Information Measure as Time Complexity

GUO ZHAO, Southwest University for Nationalities

The concept of Shannon entropy of random variables was generalized to measurable functions in general, and to simple functions with finite values in particular. It is shown that the information measure of a measurable function is related to the time complexity of search problems concerning this function. Formally, given a Turing reduction from a search problem \( f(x) = y \) to another search problem \( g(x) = z \), the expected number of queries is \( \frac{H(f)}{I(f;g)} \), where \( H(f) \) is the entropy of \( f(x) \) and \( I(f;g) \) is the average mutual information between \( f(x) \) and \( g(x) \). In the idea case, if \( \frac{H(f)}{I(f;g)} \) is polynomial in the maximal size of inputs and the problem \( g(x) = z \) can be solved in polynomial time, then the problem \( f(x) = y \) also has polynomial-time algorithm. As it turns out, our information-based complexity estimation is a natural setting in which to study the power of randomized or probabilistic algorithms. Applying to decision problems, our result provides a strong evidence that \( P = RP = BPP \).

Categories and Subject Descriptors: H. [Information Systems]: Information theory; F. [Theory of Computation]: Complexity Measures and Classes

General Terms: Algorithms, Measurement, Theory

Additional Key Words and Phrases: Shannon entropy, Mutual information, Measurable functions, Search problem, Decision problem, Turing reduction, Time complexity

1. INTRODUCTION

This paper addresses the following questions: How much information is contained in a function? Or more specifically, given a measurable function \( f : X \to \mathbb{R} \) and a random input \( x \in X \), how much information is needed to specify the exact output \( f(x) \)? Further, given a Turing reduction from given search problem \( f(x) = y_k \) to another search problem \( g(x) = z_i \) (see Goldreich 2008), how much information about \( f(x) = y_k \) can be obtained from querying the solutions of \( g(x) = z_i \)? And what is the least number of queries required to solve \( f(x) = y_k \)?

In fact, the same idea has been used by Shannon (1949) to analyze the practical secrecy of cryptosystem. Using functional notation, assume the enciphering function to be \( y = f_K(x) \), where \( K \) corresponds to the particular key being used. Given an enciphered message \( (y) \), any system can be broken, at
least in theory, by merely trying each possible key \((K)\) until the most possible solution \((x)\) is obtained. The complexity of the system can be measured by the \textbf{expected} number of trials to carry out the solution. Suppose that the entropy of key space is given by \(H(K)\), which is the average amount of information associated with the choice of keys. If each trial has \(S\) \textit{equally possible} results then the expected number of trials will be \(\frac{H(K)}{\log S}\). However, if the key space has very different probabilities, then only a small amount of information is obtained from testing a single key in complete trial and error.

Further, Shannon (1949) pointed out that the time complexity of decoding a cryptosystem is similar to the coin weighing problem (see also Erdos and Renyi 1963). A typical example is the following: if one coin in 27 is slightly lighter than the rest, what is the \textbf{least} number of weighings required to isolate it using a chemist’s balance? The correct answer is 3, obtained by first dividing the coin into three groups of 9 each. The set of coins corresponds to the set of keys, the counterfeit coin to the correct key, and the weighing procedure to a trial or test. The original uncertainty is \(\log 27\) bits, and each trial yields \(\log 3\) bits of information; thus, at least \(\frac{\log 27}{\log 3} = 3\) trials are required.

The situations of our questions are essentially the same. But the question we now face is how to measure the amount of information contained in given function (see Shah and Sharma 2000 for similar idea).

To this end, the concept of Shannon entropy of random variables was extended to measurable functions in general, and to simple functions with finite values in particular (see Patrick Billingsley 1995). Formally, let \((X, \mu)\) be a measurable space with measure \(\mu(X) < \infty\) and \(f : X \rightarrow \mathbb{R}\) be a measurable function. The key point is to approximate \(f : X \rightarrow \mathbb{R}\) by a simple function with finite values, which is obtained by partitioning the range of \(f(x)\) in the same way as computing its Lebesgue integral (see Patrick Billingsley 1995). As a result, these values, coupled with the probabilities of occurrence of their preimages, form a random variable in itself (see Fig. 1 and section 2 for details). So, it is natural to approximate the entropy of the given function by the entropy of the resulting random variable.
Specially, if \( f : X \to \mathbb{R} \) is a simple function with values \( y_1, y_2, \ldots, y_n \), then the self-information associated with the event \( \{ f = y_k \} \) is equal to
\[
I_k(f) = -\log p_k ,
\]
where \( p_k = \Pr(f = y_k) = \frac{\mu(f^{-1}(y_k))}{\mu(X)} \). Consequently, the entropy of \( f \) is defined to be \( H(f) = -\sum_{k=1}^{n} p_k \log p_k \). If the measure \( \mu(X) = 1 \), then \( f : X \to \mathbb{R} \) itself becomes into a random variable, and our definition of entropy coincides with that of Shannon (1948).

However, for cryptographic purposes, especially in Public-key cryptography, the given cipher text \( y_k = f(x) \) may convey some information about the corresponding preimage(s). As such, it may be possible that one can decrypt one cipher text \( y_k \) without inverting the enciphering function \( f : X \to \mathbb{R} \) directly. Intuitively, decrypting one cipher text \( y_k \) may not be computationally equivalent to inverting the enciphering function \( f : X \to \mathbb{R} \), which means decrypting all cipher texts.

To overcome this logical difficulty, we have to make good use of the information conveyed by the given search problem \( f(x) = y_k \). In some idea cases, the information hidden in the function \( f : X \to \mathbb{R} \) and the information about the value \( y_k \) may enable us to reduce the search problem \( f(x) = y_k \) to another search problem \( g(x) = z_i \).

A typical example is the optimization problem of real-valued functions. Given a differentiable function \( y = f(x) \) on an open interval \( (a,b) \), Fermat’s theorem states that if \( f(x) \) has a local extremum at \( x_0 \in (a,b) \), and then its derivative \( f'(x) \) must satisfy \( f'(x_0) = 0 \). By using Fermat’s theorem, the local extremum of \( f(x) \) is found by solving an equation \( f'(x_0) = 0 \). As a result, the information
of differentiability enables us to reduce the optimization problem $\min_{x \in (a,b)} f(x)$ to another search problem $f'(x_0) = 0$ (see Fig. 2 and section 3.4 for details).

Formally, assume there is a reduction from given search problem $f(x) = y_k$ to another search problem $g(x) = z_i$ (see Papadimitrou 1994). In order to quantify the average amount of information about $f(x)$ obtained from $g(x)$ we have to calculate the Mutual Information between $f(x)$ and $g(x)$ (see Thomas M. Cover, Joy A. Thomas 2006).

To this end, approximate the function $g : Y \rightarrow \mathcal{R}$ by a simple function with finite values $z_0, z_1, \ldots, z_m$. Then the mutual information $I(f;g)$ of $f(x)$ and $g(x)$ is by definition equal to the expected value of pointwise mutual information, that is

$$I(f;g) = \sum_{k=1}^{n} \sum_{i=1}^{m} \Pr(f = y_k, g = z_i) \log \frac{\Pr(f = y_k, g = z_i)}{\Pr(f = y_k) \Pr(g = z_i)}.$$  \hspace{1cm} (1)

Intuitively, mutual information measures the information that $f(x)$ and $g(x)$ share.

Given a Turing reduction from given search problem $f(x) = y_k$ to another search problem $g(x) = z_i$, the search problem $f(x) = y_k$ can be solved by querying the solutions of $g(x) = z_i$ (see Goldreich 2008). Further, if each query can be done in polynomial time, then to roughly estimate the time complexity of solving $f(x) = y_k$ we just need to estimate the expected number of queries.

Now that the self-information associated with $\{f = y_k\}$ is equal to $I_k(f) = -\log p_k$, and the amount of information about $f(x) = y_k$ provided by
each query is exactly equal to the average mutual information $I(f;g)$, the least number of queries needed to solve $f(x) = y_k$ will be

$$
\frac{I_k(f)}{I(f;g)} = \sum_{k=1}^n \sum_{i=1}^m \log \frac{\Pr(f = y_k, g = z_i)}{\Pr(f = y_k) \Pr(g = z_i)}.
$$

(2)

In theory, $\frac{I_k(f)}{I(f;g)}$ gives an upper bound to the time complexity of solving $y_k = f(x)$.

On average, the expected number of querying $g(x)$ needed to solve $f(x) = y_k$ for randomly chosen value $y_k$ is $\frac{H(f)}{I(f;g)}$. In the idea case, if $\frac{H(f)}{I(f;g)}$ is polynomial in the maximal size of inputs and the problem $g(x) = z_i$ can be solved in polynomial time, then the problem $f(x) = y_k$ also has polynomial-time algorithm.

As it turns out, our information-based complexity estimation is a natural setting in which to study the power of randomized or probabilistic algorithms. Applying to decision problems, our result provides a strong evidence that $P = RP = BPP$.

The main contributions of this paper can be summarized as follows.

— The concept of Shannon entropy of random variables was generalized to measurable functions.

— It is shown that the information measure of functions is related to the time complexity of solving search problems concerning functions.

— Our information-based complexity estimation provides a strong evidence that $P = RP = BPP$.

2. MEASURABLE FUNCTIONS

We begin with some definitions and lemmas. Those who are familiar with Shannon information theory (see Thomas M. Cover, Joy A. Thomas 2006) and measure theory (see Patrick Billingsley 1995) can skip directly to section 3.

