

SEBASE Summer School 2007
No Free Lunch, Fitness Landscape Analysis
and Complexity Theory

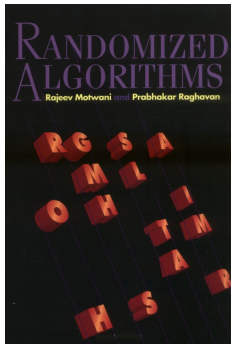
Per Kristian Lehre

July 30, 2007

Introduction and Motivation

- ▶ Randomised Search Heuristics (RSHs)
 - ▶ Vast amounts of empirical research show they are often highly successful.
 - ▶ Poor theoretical understanding.
- ▶ Existing theory
 - ▶ Schema theory (early EC theory)
 - ▶ Theoretical biology
 - ▶ Dynamical systems
 - ▶ Statistical physics
 - ▶ ...

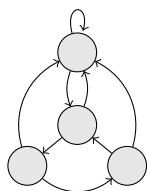
Introduction and Motivation



- ▶ Randomised Search Heuristics (RSHs)
 - ▶ Vast amounts of empirical research show they are often highly successful.
 - ▶ Poor theoretical understanding.
- ▶ Existing theory
 - ▶ Schema theory (early EC theory)
 - ▶ Theoretical biology
 - ▶ Dynamical systems
 - ▶ Statistical physics
 - ▶ ...
- ▶ RSHs from a Computer Science perspective
 - ▶ EAs and other RSHs are **randomised algorithms**
 - ▶ Existing techniques from the field of randomised algorithms can be applied to analyse RSHs.

Running Example - Computing UIOs

Unique input output sequences can be used to test whether an implementation conforms to a FSM model.



Model

Conformance
Testing



Implementation

Some questions relevant to the SBSE researcher

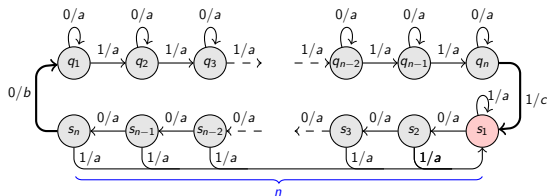
- ▶ Is random search always sufficient to find UIOs?
- ▶ How long will a given search heuristic need to find the UIO?
- ▶ How does the choice of search heuristic influence runtime?

Running Example

Theorem

On the instance class below

- ▶ The probability that random search has found a UIO for s_1 within e^{cn} steps is exponentially small $e^{\Omega(-n)}$.
- ▶ $(1+1)$ EA finds a UIO for s_1 in expected time $O(n^3)$.
- ▶ $(1+1)^*$ EA finds a UIO for s_1 within $n^{n/12}$ iterations with exponentially small probability $e^{-\Omega(n)}$.



Outline

Introduction and Motivation

- Basic Probability Theory
- The Black Box Scenario

The No Free Lunch Theorem

- Statement and proof of theorem
- Another view: Almost NFL

Fitness Landscapes

- Fitness Distance Correlation

Techniques in Computational Complexity of RSHs

- Artificial Fitness Levels
- Chernoff Bounds
- Drift Analysis
- Typical Runs

Summary

Basic Probability Theory

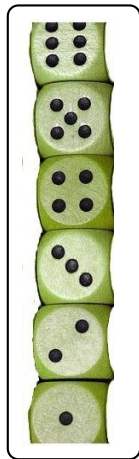
Sample space and Probability

- ▶ Ω : Sample space.
- ▶ \mathcal{F} : Allowable subsets of Ω .
- ▶ **Pr** : $\mathcal{F} \rightarrow \mathbb{R}$ probability function (satisfying probability axioms).



Basic Probability Theory

Sample space and Probability



Ω

- ▶ Ω : Sample space.
- ▶ \mathcal{F} : Allowable subsets of Ω .
- ▶ **Pr** : $\mathcal{F} \rightarrow \mathbb{R}$ probability function (satisfying probability axioms).

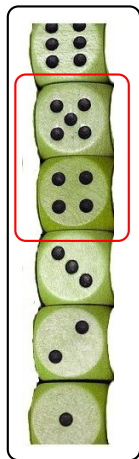
Basic Probability Theory

Sample space and Probability

- ▶ Ω : Sample space.
- ▶ \mathcal{F} : Allowable subsets of Ω .
- ▶ **Pr** : $\mathcal{F} \rightarrow \mathbb{R}$ probability function (satisfying probability axioms).

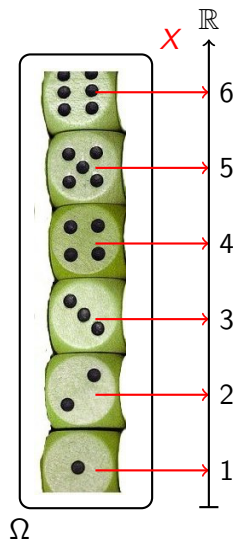
Events

- ▶ $\mathcal{E} \in \mathcal{F}$.



Ω

Basic Probability Theory



Sample space and Probability

- ▶ Ω : Sample space.
- ▶ \mathcal{F} : Allowable subsets of Ω .
- ▶ \Pr : $\mathcal{F} \rightarrow \mathbb{R}$ probability function (satisfying probability axioms).

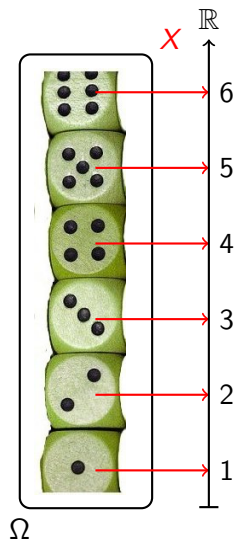
Events

- ▶ $\mathcal{E} \in \mathcal{F}$.

Random Variable

- ▶ $X : \Omega \rightarrow \mathbb{R}$.
- ▶ $X = i \iff \{x \in \Omega \mid X(x) = i\}$.

Basic Probability Theory



Sample space and Probability

- ▶ Ω : Sample space.
- ▶ \mathcal{F} : Allowable subsets of Ω .
- ▶ $\Pr : \mathcal{F} \rightarrow \mathbb{R}$ probability function (satisfying probability axioms).

Events

- ▶ $\mathcal{E} \in \mathcal{F}$.

Random Variable

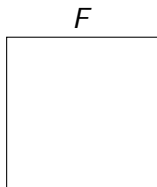
- ▶ $X : \Omega \rightarrow \mathbb{R}$.
- ▶ $X = i \iff \{x \in \Omega \mid X(x) = i\}$.

Expectation

- ▶ $\mathbf{E}[X] := \sum_i i \cdot \Pr[X = i]$.
- ▶ $\mathbf{E}[X \mid \mathcal{E}] := \sum_i i \cdot \Pr[X = i \mid \mathcal{E}]$.

The Black Box Scenario

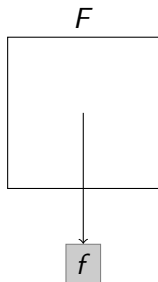
General theoretical framework for evaluating RSHs.



[Droste et al., 2006]

The Black Box Scenario

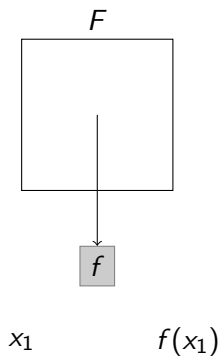
General theoretical framework for evaluating RSHs.



[Droste et al., 2006]

The Black Box Scenario

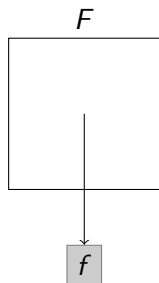
General theoretical framework for evaluating RSHs.



[Droste et al., 2006]

The Black Box Scenario

General theoretical framework for evaluating RSHs.

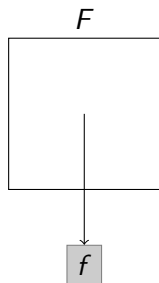


x_1	$f(x_1)$
x_2	$f(x_2)$

[Droste et al., 2006]

The Black Box Scenario

General theoretical framework for evaluating RSHs.

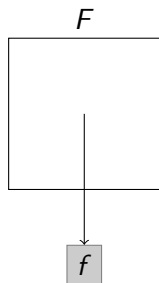


x_1	$f(x_1)$
x_2	$f(x_2)$
x_3	$f(x_3)$

[Droste et al., 2006]

The Black Box Scenario

General theoretical framework for evaluating RSHs.



- ▶ **Worst case** runtime of A on F

$$\max_{f \in F} T(A, f).$$

- ▶ **Average case** runtime of A on F

$$\sum_{f \in F} \Pr[f] \cdot T(A, f).$$

$$T(A, f) \left\{ \begin{array}{ll} x_1 & f(x_1) \\ x_2 & f(x_2) \\ x_3 & f(x_3) \\ \vdots & \vdots \\ x_k & f(x_k) \text{ optimal} \end{array} \right.$$

[Droste et al., 2006]

No Free Lunch

Theorem ([Wolpert and Macready, 1997])

*Let F be the set of all functions $f : S \rightarrow B$,
where S and B are finite sets, B totally ordered.*

*The average case runtime over F
is the same for all search heuristics.*

(Multiple evaluations of same search point counted once.)

