Average Complexity


- We have looked, mostly, at worst-case complexity: the worst behavior an algorithm exhibits.
- Many algorithms behave much better for a large number of inputs (see QuickSort, or, even, Insertion Sort).
- NP problems (which are worst-case intractable - exponential) may contain large subclasses that are well behaved (= low polynomial).
- It becomes useful and important to determine how an algorithm will behave "on average" (on even in the best case: what's the least amount of effort it will take?)
We want a high average complexity for cryptography (a bad worst case is not enough) - actually we want a high best-case complexity.

We would like a low average complexity for problems like the Hamiltonian Cycle problem (or Traveling Salesman one) - this would also imply a low best-case complexity. As it turns out HC (and other NP-complete problems) has some algorithms that are fast on average.

We would also like to find NP-complete problems with high average complexity - although in this case we just have problems for which no good average complexity algorithms have been found.
Average Complexity

Average (or Expected) Polynomial Time.

In order to talk about average case anything, we must introduce the notion of a probability distribution on the space of items we consider. The time functions we deal with, $T : \mathbb{Z}^+ \rightarrow \mathbb{Z}^+$, depend on the length of the input to a problem (in bits, say), so that it makes sense to talk about distributions in two different ways.

- A distribution $\mu : \mathbb{Z}^+ \rightarrow \mathbb{R}$ is a function s. t. $\mu(i) \geq 0 \ \forall \ i \in \mathbb{Z}^+$ and $\sum_{i \in \mathbb{Z}^+} \mu(i) = 1$.

- Since elements of $\mathbb{Z}^+$ can also be thought of as (binary) strings, it makes sense to define $\mu_n = \mu$ the conditional probability distribution obtained from $\mu$ by restricting $\mu$ to those integers whose (binary) string representations have length $n$ : $\mu_n(x) = \mu(x | |x| = n)$. 
A First Cut

**Definition 1 (Gruska):** an algorithm runs in expected polynomial time $T(n)$ over a probability distribution $\mu$ if

$$(\exists k \geq 0)(\forall n) \sum_{|x| = n} T(x)\mu_n(x) = O(n^k),$$

where $T(x)$ is the time complexity of the algorithm for input $x$ and $\mu_n$ is the conditional probability distribution of $\mu$ on strings of length $n$.

Note that $k$ is independent of $x$ and $n$, but depends only on the language being examined. Note further that $T$ has two meanings: $T(x)$ refers to the running time of a specific input instance; $T(n)$ refers to the expected time over all instances of length $n$.

As a definition, it makes sense.
Is this definition robust? : if I replace a machine with one that is polynomially faster or slower, does a polynomial expected time remain polynomial?

Ex. (Gruska): Let $\mathcal{A}$ be an algorithm that runs in polynomial time $O(n^k)$ on a $(1 - 2^{-0.1n})$ fraction of input instances of length $n$ and runs in time $2^{0.09n}$ on the remaining $2^{-0.1n}$ fraction of inputs. Hence the expected time for the algorithm (over strings of length $n$) is bounded by

$$O(n^k) \cdot (1 - 2^{-0.1n}) + 2^{0.09n} \cdot 2^{-0.1n} = O(n^k) + 2^{-0.01n} = O(n^k).$$

Now consider a quadratically slower machine ($T(n) \rightarrow T(n^2)$):

$$O(n^{2k}) \cdot (1 - 2^{-0.1n}) + (2^{0.09n})^2 \cdot 2^{-0.1n} = O(n^{2k}) + 2^{0.08n} = \Omega(2^{0.08n}).$$

Conclusion: the definition is NOT robust…
A Second Cut

What would make it robust? One observation we can make about the previous example is that the inputs that required exponential time were not "rare enough": even though their percentage among all inputs of a given length decays exponentially with length, that's not good enough. It is easy to check that even if the percentage decay proceeded as an exponential polynomial \((2^{-cn^k})\), for \(0 < c << 1, k > 0\), a polynomial change in the speed of the TM could still move us to exponential time.

