA. *Theoretical Computer Science* 276:51–81

Upper Bounds with f-Based Partitions

Theorem

Consider (1+1)-EA on $f\colon \{0,1\}^n \to \mathbb{R}$ and an f-based partition $L_0, L_1, \dots, L_k$.

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

for all $i \in \{0,1,\dots,k-1\}$.

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

Hint most often, very simple lower bounds for s_i suffice

Very Simple Example

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

First Step define f-based partition

trivial for each fitness value one L_i

$L_i := \{x \in \{0,1\}^n \mid \textsc{OneMax}(x) = i\},\ 0 \le i \le n$

Second Step find lower bounds for s_i

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

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

$\left(\left(1-\frac{1}{n}\right)^{n-1} \ge \frac{1}{e} \ge \left(1-\frac{1}{n}\right)^{n}\right)$

Third Step compute upper bound

$\mathrm{E}\left(T_{\text{(1+1)-EA},\textsc{OneMax}}\right) \le \sum\limits_{i=0}^{n-1} \frac{en}{n-i} = en \cdot \sum\limits_{i=1}^{n} \frac{1}{i} = O(n \log n)$

Example: Result for a Class of Functions

Definition

$f\colon \{0,1\}^n \to \mathbb{R}$ is called linear

$\Leftrightarrow \exists w_0, w_1, \dots, w_n \in \mathbb{R}\colon \forall x \in \{0,1\}^n\colon f(x) = w_0 + \sum\limits_{i=1}^{n} w_i \cdot x[i]$

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

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

First Step define f-based partition

$L_i := \left\{x \in \{0,1\}^n \mid \sum\limits_{j=1}^{i} w_j \le f(x) < \sum\limits_{j=1}^{i+1} w_j\right\},\ 0 \le i \le n$

Second Step find lower bounds for s_i

Observation There is always at least 1-bit-mutation for leaving L_i.

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

Third Step $\mathrm{E}\left(T_{\text{(1+1)-EA},f}\right) \le \sum\limits_{i=0}^{n-1} en = en^2$

Generalizing the Method

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

Consider $(1+\lambda)$-EA on LeadingOnes. $\left(\textsc{LeadingOnes}(x) = \sum\limits_{i=1}^{n} \prod\limits_{j=1}^{i} x[j]\right)$

First Step define f-based partition

trivial for each fitness value one L_i

$L_i := \{x \in \{0,1\}^n \mid \textsc{LeadingOnes}(x) = i\},\ 0 \le i \le n$

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

$s_i :=$ Prob (leave L_i in one generation)

Observation $\mathrm{E}\left(T_{(1+\lambda)\text{-EA},f}\right) \le \lambda \cdot \sum\limits_{i=0}^{k-1} \frac{1}{s_i}$

Jansen/De Jong/Wegener (2005): On the choice of the offspring population size in evolutionary algorithms. *Evolutionary Computation* 13(4):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.
$s_i \geq 1 - \left(1 - \frac{1}{en}\right)^{\lambda} \geq 1 - e^{-\lambda/(en)}$

Case Inspection Case 1 $\lambda \geq en$
$s_i \geq 1 - \frac{1}{e}$

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

Third Step compute upper bound

$$\mathrm{E}\left(T_{(1+\lambda)\text{-EA},\textsc{LeadingOnes}}\right) \leq \lambda \cdot \left(\left(\sum_{i=0}^{n-1} \frac{1}{1-e^{-1}}\right) + \left(\sum_{i=0}^{n-1} \frac{2en}{\lambda}\right)\right)$$
$$= O\left(\lambda \cdot \left(n + \frac{n^2}{\lambda}\right)\right) = O\left(\lambda \cdot n + n^2\right)$$

Markov Inequality and Chernoff Bounds

Theorem (Markov Inequality)

$X \geq 0$ random variable, $s > 0$
Prob $(X \geq s \cdot \mathrm{E}(X)) \leq \frac{1}{s}$

Theorem (Chernoff Bounds)

Let $X_1, X_2, \dots, X_n \colon \Omega \to \{0,1\}$ independent random variables with
$\forall i \in \{1, 2, \dots, n\} \colon 0 < \text{Prob}(X_i = 1) < 1$.
Let $X := \sum_{i=1}^{n} X_i$.

$\forall \delta > 0 \colon \text{Prob}(X > (1+\delta) \cdot \mathrm{E}(X)) < \left(\frac{e^{\delta}}{(1+\delta)^{1+\delta}}\right)^{\mathrm{E}(X)}$
$\forall 0 < \delta < 1 \colon \text{Prob}(X < (1-\delta) \cdot \mathrm{E}(X)) < e^{-\mathrm{E}(X)\delta^2/2}$

Motwani/Raghavan (1998): *Randomized Algorithms*

Some Useful Background Knowledge

a short detour into very basic probability theory

We already know, we care for $\mathrm{E}(T)$ — 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.