2.1 Random Variables

A random variable is a measurable function $\xi: X \to \mathcal{R}$ defined on the sample space $X$, endowed with probability measure, that is, $\mu(X) = 1$.

Generally, randomness means uncertainty. In theory, the degree of uncertainty about given random variable may be characterized by its entropy (see Shannon 1948). Shannon entropy plays a central role in information theory as measure of information, choice, and uncertainty. Since Shannon entropy measures the average amount of information contained in given random variable, it provides an absolute limit on the best possible expected number of choices of specifying a value of given random variable, assuming that each choice requires one unit of information.

In his 1948 paper "A Mathematical Theory of Communication", Shannon classified random variables into the discrete case and the continuous case. If
\( \xi : X \to \mathbb{R} \) is a discrete random variable with probabilities \( p_1, p_2, \cdots, p_n \), then its entropy is defined to be

\[
H(\xi) = H(p_1, p_2, \cdots, p_n) = -\sum_{k=1}^{n} p_k \log p_k .
\]  

(3)

Note that the choice of a logarithmic base corresponds to the choice of a unit for measuring information. Typically, the base is taken to be 2 and will be dropped if the context makes it clear.

As for the continuous case, to a considerable extent it can be obtained through a limiting process from the discrete case by dividing its range into a large but finite number of small regions and calculating the entropy involved on a discrete basis. As the size of the regions is decreased the entropy in general approaches as limits the proper entropy for the continuous case. However, the theory of entropy can be formulated in a completely axiomatic and rigorous manner which includes both the discrete and continuous cases. Formally, given a random variable \( \xi : X \to \mathbb{R} \) with probability density function \( p(x) \), its Shannon entropy \( H(\xi) \) can be defined by means of Riemann-Stieltjes integral

\[
H(\xi) = -\int_{-\infty}^{+\infty} p(x) \log p(x) d\mu(x) .
\]  

(4)

where \( \mu \) is the Lebesgue measure if \( \xi \) is continuous, and \( \mu \) is the counting measure in case \( \xi \) is discrete.

However, since arbitrary function may not be integrable in the sense of Riemann-Stieltjes, we have to define entropy for measurable functions on the basis of Lebesgue integral.

2.2 Measurable Functions

Let \((X, \mu)\) be a measurable space. A real-valued function \( f : X \to \mathbb{R} \) is said to be measurable in case \( \{ x \in X \mid f(x) > c \} \) is measurable for all \( c \in \mathbb{R} \).

If \( \mu(X) = 1 \), then a measurable function \( f : X \to \mathbb{R} \) is by definition a random variable. In this case, \( X \) amounts to the sample space.

It is well known that every Lebesgue measurable function is nearly continuous, as being stated by the famous Lusin’s theorem (see Patrick Billingsley 1995). On the other hand, most functions of interest in combinatorial optimization and cryptography are discrete. But we can unify both the discrete and continuous cases by invoking simple functions (see Patrick Billingsley 1995).

2.3 Simple Functions

A simple function is a real-valued function that attains only a finite number \( (n) \) of values. Formally, a simple function is a finite linear combination of characteristic functions of measurable sets. More precisely, let \((X, \mu)\) be a measurable space, \( X_1, X_2, \cdots, X_n \) be a finite partition of \( X \), and \( c_1, c_2, \cdots, c_n \) be a sequence of real numbers. A simple function is a function \( f : X \to \mathbb{R} \) of the form
\begin{equation}
    f(x) = \sum_{k=1}^{n} c_k \chi_{X_k}(x)
\end{equation}

where \( \chi_{X_k} \) is the characteristic function of the set \( X_k \).

It turns out that any measurable function can be approximated by simple functions. More precisely, we have the following lemma (see Patrick Billingsley 1995).

**Lemma 2.1.** Let \( f : X \to \mathbb{R} \) be any measurable function. Then there exists a sequence of simple functions \( f_n : X \to \mathbb{R} \) such that \( \lim_{n \to \infty} f_n(x) = f(x) \) for all \( x \in X \).

**Proof.** Without loss of generality, assume \( f : X \to \mathbb{R} \) to be non-negative. For each \( n \in \mathbb{N} \), subdivide the range of \( f : X \to [0, +\infty) \) into \( 2^{2n} + 1 \) intervals: 
\[
    \left( -\frac{k-1}{2^n}, \frac{k}{2^n} \right) \quad \text{for} \quad k = 1, 2, \ldots, 2^{2n} \quad \text{and} \quad \left[ 2^n, +\infty \right).
\]
For each \( n \in \mathbb{N} \), define the measurable sets
\[
    X_{n,k} = f^{-1}\left( \left[ -\frac{k-1}{2^n}, \frac{k}{2^n} \right) \right), \forall k = 1, 2, \ldots, 2^{2n} \quad \text{and} \quad X_{n,2^{2n}+1} = [2^n, +\infty).
\]
Consequently, define the sequence of increasing simple functions
\begin{equation}
    f_n = \sum_{k=1}^{2^{2n}+1} \frac{k-1}{2^n} \chi_{X_{n,k}}.
\end{equation}

Then it is easy to see that \( \lim_{n \to \infty} f_n(x) = f(x) \) for all \( x \in X \) (see Fig. 1).

### 3. Information Measure

Following the technical route of Shannon (1948), we shall measure the average amount of information contained in a measurable function in terms of entropy.

#### 3.1 Entropy

First of all, there is a natural way to obtain random variables from given measurable function. The key idea is that each measurable function can be approximated by a simple function with finite number of values (see Patrick Billingsley 1995). As a result, these values, coupled with the probability measure of their preimages, forms a random variable in itself. So, it is natural to approximate the entropy of the given function by the entropy of the resulting random variable.

Formally, let \( (X, \mu) \) be a measurable space with measure \( \mu(X) < \infty \) and \( f : X \to \mathbb{R} \) be a measurable function. To define the entropy for \( f : X \to \mathbb{R} \) conducts the following steps.
(1) Approximate the given function \( f : \mathbb{X} \to \mathbb{R} \) by a simple function \( f_n : \mathbb{X} \to \mathbb{R} \) with values \( y_1, y_2, \ldots, y_n \), by partitioning the range of \( f(x) \) in the same way as computing its Lebesgue integral (see the proof of Lemma 2.1).

(2) For each value \( y_k \) \((k = 1, 2, \ldots, n)\),

(a) Define the measurable set \( X_k = f_n^{-1}(y_k) = \{ x \in \mathbb{X} \mid f_n(x) = y_k \} \).

(b) Calculate \( p_k = \Pr(f = y_k) = \frac{\mu(X_k)}{\mu(X)} \), which stands for the probability that the output of a randomly chosen input \( x \) takes the value \( y_k \).

(3) Compute the sum \( H(f_n) = -\sum_{k=1}^{n} p_k \log p_k \) as the entropy of \( f_n \).

(4) Define the entropy of \( f \) to be the limit \( H(f) := \lim_{n \to \infty} H(f_n) \), if necessary.

As a natural generalization of the Shannon entropy of random variables, the entropy of measurable functions possesses well-behaved properties. To simplify notations, in the following we assume that \( f : \mathbb{X} \to \mathbb{R} \) itself is a simple function taking values \( y_1, y_2, \ldots, y_n \) with probabilities \( p_1, p_2, \ldots, p_n \), respectively, and denote its entropy by \( H(f) = H(p_1, p_2, \ldots, p_n) \).

**Maximum.** \( H(f) = H(p_1, p_2, \ldots, p_n) \leq \log n \), with equality if and only if all the values are equally likely, i.e., \( p_k = \frac{1}{n} \) for all \( k = 1, 2, \ldots, n \).

![Fig. 3. Entropy in the case of two possibilities.](source: Shannon, Bell System Technical Journal, 27, 1948.)

**Continuity.** \( \lim_{\varepsilon \to 0} H(p_1, p_2, \ldots, p_n - \varepsilon, \varepsilon) = H(p_1, p_2, \ldots, p_n) \). The entropy is continuous, so that changing the values of the probabilities by a very small amount should only change the entropy by a small amount. Specially, adding or removing a value with probability zero does not contribute to the entropy, that is, \( H(p_1, p_2, \ldots, p_n, 0) = H(p_1, p_2, \ldots, p_n) \).
**Example 3.1 (Dirichlet function).** The Dirichlet function is the characteristic function of the set of rational numbers ($\mathbb{Q}$). Formally, the Dirichlet function is defined by

$$
\chi_{\mathbb{Q}}(x) = \begin{cases} 
1, & \text{if } x \in \mathbb{Q} \\
0, & \text{if } x \notin \mathbb{Q}
\end{cases}
$$

(8)

The range of Dirichlet function contains only two values 0 and 1. It is easy to see $X_1 = \chi_{\mathbb{Q}}^{-1}(1) = \mathbb{Q}$. Since the Lebesgue measure of the set of rational numbers in any interval is zero, we have $\mu(X_1) = \mu(\mathbb{Q}) = 0$. Consequently, $p_1 = 0$, and hence $p_1 = 1 - p_0 = 1$. Thus the entropy of the Dirichlet function is

$$
H(\chi_{\mathbb{Q}}) = -(p_0 \log p_0 + p_1 \log p_1) = 0
$$

(9)

**Example 3.2 (Modular arithmetic).** Modular arithmetic is one of the foundations of number theory, and is widely used in cryptography. The modular function $MOD: \mathbb{Z} \to \mathbb{Z}_m$ is defined by $MOD(x) = x(\text{mod } m)$, where the modulus $m$ is a positive integer number.