Generalized No Free Lunch

Theorem ([Droste et al., 2002b])

*Let F be a set of functions $f : S \rightarrow B$,
where S and B are finite sets, B totally ordered.*

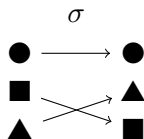
*If F is **closed under permutations**,
then the average case runtime over F
is the same for all search heuristics.*

(Multiple evaluations of same search point counted once.)

Class of functions *closed under permutations* (c.u.p.)

Definition (F c.u.p.)

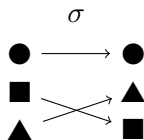
If function f is in class F ,
then for any permutation σ ,
function $f \circ \sigma$ is also in F .



Class of functions *closed under permutations* (c.u.p.)

Definition (F c.u.p.)

If function f is in class F ,
then for any permutation σ ,
function $f \circ \sigma$ is also in F .



Example

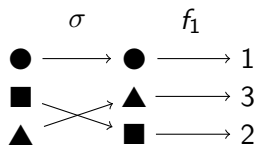
x	$f_1(x)$
●	1
■	2
▲	3

The class $\{f_1\}$ is **not** closed under permutation.

Class of functions *closed under permutations* (c.u.p.)

Definition (F c.u.p.)

If function f is in class F ,
then for any permutation σ ,
function $f \circ \sigma$ is also in F .



Example

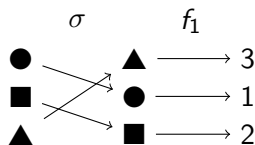
x	$f_1(x)$	$f_2(x)$
●	1	1
■	2	3
▲	3	2

The class $\{f_1, f_2\}$ is **not** closed under permutation.

Class of functions *closed under permutations* (c.u.p.)

Definition (F c.u.p.)

If function f is in class F ,
then for any permutation σ ,
function $f \circ \sigma$ is also in F .



Example

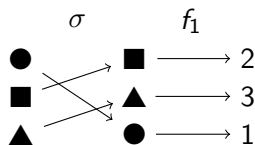
x	$f_1(x)$	$f_2(x)$	$f_3(x)$	$f_4(x)$
●	1	1	2	3
■	2	3	1	1
▲	3	2	3	2

The class $\{f_1, f_2, f_3, f_4\}$ is **not** closed under permutation.

Class of functions *closed under permutations* (c.u.p.)

Definition (F c.u.p.)

If function f is in class F ,
then for any permutation σ ,
function $f \circ \sigma$ is also in F .



Example

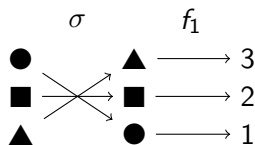
x	$f_1(x)$	$f_2(x)$	$f_3(x)$	$f_4(x)$	$f_5(x)$
●	1	1	2	3	2
■	2	3	1	1	3
▲	3	2	3	2	1

The class $\{f_1, f_2, f_3, f_4, f_5\}$ is **not** closed under permutation.

Class of functions *closed under permutations* (c.u.p.)

Definition (F c.u.p.)

If function f is in class F ,
then for any permutation σ ,
function $f \circ \sigma$ is also in F .



Example

x	$f_1(x)$	$f_2(x)$	$f_3(x)$	$f_4(x)$	$f_5(x)$	$f_6(x)$
●	1	1	2	3	2	3
■	2	3	1	1	3	2
▲	3	2	3	2	1	1

The class $\{f_1, f_2, f_3, f_4, f_5, f_6\}$ is closed under permutation.

Generalized No Free Lunch

Theorem ([Droste et al., 2002b])

*Let F be a set of functions $f : S \rightarrow B$,
where S and B are finite sets, B totally ordered.*

*If F is closed under permutations,
then the average case runtime over F
is the same for all search heuristics.*

NFL Proof Idea - 1

x	$f_1(x)$	$f_2(x)$	$f_3(x)$	$f_4(x)$	$f_5(x)$	$f_6(x)$
●	1	1	2	3	2	3
■	2	3	1	1	3	2
▲	3	2	3	2	1	1

NFL Proof Idea - 1

x	$f_1(x)$	$f_2(x)$	$f_3(x)$	$f_4(x)$	$f_5(x)$	$f_6(x)$
●	1	1	2	3	2	3
■	2	3	1	1	3	2
▲	3	2	3	2	1	1

↓

f_1

- ▶ Opponent selects function f_1 .

NFL Proof Idea - 1

x	$f_1(x)$	$f_2(x)$	$f_3(x)$	$f_4(x)$	$f_5(x)$	$f_6(x)$
●	1	1	2	3	2	3
■	2	3	1	1	3	2
▲	3	2	3	2	1	1

↓

f_1

- ▶ Opponent selects function f_1 .
- ▶ Heuristic knows fitness function is either f_1 , or f_2 , or ... or f_6 .

NFL Proof Idea - 1

x	$f_1(x)$	$f_2(x)$	$f_3(x)$	$f_4(x)$	$f_5(x)$	$f_6(x)$
●	1	1	2	3	2	3
■	2	3	1	1	3	2
▲	3	2	3	2	1	1

↓

f_1

- ▶ Opponent selects function f_1 .
- ▶ Heuristic knows fitness function is either f_1 , or f_2 , or ... or f_6 .
- ▶ Heuristic asks for the function value of ● and gets 1.
 - ▶ Only functions f_1 and f_2 consistent with $f(\bullet) = 1$.

NFL Proof Idea - 1

x	$f_1(x)$	$f_2(x)$	$f_3(x)$	$f_4(x)$	$f_5(x)$	$f_6(x)$
●	1	1	2	3	2	3
■	2	3	1	1	3	2
▲	3	2	3	2	1	1

↓

f_1

- ▶ Opponent selects function f_1 .
- ▶ Heuristic knows fitness function is either f_1 , or f_2 , or ... or f_6 .
- ▶ Heuristic asks for the function value of ● and gets 1.
 - ▶ Only functions f_1 and f_2 consistent with $f(\bullet) = 1$.
- ▶ Problem reduced to function class $F(\bullet, 1)$.

NFL Proof Idea - 2

x	$f_1(x)$	$f_2(x)$	$f_3(x)$	$f_4(x)$	$f_5(x)$	$f_6(x)$
●	1	1	2	3	2	3
■	2	3	1	1	3	2
▲	3	2	3	2	1	1

$F(x, b)$ is subset of functions f consistent with $f(x) = b$.

Two claims

- ▶ $F(x, b)$ is closed under permutations.

NFL Proof Idea - 2

x	$f_1(x)$	$f_2(x)$	$f_3(x)$	$f_4(x)$	$f_5(x)$	$f_6(x)$
●	1	1	2	3	2	3
■	2	3	1	1	3	2
▲	3	2	3	2	1	1

$F(x, b)$ is subset of functions f consistent with $f(x) = b$.

Two claims

- ▶ $F(x, b)$ is closed under permutations.
- ▶ $F(x, b)$ and $F(y, b)$ are isomorphic.

NFL Proof 1/2 - Deterministic search heuristics

- ▶ Proof by induction over the size of search space S .
 - ▶ Step 1: Show that NFL holds when $|S| = 1$.
 - ▶ Step 2: Assume that NFL holds when $|S| = N$.
Show that it also holds when $|S| = N + 1$.
- ▶ Consider any two search heuristics A and B
 - ▶ Assume A chooses x as first search point.
 - ▶ Assume B chooses y as first search point.

NFL Proof 1/2 - Deterministic search heuristics

- ▶ Proof by induction over the size of search space S .
 - ▶ Step 1: Show that NFL holds when $|S| = 1$.
 - ▶ Step 2: Assume that NFL holds when $|S| = N$.
Show that it also holds when $|S| = N + 1$.
- ▶ Consider any two search heuristics A and B
 - ▶ Assume A chooses x as first search point.
 - ▶ Assume B chooses y as first search point.

Step 1: Search space S contains only one search point.

- ▶ A and B find optimum in first iteration.
- ▶ So NFL trivially holds when $|S| = 1$.

NFL Proof 1/2 - Deterministic search heuristics

Step 2: Assume NFL holds when $|S| = N$.

$$\mathbf{E}[F | A] = \mathbf{Pr}[f(x) = b^*] \cdot 1 + \sum_{b \neq b^*} \mathbf{Pr}[f(x) = b] \cdot (1 + \mathbf{E}[F(x, b) | A]).$$

$$\mathbf{E}[F | B] = \mathbf{Pr}[f(y) = b^*] \cdot 1 + \sum_{b \neq b^*} \mathbf{Pr}[f(y) = b] \cdot (1 + \mathbf{E}[F(y, b) | B]).$$

NFL Proof 1/2 - Deterministic search heuristics

Step 2: Assume NFL holds when $|S| = N$.

$$\mathbf{E}[F | A] = \mathbf{Pr}[f(x) = b^*] \cdot 1 + \sum_{b \neq b^*} \mathbf{Pr}[f(x) = b] \cdot (1 + \mathbf{E}[F(x, b) | A]).$$

$$\mathbf{E}[F | B] = \mathbf{Pr}[f(y) = b^*] \cdot 1 + \sum_{b \neq b^*} \mathbf{Pr}[f(y) = b] \cdot (1 + \mathbf{E}[F(y, b) | B]).$$

- ▶ $F(x, b)$ and $F(y, b)$ closed under permutations (by Claim 1.)