A Very Simple Application

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

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

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

Markov Prob $(B \geq 75) =$ Prob $\left(M \geq \frac{3}{2} \cdot 50\right) \leq \frac{2}{3}$

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

Truth Prob $(B \geq 75) = \sum_{i=75}^{100} \binom{100}{i} 2^{-100}$
$= \frac{89{,}310{,}453{,}796{,}450{,}805{,}935{,}325}{316{,}912{,}650{,}057{,}057{,}350{,}374{,}175{,}801{,}344} < 0.000000282$

The Law of Total Probability

Theorem (Law of Total Probability)

Let B_i with $i \in I$ be a partition of some probability space Ω.
$\forall A \subseteq \Omega$: $\text{Prob}(A) = \sum_{i \in I} \text{Prob}(A \mid B_i) \cdot \text{Prob}(B_i)$

immediate consequence $\quad \text{Prob}(A) \geq \text{Prob}(A \mid B) \cdot \text{Prob}(B)$

Useful for lower bounds
when some event "determines" expected optimization time

A Very Simple Example

Consider (1+1)-EA on $f \colon \{0,1\}^n \to \mathbb{R}$

with $f(x) := \begin{cases} n - \frac{1}{2} & \text{if } x = 0^n \\ \text{ONEMAX}(x) & \text{otherwise} \end{cases}$.

Theorem

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

Proof.

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

Observation $\quad \mathrm{E}\left(T_{(1+1)\text{-EA}}, f \mid B\right) = n^n$
since all bits have to flip simultaneously

Law of Total Probability
$\mathrm{E}\left(T_{(1+1)\text{-EA}}, f\right) \geq n^n \cdot 2^{-n} = \left(\frac{n}{2}\right)^n$ $\square$

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

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

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

Generalization

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

S. Droste, T. Jansen, I. Wegener: On the analysis of the (1+1) Evolutionary Algorithm. Theoretical Computer Science, 2002.

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 nth Harmonic number, and

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

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

R. Motwani, P. Raghavan: Randomized Algorithms, Cambridge University Press, 1995.

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 $O(l \cdot d_{max})$ operations to reach optimum.
- Assume: expected time for each operation is at most r.
- Upper bound $O(r \cdot l \cdot d_{max})$ to obtain an optimal solution.

F. Neumann, I. Wegener: Randomized local search, evolutionary algorithms, and the minimum spanning tree problem, GECCO 2004.

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 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 $d\colon Z \to \mathbb{R}_0^+$, ($Z$ set of all populations)
with $d(P) = 0 \Leftrightarrow P$ contains optimal solution

Caution "Distance" need not be a metric!

Drift

Define distance $d\colon Z \to \mathbb{R}_0^+$, ($Z$ 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 $M := \max\{d(P) \mid P \in Z\}$,
decrease in distance $D_t := d(P_{t-1}) - d(P_t)$

Definition $\mathrm{E}(D_t \mid T \geq t)$ is called drift.

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

Drift Theorem (Upper Bound) $\Delta > 0 \Rightarrow \mathrm{E}(T) \leq M/\Delta$

He/Yao (2004): A study of drift analysis for estimating computation time of evolutionary algorithms. *Natural Computing* 3(1):21–35

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, $d\colon Z \to \mathbb{R}_0^+$ some distance measure with
$d(P) = 0 \Leftrightarrow P$ contains an optimum of f,
$M = \max\{d(P) \mid P \in Z\}$, $D_t := d(P_{t-1}) - d(P_t)$,
$\Delta := \min\{\mathrm{E}(D_t \mid T \geq t) \mid t \in \mathbb{N}_0\}$.
$\Delta > 0 \Rightarrow \mathrm{E}(T_{A,f}) \leq M/\Delta$

Proof

Observe $\quad M \geq \mathrm{E}\left(\sum\limits_{t=1}^{T} D_t\right)$

Proof of the Drift Theorem (Upper Bound)