- We must introduce requirements for the "rareness" of "hard" inputs that will guarantee polynomial invariance: average polynomial complexity should be invariant under polynomial changes to the TM speed.
Average Complexity

- Let $r : \Sigma^* \to \mathbb{R}^+$ be a function that measures the "rareness" of inputs from $\Sigma^*$, and require for "average polynomial time" to mean that
  $$(\exists k > 0)(\forall x \in \Sigma^*) \ T(x) \leq (|x| r(x))^k.$$ 

Taking the $k^{th}$ root of both sides: $T(x)^{1/k} |x|^{-1} \leq r(x)$. When we introduce a
$\delta \in (0, 1]$ we can sum
$\Sigma_{x \in \Sigma^+} T(x)^{\delta/k} |x|^{-1} \mu(x) \leq \Sigma_{x \in \Sigma^+} (T(x)^{1/k} |x|^{-1})^\delta \mu(x) \leq \Sigma_{x \in \Sigma^+} r(x)\delta \mu(x) < \infty$.

We can use this to motivate the

**Definition (L. Levin '86):** A function $f : \Sigma^* \to \mathbb{N}$ is $\mu$-average polynomial
(or polynomial on $\mu$-average), w.r.t. the probability distribution $\mu$, if
there exists $k \in \mathbb{N}^+$ s.t.
$$\sum_{x \in \Sigma^*} \frac{\mu(x)}{|x|} \left(\frac{f(x)}{|x|}\right)^{1/k} < \infty.$$
Note that raising $f(x)$ or $|x|$ to a constant power would not change the outcome (the $k$ may need to be different, but all we need is existence).

This implies that the definition is machine-independent (at least as far as polynomially faster or slower machines.

The definition is not affected by polynomial reductions (transform a problem into another by means of a recursive function of polynomial complexity).

How do we tie this to the original (flawed) definition, and how do we deal with the sequence $\{ \mu_n \mid n > 0 \}$ of conditional distributions highlighting the input strings of length $n$?
**Average Complexity**

**Definition 1**: We say that $T : \mathbb{Z}^+ \rightarrow \mathbb{Z}^+$ is polynomial on average w.r.t. $\mu$, a distribution on $\mathbb{Z}^+$, if there exists a $k \in \mathbb{Z}^+$ so that

$$
\sum_{x \in \mathbb{Z}^+} \frac{T^k(x)}{|x|} \mu(x) < \infty.
$$

**Definition 2**: We say that $T : \mathbb{Z}^+ \rightarrow \mathbb{Z}^+$ is polynomial on average w.r.t. $\mu_n$, with $\{\mu_n \mid n > 0\}$ a sequence (or ensemble) of distributions on $\mathbb{Z}^+$, if there exists a $k \in \mathbb{Z}^+$ so that

$$(\forall n) \sum_{|x|=n} T(x)^{1/k} \mu_n(x) = O(n).$$

**Question**: are the two definitions equivalent?
Proposition 1 (Impagliazzo, p. 10): if $\mu_n(x) = \mu(x \mid \mathbf{x} = n)$, then Definition 1 is equivalent to Definition 2.

Proof: see the reference given.

Why $T(x)^{1/k}$ instead of $T(x)$?

1. The example on slide #5. This new definition will still maintain a polynomial bound even after replacing a Turing machine with one polynomially slower - $k$ may have to be increased, but that's all.
2. One can prove that if functions $f$ and $g$ are $\mu$-average polynomial, then $f + g$ and $f \cdot g$ are $\mu$-average polynomial.
Explained in Y. Gurevitch's "The Challenger-Solver Game" (http://research.microsoft.com/~gurevich/Opera/85.pdf): it attempts to capture the trade-off between a measure of difficulty and the fraction of hard instances of the problem, i.e., the trade-off between a time-bound $T$ and the fraction of instances that take the algorithm more than $T$ time. The trade-off should be polynomial in $T$: only a sub-polynomial fraction of instances should require super-polynomial time; only a quasi-polynomial fraction should require more than quasi-polynomial time. Another way of stating this would be that the time to find, via random sampling, an instance requiring more than $T$ time is at least $T^{1/k}$, so the poser of the problem has no more than polynomial advantage over the solver (no advantage: finding requires $T$ time, just as much as the solution bound).
We now extend the concept of NP-completeness to average-case complexity.