1. The image of $MOD: \mathbb{Z} \to \mathbb{Z}_m$ take integers between 0 and $m-1$.
2. For each integer $0 \leq k \leq m-1$, define the set $X_k = \{x \in \mathbb{Z} | x \equiv k(\text{mod } m)\}$.
   
   Since every integer belongs to one and only one residue class modulo $m$, the corresponding probability of a random input to fall into $X_k$ is $p_k = \frac{1}{m}$.
3. The entropy of the modular arithmetic is

$$
H(MOD) = -\sum_{k=0}^{m-1} p_k \log p_k = \log m
$$

(10)

As a special case, if $m = 2$, then $H(MOD) = 1$. □

It is worth emphasis that the entropy of a function is an attribute of this function. As such, the value of the entropy conveys important information of given function. On the other hand, the computation of the entropy may also need some important information about this function. The following two examples will illustrate this.

First we shall show that calculating the entropy of a Boolean function must be at least as hard as the Boolean satisfiability problem (SAT) problem. It is well known that the SAT problem is the first problem that was proven to be $NP$-complete, independently by Cook (1971) and Levin (1973). Recall that each propositional logic formula can be expressed as a Boolean function (see Shannon 1938, 1949b).

**Example 3.3 (Boolean function).** Boolean functions play a fundamental role in questions of computational complexity theory. Formally, a Boolean function is a function $BOOL: \{0,1\}^n \to \{0,1\}$. 
(1) The measure of its domain equals $\mu(\{0,1\}^n) = 2^n$, where $\mu$ is the counting measure. The range of a Boolean function contains only two values 0 and 1.

(2) For each value $k \in \{0,1\}$, define the set $X_k = \{x \in \{0,1\}^n \mid BOOL(x) = k\}$. Then the corresponding probability is $p_k = \frac{\mu(X_k)}{2^n}$. To be precise, $p_1$ is the probability of a random input $x \in \{0,1\}^n$ to satisfy $BOOL(x) = 1$.

(3) The entropy of a Boolean function is

$$H(BOOL) = -(p_0 \log p_0 + p_1 \log p_1)$$

(11)

It is worth emphasis that to calculate the probability $p_i$ we have to count the number of satisfying assignments of given Boolean formula. This means that we have to solve the #SAT problem, which must be at least as hard as the SAT problem in the worst case (see Goldreich 2008).

To see why, note that to solve the SAT problem, we are just asked to decide whether there is some input $x \in \{0,1\}^n$ satisfying $BOOL(x) = 1$. Even in the functional SAT problem, only one satisfying assignment is needed to be found, rather than finding out all.

Example 3.4 (Euler’s Totient Function). In number theory, Euler’s totient function $\phi(x)$ counts the positive integers up to a given number $x$ that are relatively prime to $x$. Formally, $\phi(x)$ is defined as the number of integers $k \leq x$ for which the greatest common divisor $\gcd(x,k) = 1$.

(1) Approximate the domain of $\phi(x)$ by the set of natural numbers between 1 and $n$. Then the range of $\phi(x)$ consists of totient numbers up to $n$. So the counting measure of the range of $\phi(x)$ coincides with the number of totient numbers up to $n$, that is equal to $\frac{n}{\log n} \exp((C + o(1))(\log \log \log n)^2)$ for a constant $C = 0.8178143\ldots$ (see Ford 1998).

(2) For each totient number $k$ between 1 and $\phi(n)$,

(a) Define the measurable sets $X_k = \{x \leq n \mid \phi(x) = k\}$. It is easy to see that the counting measure of $X_k$ equals the multiplicity of the totient number $k$.

(b) Calculate the corresponding probability $p_k = \frac{\mu(X_k)}{n}$, which is the probability of a random integer $x \leq n$ to satisfy $\phi(x) = k$.

The difficulty in computing the entropy of Euler’s totient function lies in the fact that the behavior of the multiplicity of totient numbers is not clear (see Ford 1999). Indeed, there is a famous unsolved problems concerning the multiplicity of totient numbers, namely, Carmichael’s Conjecture. In 1907, Carmichael Conjectured that for every $k$, the equation $\phi(x) = k$ has either no solutions or at least two solutions. In other words, no totient can have multiplicity 1. In our
notations, Carmichael’s Conjecture asserts $\mu(X_k) \geq 2$ for all totient number $k$. Since Carmichael’s Conjecture remains an open problem (see Ford 1999), so far it is impossible to accurately calculate the entropy of Euler’s totient function. □

3.2 Information Amount

The entropy of a function characterizes the uncertainty about the image $f(x)$ of random input $x$. In other words, the entropy of a function $f : X \rightarrow \mathbb{R}$ is the average amount of information needed for specifying the value of a random input $x \in X$.

But, given a measurable function $f : X \rightarrow \mathbb{R}$, in complexity theory we are just interested with the amount of information associated with specifying some particular values. For example, to decode a Public-key cryptography, we have to invert the enciphering function $y = f(x)$ given a cipher text $y_k$. Obviously, the given cipher text $y_k$ will convey some information about the corresponding preimage(s). Indeed, given the enciphering function $y = f(x)$ and arbitrary cipher text $y_k$, the goal of cryptanalyst is to gain as much information as possible about the plaintext $x \in f^{-1}(y_k)$. However, for cryptographic purposes, it may be possible that one can decrypt one arbitrary cipher text $y_k$ without inverting the enciphering function $f$ directly. Intuitively, decrypting one cipher text $y_k$ may not be computationally equivalent to inverting the enciphering function $f$, which means decrypting all cipher texts. To overcome this logical difficulty, we have to make good use of the information conveyed by the given value $y_k$.

Formally, let $(X, \mu)$ be a measurable space with measure $\mu(X) < \infty$ and $f : X \rightarrow \mathbb{R}$ be a simple function with values $y_1, y_2, \ldots, y_n$. Then the self-information associated with $\{f = y_k\}$ is equal to $I_k(f) = -\log p_k$, where $p_k = Pr(f = y_k) = \frac{\mu(f^{-1}(y_k))}{\mu(X)}$. The total amount of information contained in the function $f : X \rightarrow \mathbb{R}$ with values $y_1, y_2, \ldots, y_n$ is defined to be the sum $I(f) = \sum_{k=1}^{n} I_k(f) = -\sum_{k=1}^{n} \log p_k$.

Example 3.5 (Integer Factorization). The integer factorization problem requires inverting the integer multiplication function. Formally, the integer multiplication function is a binary operation on the set of natural numbers defined by $M(u, v) = uv$, where $u, v \in \mathbb{N}$ are natural numbers.

(1) Approximate the range of $M(u, v)$ by the set of natural numbers between 1 and $n$.
(2) For each value $k$ between 1 and $n$,
(a) Define the measurable sets \( X_k = \{(u,v) \in \mathbb{N} \times \mathbb{N} | uv = k\} \). It is easy to see that the counting measure of \( X_k \) equals the number of ways that the integer \( k \) can be written as a product of two integers. In number theory, the number of divisors of an integer \( k \) is usually denoted as \( d(k) \), the divisor function. That is, \( \mu(X_k) = d(k) \). Summing over all divisor function, we get the divisor summatory function \( D(n) = \sum_{k=1}^{n} d(k) \). In big-O notation, Dirichlet demonstrated that \( D(n) = n \log n + n(2\gamma - 1) + O(\sqrt{n}) \), where \( \gamma = 0.5772\cdots \) is the Euler’s gamma constant. Improving the bound \( O(\sqrt{n}) \) in this formula is known as Dirichlet’s Divisor Problem (see Hua 1982).

(b) Calculate the corresponding probability \( p_k = \frac{\mu(X_k)}{\mu(X)} = \frac{d(k)}{D(n)} \), which is the probability of a random integer pair \((u,v)\) below the hyperbola \( uv = n \) to satisfy \( uv = k \).

(3) The self-information associated with factoring \( n \) is \( I_1(M) = -\log p_n = \log \frac{D(n)}{d(n)} \).

The difficulty in estimating the self-information associated with factoring \( n \) is that the behavior of the divisor function \( d(n) \) is irregular (see Hua Lo-keng 1982). In the extreme case, if \( n \) is a semiprime with only two prime factors, then \( d(n) = 4 \) and the self-information associated with factoring \( n \) is \( I_1(M) \approx \log \frac{n \log n}{4} \).

However, a lower bound can be obtained by invoking the following property of the divisor function: for all \( \varepsilon > 0 \), the divisor function satisfies the inequality \( d(n) < n^{\varepsilon} \) (see Hua 1982). As a result, a lower bound of the self-information associated with factoring \( n \) is given by \( \log(n^{1-\varepsilon} \log n) \).

3.3 Conditional Entropy

To solve the search problem \( f(x) = y_k \) we have to make good use of the information hidden in the function \( y = f(x) \) and the information conveyed by the given value \( y_k \).