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

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

$$
\begin{aligned}
&\geq \sum_{i=1}^{\infty} \sum_{t=i}^{\infty} \mathrm{Prob}(T=t) \cdot \mathrm{E}(D_i \mid T=t)\\
&= \sum_{i=1}^{\infty} \sum_{t=i}^{\infty} \mathrm{Prob}(T \geq i) \cdot \mathrm{Prob}(T=t \mid T \geq i) \cdot \mathrm{E}(D_i \mid T=t)\\
&= \sum_{i=1}^{\infty} \mathrm{Prob}(T \geq i) \sum_{t=i}^{\infty} \mathrm{Prob}(T=t \mid T \geq i) \cdot \mathrm{E}(D_i \mid T=t \wedge T \geq i)\\
&= \sum_{i=1}^{\infty} \mathrm{Prob}(T \geq i) \sum_{t=1}^{\infty} \mathrm{Prob}(T=t \mid T \geq i) \cdot \mathrm{E}(D_i \mid T=t \wedge T \geq i)\\
&= \sum_{i=1}^{\infty} \mathrm{Prob}(T \geq i)\, \mathrm{E}(D_i \mid T \geq i) \geq \Delta \cdot \sum_{i=1}^{\infty} \mathrm{Prob}(T \geq i) = \Delta \cdot \mathrm{E}(T)
\end{aligned}
$$

thus $\quad \mathrm{E}(T) \leq \frac{M}{\Delta}$ $\square$

A Simple Application

Consider $(1, n)$-EA on LeadingOnes

Theorem

$\mathrm{E}\left(T_{(1,\,n)\text{-EA},\textsc{LeadingOnes}}\right) = O(n^2)$

Proof.

$d(x) := n - \textsc{LeadingOnes}(x) \quad \rightsquigarrow \quad M = n$

$$
\begin{aligned}
&\mathrm{E}(d(x_{t-1}) - d(x_t) \mid T > t)\\
\geq\ & 1 \cdot \left(1 - \left(1 - \frac{1}{en}\right)^n\right) - n \cdot \left(1 - \left(1 - \frac{1}{n}\right)^n\right)^n\\
=\ & \Omega(1)
\end{aligned}
$$

thus $\quad \mathrm{E}(T) = O(n)$

thus $\quad \mathrm{E}\left(T_{(1,\,n)\ \text{EA},\textsc{LeadingOnes}}\right) = n \cdot \mathrm{E}(T) = O(n^2)$ $\square$

Another Example

Consider (1+1)-EA on linear function $f\colon \{0,1\}^n \to \mathbb{R}$
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
$M = \max\{d(x) \mid x \in \{0,1\}^n\} = \ln\left(1 + \frac{3}{2}n\right) = \Theta(\ln n)$

He/Yao (2004): A study of drift analysis for estimating computation time of evolutionary algorithms. *Natural Computing* 3(1):21–35

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 $\mathsf{E}\left(d(x_{t-1}) - d(x_t) \mid T \geq t\right)$

Observe minimal for $x_{t-1} = 011\cdots1$ or $\underbrace{11\cdots1}_{\text{left}}\underbrace{01\cdots1}_{\text{right}}$

Consider separately and do tedious calculations…

Calculation for $011\cdots1$

$$\begin{aligned}
&\mathsf{E}\left(d(x_{t-1}) - d(x_t) \mid T \geq t\right)\\
= {} & \frac{1}{n}\left(1-\frac{1}{n}\right)^{n-1}(\ln(3)-\ln(1))\\
&+\binom{n/2}{1}\left(\frac{1}{n}\right)^2\left(1-\frac{1}{n}\right)^{n-2}(\ln(3)-\ln(1+1))\\
&-\sum_{b_r=3}^{n/2}\binom{n/2}{b_r}\left(\frac{1}{n}\right)^{1+b_r}\left(1-\frac{1}{n}\right)^{n-b_r-1}(\ln(1+b_r)-\ln(3))\\
&-\sum_{b_l=1}^{(n/2)-1}\sum_{b_r=0}^{n/2}\binom{(n/2)-1}{b_l}\binom{n/2}{b_r}\left(\frac{1}{n}\right)^{1+b_l+b_r}\left(1-\frac{1}{n}\right)^{n-b_l-b_r-1}\\
&\quad(\ln(1+2b_l+b_r)-\ln(3))\\
= {} & \Omega\left(\frac{1}{n}\right)
\end{aligned}$$

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

$$\begin{aligned}
&\mathsf{E}\left(d(x_{t-1}) - d(x_t) \mid T \geq t\right)\\
= {} & \frac{1}{n}\left(1-\frac{1}{n}\right)^{n-1}(\ln(2)-\ln(1))\\
&-\binom{n/2}{1}\left(\frac{1}{n}\right)^2\left(1-\frac{1}{n}\right)^{n-2}(\ln(1+2)-\ln(2))\\
&-\sum_{b_r=2}^{(n/2)-1}\binom{(n/2)-1}{b_r}\left(\frac{1}{n}\right)^{1+b_r}\left(1-\frac{1}{n}\right)^{n-b_r-1}(\ln(1+b_r)-\ln(2))\\
= {} & \Omega\left(\frac{1}{n}\right)
\end{aligned}$$

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

We have

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

together $\quad \mathrm{E}\left(T_{(1+1)\text{ EA},f}\right) = O(n \log n)$ for any linear f

Drift Analysis of Lower Bounds

We have $\quad$ drift analysis for upper bounds

How can we obtain lower bounds when analyzing drift?

