

We want to maximize  $f : \{0, 1\}^n \rightarrow \mathbb{R}$ .

## I. Randomized Local Search

1. Sample  $x \in \{0, 1\}^n$  uar.
2. For  $t = 1, 2, \dots$  do
  - sample  $i \in [n]$  uar
  - create  $y$  such that for all  $j \in [n]$ ,  $y_j = \begin{cases} 1-x_j & \text{if } j=i \\ x_j & \text{otherwise} \end{cases}$
  - evaluate  $f(y)$
  - selection  $x \leftarrow y$  iff  $f(y) \geq f(x)$

We remind the **OneMax** objective function **OneMax** :  $\begin{cases} \{0, 1\}^n \rightarrow [0, n] \\ x \mapsto \sum_{i=1}^n x_i \end{cases}$

### Prop (Upper bound for RLS on OneMax)

From last lecture:  $\mathbb{E}[T(\text{RLS}, \text{OneMax})] \leq (1 + o(1))n \log n$ .

### Prop (Lower bound for RLS on OneMax)

We have  $\mathbb{E}[T(\text{RLS}, \text{OneMax})] = \Omega(n \log n)$ .

#### Proof.

With probability  $\geq \frac{1}{2}$ , RLS starts in a point  $x$  with  $\text{OneMax}(x) \leq \frac{n}{2}$ .

In each iteration, RLS can only go from  $\text{OneMax}(x)$  to  $\text{OneMax}(x)$  or  $\text{OneMax}(x) + 1$ .

Probability to go to  $\text{OneMax}(x) + 1$  is  $\frac{n - \text{OneMax}(x)}{n}$ .

Hence, the expected optimization time is at least  $\frac{1}{2} \sum_{i=\frac{n}{2}}^{n-1} \frac{n}{n-i} = \frac{1}{2}(1 + o(1))nH_{\frac{n}{2}} = \Omega(n \log n)$ .

## I.1. Fitness level method

Let  $f : S \rightarrow \mathbb{R}$ .

### Definition (Fitness partition)

We call  $L_1, \dots, L_m$  a *fitness partition* iff

1.  $L_1 \sqcup \dots \sqcup L_m = S$
2.  $\forall i < j, \forall x \in L_i, \forall y \in L_j, f(y) > f(x)$
3.  $\forall x \in L_m, f(x) = \max_{s \in S} f(s)$

Let  $A$  be an *elitist* (greedy) algorithm optimizing  $f$ .

For all  $i$ , let  $p_i$  be a lower bound for the probability that algo  $A$  starting in  $x \in L_i$  samples a solution  $y \in \bigcup_{j \geq i} L_j$ .

### Prop (Upper bound for A)

The expected optimization time of  $A$  on  $f$  is at most  $\sum_{i=1}^{m-1} \frac{1}{p_i}$ , that is  $\mathbb{E}[T(A, f)] \leq \sum_{i=1}^{m-1} \frac{1}{p_i}$ .

We introduce two new objective functions.

**Definition (Leading ones)**

$$\text{LeadingOnes} : \begin{cases} \{0,1\}^n \rightarrow [0,n] \\ x \mapsto \max\{i \in [0,n] \mid \forall j \leq i, x_j = 1\} \end{cases}$$

**Example (Leading ones)**

$$\text{LeadingOnes}\left(\underbrace{1110}_{\text{tail}}11010\right) = 3$$

**Definition (Binary value)**

$$\text{BinaryValue} : \begin{cases} \{0,1\}^n \rightarrow \mathbb{R} \\ x \mapsto \sum_{i=1}^n 2^{n-i} x_i \end{cases}$$

**Example (Binary value)**

$$\text{BinaryValue}(01001) = 9$$

**Prop (Upper bound for LeadingOnes)**

$$\mathbb{E}[T(\text{RLS}, \text{LeadingOnes})] = \Theta(n^2)$$

**Proof.**

Use the fitness prop above with  $p_i = \frac{1}{n}$  (that only gives  $O(\cdot)$  and not  $\Theta(\cdot)$ ).

**Prop (Upper bound for BinaryValue)**

$$\mathbb{E}[T(\text{RLS}, \text{BinaryValue})] = \Theta(n \log n)$$

**Proof.**

Same as for OneMax (not using fitness levels because there are too many).

## II. The $(1 + 1)$ Evolutionary Algorithm

With *mutation rate*  $p \in [0, 1]$ .

1. Sample  $x \in \{0, 1\}^n$  uar.
2. For  $t = 1, 2, \dots$  do
  - create  $y \in \{0, 1\}^n$  by setting for all  $i \in [n]$ ,  $y_i = \begin{cases} 1-x_i & \text{with probability } p \\ x_i & \text{otherwise} \end{cases}$
  - evaluate  $f(y)$
  - $x \leftarrow y$  iff  $f(y) \geq f(x)$

**Remark:**

- if  $p = \frac{1}{2}$ : uniform search
- if  $p = \frac{1}{n}$ : in expectation we behave like RLS

**Lemma**

Let  $f : \{0, 1\}^n \rightarrow \mathbb{R}$ .

Let  $(X^i)_{i \in \mathbb{N}}$  be the search trajectory of the  $(1 + 1)$ -EA maximizing  $f$ .

Then  $\mathbb{P}[X^t \in \operatorname{argmax} f] \xrightarrow{t \rightarrow \infty} 1$ .