To illustrate, consider the famous subset sum problem, which can be stated as follows: given a set of non-zero integers \( a_1, a_2, \cdots, a_n \in \mathbb{Z} \), is there a non-empty subset that adds up to 0 (see Karp 1972)? The corresponding subset sum function is given by

\[
SUM(x) = a_1 x_1 + a_2 x_2 + \cdots + a_n x_n, x_i \in \{0,1\}, i = 1,2, \cdots, n.
\] (12)
In terms of the subset sum function, the subset sum problem amounts to the existence of some \( x \in \{0,1\}^n \) such that \( \text{SUM}(x) = 0 \). But if \( a_1 + a_2 + \cdots + a_n \neq 0 \), then the assumption of \( \text{SUM}(x) = 0 \) implies that the components of \( x \) can not be all positive. As such, the search space may be dramatically narrowed down. However, due to lack of information about the distribution of the conditional probabilities under the condition \( \text{SUM} = 0 \), it seems difficult to analyse the conditional entropy in detail theoretically (see Yao 1982 for discussions).

In theory, the information conveyed by the given value \( y_k \) enables us to reduce a search problem to the corresponding decision problem (see Rich 2007; Goldreich 2008).

**Search problem.** Given the function relation \( f : X \to \mathbb{R} \) and arbitrary value \( y_k \in Y \), find some \( x \in X \) such that \( f(x) = y_k \).

**Decision problem.** Given the function relation \( f : X \to \mathbb{R} \) and a value \( y_k \in \mathbb{R} \), decide whether there exists \( x \in X \) satisfying \( f(x) = y_k \).

Decision problems are important in that most well-studied complexity classes are defined as classes of decision problems. But, this kind of reduction may not make full of the information hidden in the function \( y = f(x) \).

To illustrate, consider the optimization problem of a real-valued function \( y = f(x) \) on an open interval \((a,b)\). Fermat’s theorem states that if \( f(x) \) is differentiable and has a local extremum at \( x_0 \in (a,b) \), then its derivative \( f'(x) \) must satisfy \( f'(x_0) = 0 \). By using Fermat’s theorem, the local extremum of \( f(x) \) is found by solving an equation \( f'(x_0) = 0 \). As a result, the information of differentiability enable us to reduce the optimization problem \( \min_{x \in (a,b)} f(x) \) to another search problem \( f'(x_0) = 0 \) (see Fig. 2).

In some idea cases, the information hidden in the function \( f : X \to \mathbb{R} \) may enable us to transform the search problem \( f(x) = y_k \) into another search problem \( g(x) = z_i \) (see Karp 1972 for examples). As such, to quantify the average amount of information about \( f(x) \) obtained from \( g(x) \) we have to calculate the conditional entropy \( H(f \mid g) \) (see Shannon 1948).

To this end, approximate given function \( f : X \to \mathbb{R} \) by a simple function taking values \( y_0, y_1, \cdots, y_n \) and approximate \( g : Y \to \mathbb{R} \) by a simple function with values \( z_0, z_1, \cdots, z_m \). The corresponding probabilities \( \Pr(f = y_k) \) and \( \Pr(g = z_i) \) are calculated in the same way as described in section 3.1.

Firstly, consider the entropy of \( H(f \mid g = z_i) \) conditioned on \( g(x) \) taking the value \( z_i \).
\[ H(f \mid g = z_j) = \sum_{k=1}^{n} \Pr(f = y_k \mid g = z_j) \log \frac{1}{\Pr(f = y_k \mid g = z_j)}. \] (13)

Secondly, the condition entropy \( H(f \mid g) \) is the weighted sum of \( H(f \mid g = z_j) \) for each possible value of \( g(x) \)

\[
H(f \mid g) = \sum_{j=1}^{m} \Pr(g = z_j) H(f \mid g = z_j)
\]

\[
= \sum_{j=1}^{m} \Pr(g = z_j) \sum_{k=1}^{n} \Pr(f = y_k \mid g = z_j) \log \frac{1}{\Pr(f = y_k \mid g = z_j)}.
\] (14)

Example 3.6 (The class \( \mathcal{RP} \)). One of the important questions in complexity theory is whether randomized or probabilistic algorithms are more powerful than their deterministic counterparts (see Papadimitriou 1993). The most famous concrete formulation of this question regards the power of the class \( \mathcal{RP} \), the set of decision problems that have Randomized, Polynomial time algorithms (see Rich 2007, section 30.2; Papadimitriou 1993).

For convenience, we shall define decision problems in terms of formal languages. Formally, a language \( L \) over the alphabet \( \{0, 1\} \) belongs to \( \mathcal{RP} \) if and only if there exists some randomized Turing machine that runs in polynomial time for deciding \( L \) such that: if \( w \notin L \) then all computation halts with rejection, if \( w \in L \) then \( M \) accepts \( w \) with probability \( 1 - \varepsilon \) for arbitrarily small constant \( 0 < \varepsilon < \frac{1}{2} \). It is worth emphasis that \( \varepsilon \) must be independent of the input to the algorithm.

It is clear that the class \( \mathcal{RP} \) lies somewhere between \( \mathcal{P} \) and \( \mathcal{NP} \) (see Papadimitriou 1993). One of the most famous problems that was known to be in \( \mathcal{RP} \) is the problem of determining whether a given number is (not) a prime number. However, Agrawal, Kayal and Saxena (2004) have shown that Primality test turns out to be in \( \mathcal{P} \) (see section 4.2 for details). Indeed, Now it is widely conjectured, but unproven, that \( \mathcal{RP} \) (see Papadimitriou 1993).

Traditionally, the fundamental paradigm in derandomization is to trade hardness for randomness (see Yao 1982). Based on the hardness-randomness tradeoffs, it is widely conjectured that efficient probabilistic algorithm can be transformed into an efficient deterministic algorithm without a significant increase in the running time (see Yao 1982).

However, since randomness means uncertainty, it turns out that our definition of entropy is a natural setting in which to study randomization. To see this, take a language \( L \subseteq \{0, 1\}^* \) as fixed and consider its characteristic function
\[
\chi_L(w) = \begin{cases} 
1, & \text{if } w \in L \\
0, & \text{if } w \notin L.
\end{cases}
\] (15)

Let \(p_1\) be the probability of acceptance, and \(p_0 = 1 - p_1\) be the probability of rejection. Then the entropy of the given language \(L\) equals
\[
H(\chi_L) = -(p_0 \log p_0 + p_1 \log p_1).
\]

On the other hand, the definition of the class \(RP\) gives rise to a function \(\chi^e_L\) from \(\{0,1\}^*\) to the set \(\{0,1\}\), with probabilities \(\Pr(\chi^e_L = 1) = p_1(1 - \varepsilon)\) and \(\Pr(\chi^e_L = 0) = p_0 + p_1 \varepsilon\). So the entropy of \(\chi^e_L\) is
\[
H(\chi^e_L) = -[(p_0 + p_1 \varepsilon) \log(p_0 + p_1 \varepsilon) + p_1(1 - \varepsilon) \log p_1(1 - \varepsilon)].
\] (16)

The language \(L\) in the class \(RP\) yields the following conditional probabilities
\[
\begin{align*}
\Pr(\chi^e_L = 0 | \chi_L = 0) &= 1, & \Pr(\chi^e_L = 1 | \chi_L = 0) &= 0, \\
\Pr(\chi^e_L = 0 | \chi_L = 1) &= \varepsilon, & \Pr(\chi^e_L = 1 | \chi_L = 1) &= 1 - \varepsilon.
\end{align*}
\] (17)

Then the conditional entropy \(H(\chi^e_L | \chi_L)\) is by definition equal to
\[
\begin{align*}
H(\chi^e_L | \chi_L) &= \Pr(\chi_L = 0) H(\chi^e_L | \chi_L = 0) + \Pr(\chi_L = 1) H(\chi^e_L | \chi_L = 1) \\
&= \Pr(\chi_L = 0) \Pr(\chi^e_L = 0 | \chi_L = 0) \log \frac{1}{\Pr(\chi^e_L = 0 | \chi_L = 0)} \\
&\quad + \Pr(\chi_L = 0) \Pr(\chi^e_L = 1 | \chi_L = 0) \log \frac{1}{\Pr(\chi^e_L = 1 | \chi_L = 0)} \\
&\quad + \Pr(\chi_L = 1) \Pr(\chi^e_L = 0 | \chi_L = 1) \log \frac{1}{\Pr(\chi^e_L = 0 | \chi_L = 1)} \\
&\quad + \Pr(\chi_L = 1) \Pr(\chi^e_L = 1 | \chi_L = 1) \log \frac{1}{\Pr(\chi^e_L = 1 | \chi_L = 1)} \\
&= -p_1[\varepsilon \log \varepsilon + (1 - \varepsilon) \log(1 - \varepsilon)].
\end{align*}
\] (18)

It is easy to see that \(\lim_{\varepsilon \to 0} H(\chi^e_L | \chi_L) = 0\).  

### 3.4 Mutual Information

Now, assume that there is a Turing reduction from the search problem \(f(x) = y_k\) to another search problem \(g(x) = z_i\) (see Karp 1972 for examples). Generally, this reduction is enabled by the information hidden in the function \(f : X \to \mathfrak{R}\). As such, to estimate of the complexity of solving \(f(x) = y_k\) we have to measure
the Mutual Information between \( f(x) \) and \( g(x) \) (see Thomas M. Cover, Joy A. Thomas 2006).