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

Remember $\quad M \geq \mathrm{E}\left(\sum\limits_{t=1}^{T} D_t\right) = \cdots = \sum\limits_{i=1}^{\infty} \mathrm{Prob}\left(T \geq i\right) \mathrm{E}\left(D_i \mid T \geq i\right)$

$\geq \Delta \cdot \sum\limits_{i=1}^{\infty} \mathrm{Prob}\left(T \geq i\right) = \Delta \cdot \mathrm{E}\left(T\right)$

with

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

Modification for a Lower Bound Technique

observation $\quad$ 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 $\Delta_l = \min\{\mathrm{E}\left(d(P_{t-1}) - d(P_t) \mid T \geq t\right)\}$

clearly $\quad$ for lower bound $\Delta_u = \max\{\mathrm{E}\left(d(P_{t-1}) - d(P_t) \mid T \geq t\right)\}$
sensible and sufficient for "$\leq$"

clearly $\quad$ for lower bound instead of M $\min\{d(P) \mid P \in Z\}$
possible and sufficient for "$\leq$",
but pointless, since $\min\{d(P) \mid P \in Z\} = 0$

He/Yao (2004): A study of drift analysis for estimating computation time of evolutionary algorithms. *Natural Computing* 3(1):21–35

Closing the Distance

clearly $\quad \mathrm{E}\left(\sum\limits_{t=1}^{T} D_t\right)$ fixed, if initial population is known

thus $\quad$ lower bound on $d(P_0)$ yields lower bound on $\mathrm{E}\left(T\right)$

making this concrete

- $\mathrm{E}\left(T \mid d(P_0) \geq M_u\right) \geq M_u/\Delta_u$
- $\mathrm{E}\left(T\right) \geq \mathrm{Prob}\left(d(P_0) \geq M_u\right) \cdot \mathrm{E}\left(T \mid d(P_0) \geq M_u\right) \geq \mathrm{Prob}\left(d(P_0) \geq M_u\right) \cdot M_u/\Delta_u$
- $\mathrm{E}\left(T\right) \geq \sum \mathrm{Prob}\left(d(P_0) \geq d\right) \cdot d/\Delta_u \geq \mathrm{E}\left(d(P_0)\right)/\Delta_u$

thus $\quad$ 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 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?

Standard Bit Mutations on Uniformly Distributed Bits

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

clearly holds for $t = 0$

$$\begin{aligned}
\text{Prob}\,(x_t = x) &= \sum_{x' \in \{0,1\}^n} \text{Prob}\,((x_{t-1} = x') \wedge (\text{mut}(x') = x)) \\
&= \sum_{x' \in \{0,1\}^n} \text{Prob}\,(x_{t-1} = x') \cdot \text{Prob}\,(\text{mut}(x') = x) \\
&= \sum_{x' \in \{0,1\}^n} 2^{-n} \cdot \text{Prob}\,(\text{mut}(x') = x) \\
&= 2^{-n} \sum_{x' \in \{0,1\}^n} \text{Prob}\,(\text{mut}(x) = x') \\
&= 2^{-n} \quad \square
\end{aligned}$$

Expeced Decrease in Distance on LEADINGONES

Note decrease in distance $\widehat{=}$ 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

Expected Increase in Fitness and Expected Intial Distance

$$\begin{aligned}
&\text{E}\,(\text{increase in fitness}) \\
&= \sum_{i=1}^{n} i \cdot \text{Prob}\,(\text{fitness increase} = i) \\
&\leq \sum_{i=1}^{n} i \cdot \frac{1}{n} \cdot 2^{-i} \leq \frac{1}{n} \sum_{i=1}^{\infty} \frac{i}{2^i} = \frac{2}{n}
\end{aligned}$$

$$\begin{aligned}
\text{E}\,(d(x_0)) &= n - \sum_{i=1}^{n} i \cdot \text{Prob}\,(\text{LEADINGONES}(x_0) = i) \\
&= n - \sum_{i=1}^{n} \frac{i}{2^{i+1}} \geq n - \frac{1}{2} \sum_{i=1}^{\infty} \frac{i}{2^i} = n - 1
\end{aligned}$$

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

thus $\text{E}\left(T_{(1+1)\ \text{EA},\text{LEADINGONES}}\right) = \Theta(n^2)$

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 $2m(n-1)$.

