I. Black-box complexity

Definition (Black-box complexity)

Let $\mathcal{A}$ be a collection/set of algorithms, and let $\mathcal{F} \subseteq \{f : S \to \mathbb{R}\}$ be a set of functions defined over the *search space* (sometimes *design space*) S.

The *$\mathcal{A}$-black-box complexity of $\mathcal{F}$* is defined as

$$\mathcal{A}\text{-BBC}(\mathcal{F}) = \inf_{A \in \mathcal{A}} \sup_{f \in \mathcal{F}} \mathbb{E}[T(A, f)]$$

the best worst-case expected runtime that an algorithm $A \in \mathcal{A}$ can achieve on the collection $\mathcal{F}$.

Reminder: $T(A, f)$ denotes the number of evaluations that A performs until it queries an element in argmax f.

A general upper bound?
Let S be a finite set and $\mathcal{F} = \{f : S \to \mathbb{R}\}$.
Let $\mathcal{A}$ be the set of all (possibly randomized) algorithms.

$$\begin{aligned}\mathcal{A}\text{-BBC}(\mathcal{F}) &\leq |S| \text{ by enumerating the elements in } S \text{ and query one after the other} \\ &\leq \frac{|S|}{2} \text{ randomly}\end{aligned}$$

Is this tight?
Yes, it can be tight.

Example (Tight collections)

- $\mathcal{F}$ a collection of completely random functions
- needle in-the-hay stack $f_z : S \longrightarrow R, x \mapsto \begin{cases} 1 \text{ if } x=z \\ 0 \text{ otherwise} \end{cases}$

Theorem (No free lunch theorem)

This theorem essentially tells us that all algorithms have equal performance after averaging, over all possible functions $\{f : S \to \mathbb{R}\}$.

Hopefully in practice we do **not** look at all possible functions when optimizing a relevant problem.

An upper bound for a single problem?
Let $f : S \to \mathbb{R}$ and $\mathcal{F} = \{f\}$.
Then, $\mathcal{A}\text{-BBC}(\mathcal{F}) = 1$.
Proof: take $s \in \text{argmax } f$. Let A be the algorithm that queries s in the first iteration.

Unfortunately this is not useful. Instead, we will typically look at collections of functions. Recall from lecture 1, we looked at functions $f_z : \{0,1\}^n \to [0,n], x \mapsto |\{i \in [n] \mid x_i = z_i\}|$. We came from `OneMax`, then said that RLS behaves exactly the same on every function $f_z, z \in \{0,1\}^n$ (in particular $f_{(1,\ldots,1)}$ which is `OneMax`).

Alternative to looking at $\{f_z \mid z \in \{0,1\}^n\}$ is to **restrict** the set of admissible algorithms. One popular way of restricting the class of admissible algorithms is to look only at *k-ary unbiased* black-box optimization algorithms. These algorithms use only so-called k-ary unbiased *variation operators.*

Definition (k-ary unbiased variation operators (for $S = \{0,1\}^n$))

- *k-ary*: the output of the variation operator depends on at most k points. That is, we can describe it as a distribution $D(\cdot \mid x^1, ..., x^k)$ where the x^i's can but do not have to be the k most recently queried ones.
- *unbiased*:
  1. `XOR`-invariant:
  $$\forall y, z \in \{0,1\}^n, D(y \mid x^1, ..., x^k) = D(y \oplus z \mid x^1 \oplus z, ..., x^k \oplus z)$$
  2. permutation invariance:
  $$\forall y \in \{0,1\}^n, \forall \sigma \in \mathcal{S}_n, D(y, x^1, ..., x^k) = D(\sigma(y) \mid \sigma(x^1), ..., \sigma(x^k))$$
  $$\text{where } \sigma(x) = (x_{\sigma(1)}, ..., x_{\sigma(n)})$$

Prop (Black-box complexity of unbiased algorithms (no proof in lecture))

Let $\mathcal{U}$ be the set of of all unbiased black-box algorithms.
Let $f : \{0,1\}^n \to \mathbb{R}$.
Then $\mathcal{U}\text{-BBC}(\{f\}) = \mathcal{A}\text{-BBC}(\{f_{z,\sigma} : \{0,1\}^n \to \mathbb{R}, x \mapsto f(\sigma(x \oplus z))\})$.