To this end, define the pointwise mutual information to be

\[
I(f = y_k; g = z_i) = \log \frac{\Pr(f = y_k, g = z_i)}{\Pr(f = y_k) \Pr(g = z_i)}
\]

\[
= \log \frac{\Pr(f = y_k \mid g = z_i)}{\Pr(g = z_i)}
\]

Then the mutual information of \( f(x) \) and \( g(x) \) is by definition equal to the expected value of these pointwise mutual information, that is

\[
I(f; g) = \sum_k \sum_i \Pr(f = y_k, g = z_i) I(f = y_k; g = z_i)
\]

\[
= \sum_k \sum_i \Pr(f = y_k, g = z_i) \log \frac{\Pr(f = y_k, g = z_i)}{\Pr(f = y_k) \Pr(g = z_i)}.
\]  

Intuitively, mutual information measures the information that \( f(x) \) and \( g(x) \) share. For example, if \( f(x) \) and \( g(x) \) take values independently, then knowing values of \( g(x) \) does not give any information about \( f(x) \) and vice versa, so their mutual information is zero.

**Example 3.7 (Fermat’s Theorem).** Given a differentiable function \( y = f(x) \) on an open interval \((a, b)\), Fermat’s theorem asserts that the optimization problem of \( f(x) \) can be reduced to the search problem \( f'(x_0) = 0 \). We shall calculate the mutual information \( I(f; f') \).

Assume that the objective function \( f(x) \) has \( r + 1 \) local extrema and denote them to be \( y_0, y_1, \ldots, y_r \). Similarly, denote the derivative at the stationary point to be \( z_0 = f'(x_0) = 0 \). Then Fermat’s theorem says that the conditional probability \( \Pr(f' = z_0 \mid f = y_k) = 1 \) for all \( k = 0, 1, \ldots, r \). As for \( k > r \) and \( i > 0 \), we have

\[
\Pr(f = y_k, f' = z_i) = \Pr(f = y_k) \Pr(f' = z_i).
\]  

So the pointwise mutual information for \( k > r \) and \( i > 0 \) satisfy

\(^1\) If \( f(x) \) is continuous, then the event \( \{ f = y_k \} \) should be replaced by \( \{ | f - y_k | < \varepsilon \} \) for suitable \( \varepsilon > 0 \). But in the following we still use \( \{ f = y_k \} \) in order to simplify notations.
Information Measure as Computational Complexity

\[ I(f = y_k; f' = z_i) = \log \frac{\Pr(f = y_k, g = z_i)}{\Pr(f = y_k) \Pr(g = z_i)} = 0. \] (22)

As a result, the mutual information of \( f(x) \) and \( f'(x) \) is by definition equal to

\[ I(f; f') = \sum_{k=1}^{r} \sum_{i=1}^{m} \Pr(f = y_k, f' = z_i) \log \frac{\Pr(f = y_k, f' = z_i)}{\Pr(f = y_k) \Pr(f' = z_i)}. \]

Specially, if \( r = 0 \) then \( y_0 \) is the only global extremum of \( f(x) \). In this case the mutual information of \( f(x) \) and \( f'(x) \) becomes into

\[ I(f; f') = \Pr(f = y_0) \log \frac{1}{\Pr(f' = 0)}. \] (24)

4. TIME COMPLEXITY

Each algorithm uses a sequence of elementary operations to complete the desired work, no matter how complex the algorithm is. With regards to a Turing machine, an elementary operation can be defined as one move of its tape (see Turing 1948). Indeed, in his 1948 essay Turing wrote that: “...However, the tape can be moved back and forth through the machine, this being one of the elementary operations of the machine”. The time complexity of an algorithm is a measure of how many times the tape moves when the machine is started on some input.

![Turing Machine Diagram](image-url)

**Fig. 4. A Turing Machine.**

On the other hand, the amount of information associated with one elementary operation is equal to the additional amount of information required to specify the state of the tape after one move of it. If we assume that one elementary operation yields one unit of information on a Turing machine, then the time
complexity coincides with the total amount of information to be obtained (see Shah and Sharma 2000 for similar idea).

In fact, the same idea has been used by Shannon (1949) to analyze the practical secrecy of cryptosystem. Further, Shannon (1949) pointed out that the time complexity of breaking a cryptosystem is similar to the coin weighing problem (see also Erdos and Renyi 1963). Essentially, the time complexity of decoding a cryptosystem is analogous to the work done against the gravitational force. To raise an object of mass $m$ upward a vertical height $h$, the work done against the gravitational force can not be less than $mgh$, where $g$ is the gravitational acceleration (see Feynman et al. 2013).

In view of this, the question of time complexity is to measure the amount of information contained in given function $y = f(x)$. As such, the time complexity of solving a search problem $f(x) = y_k$ can be measured in terms of the amount of information associated with the function in question.

### 4.1 Worst-case Complexity

To be precise, assume there is a Turing reduction from given search problem $f(x) = y_k$ to another search problem $g(x) = z_i$ (see Goldreich 2008). Then the search problem $f(x) = y_k$ can be solved by querying the solutions of $g(x) = z_i$. Further, if each query can be done in polynomial time, then to estimate the worst-case complexity of solving $f(x) = y_k$ we just need to estimate the expected number of queries. Obviously, to achieve the best performance, any algorithm for solving $f(x) = y_k$ must consider the tradeoff between the number of queries and the complexity of each query.

Now that the self-information associated with $\{f = y_k\}$ is equal to

$$I_k(f) = -\log p_k,$$

where $p_k = \Pr(f = y_k)$, and the average amount of information about $f(x) = y_k$ provided by each query is exactly equal to the average mutual information $I(f;g)$, the least number of queries needed by solving $f(x) = y_k$ will be

$$\frac{I_k(f)}{I(f;g)} = \frac{-\log \Pr(f = y_k)}{\sum_{k=1}^{n} \sum_{m=1}^{m} \Pr(f = y_k, g = z_i) \log \frac{\Pr(f = y_k, g = z_i)}{\Pr(f = y_k) \Pr(g = z_i)}}. \quad (25)$$

In theory, $\frac{I_k(f)}{I(f;g)}$ gives an upper bound to the time complexity of solving $f(x) = y_k$.

On average, the expected number of queries needed to solve $f(x) = y_k$ for randomly chosen value $y_k$ is $\frac{H(f)}{I(f;g)}$. Since $I(f;g) \leq \min(H(f), H(g))$, we
always have \( \frac{H(f)}{I(f;g)} \geq 1 \), with equality only if \( I(f;g) = H(f) \). In such a case the conditional entropy satisfies \( H(f | g) = H(g | f) = 0 \).

In some idea cases, there may be a Turing reduction from given search problem \( f(x) = y_i \) to another search problem \( g(x) = z_i \) such that \( \frac{H(f)}{I(f;g)} \) is polynomial in the maximal size of inputs. In this case if the search problem \( g(x) = z_i \) can be solved in polynomial time, then so is the search problem \( f(x) = y_i \).

As it turns out, our information-based complexity theory is a natural setting in which to study the power of randomized or probabilistic algorithms of decision problems. To justify the usefulness of our framework, we shall consider the complexity of the class \( BPP \), consisting of decision problems that have Bounded-error, Probabilistic, Polynomial time algorithms (see Rich 2007, section 30.2; Papadimitriou 1993). Indeed, \( BPP \) is one of the largest classes of problems of practical importance, meaning most problems of interest in \( BPP \) has efficient probabilistic algorithms. But there are two big unknowns concerning the class \( BPP \). One unknown is the relationship between \( BPP \) and \( NP \). The other unknown is whether \( P = BPP \), which is widely conjectured to be true.

**Example 4.1 (The class \( BPP \).** Formally, a language \( L \) over the alphabet \( \{0,1\} \) belongs to the class \( BPP \) if and only if there exists some probabilistic Turing machine that runs in polynomial time on all inputs and that decides \( L \) with false probability \( \varepsilon \) for arbitrarily small constant \( 0 < \varepsilon < \frac{1}{2} \). It is worth noting that the definition of \( BPP \) is symmetric.

Now assume that the characteristic function of \( L \) is \( \chi_L \) with the probability of acceptance \( p_1 \) and the probability of rejection \( p_0 \). Then the entropy of the given language \( L \) equals \( H(\chi_L) = -(p_0 \log p_0 + p_1 \log p_1) \).

Now \( L \in BPP \) means that there are two kinds of error: to accept words that should be rejected (0 to 1), or to reject words that should be accepted (1 to 0). This gives rise to a \( \{0,1\}\)-valued function \( \chi^\varepsilon_L \) on \( \{0,1\}^* \), with probabilities \( \Pr(\chi^\varepsilon_L = 0) = p_0(1 - \varepsilon) + p_1 \varepsilon \) and \( \Pr(\chi^\varepsilon_L = 1) = p_0 \varepsilon + p_1(1 - \varepsilon) \). So the entropy of the function \( \chi^\varepsilon_L \) is

\[
H(\chi^\varepsilon_L) = [p_0(1 - \varepsilon) + p_1 \varepsilon] \log \frac{1}{p_0(1 - \varepsilon) + p_1 \varepsilon} + [p_0 \varepsilon + p_1(1 - \varepsilon)] \log \frac{1}{p_0 \varepsilon + p_1(1 - \varepsilon)} (26)
\]

We shall calculate the average mutual information via

\[
I(\chi_L; \chi^\varepsilon_L) = H(\chi^\varepsilon_L) - H(\chi^\varepsilon_L | \chi_L). \tag{27}
\]
To do this, we need to compute the conditional entropy $H(\chi^e_L \mid \chi_L)$, which in turn needs the following conditional probabilities

$$
\begin{align*}
\Pr(\chi^e_L = 0 \mid \chi_L = 0) &= 1 - \varepsilon, \\
\Pr(\chi^e_L = 1 \mid \chi_L = 0) &= \varepsilon, \\
\Pr(\chi^e_L = 0 \mid \chi_L = 1) &= \varepsilon, \\
\Pr(\chi^e_L = 1 \mid \chi_L = 1) &= 1 - \varepsilon.
\end{align*}
$$