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

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: $\textbf{Prob}\left(T_{\mathrm{EA},f} \le \sum_{i=1}^{k} T_i\right) \ge 1 - \sum_{i=1}^{k} p_i$

Example: Plateaus

Definition

$$\text{Plateau}(x) := \begin{cases} n - \mathit{OneMax}(x) & : \quad x \notin \{1^i 0^{n-i}, 0 \le i \le n\} \\ n+1 & : \quad x \in \{1^i 0^{n-i}, 0 \le i < n\} \\ n+2 & : \quad x = 1^n. \end{cases}$$

Upper bound (RLS)

- Solution with fitness $\ge n+1$ 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"

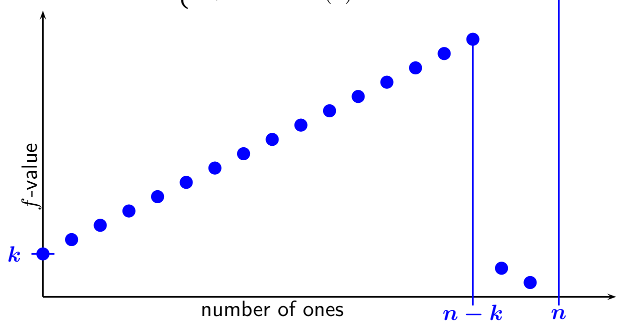
In this case, a failure in any phase can be described as a restart.

This yields: $\textbf{E}\left(T_{\mathrm{EA},f}\right) \le \frac{\sum_{i=1}^{k} T_i}{1 - \sum_{i=1}^{k} p_i}$

A Concrete Example

$\text{JUMP}_k(x)\colon \{0,1\}^n \to \mathbb{R}$ with $k \in \{1, 2, \dots, n\}$

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



GA on JUMP_k

Theorem

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

Method of Proof: Typical Run

Jansen/Wegener (2002): On the analysis of evolutionary algorithms - a proof that crossover really can help. *Algorithmica* 34(1):47–66

A Steady State GA

(μ+1)-EA with prob. p_c for uniform crossover

1. **Initialization**
   Choose $x_1, \dots, x_\mu \in \{0,1\}^n$ uniformly at random.
2. **Selection and Variation**
   With probability p_c:
   Select z_1 and z_2 independently from $x_1, \dots, x_\mu$.
   $z :=$ uniform crossover(z_1, z_2)
   $y :=$ standard $1/n$ bit mutation(z)
   Otherwise:
   Select z from $x_1, \dots, x_\mu$.
   $y :=$ standard $1/n$ bit mutation(z)
3. **Selection for Replacement**
   If $f(y) \geq \min\{f(x_1), \dots, f(x_\mu)\}$
   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

OPT: $n + k \in \{\text{JUMP}_k(x_1), \dots, \text{JUMP}_k(x_\mu)\}$

i	C_{i-1}	C_i	T_i
1	$\emptyset$	$\min\{\text{JUMP}_k(x_1), \dots, \text{JUMP}_k(x_\mu)\} \geq n$	$O(\mu n \log n)$
2	C_1	$\left(\forall j \in \{1, \dots, n\}\colon \sum_{h=1}^{\mu} (1 - x_h[j]) \leq \frac{\mu}{4k}\right) \vee$ OPT	$O(\mu n^2 k)$
3	C_2	OPT	$O(2^{2k}/p_c)$

Phase 1: Towards the Gap

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

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

For larger μ, 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 Prob$\left(A_z^+\right)$ and Prob$\left(A_z^-\right)$.

A Closer Look at A_z^+

"Smaller/Simpler" Events:

event	description	probability
B_z	do crossover	p_c
C_z	at selection for replacement,	
	select x with 1 at first position	$(\mu - z)/\mu$
D_z	at selection for reproduction,	
	select parent with 0 at first position	z/μ
E_z	no mutation at first position	$1 - \frac{1}{n}$
$F_{z,i}^+$	out of $k-1$ 0-bits i mutate and	
	out of $n-k$ 1-bits i mutate	$\binom{k-1}{i}\binom{n-k}{i}\left(\frac{1}{n}\right)^{2i}\left(1-\frac{1}{n}\right)^{n-2i}$
$G_{z,i}^+$	out of k 0-bits i mutate and	
	out of $n-k-1$ 1-bits $i-1$ mutate	$\binom{k}{i}\binom{n-k-1}{i-1}\left(\frac{1}{n}\right)^{2i-1}\left(1-\frac{1}{n}\right)^{n-2}$