More precisely, for all  $f$ , we always have  $\mathbb{E}[T((1 + 1)\text{-EA}, f)] = O\left(\left(\frac{1}{p}\right)^n\right)$ .

Is this tight? Can we design a function  $f : \{0, 1\}^n \rightarrow \mathbb{R}$  such that  $\mathbb{E}[T((1 + 1)\text{-EA}, f)] = \Omega\left(\frac{1}{p^n}\right)$ ?  
Yes! We can use the needle below.

**Needle** :  $f(x) = \begin{cases} 1 & \text{for } x = 1 \dots 1 \\ 0 & \text{otherwise} \end{cases}$

Many variants possible (even possible to lead the algo in the wrong direction).

Carola presented an example (using a quadratic form) to trick only half of the people. Obtaining the following result :  $\mathbb{P}[T \leq n \log n] = \frac{1}{2}$  and  $\mathbb{P}[T \geq n^n] = \frac{1}{2}$

Now, let's try to establish upper bounds.

**Prop (Upper bound for  $(1 + 1)$ -EA on OneMax)**

$$\mathbb{E}[T((1 + 1)\text{-EA}, \text{OneMax})] \leq (1 + o(1))en \log n$$

**Proof.**

For **OneMax**, we can only consider 1-bit flips (since the algorithm is elitist).

That leads to  $p_i \geq (n - i) \cdot \frac{1}{n} \cdot \left(1 - \frac{1}{n}\right)^{n-1} \simeq \frac{n-i}{n} \cdot \frac{1}{e}$ . Then  $\mathbb{E}[T((1 + 1)\text{-EA}, \text{OneMax})] \leq \sum_{i=0}^{n-1} e \frac{n}{n-i} = (1 + o(1))en \log n$

**Prop (Upper bound for  $(1 + 1)$ -EA on LeadingOnes)**

$$\mathbb{E}[T((1 + 1)\text{-EA}, \text{LeadingOnes})] \leq (1 + o(1))en^2$$

**Proof.**

Same as above.

**Lemma (Lower bound for  $(1 + 1)$ -EA)**

The expected optimization time of the  $(1 + 1)$ -EA on any function  $f : \{0, 1\}^n \rightarrow \mathbb{R}$  with unique global optimum is  $\Omega(n \log n)$ .

**Proof.**

In the initial solution, with probability at least  $\frac{1}{2}$ , half of the bits are incorrect.

The probability that at least one of them has not been flipped in the first  $t$  iterations equals  $1 - \left(1 - \left(1 - \frac{1}{n}\right)^t\right)^{\frac{n}{2}}$ . With  $t = \frac{1}{3}n \log n$ , the probability is constant.

Let's get an upper bound for  $(1 + 1)$ -EA on **BinaryValue**.

⚠ We can accept  $y$  with  $\text{BinaryValue}(y) \geq \text{BinaryValue}(x)$  but  $d(y, \text{opt}) > d(x, \text{opt})$  (this is not the case for **OneMax**), e.g.:

- $\text{BinaryValue}(10\dots 0) = 2^{n-1}$
- $\text{BinaryValue}(01\dots 1) < 2^{n-1}$

**Theorem (Later with Benjamin)**

The expected optimization time of the  $(1 + 1)$ -EA on any *linear* function  $f : \begin{cases} \{0,1\}^n \rightarrow \mathbb{R} \\ x \mapsto \sum_{i=1}^n w_i x_i \end{cases}$  is  $\Theta(n \log n)$ .

**Prop (Upper bound for  $(1 + 1)$ -EA on BinaryValue)**

$$\mathbb{E}[T((1 + 1)\text{-EA}, \text{BinaryValue})] = O(n^2)$$

**Proof.**

Use fitness partition as in `LeadingOnes`:

- $L_n = \{(1 \dots 1)\}$
- $L_i = \{x \mid \forall j \leq i, x_j = 1\} \setminus \bigcup_{j > i} L_j$

For  $x \in L_i$ , the probability that the  $(1 + 1)$ -EA with mutation probability  $p + \frac{1}{n}$  samples a solution  $y \in \bigcup_{j > i} L_j$  is at least  $\frac{1}{n} \left(1 - \frac{1}{n}\right)^{n-1} \simeq \frac{1}{n} \cdot \frac{1}{e^n}$  (since we need to flip the specific bit in position  $i + 1$ ).

Using fitness level method gives  $\mathbb{E}[T] \leq \sum_{i=0}^{n-1} en = en^2$ .

**III. Project**

Based on *submodular problems*:  $f(A \cup B) \stackrel{?}{\geq} f(A \cap B) \geq f(A) + f(B)$ .

The value of an item depends on the amount of that item we already have.

$\forall X \subseteq Y, \forall z \notin Y, f(X \cup \{z\}) - f(X) \geq f(Y \cup \{z\}) - f(Y)$  (think of chocolate somehow).

**Theory:** we often study worst-case expected optimization time.

**Real life:** we often have a fixed budget, and we care much more about the actual distribution (**use box plots or similar**).

Metric: use **function evaluations** (not CPU time or something else, ...).

Attainment function (EAF): a grid (heatmap) where each cell is  $(t, f(t)) =$

$\mathbb{P}[\text{Algo finds within the first } t \text{ iterations a solution of quality at least } f(t)]$

Performance is way more than “the expectation” and some box plots: we want to gain a deep understanding to be able to adapt algorithms given on the specific instances shape.