- For `OneMax`, we did this (σ does not have any impact).
- For `LeadingOnes`, the permutations do matter: if we want to study $\mathcal{U}$-BBC(`LeadingOnes`), we need to consider the functions

$$\texttt{LeadingOnes}_{z,\sigma} : \begin{cases} \{0,1\}^n \to [0,n] \\ x \mapsto \max\{i \in [0,n] \mid \forall j < i, x_{\sigma(j)} \oplus z_{\sigma(j)} = 1\} \end{cases}$$

Example (Arity and unbiasedness of operators)

- **uniform sampling**: unbiased, 0-ary (the one and only 0-ary unbiased variation operator)
- sample $(0, ..., 0)$: 0-ary but not unbiased: not `XOR` invariant:
$$\mathbb{P}((0, ..., 0)) = 1 \neq \mathbb{P}((0, ..., 0) \oplus (1, ..., 1)) = 0$$
- **flip$_l$** operator: given a point x, create y by first sampling uar without replacement l indices $i_1, ... i_l \in [n]$ and then setting
$$y_j = \begin{cases} 1 - x_j \text{ if } j \in \{i_1, ..., i_l\} \\ x_j \text{ otherwise} \end{cases}$$
This operator is 1-ary and unbiased.

Note: RLS uses flip$_1$.

Definition (Alternative definition of unbiasedness)

A k-ary variation distribution operator D is unbiased if and only if it is *Hamming invariant i.e.*
$$\forall y, z \in \{0,1\}^n, \forall i \in [k], d(y, x^i) = d(z, x^i) \Rightarrow D(y \mid x^1, ..., x^k) = D(z \mid x^1, ..., x^k)$$

This gives us a nice characterization of unbiased operators.

Prop (Characterization for unary unbiased variation operators)

Every unary unbiased variation operator can be described by a probability distribution prob over the possible search radius.

The operator first samples the radius $k \underset{\text{prob}}{\sim} [0, n]$, then we apply the flip_k operator.

A few more unary examples:

- $\mathbf{(1+1)}$**-EA** uses the standard bit mutation operator sbm_p, which flips each bit independently with probability p. It is unbiased (use the previous characterization with $\text{prob} = \mathcal{B}(n, p)$).
- $\textbf{invert}_l$: flips entry in position l: unary and not unbiased.
- $\textbf{flip}_{\textbf{3-ones}}$: takes x and flips exactly 3 positions whose entry is 1, chosen uar: not unbiased.

Now, some binary operators:

- **uniform crossover**: from $x, y \in \{0,1\}^n$, create z by setting $z_i = \begin{cases} x_i \text{ with proba } \frac{1}{2} \\ y_i \text{ otherwise} \end{cases}$. It is unbiased.
- **biased crossover** with bias $c \in]0, \frac{1}{2}[$: pick x_i with proba c - it is still unbiased!
- $\boldsymbol{k}$**-point crossover**: given $x, y \in \{0,1\}^n$, create z by first sampling uar without replacement $i_1, ..., i_k \in [n]$ and setting

$$\begin{aligned} x &= \underline{011} \mid 110 \mid \underline{...} \mid ... \\ y &= 101 \mid \underline{010} \mid ... \mid \underline{...} \\ z &= 011 \mid 010 \mid ... \mid ... \end{aligned}$$

  It is not unbiased (it is not permutation invariant).
- **majority**: given $x^1, ..., x^k$ create y by setting

$$y_i = \begin{cases} \text{maj}\left(x_i^1, ..., x_i^k\right) \text{ if it exists} \\ 1 \text{ with probability } \frac{1}{2} \\ 0 \text{ else} \end{cases}$$

  It is also unbiased.