Observe:

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

A Still Closer Look at A_z^+

Using

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

together with

Prob $(B_z) = p_c$

Prob $(C_z) = \frac{\mu - z}{\mu}$

Prob $(D_z) = \frac{z}{mu}$

Prob $(E_z) = 1 - \frac{1}{n}$

Prob $\left(F_{z,i}^+\right) = \binom{k-1}{i}\binom{n-k}{i}\left(\frac{1}{n}\right)^{2i}\left(1-\frac{1}{n}\right)^{n-2i}$

Prob $\left(G_{z,i}^+\right) = \binom{k}{i}\binom{n-k-1}{i-1}\left(\frac{1}{n}\right)^{2i-1}\left(1-\frac{1}{n}\right)^{n-2i}$

yields some bound on Prob (A_z^+).

A Closer Look at A_z^-

"Smaller/Simpler" Events:

event	description	probability
B_z	do crossover	p_c
C_z	at selection for replacement, select x with 1 at first position	$(\mu - z)/\mu$
D_z	at selection for reproduction, select parent with 0 at first position	z/μ
E_z	no mutation at first position	$1 - \frac{1}{n}$
$F_{z,i}^-$	out of $k-1$ 0-bits $i-1$ mutate and out of $n-k$ 1-bits i mutate	$\binom{k-1}{i-1}\binom{n-k}{i}\left(\frac{1}{n}\right)^{2i-1}\left(1-\frac{1}{n}\right)^{n-2}$
$G_{z,i}^-$	out of k 0-bits i mutate and out of $n-k-1$ 1-bits i mutate	$\binom{k}{i}\binom{n-k-1}{i}\left(\frac{1}{n}\right)^{2i-1}\left(1-\frac{1}{n}\right)^{n-2}$

Observe:

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

A Still Closer Look at A_z^-

Using

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

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 $z \geq \frac{\mu}{8k}$:

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

Bias Towards 1-Bits

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

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

E (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.

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

Prob (crossover) $= p_c$

Prob (create 1^n with uniform crossover) $= (1/2)^{2k}$

Prob (no mutation) $= (1 - 1/n)^n$

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

Together:
Prob (find optimum in current generation) $= \Omega(p_c \cdot 2^{-2k})$

Concluding Phase 3

We have
Prob (find optimum in current generation) $= \Omega(p_c \cdot 2^{-2k})$

Prob (find optimum in $c_3 2^{2k}/p_c$ 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:
$O(\mu n \log n) + O(\mu n^2 k) + O(2^{2k}/p_c) = O(\mu n^2 k + 2^{2k}/p_c)$

Sum of Failure Probabilities: $\varepsilon + e^{-\Omega(\log^2 n)} + \varepsilon' \leq \varepsilon^* < 1$

$\mathrm{E}\left(T_{\mathrm{GA}(\mu,\, p_c)}\right) = O(\mu n^2 k + 2^{2k}/p_c)$ $\square$

Black Box Optimization

Setting

- Given two finite spaces S and R.
- Find for a given function $f: S \to R$ an optimal solution.
- Count number of fitness evalutions.
- 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 $f: S \to R$ 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 $f: S \to R$ is the same for A and A'.

D.H. Wolpert, W. G. Macready: No Free Lunch Theorems for Optimization, IEEE Transactions on Evolutionary Computation, 1997.

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 szenarios.

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.

classical algorithms	black box algorithms
problem class known	problem class known
problem instance known	problem instance unknown

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

optimization time of A on $f \in \mathcal{F}$:
$T_{A,f} = \min\{t \mid f(x_t) = \max\{f(x) \in S\}\}$

worst case expected optimization time of A on $\mathcal{F}$:
$T_{A,\mathcal{F}} = \max\{\mathrm{E}(T_{A,f}) \mid f \in \mathcal{F}\}$

black box complexity of F:
$B_{\mathcal{F}} = \min\{T_{A,\mathcal{F}} \mid A \text{ is black box algorithm for } \mathcal{F}\}$

Droste/Jansen/Wegener (2006): Upper and lower bounds for randomized search heuristics in black-box optimization. *Theory of Computing Systems* 39(4):525–544

Comparison With Computational Complexity

$\mathcal{F} :=$
$$\left\{ f \colon \{0,1\}^n \to \mathbb{R} \mid f(x) = w_0 + \sum_{i=1}^{n} w_i x_i + \sum_{1 \le i < j \le n} w_{i,j} x_i x_j \right\}$$
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: $B_{\mathcal{F}} = O(n^2)$

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

From Functions to Classes of Functions

Observation: $\forall \mathcal{F}\colon B_{\mathcal{F}} \leq |\mathcal{F}|$

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 $f\colon \{0,1\}^n \to \mathbb{R}$, we define $f^* := \{f_a \mid a \in \{0,1\}^n\}$
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 $\mathcal{F} \subseteq \{f\colon \{0,1\}^n \to \mathbb{R}\}$, $B_{\mathcal{F}} \leq 2^{n-1} + 1/2$ holds.