NFL Proof 1/2 - Deterministic search heuristics

Step 2: Assume NFL holds when $|S| = N$.

$$\mathbf{E}[F | A] = \mathbf{Pr}[f(x) = b^*] \cdot 1 + \sum_{b \neq b^*} \mathbf{Pr}[f(x) = b] \cdot (1 + \mathbf{E}[F(x, b) | A]).$$

$$\mathbf{E}[F | B] = \mathbf{Pr}[f(y) = b^*] \cdot 1 + \sum_{b \neq b^*} \mathbf{Pr}[f(y) = b] \cdot (1 + \mathbf{E}[F(y, b) | B]).$$

- ▶ $F(x, b)$ and $F(y, b)$ closed under permutations (by Claim 1.)
- ▶ $F(x, b)$ and $F(y, b)$ the same problem. (Claim 2.)

NFL Proof 1/2 - Deterministic search heuristics

Step 2: Assume NFL holds when $|S| = N$.

$$\mathbf{E}[F | A] = \Pr[f(x) = b^*] \cdot 1 + \sum_{b \neq b^*} \Pr[f(x) = b] \cdot (1 + \mathbf{E}[F(x, b) | A]).$$

$$\mathbf{E}[F | B] = \Pr[f(y) = b^*] \cdot 1 + \sum_{b \neq b^*} \Pr[f(y) = b] \cdot (1 + \mathbf{E}[F(y, b) | B]).$$

- ▶ $F(x, b)$ and $F(y, b)$ closed under permutations (by Claim 1.)
- ▶ $F(x, b)$ and $F(y, b)$ the same problem. (Claim 2.)
- ▶ $\mathbf{E}[F(x, b) | A] = \mathbf{E}[F(y, b) | B]$. (Induction hypothesis.)

NFL Proof 1/2 - Deterministic search heuristics

Step 2: Assume NFL holds when $|S| = N$.

$$\mathbf{E}[F | A] = \Pr[f(x) = b^*] \cdot 1 + \sum_{b \neq b^*} \Pr[f(x) = b] \cdot (1 + \mathbf{E}[F(x, b) | A]).$$

$$\mathbf{E}[F | B] = \Pr[f(y) = b^*] \cdot 1 + \sum_{b \neq b^*} \Pr[f(y) = b] \cdot (1 + \mathbf{E}[F(y, b) | B]).$$

- ▶ $F(x, b)$ and $F(y, b)$ closed under permutations (by Claim 1.)
- ▶ $F(x, b)$ and $F(y, b)$ the same problem. (Claim 2.)
- ▶ $\mathbf{E}[F(x, b) | A] = \mathbf{E}[F(y, b) | B]$. (Induction hypothesis.)

$$\implies \mathbf{E}[F | A] = \mathbf{E}[F | B] = \mathbf{E}[F].$$

NFL Proof 2/2 - Randomised search heuristics

Finite number m of deterministic search heuristics H_1, H_2, \dots, H_m .

Randomised search heuristic

- ▶ decisions sometimes based on outcome of a coin toss
- ▶ start by making all coin tosses, then proceed “deterministically” according to outcomes of coin tosses
- ▶ i.e. choose probabilistically a deterministic heuristic H_i

NFL Proof 2/2 - Randomised search heuristics

Finite number m of deterministic search heuristics H_1, H_2, \dots, H_m .

Randomised search heuristic

- ▶ decisions sometimes based on outcome of a coin toss
- ▶ start by making all coin tosses, then proceed “deterministically” according to outcomes of coin tosses
- ▶ i.e. choose probabilistically a deterministic heuristic H_i

$$\begin{aligned}\mathbf{E}[F | A] &= \sum_{i,j} \mathbf{Pr}[A \text{ uses } H_i \cap \text{opponent chooses } f_j] \cdot T[H_i, f_j] \\ &= \sum_i \mathbf{Pr}[A \text{ uses } H_i] \cdot \sum_j \mathbf{Pr}[\text{opponent chooses } f_j] \cdot T[H_i, f_j] \\ &= \sum_i \mathbf{Pr}[A \text{ uses } H_i] \cdot \mathbf{E}[F | H_i] \\ &= \mathbf{E}[F].\end{aligned}$$

How realistic is the NFL scenario?

Assume the class F of fitness functions $f : A \rightarrow B$, where

- ▶ A is the set of bitstrings of length 100.
- ▶ B is the set of integers represented by 32 bits.

How many different fitness functions are there in this class?

$$|F| = |B|^{|A|} = 2^{32 \cdot 2^{100}}.$$

How large are the programs that implement these functions?

How realistic is the NFL scenario?

Assume the class F of fitness functions $f : A \rightarrow B$, where

- ▶ A is the set of bitstrings of length 100.
- ▶ B is the set of integers represented by 32 bits.

How many different fitness functions are there in this class?

$$|F| = |B|^{|A|} = 2^{32 \cdot 2^{100}}.$$

How large are the programs that implement these functions?

By a simple counting argument, the length of at least half of the fitness functions must have shortest programs of length at least

$$\log |F| - 1 \text{ bits} \geq 32 \cdot 2^{100} - 1 \text{ bits} \geq 10^{20} \text{ terabytes.}$$

⇒ Conditions in NFL theorem rarely hold in practice.

Almost No Free Lunch

Theorem ([Droste et al., 2002b])

Let H be a randomised search heuristic and

$$f : \{0, 1\}^n \rightarrow \{0, 1, \dots, N - 1\}.$$

Then there exist at least $N^{2^{n/3}-1}$ functions

$$f^* : \{0, 1\}^n \rightarrow \{0, 1, \dots, N\}$$

which agree with f on all but at most $2^{n/3}$ inputs such that

- ▶ H does find the optimum of f^* within $2^{n/3}$ steps with a probability bounded above by $2^{-n/3}$.
- ▶ Exponentially many of these functions have the additional property that their evaluation time, circuit size representation, and Kolmogorov complexity is only by an additive term of $O(n)$ larger than the corresponding complexity of f .

NFL - Conclusion

- ▶ No single best search heuristic on **all** problems.
- ▶ In *design* and *analysis* of search heuristics, it is necessary to consider function classes that are not *c.u.p.*
 - ▶ assume a certain type of “fitness landscape”
 - ▶ assume a certain type of “structural” property.
- ▶ Runtime differences still possible on subclasses.
 - ▶ F *c.u.p.*, $F_1 \cup F_2 = F$ and $F_1 \cap F_2 = \emptyset$.
 - ▶ A outperforms B on $F_1 \implies B$ outperforms A on F_2 .
- ▶ NFL conditions will not occur in practice.
- ▶ Almost NFL in restricted scenario
 - ▶ Small modifications to an easy function can make it hard.

Fitness Landscapes

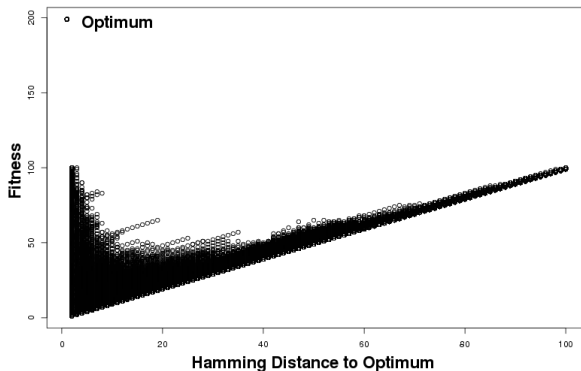


- ▶ Landscapes as a metaphor for explaining dynamics of evolutionary algorithms (or other search heuristics).

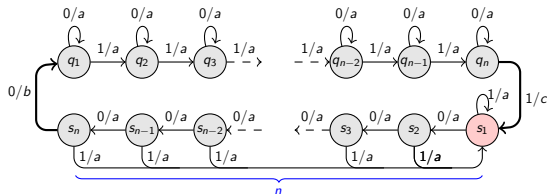
Fitness Landscapes - Fitness Distance Correlation

- ▶ Intuition: In “nice” fitness landscapes, distance to optimum correlates with fitness.
- ▶ FDC plott: Distance to optimum vs. fitness value [Jones and Forrest, 1995].