I.1. Black-box complexity of `OneMax`

I.1.1. Lower bound

Technique:

1. bound the average deterministic BBC
2. use the so-called *Yao's minimax principle*

Let's look at deterministic algorithms for solving an arbitrary function $\texttt{OneMax}_z$.

Every deterministic algorithm can be expressed as a decision tree.

Lemma

Let $\mathcal{D}$ be the set of all deterministic algorithms operating on functions $f : S \to \mathbb{R}$.
Let $\mathcal{F} \subseteq \{f : S \to \mathbb{R}\}$ such that for all $s \in S$, there is a function f_s whose unique global maximum is reached in s, that is $\{s\} = \text{argmax } f_s$.
If for every s it holds that $\left| \left\{ f_{s(x)} \mid x \in S \right\} \right| \leq k$ (at most k branching in our decision tree), then the best possible complexity that a deterministic algorithm can achieve on a uniformely chosen function f_s is at least $\log_k(|S|) - 1$.

Think of $\texttt{OneMax}_z$, which reaches its unique optimal value in z. Moreover, we have $k = n + 1$ possible answers.

Proof (sketch).
We need to place all $s \in S$ somewhere in the decision tree.

In the first j levels of the decision tree, we can place at most $\sum_{i=1}^{j} k^{i-1}$.
We need this sum to be at least $|S|$: essentially, we need at least $\log_k(|S|)$ levels. The average height (level) at which we can find a uniformely chosen $s \in S$ is at least $\log_k(|S|) - 1$.

Conclusion: we have a lower bound for all *deterministic* algorithms on a function chosen uar.

> **Theorem (Yao's (minimax) principle (FOCS, 77))**
> This principle essentially tells us that the **best worst-case complexity of a randomized algorithm** over a set of functions $\mathcal{F}$ **cannot be better** than the best worst-case expected complexity of a **deterministic** algorithm over a **randomly chosen function** $f \in \mathcal{F}$.

We need to define difficult distributions from which we sample the function $f \in \mathcal{F}$.

Going back to $\texttt{OneMax}$.
Consider uniform distribution over $\{f_z \mid z \in \{0,1\}^n\}$. What could be the best possible complexity of a deterministic algorithm over randomly chosen f_z?
By our lemma, this complexity is at least $\log_{n+1}(2^n) - 1 \simeq \frac{n}{\log_2 n}$.

Conclusion: the black-box complexity of $\texttt{OneMax}$ wirt. all randomized algorithms is at least $\frac{n}{\log n}$.

Is this tight? Yes! See Erdös-Rényi 1983, Lindström, 1983.

[Sketch of the proof by Erdös-Rényi, involving lots of pre-computations (inefficient in practice).]

⚠️ **Black-box complexity has nothing to do with classical complexity!**

For example, NP-hard problems can have polynomial BBC.

> **Example (Black-box complexity $\neq$ classical complexity)**
> MaxClique is NP-hard.
> Black-box queries are $S \subseteq V$ and $f(S) = \begin{cases} 0 \text{ if } S \text{ is not a clique} \\ |S| \text{ otherwise} \end{cases}$.
> We have $\mathrm{BBC}(\mathrm{MaxClique}) \leq \binom{|V|}{2}$:
> 1. Query every possible edge $\{v, w\} \subseteq V^2$
> 2. Reconstruct the set of edges E
> 3. Offline computation of MaxClique (expensive because NP-hard but we don't care because we aren't making any query)

> **Theorem (Black-box complexity of $\texttt{OneMax}$)**
> $\mathcal{A}\text{-BBC}(\texttt{OneMax}) = \Theta\left(\frac{n}{\log n}\right)$ **BUT** 1-ary unbiased $\mathcal{A}\text{-BBC}(\texttt{OneMax}) = \Theta(n \log n)$