So the conditional entropy $H(\chi^e_L \mid \chi_L)$ is by definition equal to

$$
H(\chi^e_L \mid \chi_L) = \Pr(\chi_L = 0)H(\chi^e_L \mid \chi_L = 0) + \Pr(\chi_L = 1)H(\chi^e_L \mid \chi_L = 1)
$$

$$
= \Pr(\chi_L = 0)\Pr(\chi^e_L = 0 \mid \chi_L = 0)\log \frac{1}{\Pr(\chi^e_L = 0 \mid \chi_L = 0)}
$$

$$
+ \Pr(\chi_L = 0)\Pr(\chi^e_L = 1 \mid \chi_L = 0)\log \frac{1}{\Pr(\chi^e_L = 1 \mid \chi_L = 0)}
$$

$$
+ \Pr(\chi_L = 1)\Pr(\chi^e_L = 0 \mid \chi_L = 1)\log \frac{1}{\Pr(\chi^e_L = 0 \mid \chi_L = 1)}
$$

$$
+ \Pr(\chi_L = 1)\Pr(\chi^e_L = 1 \mid \chi_L = 1)\log \frac{1}{\Pr(\chi^e_L = 1 \mid \chi_L = 1)}
$$

$$
= p_0[(1 - \varepsilon)\log \frac{1}{1 - \varepsilon} + \varepsilon \log \frac{1}{\varepsilon}] + p_1[(1 - \varepsilon)\log \frac{1}{1 - \varepsilon} + \varepsilon \log \frac{1}{\varepsilon}]
$$

$$
= (1 - \varepsilon)\log \frac{1}{1 - \varepsilon} + \varepsilon \log \frac{1}{\varepsilon}
$$

As a result, the mutual information of $\chi_L$ and $\chi^e_L$ equals

$$
I(\chi_L; \chi^e_L) = H(\chi^e_L) - H(\chi^e_L \mid \chi_L)
$$

$$
= [p_0(1 - \varepsilon) + p_1\varepsilon]\log \frac{1}{p_0(1 - \varepsilon) + p_1\varepsilon}
$$

$$
+ [p_0\varepsilon + p_1(1 - \varepsilon)]\log \frac{1}{p_0\varepsilon + p_1(1 - \varepsilon)}
$$

$$
- [(1 - \varepsilon)\log \frac{1}{1 - \varepsilon} + \varepsilon \log \frac{1}{\varepsilon}]
$$

Note that the mutual information $I(\chi_L; \chi^e_L)$ is completely determined by the acceptance probability $p_1$, which is an attribute the language, and the false probability $\varepsilon$, which is an attribute of the probabilistic Turing machine. As a result, $I(\chi_L; \chi^e_L)$ is a constant for a given language $L \in BPP$. 
By the **CONTINUITY** properties of the entropy measure, we have
\[
\lim_{\varepsilon \to 0} I(\chi_L; \chi_L^\varepsilon) = H(\chi_L).
\]
Consequently, given language \( L \in BPP \), the expected number of queries
\[
\frac{H(\chi_L)}{I(\chi_L; \chi_L^\varepsilon)}
\]
approaches 1 as \( \varepsilon \to 0 \). So the time complexity of deciding \( L \) is equal to \( \frac{H(\chi_L)}{I(\chi_L; \chi_L^\varepsilon)} \) times the time complexity of each query. Since \( L \in BPP \), each query can be done within polynomial time. As a result, any language \( L \) in the class \( BPP \) can be decided by a deterministic Turing machine within polynomial time. \( \square \)

**Example 4.2 (The class \( PP \)).** A language \( L \) over the alphabet \( \{0,1\} \) belongs to the class \( PP \) if and only if there exists some probabilistic Turing machine \( M \) that runs in polynomial time on all inputs and that decides \( L \) with false probability of less than \( \frac{1}{2} \). That is, if \( w \notin L \) then \( M \) rejects \( w \) with probability \( \frac{1}{2} + \varepsilon \), and if \( w \in L \) then \( M \) accepts \( w \) with probability \( \frac{1}{2} + \varepsilon \), where \( 0 < \varepsilon < \frac{1}{2} \) may depend on the size of inputs.

Now assume the characteristic function of \( L \) to be \( \chi_L \) with the probability of acceptance \( p_1 \) and the probability of rejection \( p_0 \). Then the entropy of the given language \( L \) equals \( H(\chi_L) = -(p_0 \log p_0 + p_1 \log p_1) \).

ON the other hand, \( L \in PP \) gives rise to a function \( \chi_L^\varepsilon \) from \( \{0,1\}^* \) to the set \( \{0,1\} \) with probabilities
\[
\Pr(\chi_L^\varepsilon = 0) = p_0 \left(\frac{1}{2} + \varepsilon\right) + p_1 \left(\frac{1}{2} - \varepsilon\right) = \frac{1}{2} + \varepsilon(p_0 - p_1)
\]
\[
\Pr(\chi_L^\varepsilon = 1) = p_0 \left(\frac{1}{2} - \varepsilon\right) + p_1 \left(\frac{1}{2} + \varepsilon\right) = \frac{1}{2} + \varepsilon(p_1 - p_0) \quad \text{(31)}
\]
So the entropy of the function \( \chi_L^\varepsilon \) is
\[
H(\chi_L^\varepsilon) = \left[\frac{1}{2} + \varepsilon(p_0 - p_1)\right] \log \frac{1}{\frac{1}{2} + \varepsilon(p_0 - p_1)} + \left[\frac{1}{2} + \varepsilon(p_1 - p_0)\right] \log \frac{1}{\frac{1}{2} + \varepsilon(p_1 - p_0)} \quad \text{(32)}
\]

The language \( L \in PP \) yields the following conditional probabilities
\[
\Pr(\chi_L^\varepsilon = 0 | \chi_L = 0) = \frac{1}{2} + \varepsilon, \quad \Pr(\chi_L^\varepsilon = 1 | \chi_L = 0) = \frac{1}{2} - \varepsilon,
\]
\[
\Pr(\chi_L^\varepsilon = 0 | \chi_L = 1) = \frac{1}{2} - \varepsilon, \quad \Pr(\chi_L^\varepsilon = 1 | \chi_L = 1) = \frac{1}{2} + \varepsilon. \quad \text{(33)}
\]
So the conditional entropy $H(\chi_L^c | \chi_L)$ is by definition equal to

$$H(\chi_L^c | \chi_L) = \Pr(\chi_L = 0)H(\chi_L^c | \chi_L = 0) + \Pr(\chi_L = 1)H(\chi_L^c | \chi_L = 1)$$

$$= \frac{1}{\Pr(\chi_L = 0)} \log \frac{1}{\Pr(\chi_L = 0)} + \frac{1}{\Pr(\chi_L = 1)} \log \frac{1}{\Pr(\chi_L = 1)}$$

$$= p_0[(\frac{1}{2} + \epsilon) \log \frac{1}{2 + \epsilon} + (\frac{1}{2} - \epsilon) \log \frac{1}{2 - \epsilon}]$$

$$+ p_1[(-\frac{1}{2} - \epsilon) \log \frac{1}{2 - \epsilon} + (\frac{1}{2} + \epsilon) \log \frac{1}{2 + \epsilon}]$$

$$= (\frac{1}{2} - \epsilon) \log \frac{1}{2 - \epsilon} + (\frac{1}{2} + \epsilon) \log \frac{1}{2 + \epsilon}.$$  \hspace{1cm} (34)

Now that $\lim_{\epsilon \to 0} H(\chi_L^c) = \lim_{\epsilon \to 0} H(\chi_L^c | \chi_L) = 1$, the mutual information of $\chi_L$ and $\chi_L^c$ satisfies

$$\lim_{\epsilon \to 0} I(\chi_L; \chi_L^c) = \lim_{\epsilon \to 0} H(\chi_L^c) - \lim_{\epsilon \to 0} H(\chi_L^c | \chi_L) = 0.$$  \hspace{1cm} (35)

In such a case, each query of $\chi_L^c$ reveals nearly no information about the language. This means that in the worst case an exponential number of repetitions of the probabilistic algorithm may be required in order to determine the correct answer with reasonable confidence (see Papadimitriou 1994).