We start by generalizing what we mean by a decision problem: a distributional decision problem (ddp) is a pair \((L, \mu)\) where \(L \subseteq \Sigma^*\) is a language \(L = \{x \mid \mu(x) > 0\}\), and \(\mu\) is a probability distribution over \(\Sigma^+\).

A language \(L\) is decidable in average polynomial time w.r.t. a distribution \(\mu\) if it can be decided by a deterministic algorithm whose time complexity is bounded from above by a \(\mu\)-average polynomial function.

**AP** (Average Polynomial) is the class of ddps \((L, \mu)\) where \(L\) is decidable in \(\mu\)-average polynomial time.

**ANP** (Average Non-deterministic Polynomial) is the class of ddps \((L, \mu)\) where \(L\) is decidable in \(\mu\)-average polynomial time on a non-deterministic Turing machine.
There are two critical notions associated with the P vs NP problem, and those need to be properly translated into the world of distributional decision problems: reducibility and completeness.

We start by introducing some ideas to support a useful notion of reducibility.

- A specialized definition of polynomial-time computable (approximable?): a function $f : \Sigma^+ \rightarrow [0, 1]$ (note: $[0, 1] \subset \mathbb{R}$) is polynomial-time-computable if there is a Multi-tape Turing Machine which, for every input $x \in \Sigma^+$, outputs a finite binary fraction $y$, in time polynomial in $|x|$ and $k$, s.t. $|f(x) - y| \leq 2^{-k}$. Note that distributions map $\Sigma^+ \rightarrow [0, 1]$. 
Average Complexity

Reducibility & Completeness

- We consider only distributions \( \mu \) s.t. the cumulative distribution function \( \mu^*(x) = \sum_{y \leq x} \mu(y) \) is polynomial-time computable. This is a strong requirement, since there are exponentially many strings over which the sum is taken. Note that if \( \mu^*(x) \) is polynomial-time computable, so is \( \mu(x) \) (Proof?).

- From now on a distribution \( \mu \) will be said to be polynomial-time computable, if both \( \mu \) and \( \mu^* \) are polynomial-time computable.

- From NP-completeness we have the notion of many-to-one polynomial-time reducibility from one decision problem to another. Any reduction function for our average case study will have to satisfy this condition and something else.
Average Complexity

Reducibility & Completeness

We require that AP be closed under reduction: more specifically, if \((L_1, \mu_1)\) and \((L_2, \mu_2)\) are ddps and \(f : (L_1, \mu_1) \rightarrow (L_2, \mu_2)\) is a many-to-one polynomial-time reduction, then \((L_2, \mu_2) \in \text{AP} \Rightarrow (L_1, \mu_1) \in \text{AP}\).

- We make use of the "usual worst-case reduction" with the requirement that it does not reduce "high density" or frequent (as in \(\mu_1(x)\)) instances of \(L_1\) into "low density" or rare ones of \(L_2\).

- In more precise terms, define \(f(\mu_1)\) to be the distribution
  \[
  f(\mu_1)(y) \equiv \sum_{f(x)=y} \mu_1(x),
  \]
  where the distribution \(f(\mu_1)\) must be bounded above, within a polynomial factor, by \(\mu_2\).
Average Complexity

Reducibility & Completeness

Definition (Gruska):

1. Let $\mu$ and $\nu$ be probability distributions on $\Sigma^*$. We say that $\mu$ is dominated by $\nu$, denoted by $\mu \preceq \nu$, if there exists a polynomial $p$ s.t. $\mu(x) \leq p(|x|)\nu(x) \ \forall \ x \in \Sigma^*$.

2. Let $\mu$ and $\nu$ be probability distributions on strings of languages $L_1$ and $L_2$, respectively, and $f$ a many-to-one reduction from $L_1$ to $L_2$. We say that $\mu_1$ is dominated by $\mu_2$ w.r.t. $f$ ($\mu \preceq_f \nu$) if there exists a distribution $\mu_1'$ on $\Sigma^+$ s.t. $\mu_1 \preceq \mu_1'$ and $\mu_2(y) = f(\mu_1')(y) \ \forall \ y \in \text{range}(f)$.