Example (Our FSM instance)



Fitness Distance Correlation Plot - Example

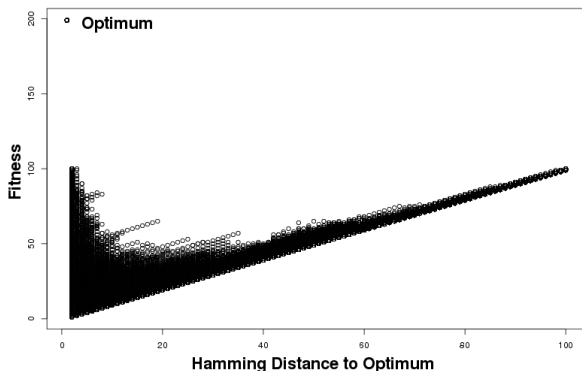


$$\text{ONEMAX}(x) := x_1 + x_2 + \dots + x_n.$$

$$\text{LEADINGZEROS}(x) := \sum_{i=1}^n \prod_{j=1}^i (1 - x_j).$$

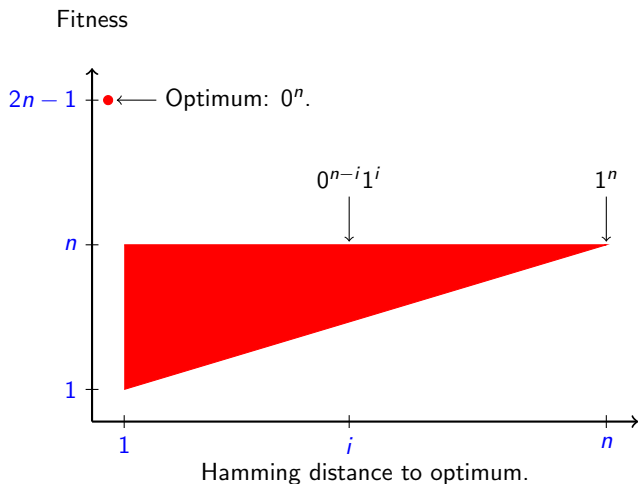
$$f_R(x) = \begin{cases} 2n - 1 & \text{if } x = 0^n, \quad \text{and} \\ \text{LEADINGZEROS}(x) + \text{ONEMAX}(x) & \text{otherwise.} \end{cases}$$

Fitness Distance Correlation Plot - Example



$$f_R(x) = \begin{cases} 2n - 1 & \text{if } x = 0^n, \quad \text{and} \\ \text{LEADINGZEROES}(x) + \text{ONEMAX}(x) & \text{otherwise.} \end{cases}$$

Fitness Distance Correlation Plot - Example



$$f_R(x) = \begin{cases} 2n - 1 & \text{if } x = 0^n, \quad \text{and} \\ \text{LEADINGZEROES}(x) + \text{ONEMAX}(x) & \text{otherwise.} \end{cases}$$

Part II

Techniques for analysing Time Complexity of RSHs.

Goals and Methods

Research Objectives

- ▶ Probability of reaching optimum in the limit.
 - ▶ Mostly of interest to theoreticians.
 - ▶ Trivial in discrete search spaces when using elitism and a sensible mutation operator.
- ▶ Time to reach the optimum.

General approach

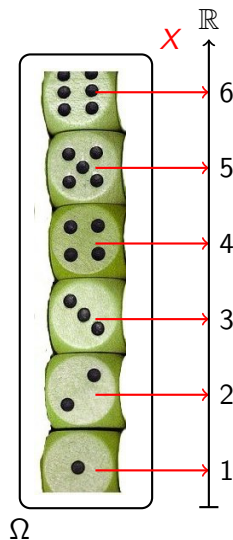
- ▶ Start by developing mathematical techniques for
 1. analysing simple algorithms,
 2. on simple problems.
- ▶ Proceed by using and extending these techniques for
 1. more complicated algorithms, and
 2. on combinatorial and real-world problems.

State of the Art in Computational Complexity of RSHs

ONEMAX	(1+1) EA	$O(n \log n)$	[Mühlenbein, 1992]
	1-ANT, $\rho = \Omega(n^{\epsilon-1})$	$O(n^2)$ w.h.p.	[Neumann and Witt, 2006]
Linear Functions	(1+1) EA	$\Theta(n \log n)$	[Droste et al., 2002a] and [He and Yao, 2003]
	cGA	$\Theta(n^{2+\epsilon})$, $\epsilon > 0$ const.	[Droste, 2006]
Max. Matching	(1+1) EA	$e^{\Omega(n)}$, PRAS	[Giel and Wegener, 2003]
Sorting	(1+1) EA	$\Theta(n^2 \log n)$	[Scharnow et al., 2002]
Shortest Path	(1+1) EA	$e^{\Omega(n)}$	[Scharnow et al., 2002]
	MO (1+1) EA	$O(n^3)$	[Scharnow et al., 2002]
Min. Spanning Tree	(1+1) EA	$\Theta(m^2(\log n + \log w_{max}))$	[Neumann and Wegener, 2007]
	(1+ λ) EA	$O(n(\log n + \log w_{max}))$	[Neumann and Wegener, 2007]
Max. Clique (rand. plan.)	(1+1) EA	$\Theta(n^5)$	[Storch, 2006]
	(16n+1) RLS	$\Theta(n^{5/3})$	[Storch, 2006]
Eulerian Cycle	(1+1) EA	$\Theta(m^2 \log m)$	[Doerr et al., 2007]
Partition	(1+1) EA	PRAS, avg.	[Witt, 2005]
Vertex Cover	(1+1) EA	$e^{\Omega(n)}$, arb. bad approx.	[Friedrich et al., 2007] and [Oliveto et al., 2007a]
	SEMO	Pol. $O(\log n)$ -approx.	[Friedrich et al., 2007]
UIO / FSM conf. test.	(1+1) EA	$e^{\Omega(n)}$	[Lehre and Yao, 2007]

See recent survey [Oliveto et al., 2007b].

Basic Probability Theory



Sample space and Probability

- ▶ Ω : Sample space.
- ▶ \mathcal{F} : Allowable subsets of Ω .
- ▶ $\Pr : \mathcal{F} \rightarrow \mathbb{R}$ probability function (satisfying probability axioms).

Events

- ▶ $\mathcal{E} \in \mathcal{F}$.

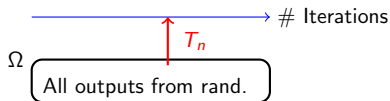
Random Variable

- ▶ $X : \Omega \rightarrow \mathbb{R}$.
- ▶ $X = i \iff \{x \in \Omega \mid X(x) = i\}$.

Expectation

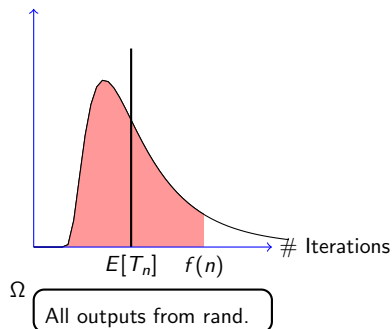
- ▶ $\mathbf{E}[X] := \sum_i i \cdot \Pr[X = i]$.
- ▶ $\mathbf{E}[X \mid \mathcal{E}] := \sum_i i \cdot \Pr[X = i \mid \mathcal{E}]$.

Runtime Analysis of Randomised Search Heuristics



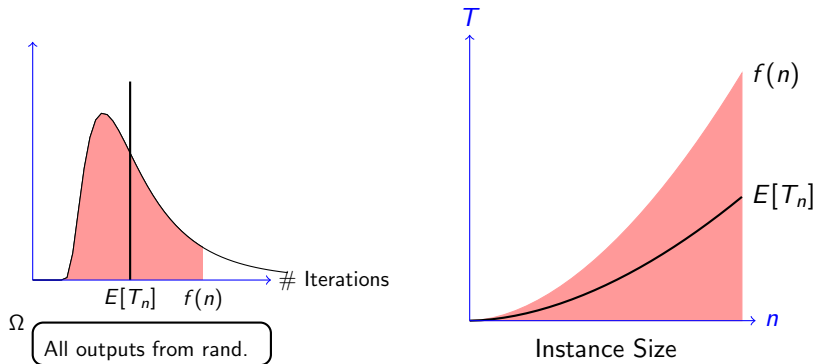
- ▶ The **runtime** T_n of a randomised search heuristics is a **random variable** over the possible outcomes of the random number generator.

Runtime Analysis of Randomised Search Heuristics



- ▶ The **runtime** T_n of a randomised search heuristics is a **random variable** over the possible outcomes of the random number generator.
- ▶ **Runtime analysis**: Find properties of the distribution of T_n ,
 - ▶ Expected runtime $\mathbf{E}[T_n]$.
 - ▶ Success probability $\mathbf{Pr}[T_n \leq f(n)]$.

Runtime Analysis of Randomised Search Heuristics



- ▶ The **runtime** T_n of a randomised search heuristics is a **random variable** over the possible outcomes of the random number generator.
- ▶ **Runtime analysis**: Find properties of the distribution of T_n ,
 - ▶ Expected runtime $\mathbf{E}[T_n]$.
 - ▶ Success probability $\mathbf{Pr}[T_n \leq f(n)]$.
- ▶ as a function of the problem instance size n .

Basic probability theory - Geometric Distribution

In each iteration, an event happens with probability p .

What is the number of iterations until the event happens?

Geometric Distribution

A random variable X is *geometrically* distributed with param. p if

$$\Pr[X = n] = (1 - p)^{n-1} \cdot p.$$

The expectation of X is $\mathbf{E}[X] = 1/p$.

Basic probability theory - Geometric Distribution

In each iteration, an event happens with probability p .

What is the number of iterations until the event happens?

Geometric Distribution

A random variable X is *geometrically* distributed with param. p if

$$\Pr[X = n] = (1 - p)^{n-1} \cdot p.$$

The expectation of X is $\mathbf{E}[X] = 1/p$.