To see this, note that $\log(1 + x) \approx x$ for $|x| < 1$. Consequently, it is routine to check that
Information Measure as Computational Complexity

\[ H(\chi_L^c | \chi_L) = \left(\frac{1}{2} - \varepsilon\right) \log \frac{1}{2 - \varepsilon} + \left(\frac{1}{2} + \varepsilon\right) \log \frac{1}{2 + \varepsilon} \]

\[ = -\left[\left(\frac{1}{2} - \varepsilon\right) \log \frac{1}{2} (1 - 2\varepsilon) + \left(\frac{1}{2} + \varepsilon\right) \log \frac{1}{2} (1 + 2\varepsilon)\right] \]

\[ = 1 - \left[\left(\frac{1}{2} - \varepsilon\right) \log (1 - 2\varepsilon) + \left(\frac{1}{2} + \varepsilon\right) \log (1 + 2\varepsilon)\right] \]

\[ \approx 1 - \left[\left(\frac{1}{2} - \varepsilon\right) \times (-2\varepsilon) + \left(\frac{1}{2} + \varepsilon\right) \times (2\varepsilon)\right] \]

\[ = 1 - 4\varepsilon^2 \]

and

\[ H(\chi_L^c) = \left[\frac{1}{2} + \varepsilon(p_0 - p_1)\right] \log \frac{1}{2 + \varepsilon(p_0 - p_1)} + \left[\frac{1}{2} + \varepsilon(p_1 - p_0)\right] \log \frac{1}{2 + \varepsilon(p_1 - p_0)} \]

\[ = -\left[\frac{1}{2} + \varepsilon(p_0 - p_1)\right] \log \frac{1}{2} [1 + 2\varepsilon(p_0 - p_1)] - \left[\frac{1}{2} + \varepsilon(p_1 - p_0)\right] \log \frac{1}{2} [1 + 2\varepsilon(p_1 - p_0)] \]

\[ = 1 - \left[\left(\frac{1}{2} + \varepsilon(p_0 - p_1)\right) \log [1 + 2\varepsilon(p_0 - p_1)] - \left[\frac{1}{2} + \varepsilon(p_1 - p_0)\right] \log [1 + 2\varepsilon(p_1 - p_0)]\right] \]

\[ \approx 1 - \left[\frac{1}{2} + \varepsilon(p_0 - p_1)\right] \times 2\varepsilon(p_0 - p_1) - \left[\frac{1}{2} + \varepsilon(p_1 - p_0)\right] \times 2\varepsilon(p_1 - p_0) \]

\[ = 1 - 4\varepsilon^2 (p_0 - p_1)^2 \]

Consequently, we obtain an approximation of the average mutual information

\[ I(\chi_L; \chi_L^c) = H(\chi_L^c) - H(\chi_L^c | \chi_L) \approx 4\varepsilon^2 [1 - (p_0 - p_1)^2] \]. \quad (38) \]

To get the desired result, assume that \( \varepsilon = \frac{1}{2^n} \). In this case a word \( w \in L \) has an acceptance probability of \( \frac{1}{2} + \frac{1}{2^n} \), with just two more accepting computations than rejecting computations. Then the expected number of repetitions of the probabilistic algorithm is approximately equal to

\[ \frac{H(\chi_L)}{I(\chi_L; \chi_L^c)} \approx \frac{H(\chi_L)}{4\varepsilon^2 [1 - (p_0 - p_1)^2]} = \frac{H(\chi_L)}{1 - (p_0 - p_1)^2} \times 2^{2n-2}. \quad (39) \]

4.2 Average-case Complexity

Some NP-complete problems are easy on average with respect to a natural probability distribution on inputs. In this case, the average-case complexity of a problem is a more significant measure than its worst-case complexity (see Levin...
1986). For example, in cryptographic applications we want a guarantee that the average-case complexity of every algorithm which "breaks" the cryptographic scheme is inefficient.

Interestingly, within our framework it is easy to give a condition for inverting a candidate one-way function \( y = f(x) \) to be difficult in the average-case. To do this, consider any Turing reduction from given search problem \( f(x) = y_k \) to any other search problem \( g(x) = z_i \). Then the expected number of querying \( g(x) \) needed to solve \( f(x) = y_k \) is \( \frac{I_k(f)}{I(f;g)} \). At the same time, the expected number of querying \( g(x) \) needed by solving \( f(x) = y_k \) for randomly chosen value \( y_k \) is \( \frac{H(f)}{I(f;g)} \). So the worst-case complexity of inverting \( y = f(x) \) coincides with its average-case complexity if \( H(f) = I_k(f) \) for any value \( y_k \). In such a case, all the values of \( y = f(x) \) are equally possible.

As applications, we shall show that several candidate one-way functions are hard on average if the parameters are carefully chosen.

**Example 4.3 (Discrete Logarithm).** The discrete logarithm requires inverting a modular exponentiation function. Formally, a modular exponentiation function is \( EXP(x) = b^x \pmod{n} \), where the base \( b \geq 2 \) is a primitive root modulo \( n \). In other words, \( b \) is a generator of the multiplicative group of integers modulo \( n \).

1. Since \( b \) is a generator of the multiplicative group of integers modulo \( n \), \( b^x \pmod{n} \) ranges from 0 to \( n-1 \), and hence the measure of the range equals \( n \).
2. For value \( k \) between 0 and \( n-1 \), define the set \( X_k = \{ x \mid b^x \equiv k \pmod{n} \} \). Note that \( b \) is a primitive root modulo \( n \) implies that \( X_k \) contains only one element for each value \( k \) (see Hua 1982). Consequently, the probability of a random input to satisfy \( b^x \equiv k \pmod{n} \) is \( p_k = \frac{1}{n} \).
3. The entropy of modular exponentiation function is

   \[
   H(EXP) = -\sum_{k=0}^{n-1} p_k \log p_k = \log n \quad (40)
   \]

4. To compute discrete logarithm, we are required to find out \( x < n \) such that \( b^x \equiv k \pmod{n} \). The self-information associated with the event \( \{ EXP = k \} \) is

   \[
   I_k(EXP) = -\log p_k = \log n = H(EXP). \quad \square
   \]

**Example 4.4 (Rabin Function).** The Rabin function is defined by squaring modulo \( n = q_1 q_2 \), where \( q_1 \) and \( q_2 \) are distinct odd prime numbers. Formally,
Information Measure as Computational Complexity

\[ RABIN(x) = x^2 \mod n \]. It is well known that inverting the Rabin function is computationally equivalent to integer factorization in the sense of polynomial-time reduction (see Rabin 1979).

(1) It suffices to restrict the domain of \( RABIN(x) \) to a reduced residue system modulo \( n \). So the counting measure of the domain of \( RABIN(x) \) equals \( \varphi(n) = (q_1 - 1)(q_2 - 1) \), where \( \varphi(\cdot) \) is the Euler's totient function. The range of \( RABIN(x) \) consists of all quadratic residue modulo \( n \), and hence the measure of the range equals \( \frac{\varphi(n)}{4} \) (see Hua 1982).

(2) For each quadratic residue \( y_k \), define the set \( X_k = \{ x \mid x^2 \equiv y_k \mod n \} \). Since \( n = q_1q_2 \) is a semiprime, \( X_k \) exactly has 4 elements (see Hua 1982). Consequently, the corresponding probability is \( p_k = \frac{4}{\varphi(n)} \).

(3) The entropy of Rabin function is

\[
H(RABIN) = -\sum_{k=0}^{\varphi(n)} p_k \log p_k = \log \frac{\varphi(n)}{4}
\] (41)

(4) To decode the Rabin cryptosystem, we are required to find out \( x < n \) such that \( x^2 \equiv y_k \mod n \). The self-information associated with the event \( \{ RABIN = y_k \} \) is \( I_k(RABIN) = -\log p_k = \log \frac{\varphi(n)}{4} = H(RABIN) \). □

Example 4.5 (RSA function). Formally, the RSA function is defined to be \( RSA(x) = x^e \mod n \), where \( n = q_1q_2 \), the product of distinct odd prime numbers, and \( e \geq 3 \) is relatively prime to \( \varphi(n) = (q_1 - 1)(q_2 - 1) \) (see Rivest, Shamir, and Adleman 1978).

(1) It suffices to restrict the domain of \( RSA(x) \) to a reduced residue system modulo \( n \). So the counting measure of the domain of \( RSA(x) \) equals \( \varphi(n) = (q_1 - 1)(q_2 - 1) \). The range of \( RSA(x) \) consists of all \( e \)-th power residue modulo \( n \), and hence the measure of the range equals \( \varphi(n) \).

(2) For each \( e \)-th residue \( y_k \), define the set \( X_k = \{ x < n \mid x^e \equiv y_k \mod n \} \). Since \( e \) is relatively prime to \( \varphi(n) \), the congruence equation \( x^e \equiv k \mod n \) has exactly 1 solution. Consequently, the corresponding probability is \( p_k = \frac{1}{\varphi(n)} \).

(3) The entropy of RSA function is

\[
H(RSA) = -\sum_{k=0}^{\varphi(n)} p_k \log p_k = \log \varphi(n)
\] (42)
(4) To decode the RSA cryptosystem, we are required to find out \( x < n \) such that \( x^e \equiv y_k \pmod{n} \). The self-information associated with the event \( \{RSA = y_k\} \) is \( I_k(RSA) = -\log p_k = \log \varphi(n) = IH(RSA) \).

But, the RSA cryptosystem would be broken if the number \( n \) could be factored or if \( \varphi(n) \) could be computed without factoring \( n \). Indeed, the RSA function is widely conjectured to be trapdoor one-way (see Diffie and Hellman 1976). The trapdoor is the information about \( d \), the multiplicative inverse of \( e \) modulo \( \varphi(n) \). In fact, the prime factors \( q_1, q_2 \) and the totient number \( \varphi(n) \) are also part of the trapdoor information because they can be used to calculate \( d \). If the factorization \( n = q_1 q_2 \) is known, Euler's totient function \( \varphi(n) = (q_1-1)(q_2-1) \) can be computed, then we can solve for \( d \) given \( ed \equiv 1 \pmod{\varphi(n)} \). Using this trapdoor information, we can easily recover \( x \) from \( y_k = x^d \pmod{n} \) via computing \( x = y_k^d \pmod{n} \). 

5. INFORMATION MEASURE OF EQUATIONS

In this section, we roughly discuss the amount of information contained in an equation. Further studies of this topic will be one direction of future researches.

Since the information of solutions of given equation must be contained in the equation itself, the information measure of equations may also be related with other important problems.