Proof

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

$$B_{\mathcal{F}} \leq \sum_{i=1}^{2^n} i \cdot 2^n$$
$$= \frac{2^n(2^n+1)}{2^{n+1}} = 2^{n-1} + \frac{1}{2}$$ □

An Important Tool

very powerful general tool for lower bounds known

Theorem (Yao's Minimax Principle)

For all distributions p over $\mathcal{I}$ and all distributions q over $\mathcal{A}$:
$\min_A \mathrm{E}\left(T_{A,I_p}\right) \leq \max_I \mathrm{E}\left(T_{A_q,I}\right)$

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.

B_{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 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 Prob $(T = t) = 2^{-n}$ for all $t \in \{1, \ldots, 2^n\}$.
This implies $\mathrm{E}(T) = \sum_{i=1}^{2^n} i \cdot 2^n = \frac{2^n(2^n+1)}{2^{n+1}} = 2^{n-1} + \frac{1}{2}$. □

Remark We already knew this from NFL.

B_{OneMax^*}

Theorem

$B_{\text{OneMax}^*} = \Omega(n/\log n)$

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
$(\log_{n+1} 2^n) - 1$
$= \frac{n}{\log_2(n+1)} = \Omega(n/\log n)$. □

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

Unimodal Functions

Consider $f\colon \{0,1\}^n \to \mathbb{R}$.

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

Unimodal functions

class of unimodal functions:
$\mathcal{U} := \{f\colon \{0,1\}^n \to \mathbb{R} \mid f \text{ unimodal}\}$

What is $B_{\mathcal{U}}$?

We want to find a lower bound on $B_{\mathcal{U}}$.

Remember: For any point not optimal under a unimodal function, there exists a path to the global optimum

Definition: l points $p_1, p_2, \ldots, p_l$ with $\mathrm{H}(p_i, p_{i+1}) = 1$ for all $1 \leq i < l$ form a path of length l.

Droste/Jansen/Wegener (2006): Upper and lower bounds for randomized search heuristics in black-box optimization. *Theory of Computing Systems* 39(4):525–544

Path Functions

Consider the following functions:

$P := (p_1, p_2, \ldots, p_{l(n)})$ with $p_1 = 1^n$ 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.

$\mathcal{P}_{l(n)} := \{f_P \mid P \text{ has length } l(n)\}$

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 $p_i \in \{x \mid \mathrm{H}(x, p_{i-1}) = 1\}$ 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.

A lower bound on $B_{\mathcal{U}}$

Theorem: $\forall \delta$ with $0 < \delta < 1$ constant: $B_{\mathcal{U}} > 2^{n^\delta}$.

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

$$\mathrm{Prob}\,(f) := \begin{cases} p & \text{if } f \in \mathcal{P}_{l(n)} \text{ and } P \text{ is constructed with prob. } p, \\ 0 & \text{otherwise.} \end{cases}$$

Our Proof Strategy

We need to prove that

an optimal deterministic algorithm

needs on average more than 2^{n^δ} 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 froms 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?

Distance Between Points on the Path

Lemma

$\forall \beta > 0$ *constant*: $\exists \alpha(\beta) > 0$ *constant*: $\forall i \leq l(n) - \beta n$:
$\forall j \geq \beta n$: *Prob* $(\mathrm{H}(p_i, p_{i+j}) \leq \alpha(\beta) n) = 2^{-\Omega(n)}$

Proof: Due to symmetry:
Considering $i = 1$ and some $j \geq \beta n$ suffices.

$H_t := \mathrm{H}(p_1, p_t)$

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:

- $H_{t+1} \in \{H_t - 1, H_t + 1\}$
- Prob $(H_{t+1} = H_t + 1) = 1 - H_t/n$
- Prob $(H_{t+1} = H_t - 1) = H_t/n$

Proof of Lemma Continued

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

Observations:

- $\gamma \leq 1/10$
- $\gamma \geq 5\alpha(\beta)$
- γ bounded below and above by positive constants

Consider the last γn 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, $H_j \geq \underbrace{2\gamma n}_{\text{in the beginning}} - \overbrace{\gamma n}^{\text{number of steps}} = \gamma n > \alpha(\beta) n.$

Proof of Lemma Continued

Case 2: $H_t < 2\gamma n$

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

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

Define independent random variable $S_t, S_{t+1}, \dots, S_j \in \{0, 1\}$ with Prob $(S_k = 1) = 7/10$.