============== FINALLY ===============

Definition: a ddp $(L_1, \mu_1)$ is polynomial-time reducible to $(L_2, \mu_2)$ if there is a polynomial-time computable reduction $f$ s.t. $L_1$ is many-to-one reducible to $L_2$ via $f$ and $\mu_1 \preceq_f \mu_2$. 
Average Complexity

Reducibility & Completeness

**Restatement**: a polynomial reduction from a ddp \((L, \mu), L \subseteq \Sigma^*\), to a ddp \((L', \mu')\), is a polynomial-time reduction \(f\) of \(L\) to \(L'\) s.t. \(\exists l \in \mathbb{Z}^+\) s.t. \(\forall x \in \Sigma^*\),

\[
\mu'(x) \geq \frac{1}{|x|^l} \sum_{y \in f^{-1}(x)} \mu(y),
\]

i.e., the distribution \(\mu'\) should be nowhere more than polynomially smaller than the distribution induced by \(\mu\).

We can now state two results that drove the previous definitions.

1. If a ddp \((L_1, \mu_1)\) is polynomial-time reducible to \((L_2, \mu_2)\), and \((L_2, \mu_2) \in \text{AP}\), then \((L_1, \mu_1) \in \text{AP}\).
2. Polynomial-time reductions on ddps are transitive.
Average Complexity

Reducibility & Completeness

**Definition**: DNP denotes the class of ddps \((L, \mu)\) s.t. \(L \in \text{NP}\) and \(\mu \preceq \nu\) for some polynomial-time computable distribution \(\nu\).

**Note**: DNP is a proper subclass of ANP. Just compare the definitions: we have a strictly stronger condition on \(\mu\). This class seems to be, at the moment, the right framework for average-case NP-completeness.

**Definition**: a ddp \((L, \mu)\) is average-case NP-complete (or DNP-complete) if \((L, \mu) \in \text{DNP}\) and every ddp in DNP is polynomial-time reducible to it.

We now have definitions for both reducibility and completeness. What can we do with them? Is the second definition meaningful?
Distributions - how do you choose one?

The major problem is that we do not have any good idea (at the moment, at least) on how to choose the distribution. Choosing a uniform distribution is not possible: all of the languages of interest have infinite cardinality, so there is no meaningful "constant value" \( \mu \) (every string has the same probability of being chosen) available.

**Definition**: a polynomial-time computable distribution \( \mu \) on \( \Sigma^+ \) is called uniform if there is a function \( \rho : \mathbb{N} \to \mathbb{R} \) s.t. \( \forall x \in \Sigma^+, \mu(x) = \rho(|x|)2^{-|x|} \), where \( \Sigma_n \rho(n) = 1 \) and there is a polynomial \( p \) s.t. \( \rho(n) \geq 1/p(n) \) for all large \( n \).
Average Complexity

Distributions - how do you choose one?

The idea is to choose an integer (the length of the binary string) at random ("uniformly"?), and then to choose a string at random. Having chosen a length $n$, choosing the string uniformly would have probability $2^{-n}$ for each of the $2^n$ binary strings of that length. The function $\rho$, and the conditions on it, guarantee to items:

- The sum $\sum_x \mu(x) = 1$;
- $\mu(x)$ does not decay "too fast" as the size of $x$ increases.

With these conditions, several problems have been shown to be DNP-complete: bounded halting, bounded Post Correspondence Problem, bounded tiling, etc.
Distributions - how do you choose one?

An example of such a function $\rho(n)$ and the associated "uniform" distribution would be:

- $\rho(n) = 6/(\pi^2n^2)$
- $\sum_n \rho(n) = 1$, where the result follows from a summation in Calculus.
- $\mu(x) = \rho(|x|)2^{-|x|}$.