**Example 5.1 (Differential equations).** Consider a differential equation of the first order

\[
\frac{dy}{dx} = f(x). \tag{43}
\]

If \( f(x) \) is Lebesgue measurable, then this differential equation determines one antiderivative \( y = F(x) \), up to a constant of integration (see Patrick Billingsley 1995). In this case, the entropy of the derivative \( f(x) \) is well-defined and can be calculated by partitioning the range of \( f(x) \) in the same way as computing its Lebesgue integral (see section 3.1). In turn, the entropy of the derivative \( f(x) \) contains important information about the antiderivative \( F(x) \).

On the other hand, Liouville’s theorem states that the antiderivatives of certain elementary functions cannot themselves be expressed as elementary functions. A typical example is when \( f(x) = e^{-x^2} \). Within our framework, it seems that this phenomenon may be attributed to the amount of information contained in the antiderivative in that being an elementary function places an important restriction on the amount of information contained in it.

**Example 5.2 (Algebraic equations).** A polynomial equation of degree \( n \) is an equation of the form
Information Measure as Computational Complexity

\[ a_n x^n + a_{n-1} x^{n-1} + \cdots + a_1 x + a_0 = 0. \quad (44) \]

The fundamental theorem of algebra states that every polynomial equation of degree \( n \) has exactly \( n \) roots, counted with multiplicity. On the other hand, the celebrated Abel–Ruffini theorem states that, for \( n \geq 5 \) there is no algebraic solution to polynomial equations with arbitrary coefficients. Galois independently proved the theorem by establishing a connection between field theory and group theory (see Jacobson 1974).

Within our framework, it seems that the Abel–Ruffini–Galois impossibility theorem may have connection with the amount of information contained in a given equation. Intuitively, a polynomial function with higher degree usually behaves in a more complex way, which in turn means that their zero points distribute in a more random way. As a result, polynomial equations with higher degree usually contain more information.

To illustrate, consider a quadratic equation

\[ ax^2 + bx + c = 0. \quad (45) \]

If \( a \neq 0 \), then the quadratic formula gives rise to a vector function from \( \mathbb{R}^3 \) to \( \mathbb{R}^2 \)

\[
(a, b, c) \mapsto \left( \frac{-b - \sqrt{b^2 - 4ac}}{2a}, \frac{-b + \sqrt{b^2 - 4ac}}{2a} \right). \quad (46)
\]

Then the information contained in the quadratic equation can be defined to be the information contained in this vector function. \( \square \)

**Example 5.3 (System of linear equations).** A general system of \( m \) linear equations with \( n \) unknowns can be written as a matrix equation of the form

\[ A_{m \times n} x = b. \quad (47) \]

If the coefficient matrix is square \( (m = n) \) and has full rank, then the system has a unique solution given by

\[ x = A^{-1} b. \quad (48) \]

In this case, we obtain a vector function \( (A, b) \mapsto A^{-1} b \) from \( \mathbb{R}^{n^2 + n} \) to \( \mathbb{R}^n \).

It seems that the amount of information contained in this function may be related to the time complexity of algorithms for solving \( Ax = b \). Indeed, one of the biggest unsolved problems in numerical analysis is whether the complexity of solving a linear system \( A_{m \times n} x = b \) can be reduced to \( O(n^2) \) (see Trefethen 2012). \( \square \)

**Example 5.4 (Elliptic curves).** An elliptic curve can be written as a plane algebraic curve defined by a Weierstrass equation over a (finite) field
The use of elliptic curves in cryptography was suggested independently by Neal Koblitz (1987) and Miller (1985). The amount of information contained in an elliptic curve may be important to Elliptic curve cryptography. □

6. CONCLUSIONS

The concept of Shannon entropy of random variables was generalized to measurable functions in general, and to simple functions with finite values in particular. The key point is to approximate a measurable function by a simple function with finite values, which is obtained by partitioning its range in the same way as computing the Lebesgue integral of it. As a result, these values, coupled with the probability measure of their preimages, forms a random variable in itself. So, it is natural to approximate the entropy of the given function by the entropy of the resulting random variable. Being an attribute of given function, the entropy conveys important information of this function. On the other hand, the computation of the entropy may also need some important information about the function in question. For example, it is shown that calculating the entropy of a Boolean function must be at least as hard as the Boolean satisfiability problem (SAT) problem.

It turns out that the information measure of measurable functions is related to the time complexity of search problems. Formally, given a Turing reduction from a search problem \( f(x) = y \) to another search problem \( g(x) = z \), the expected number of queries is equal to \( \frac{H(f)}{I(f;g)} \), where \( H(f) \) is the entropy of \( f(x) \) and \( I(f;g) \) is the average mutual information between \( f(x) \) and \( g(x) \).

In the idea case, if \( \frac{H(f)}{I(f;g)} \) is polynomial in the maximal size of inputs and the problem \( g(x) = z \) can be solved in polynomial time, then the problem \( f(x) = y \) also has polynomial-time algorithm.

It is shown that our information-based complexity estimation is a natural setting in which to study the power of randomized or probabilistic algorithms. Applying to decision problems, our result provides a strong evidence that \( P = RP = BPP \).

The main contributions of this paper can be summarized as follows.
—The concept of Shannon entropy of random variables was generalized to measurable functions.
—It is shown that the information measure of functions is related to the time complexity of solving search problems concerning functions.
—Our information-based complexity estimation provides a strong evidence that \( P = RP = BPP \).

ACKNOWLEDGMENTS
The authors would like to thank Professor Gaoping Li and Doctor Yihong Chen for helpful discussions.
REFERENCES

Agrawal, Manindra; Kayal, Neeraj; Saxena, Nitin. 2004. PRIMES is in P. Annals of Mathematics 160 (2): 781–793.

Cook, Stephen. 1971. The complexity of theorem proving procedures. In Proceedings of the Third Annual ACM Symposium on Theory of Computing, 151–158.

Thomas M. Cover, Joy A. Thomas. (2006). Elements of Information Theory (2nd edition). John Wiley & Sons, Inc.

Diffie, W., Hellman, M. 1976. New directions in cryptography. IEEE Transactions on Information Theory 22 (6): 644–654.

Erdos P. and Renyi A. (1963). On two problems of information theory, Publ. Hung. Acad. Sci. 8: 241-254.

Ford, Kevin R., Leighton, R. and Sands M. 2013. The Feynman Lectures on Physics, Volume I (online edition). The Feynman Lectures Website. http://feynmanlectures.caltech.edu/I_toc.html

Ford, Kevin (1998). The distribution of totients. Ramanujan J. 2 (1-2): 67–151.

Ford, Kevin (1999), The number of solutions of \( \phi(x) = m \), Annals of Mathematics 150 (1): 283–311.

Hua Lo-keng.1982. Introduction to Number Theory. Springer-Verlag.

Goldreich, Oded (2008), Computational Complexity: A Conceptual Perspective, Cambridge University Press.

Jacobson, N. Basic algebra I, W. H. Freeman and Co., San Francisco 1974.

Karp, Richard M. 1972. Reducibility Among Combinatorial Problems. In R. E. Miller and J. W. Thatcher (editors). Complexity of Computer Computations. New York: Plenum. pp. 85–103.

Koblitz, N. (1987). Elliptic curve cryptosystems. Mathematics of Computation 48 (177): 203–209.

Levin L. (1973). Universal search problems (in Russian), Problemy Peredachi Informatsii 9: 265–266. English translation in B. A. Trakhtenbrot, A survey of Russian approaches to Perbor (brute-force search) algorithms, Annals of the History of Computing 6 (1984): 384-400.

Miller, V. (1985). Use of elliptic curves in cryptography. CRYPTO. Lecture Notes in Computer Science 85: 417–426.

Papadimitriou Christos. (1994). Computational Complexity (1st edition). Addison Wesley.

Patrick Billingsley. (1995). Probability and Measure (3rd edition). Wiley Series in Probability and Statistics.

Rabin, Michael. (1979). Digitalized Signatures and Public-Key Functions as Intractable as Factorization. MIT Laboratory for Computer Science.

Rich, Elaine A. 2007. Automata, Computability and Complexity: Theory and Applications. Prentice-Hall.

Rivest, R.; Shamir, A.; Adleman, L. (1978). A Method for Obtaining Digital Signatures and Public-Key Cryptosystems. Communications of the ACM 21 (2): 120–126.

Shah Devavrat; Sharma, Mayank. Entropy and Complexity. HP Labs Technical Reports. HPL-BRIMS-2000-32.

Shannon C. (1938). A Symbolic Analysis of Relay and Switching Circuits. Trans. AIEE 57 (12): 713–723.

Shannon, Claude E. 1948. A Mathematical Theory of Communication. Bell System Technical Journal 27 (3), 379–423.

Shannon, Claude. 1949. Communication Theory of Secrecy Systems. Bell System Technical Journal, vol. 28(4), 656–715.

Shannon C. (1949b). The synthesis of two-terminal switching circuits. Bell System Technical Journal 28: 59–98.

Trefethen Nick. (2012). The smart money's on numerical analysis. SIAM News, Volume 45(9).

Turing, Alan. 1948. Intelligent Machinery. Reprinted in “Cybernetics: Key Papers.” Ed. C.R. Evans and A.D.J. Robertson. Baltimore: University Park Press, 1968.

Yao, A.C. 1982. Theory and Applications of Trapdoor Functions. Proc. of the 23rd IEEE Symp. on Foundation of Computer Science. 80-91.