Example (Runtime of Random Search in UIO problem)

- ▶ Only one global optimum among bitstrings of length n .
- ▶ Probability of finding optimum in any iteration is $\left(\frac{1}{2}\right)^n$.
- ▶ Expected time to find optimum is $\mathbf{E}[T] = 2^n$.
- ▶ Success probability within e^{cn} steps is

$$\Pr[T_n \leq e^{cn}] \leq e^{cn} \cdot \left(\frac{1}{2}\right)^n = e^{-\Omega(n)}, \text{ for small } c.$$

Outline of techniques in Part II

- ▶ Artificial Fitness Levels
- ▶ Chernoff Bounds
- ▶ Drift Analysis
- ▶ Typical Runs

Simple Randomised Search Heuristics (RSHs)

(1+1) EA

Choose x uniformly from $\{0, 1\}^n$.

Repeat

$x' := x$.

Flip each bit of x' with probability $1/n$.

If $f(x') \geq f(x)$,
then $x := x'$.

Simple Randomised Search Heuristics (RSHs)

$(1+1)^*$ EA

Choose x uniformly from $\{0, 1\}^n$.

Repeat

$x' := x$.

Flip each bit of x' with probability $1/n$.

If $f(x') > f(x)$,
then $x := x'$.

Simple Randomised Search Heuristics (RSHs)

RLS

Choose x uniformly from $\{0, 1\}^n$.

Repeat

$x' := x$.

Flip one bit position chosen uniformly at random.

If $f(x') \geq f(x)$,
then $x := x'$.

Simple Randomised Search Heuristics (RSHs)

RLS

Choose x uniformly from $\{0, 1\}^n$.

Repeat

$x' := x$.

Flip one bit position chosen uniformly at random.

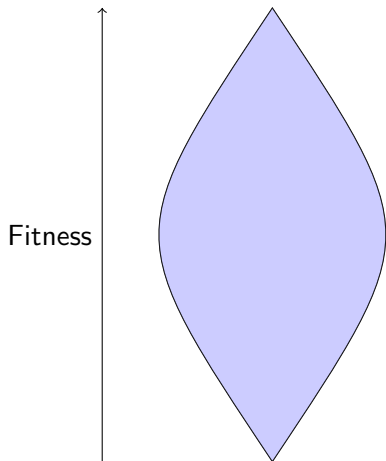
If $f(x') \geq f(x)$,
then $x := x'$.

Why analyse such simple RSHs? What about GAs?

- ▶ Runtime analysis difficult. Necessary to understand a simple EA as a first step.
- ▶ Needed to understand when populations are necessary.
- ▶ More complex EAs analysed once (1+1) EA understood.

Artificial Fitness Levels - Upper bounds

Search space partitioned into m subsets A_1, A_2, \dots, A_m , with increasing fitness, *i.e.* $f(A_i) < f(A_j)$ for all $i < j$, and $f(A_m)$ optimal.



p_i : Probability of jumping from A_i to any A_j , $i < j$.

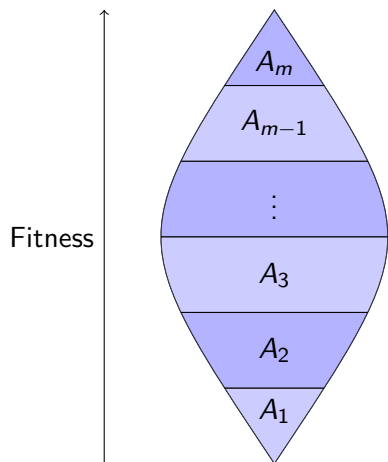
T_i : Time to jump from A_i to any A_j , $i < j$.

Expected runtime

$$\begin{aligned}\mathbf{E}[T] &\leq \mathbf{E}[T_1 + T_2 + \dots + T_m] \\ &= \mathbf{E}[T_1] + \mathbf{E}[T_2] + \dots + \mathbf{E}[T_m] \\ &= 1/p_1 + 1/p_2 + \dots + 1/p_m.\end{aligned}$$

Artificial Fitness Levels - Upper bounds

Search space partitioned into m subsets A_1, A_2, \dots, A_m , with increasing fitness, i.e. $f(A_i) < f(A_j)$ for all $i < j$, and $f(A_m)$ optimal.



p_i : Probability of jumping from A_i to any $A_j, i < j$.

T_i : Time to jump from A_i to any $A_j, i < j$.

Expected runtime

$$\begin{aligned} \mathbf{E}[T] &\leq \mathbf{E}[T_1 + T_2 + \dots + T_m] \\ &= \mathbf{E}[T_1] + \mathbf{E}[T_2] + \dots + \mathbf{E}[T_m] \\ &= 1/p_1 + 1/p_2 + \dots + 1/p_m. \end{aligned}$$

Artificial fitness levels - Example

Our example function

$$f_R(x) = \begin{cases} 2n - 1 & \text{if } x = 0^n, \quad \text{and} \\ \text{LEADINGZEROS}(x) + \text{ONEMAX}(x) & \text{otherwise,} \end{cases}$$

$$\text{ONEMAX}(x) := x_1 + x_2 + \cdots + x_n.$$

Artificial fitness levels - Example

Partitioning of search space in fitness levels

$$\text{ONEMAX}(x) := x_1 + x_2 + \dots + x_n.$$

A_i : all bitstrings with i 1-bits.

p_i : probability of increasing number of 1-bits
(at least prob. of flipping one 0-bit, and no other bits)

$$p_i \geq \underbrace{(n-i)}_{\text{\#0-bits}} \cdot \frac{1}{n} \cdot \underbrace{\left(1 - \frac{1}{n}\right)^{n-1}}_{\geq 1/e} \geq \frac{n-i}{en}.$$

Artificial fitness levels - Example

Partitioning of search space in fitness levels

$$\text{ONEMAX}(x) := x_1 + x_2 + \dots + x_n.$$

A_i : all bitstrings with i 1-bits.

p_i : probability of increasing number of 1-bits
(at least prob. of flipping one 0-bit, and no other bits)

$$p_i \geq \underbrace{(n-i)}_{\text{\#0-bits}} \cdot \frac{1}{n} \cdot \underbrace{\left(1 - \frac{1}{n}\right)^{n-1}}_{\geq 1/e} \geq \frac{n-i}{en}.$$

Expected runtime

$$\mathbf{E}[T_{\text{ONEMAX}}] \leq \sum_{i=0}^{n-1} \frac{en}{n-i} = en \sum_{i=1}^n \frac{1}{i} = O(n \ln n).$$

Behaviour of randomised algorithms not unpredictable

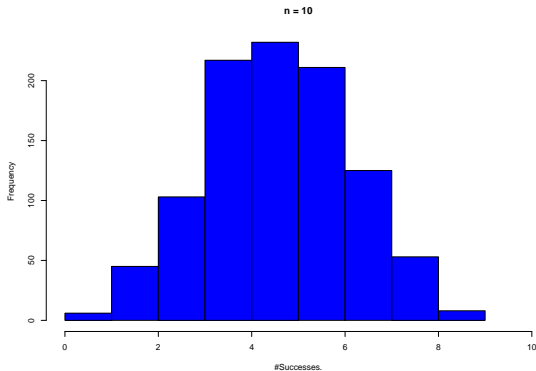
Example

Initial search point in $(1+1)$ EA chosen uniformly at random among bitstrings of length n . How many 1-bits in initial search point of $(1+1)$ EA?

Behaviour of randomised algorithms not unpredictable

Example

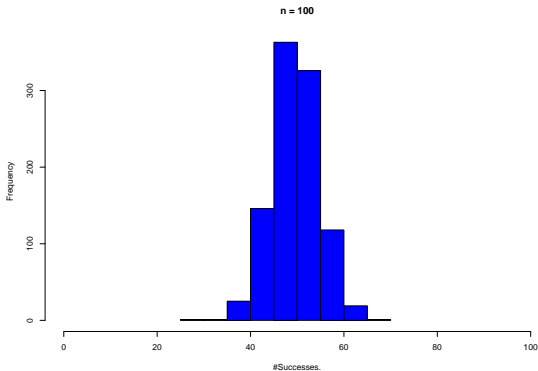
Initial search point in (1+1) EA chosen uniformly at random among bitstrings of length n . How many 1-bits in initial search point of (1+1) EA?



Behaviour of randomised algorithms not unpredictable

Example

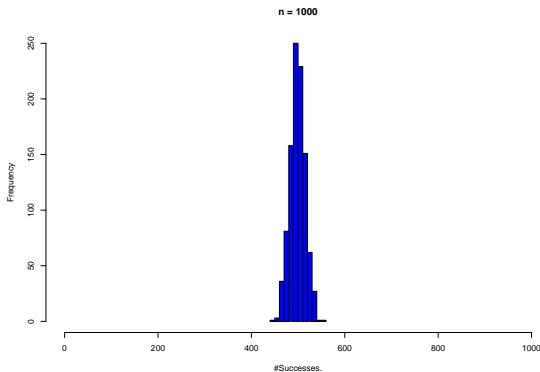
Initial search point in (1+1) EA chosen uniformly at random among bitstrings of length n . How many 1-bits in initial search point of (1+1) EA?