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

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

Since

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

Proof of Lemma Continued

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

Apply Chernoff Bounds:

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

Prob $\left(S < \frac{3}{5}\gamma n\right)$
$=$ Prob $\left(S < \left(1 - \frac{1}{7}\right) \frac{7}{10}\gamma n\right)$
$< e^{-(7/10)\gamma n (1/7)^2/2} = e^{-(1/140)\gamma n} = 2^{-\Omega(n)}$ $\square$

True Path Length

Lemma with $\beta = 1$ yields:
Prob (return to path after n steps) $= 2^{-\Omega(n)}$

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

Prob $(l'(n) \geq l(n)/n) = 1 - 2^{-\Omega(n)}$

We can prove at best lower bound of
$\frac{l'(n)-n+1}{n} > \frac{l(n)}{n^2} - 1 > 2^{n^\delta}$.

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: $\mathsf{H}(p_i, x) \leq \alpha(1)n$

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

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

Consider random path construction starting in p_i.

Similar to Lemma:
Prob (hit x) $= 2^{-\Omega(n)}$

Later steps

$N \neq \emptyset$

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

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_{far}

Clearly, optimal deterministic algorithm avoid N_{far}.

Thus, we are interested in Prob $(A \mid E)$
$= \frac{\text{Prob}(A \cap E)}{\text{Prob}(E)} \leq \frac{\text{Prob}(A)}{\text{Prob}(E)}$.

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

Repeat Case 1. $\square$

Conclusions

. . . 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

Overview of Known Results

Are there just these methods and results for toy examples?

Is there nothing really cool, interesting, and useful?

By these and other methods there are results for evolutionary algorithms for

- "real" cominatorial optimization problems
  - Euler circuits, Ising model, longest common subsequences
  - maximum cliques, maximum matchings, minimum spanning trees
  - shortest paths, sorting, partition
- "advanced" evolutionary algorithms
  - coevolutionary algorithms, memetic algorithms
  - with crossover, different (offspring) population sizes, problem-specific variation operators
- other randomized search heuristics
  - ant colony optimization
  - estimation of distribution algorithms

References for Overview of Known Results

- F. Neumann (2007): Expected runtimes of evolutionary algorithms for the Eulerian cycle problem. *Computers and Operations Research*. To appear.
- B. Doerr, D. Johannsen (2007): Adjacency list matchings - an ideal genotype for cycle covers. *GECCO*. To appear.
- S. Fischer, I. Wegener (2005): The Ising model on the ring: mutation versus recombination. *Theoretical Computer Science* 344:208–225.
- D. Sudholt (2005): Crossover is provably essential for the Ising Model on Trees. In *GECCO*, 1161–1167.
- T. Jansen, D. Weyland (2007): Analysis of evolutionary algorithms for the longest common subsequence problem. In *GECCO*. To appear.
- T. Storch (2006): How randomized search heuristics find maximum cliques in planar graphs. In *GECCO*, 567–574.
- O. Giel, I. Wegener (2003): Evolutionary algorithms and the maximum matching problem. In 20th Annual Symposium on Theoretical Aspects of Computer Science (STACS), 415–426.
- F. Neumann, I. Wegener (2004): Randomized local search, evolutionary algorithms, and the minimum spanning tree problem. In *GECCO*, 713–724.
- J. Scharnow, K. Tinnefeld, I. Wegener (2004): The analysis of evolutionary algorithms on sorting and shortest paths problems. *Journal of Mathematical Modelling and Algorithms* 3:349–366.
- C. Witt (2005): Worst-case and average-case approximations by simple randomized search heuristics. In *22nd Annual Symposium on Theoretical Aspects of Computer Science (STACS)*, 44–56.
- T. Jansen, R. P. Wiegand (2004): The cooperative coevolutionary (1+1) EA. *Evolutionary Computation* 12(4):405–434.
- D. Sudholt (2006): Local search in evolutionary algorithms: the impact of the local search frequency. In *17th International Symposium on Algorithms and Computation (ISAAC)*, 359–368.
- R. Watson, T. Jansen (2007): A building-block royal road where crossover is provably essential. In *GECCO*. To appear.
- B. Doerr, F. Neumann, D. Sudholt, C. Witt (2007): On the Runtime Analysis of the 1-ANT ACO Algorithm. *GECCO*. To appear.
- S. Droste (2005): Not all linear functions are equally difficult for the compact genetic algorithm. In *GECCO*, 679–686.