We can choose such functions $\rho$ in different ways, biasing our "uniformity" towards short or long strings. The reciprocal polynomial requirement simply guarantees that long strings still have a "reasonable probability" of being chosen, but becoming quickly smaller as the length increases.
A Problem and its Average Case Counterpart: Bounded Halting

**Thm.** The language

\[ L_{halt}^* = \{(M, w\#^n) \mid M \text{ is a one-tape, one head NTM that accepts } w \text{ in at most } n \text{ steps - and } \# \text{ is a marker } \notin \Gamma^*} \]

is NP-complete.

**Proof.** There are two parts to this proof:

1. \[ L_{halt}^* \in \text{NP}. \] Construct an NTM \( M_0 \) that
   a. Checks whether \( w' = (M, w\#^n) \) for some TM M and input \( w \) of M. If not it rejects \( w' \). This can be done in polynomial time. If \( w' \) is of the appropriate form,
   b. \( M_0 \) computes the max cardinality of the range of the transition relation - say \( k > 0 \). This can be done by one scan through the instruction set if we assume it to use appropriate representation.
b. (Cont.) For ex.: each "instruction" looks like

\[110^{n_1}10^{n_2}10^{m_{11}}10^{m_{12}}10^{m_{11}}1\ldots10^{m_{k1}}10^{m_{k2}}10^{m_{k3}}11\]

with a different polynomial overhead if the instructions are scattered. Let \(\eta_1, \ldots, \eta_k\) be \(k\) symbols not yet used and let \(M_0\) generate, randomly, the string \(\eta_{i1}\ldots\eta_{il}, l \leq n\) and consider \((M, w\#^{n}\eta_{i1}\ldots\eta_{il})\). Let \(M_0\) simulate the execution of the instructions of \(M\) on \(w\), as indicated by the string \(\eta_{i1}\ldots\eta_{il}\), starting from the initial configuration. Accept if \(M\) accepts, reject otherwise. Thus \(L_{halt}^* \in \text{NP}\).

2. For every \(L \in \text{NP}\) (\(L \subset \Gamma^*\)), \(L \leq_m^P L_{halt}^*\).

\textbf{Pf:} Since \(L \in \text{NP}\), there exists an NTM \(M_L\) and a polynomial \(p_L\) such that \(M_L\) accepts \(L\) in time \(p_L(n)\). Let \(\tau_L\) be a function on \(\Gamma^*\), \(\tau_L(w) = (M, w\#^{p_L(|w|)})\). Since \(p_L\) is a polynomial, \(\tau_L(w)\) can be computed in polynomial time. We also have that \(w \in L \iff \tau_L(w) \in L_{halt}^*\).
A Problem and its Average Case Counterpart: Bounded Halting

We can reformulate BHP (the Bounded Halting Problem) as RBHP (Randomized BHP): we can use a distribution based on the $\rho$ function defined a few slides back:

- given: a Turing Machine $M$;
- input: a string $w\#^n$, with $n > |w|$;
- question: is there a halting computation of $M$ on $w$ with at most $n$ steps?

- probability distribution: pick $n$ randomly; pick $k < n$ randomly according to a uniform distribution (probability of a specific $k = 1/n$); pick a specific string $w$, $|w| = k$, with associated probability $2^{-|w|}$. The probability is proportional to $n^3 2^{-|w|}$. 

Average Complexity

Summary of results

- Just as for NP-completeness, all known pairs of ANP-complete problems are polynomially isomorphic \((A \leq^P_m B \land B \leq^P_m A)\).

- To define average-case NP-completeness we could use average polynomial-time reductions rather than polynomial-time reductions (worst-case). Using average polynomial-time reductions one can define completeness for ANP (Average Non-deterministic Polynomial). All average-case NP-complete problems are also average polynomial-time complete for ANP. It can be shown, on the other hand, that there are ddps that are not in DNP, but are average polynomial-time complete for problems in ANP with polynomial-time computable distributions.
Average Complexity

Summary of results

- It has been shown that there are problems not in P but in AP under any polynomial-time computable distribution. However, if a problem is in AP under every exponential-time computable distribution, then it must be in P.

Unfortunately, the proofs of all the results are somewhat more complex: the presence of probability distributions makes life harder, but it makes some of the hierarchies "finer".