Behaviour of randomised algorithms not unpredictable

Example

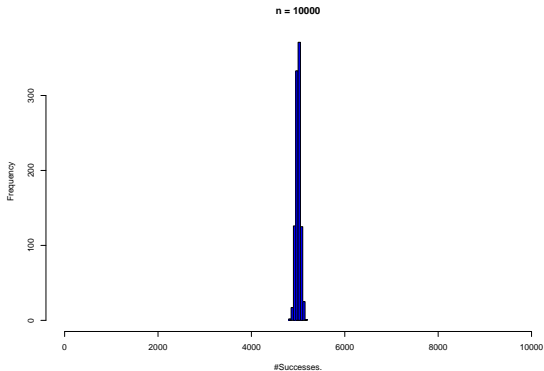
Initial search point in (1+1) EA chosen uniformly at random among bitstrings of length n . How many 1-bits in initial search point of (1+1) EA?



Behaviour of randomised algorithms not unpredictable

Example

Initial search point in (1+1) EA chosen uniformly at random among bitstrings of length n . How many 1-bits in initial search point of (1+1) EA?

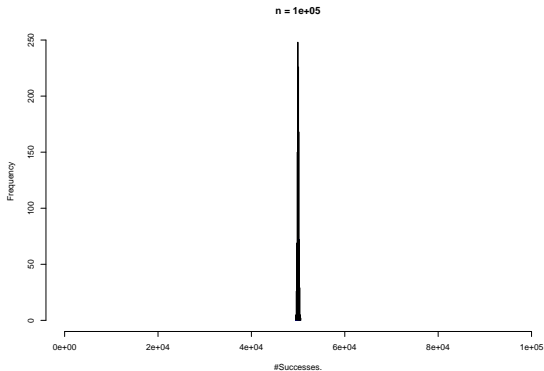


Behaviour of randomised algorithms not unpredictable

Example

Initial search point in (1+1) EA chosen uniformly at random among bitstrings of length n . How many 1-bits in initial search point of (1+1) EA?

- For high values of n , very close to $n/2$.



Chernoff Bounds

Theorem

Let X_1, X_2, \dots, X_n be n independent Poisson trials¹ with

$$\Pr[X_i = 1] = p_i \quad \text{and} \quad \Pr[X_i = 0] = (1 - p_i).$$

Define the random variable X as

$$X := X_1 + X_2 + \dots + X_n.$$

Then for any ε , $0 < \varepsilon \leq 1$,

$$\Pr[X \geq (1 + \varepsilon) \cdot \mathbf{E}[X]] \leq \exp(-\mathbf{E}[X] \cdot \varepsilon^2/3), \quad \text{and}$$

$$\Pr[X \leq (1 - \varepsilon) \cdot \mathbf{E}[X]] \leq \exp(-\mathbf{E}[X] \cdot \varepsilon^2/2).$$

[Motwani and Raghavan, 1995]

¹Not to be confused with Poisson distribution.

Example of use

Example (Lower bound of $(1+1)$ EA on ONEMAX.)

$(1+1)$ EA chooses initial search point of length n at random.

Goal 1: Whp, the initial search point will not be too close to the optimum (e.g. has less than $2n/3$ 1-bits).

Goal 2: Wcp, there is at least one 0-bit left after $(n - 1) \ln n$ iterations.

Example of use

Example (Lower bound of (1+1) EA on ONEMAX.)

(1+1) EA chooses initial search point of length n at random.

Goal 1: Whp, the initial search point will not be too close to the optimum (e.g. has less than $2n/3$ 1-bits).

Goal 2: Wcp, there is at least one 0-bit left after $(n-1) \ln n$ iterations.

1) Let $X_i := 1$ iff a 1-bit in position i .

With $p_i = 1/2$, we expect $\mathbf{E}[X] = n/2$ number of 1-bits.

By setting $\varepsilon = 1/3$, the Chernoff bound now gives us

$$\Pr \left[X \geq \left(1 + \frac{1}{3}\right) \cdot \frac{n}{2} \right] \leq \exp \left(-\frac{n}{2} \cdot \left(\frac{1}{3}\right)^2 / 3 \right) = e^{-\Omega(n)}.$$

Example of use

Example (Lower bound of (1+1) EA on ONEMAX.)

(1+1) EA chooses initial search point of length n at random.

Goal 1: Whp, the initial search point will not be too close to the optimum (e.g. has less than $2n/3$ 1-bits).

Goal 2: Wcp, there is at least one 0-bit left after $(n-1) \ln n$ iterations.

1) Let $X_i := 1$ iff a 1-bit in position i .

With $p_i = 1/2$, we expect $\mathbf{E}[X] = n/2$ number of 1-bits.

By setting $\varepsilon = 1/3$, the Chernoff bound now gives us

$$\Pr \left[X \geq \left(1 + \frac{1}{3}\right) \cdot \frac{n}{2} \right] \leq \exp \left(-\frac{n}{2} \cdot \left(\frac{1}{3}\right)^2 / 3 \right) = e^{-\Omega(n)}.$$

2) At least one 0-bit not mutated in $(n-1) \ln n$ iterations.

$$1 - \left(1 - \left(1 - \frac{1}{n} \right)^{(n-1) \ln n} \right)^{n/3} \geq 1 - \left(1 - \frac{1}{n} \right)^{n/3} \geq 1 - 1/e^{1/3}.$$

Chernoff Bounds - Lower bound for LEADINGZEROS

$$\text{LEADINGZEROS}(x) := \sum_{i=1}^n \prod_{j=1}^i (1 - x_j).$$

$x = \overbrace{000000000000000000}^{\text{Leading 0-bits.}} \mathbf{1} \overbrace{*****}^{\text{Random bitstring.}}$
First 1-bit.

- ▶ By Chernoff, whp., more than $n/3$ 1-bits.
- ▶ An iteration of RLS *successful* if first 1-bit is flipped. $p = 1/n$.
- ▶ Expected number of successful steps in $n^2/4$ iterations is

$$\mathbf{E}[X] = 1/n \cdot n^2/4 = n/4.$$

Chernoff Bounds - Lower bound for LEADINGZEROS

$$\text{LEADINGZEROS}(x) := \sum_{i=1}^n \prod_{j=1}^i (1 - x_j).$$

$x = \overbrace{000000000000000000}^{\text{Leading 0-bits.}} \underbrace{1 \text{ * * * * * * * * * * * * * * * * }_{\text{Random bitstring.}}$

First 1-bit.

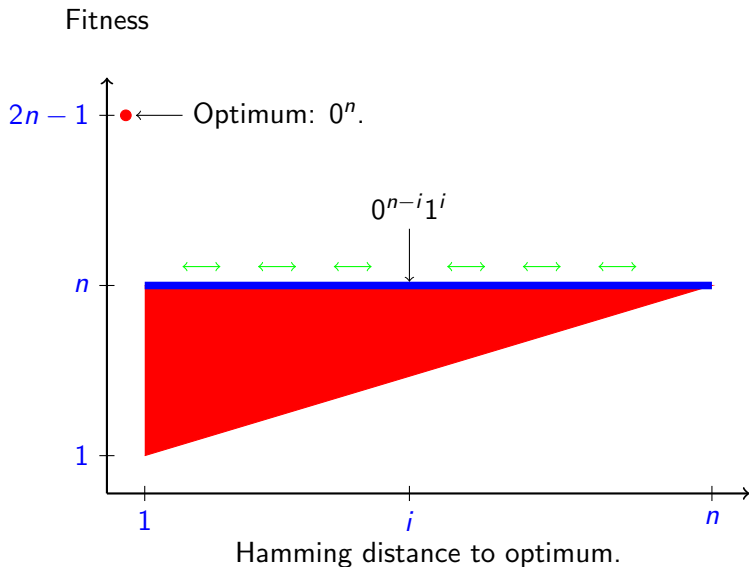
- ▶ By Chernoff, whp., more than $n/3 = (1 + 1/3) \cdot n/4$ 1-bits.
- ▶ An iteration of RLS *successful* if first 1-bit is flipped. $p = 1/n$.
- ▶ Expected number of successful steps in $n^2/4$ iterations is

$$\mathbf{E}[X] = 1/n \cdot n^2/4 = n/4.$$

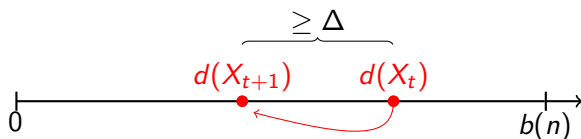
- ▶ Chernoff bounds the success probability within $n^2/4$ steps by $\Pr[X \geq (1 + 1/3) \cdot n/4] \leq \exp(-n/4 \cdot (1/3)^2/3) = e^{-\Omega(n)}$.
(Same result for (1+1) EA using a similar argument.)

Potential Functions and Drift Analysis

On the plateau, the fitness value gives no information about distance to optimum.



Drift Analysis - Upper bounds



Theorem

Let X_1, X_2, \dots be a stochastic process over S , and $d : S \rightarrow \mathbb{R}_0^+$ a distance function on S .

Define T to be the first time t such that $d(X_t) = 0$.

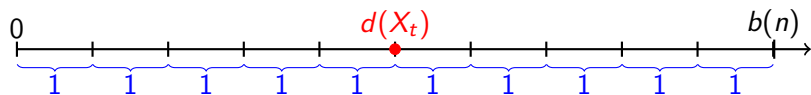
If there exists a constant $\Delta > 0$ such that,

1. $\forall t \geq 0 : \Pr [d(X_t) < b(n)] = 1$, and
2. $\forall t \geq 0 : \mathbf{E} [d(X_t) - d(X_{t+1}) \mid T > t] \geq \Delta$,

then

$$\mathbf{E} [T] \leq b(n)/\Delta.$$

Drift analysis on plateau (RLS for simplicity)

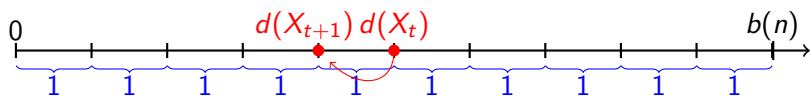


$$x = 0000011111$$

Expected drift

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

Drift analysis on plateau (RLS for simplicity)



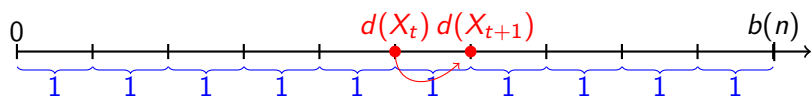
$$x = 0000011111$$

$$x' = 00000\mathbf{0}11111$$

Expected drift

$$\Delta = \mathbf{E} [d(X_t) - d(X_{t+1})] = \mathbf{1/n} \cdot \mathbf{1+}$$

Drift analysis on plateau (RLS for simplicity)



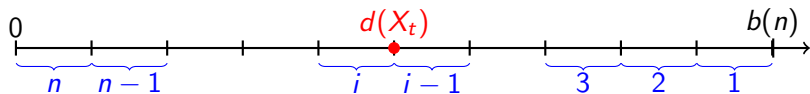
$$x = 0000011111$$

$$x' = 0000\mathbf{1}11111$$

Expected drift

$$\Delta = \mathbf{E} [d(X_t) - d(X_{t+1})] = 1/n \cdot 1 + \mathbf{1/n} \cdot (-1) = 0.$$

Drift analysis on plateau - Refined distance measure



Expected drift

$$\Delta_{\text{RLS}} = \mathbf{E} [d(X_t) - d(X_{t+1})] = (1/n) \cdot i - (1/n) \cdot (i - 1) = (1/n).$$

Maximal distance

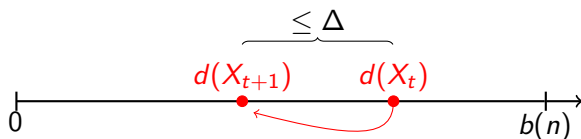
$$b(n) = \sum_{i=1}^n i = O(n^2).$$

Upper bound on Expected runtime

$$\mathbf{E} [T_{\text{RLS}}] \leq b(n)/\Delta_{\text{RLS}} = O(n^3).$$

(Same upper bound on plateau for (1+1) EA.)

Drift Analysis - Lower bounds



Theorem

Let X_1, X_2, \dots be a stochastic process over S , and $d : S \rightarrow \mathbb{R}_0^+$ a distance function on S .

Define T to be the first time t such that $d(X_t) = 0$.

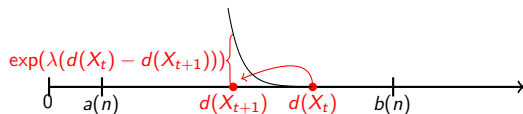
If there exists a constant $\Delta > 0$ such that,

1. $\Pr [d(X_0) \geq b(n)] = 1$, and
2. $\forall t \geq 0 : \mathbf{E} [d(X_t) - d(X_{t+1}) \mid T > t] \leq \Delta$,

then

$$\mathbf{E} [T] \geq b(n)/\Delta.$$

Drift Analysis - Exponential lower bounds



Theorem

Let $a(n)$ and $b(n)$ be two functions, with $0 < a(n) < b(n)$. If

- 1 $\Pr[d(X_0) \geq b(n)] = 1$, and
- 2 $b(n) - a(n) = \Omega(n)$,

and there exists const. $\lambda > 0$, $D \geq 1$ and a poly. $p(n) > 0$ s.t. $\forall t > 0$,

- 3 $\mathbf{E}[e^{-\lambda(d(X_{t+1}) - d(X_t))} \mid X_t, a(n) < d(X_t) < b(n)] \leq 1 - 1/p(n)$, and
- 4 $\mathbf{E}[e^{-\lambda(d(X_{t+1}) - b(n))} \mid X_t, b(n) \leq d(X_t)] \leq D$,

then for any time bounds $B > 0$,

$$\Pr[T \leq B] \leq e^{\lambda(a(n) - b(n))} \cdot B \cdot D \cdot p(n).$$

Typical Runs - Motivation

On many fitness functions, the EA behaves as follows

- ▶ Process typically goes through several distinctive phases.
- ▶ In some rare cases, the process can enter certain states which complicates the analysis tremendously.

We would like to

- ▶ Avoid detailed analysis of the difficult states.
- ▶ Analyse each phase seperately with the most appropriate technique.

Typical Runs - Success Probability

Phase 1: From the initial population, the EA will satisfy condition C_1 in at most T_1 steps with probability p_1 .

Phase 2: From condition C_1 , the EA will satisfy condition C_2 in at most T_2 steps with probability p_2 .

...

Phase k : From condition C_{k-1} , the EA will reach global optimum in at most T_k steps with probability p_k .

Success probability (lower bounds)

$$\Pr \left[T \leq \sum_{i=1}^k T_i \right] \geq 1 - \sum_{i=1}^k (1 - p_i).$$

[Jansen and Neumann, 2007]

Typical Runs - Expected Time

Given that some failure event F does not happen, then

Phase 1: From the initial population, the EA will satisfy condition C_1 in expected time $\mathbf{E} [T_1 | \bar{F}]$.

Phase 2: From condition C_1 , the EA will satisfy condition C_2 in expected time $\mathbf{E} [T_2 | \bar{F}]$.

...

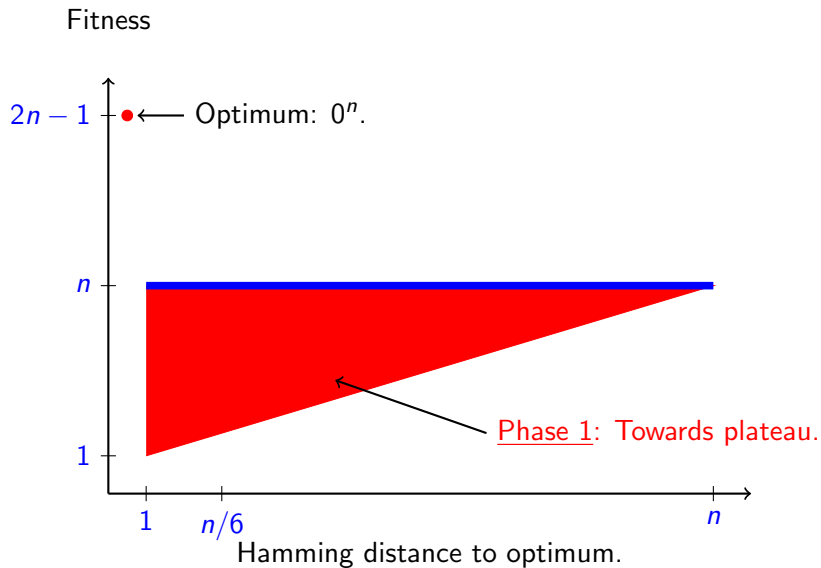
Phase k : From condition C_{k-1} , the EA will reach global optimum in expected time $\mathbf{E} [T_k | \bar{F}]$.

Expected runtime

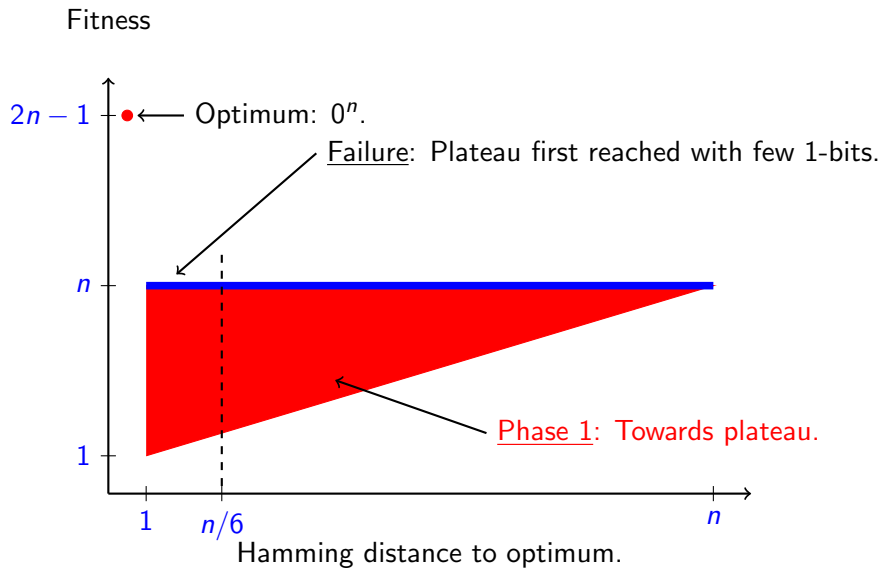
$$\mathbf{E} [T] = (1 - \mathbf{Pr} [F]) \cdot \sum_{i=1}^k \mathbf{E} [T_i | \bar{F}] + \mathbf{Pr} [F] \cdot \mathbf{E} [T | F].$$

If $\mathbf{Pr} [F]$ is small (ie exponentially small), then a very rough (polynomial) estimate of $\mathbf{E} [T | F]$ often suffices.

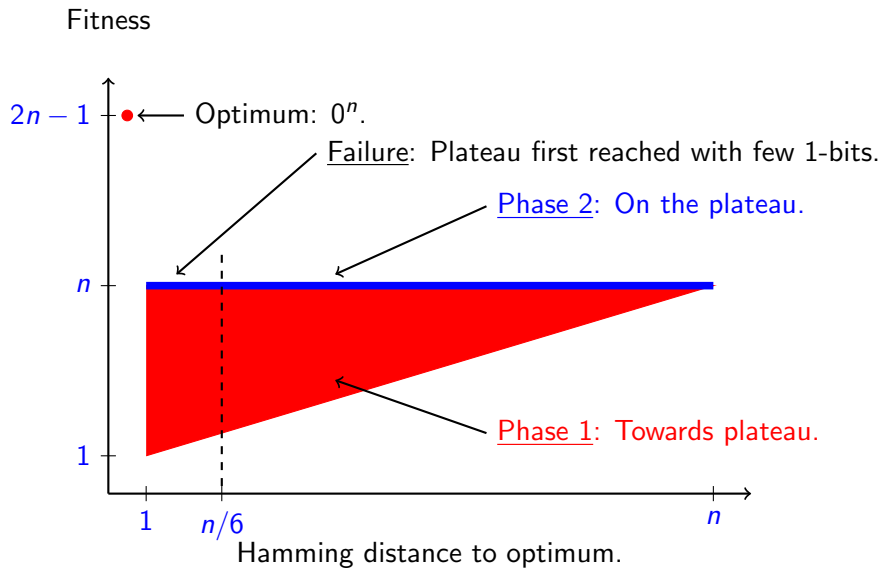
Tying it all together using Typical Runs



Tying it all together using Typical Runs



Tying it all together using Typical Runs



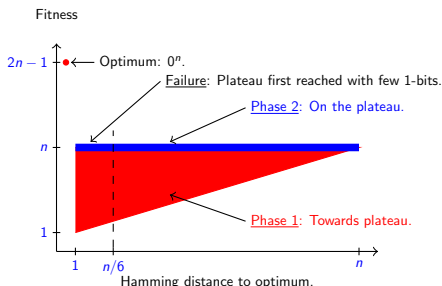
Tying it all together using typical runs - (1+1) EA

Phase 1: From any initial search point, (1+1) EA will reach the plateau in expected time $O(n \log n)$.

Phase 2: Starting anywhere on the plateau, (1+1) EA will reach the global optimum in $O(n^3)$ iterations.
(Not necessary to define failure event.)

Theorem

(1+1) EA finds the global optimum in expected time $O(n^3)$.



Tying it all together using Typical Runs - $(1+1)^*$ EA

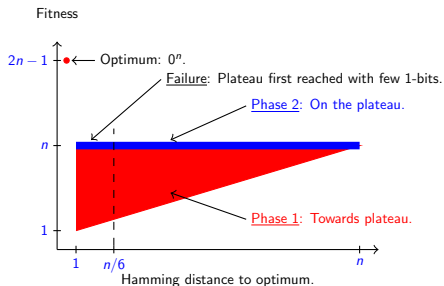
Phase 1: From the initial search point, $(1+1)^*$ EA will reach the plateau with less than $5n/6$ leading 0-bits in at most $n^2/12$ steps with probability $1 - e^{-\Omega(n)}$.

Phase 2: Starting with less than $5n/6$ leading 0-bits on the plateau, the $(1+1)^*$ EA will need at least $n^{n/12}$ iterations to reach the global optimum with probability $1 - e^{-\Omega(n)}$.

Failure: Plateau first reached with more than $5n/6$ leading 0-bits. (Estimation of failure probability not shown here.)

Theorem

$(1+1)^*$ EA finds the global optimum within $n^{n/12}$ iterations with exponentially small probability $e^{-\Omega(n)}$.



Summary of Techniques

Techniques presented today

- ▶ Artificial Fitness Levels
- ▶ Chernoff Bounds
- ▶ Drift Analysis
- ▶ Typical Runs

Summary of Techniques

Techniques presented today

- ▶ Artificial Fitness Levels
- ▶ Chernoff Bounds
- ▶ Drift Analysis
- ▶ Typical Runs

Other techniques

- ▶ Exact Markov Chain models
- ▶ Coupon Collector's problem
- ▶ Gambler's ruin problem (simpler variant of drift)
- ▶ Electrical resistive networks (for random walks)
- ▶ Family trees (for populations w/o crossover).

Summary

- ▶ Black Box Scenario
- ▶ (Almost) No Free Lunch
- ▶ Fitness Landscapes
 - ▶ FDC plot not robust measure of difficulty.
- ▶ Computational Complexity of RSHs
 - ▶ Expected Runtime and Success Probability
 - ▶ Overview of runtime results on combinatorial problems.
 - ▶ Techniques in runtime analysis of RSHs

Questions?

Thank you for your attention!

References I



Doerr, B., Klein, C., and Storch, T. (2007).
Faster evolutionary algorithms by superior graph representation.
In Proceedings of the 1st IEEE Symposium on Foundations of Computational Intelligence (FOCI'2007), pages 245–250.



Droste, S. (2006).
A rigorous analysis of the compact genetic algorithm for linear functions.
Natural Computing, 5(3):257–283.
[10.1007/s11047-006-9001-0](https://doi.org/10.1007/s11047-006-9001-0).



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



Droste, S., Jansen, T., and Wegener, I. (2002b).
Optimization with randomized search heuristics—the (a) nfl theorem, realistic scenarios, and difficult functions.
Theoretical Computer Science, 287(1):131–144.



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

References II



Friedrich, T., Hebbinghaus, N., Neumann, F., He, J., and Witt, C. (2007).
Approximating covering problems by randomized search heuristics using
multi-objective models.
In *GECCO '07: Proceedings of the 9th annual conference on Genetic and
evolutionary computation*, pages 797–804, New York, NY, USA. ACM Press.



Giel, O. (2005).
Zur Analyse von randomisierten Suchheuristiken und Online-Heuristiken.
PhD thesis, Universität Dortmund.



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



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



He, J. and Yao, X. (2003).
Towards an analytic framework for analysing the computation time of
evolutionary algorithms.
Artificial Intelligence, 145(1-2):59–97.

References III



Jansen, T. and Neumann, F. (2007).
Computational complexity and evolutionary computation.
In *GECCO '07: Proceedings of the 2007 GECCO conference companion on Genetic and evolutionary computation*, pages 3225–3250, New York, NY, USA. ACM Press.



Jones, T. and Forrest, S. (1995).
Fitness distance correlation as a measure of problem difficulty for genetic algorithms.
In *Proceedings of the Sixth International Conference on Genetic Algorithms*, pages 184–192, San Francisco, CA. Morgan Kaufmann.



Lehre, P. K. and Yao, X. (2007).
Runtime analysis of (1+1) ea on computing unique input output sequences.
In *Proceedings of 2007 IEEE Congress on Evolutionary Computation (CEC'07)*.



Motwani, R. and Raghavan, P. (1995).
Randomized Algorithms.
Cambridge University Press.



Mühlenbein, H. (1992).
How genetic algorithms really work I. Mutation and Hillclimbing.
In *Proceedings of the Parallel Problem Solving from Nature 2, (PPSN-II)*, pages 15–26. Elsevier.

References IV



Neumann, F. and Wegener, I. (2007).
Randomized local search, evolutionary algorithms, and the minimum spanning tree problem.

Theoretical Computer Science, 378(1):32–40.



Neumann, F. and Witt, C. (2006).
Runtime analysis of a simple ant colony optimization algorithm.

In *In Proceedings of The 17th International Symposium on Algorithms and Computation (ISAAC 2006)*, number 4288 in LNCS, pages 618–627.



Oliveto, P. S., He, J., and Yao, X. (2007a).
Evolutionary algorithms and the vertex cover problem.

In *In Proceedings of the IEEE Congress on Evolutionary Computation (CEC'07)*.



Oliveto, P. S., He, J., and Yao, X. (2007b).
Time complexity of evolutionary algorithms for combinatorial optimization: A decade of results.

To appear in *International Journal of Automation and Computing*.



Scharnow, J., Tinnefeld, K., and Wegener, I. (2002).
Fitness landscapes based on sorting and shortest paths problems.

In *Proceedings of 7th Conf. on Parallel Problem Solving from Nature (PPSN–VII)*, number 2439 in LNCS, pages 54–63.

References V



Storch, T. (2006).

How randomized search heuristics find maximum cliques in planar graphs.
In *GECCO '06: Proceedings of the 8th annual conference on Genetic and evolutionary computation*, pages 567–574, New York, NY, USA. ACM Press.



Witt, C. (2005).

Worst-case and average-case approximations by simple randomized search heuristics.
In *In Proceedings of the 22nd Annual Symposium on Theoretical Aspects of Computer Science (STACS'05)*, number 3404 in LNCS, pages 44–56.



Wolpert, D. H. and Macready, W. G. (1997).

No free lunch theorems for optimization.
IEEE Transactions on Evolutionary Computation, 1(1):67–82.