On a paper by Leonard Adleman – k-potency

Working paper

Comments welcome; my email is: armandobcm@yahoo.com

Armando B. Matos

November 4, 2013

Abstract

In [1] the author introduces the related concepts of k-potent numbers and inflating functions, using as main example the factorization problem. We analyze that paper and discuss the possibility of generalizing these concepts to other classes of problems. In particular, we ask if there are three or more natural "decision" or "result" problem classes, such that one of the has intermediate k-potency. In terms of the relationship between classes of problems we have a mathematical pre-order relation, where the non-symmetry is essential for the theory of public-key cryptography.

A physical interpretation of k-potency based on the Landauer principle is also discussed.

1 Introduction

In this note we analyze the ideas and concepts presented in the early paper [1]. However, randomized algorithms and randomized classes are not discussed. We often quote or transcribe parts of [1].

It is interesting to notice that [3], a "reference" in Kolmogorov complexity, mentions [1] twice, first as a precursor of time bounded Kolmogorov complexity:

> Page 596:
> "These papers, and [L.A. Levin, Problems Inform. Transmission, 9(1973), 265–266; R.P. Daley, Theoret. Comput. Sci., 4(1977), 301–309; L.M. Adleman, "Time, space, and randomness," LCS Report TM-131, 1979, MIT], are significant early documents of resource-bounded Kolmogorov complexity."

and later as the reference that introduces the concept of "potential":

> Page 599:
> "Related work on a notion of 'potential' is based on L.M. Adleman [Ibid.] and is not treated in this edition".

1.1 On "decision" and "result" problems

An NP problem Π is characterized by a polynomial time (P-time) function $\pi(x, y)$ with codomain $\{0, 1\}$ (or $\{\text{FALSE}, \text{TRUE}\}$), where x is the instance of the problem and y is the witness.

To *solve* the problem in P-time is to find a polynomial time algorithm $A(x)$ that, given the input x, answers

$$A(x) = \begin{cases} 0 & \text{if} \quad \neg\exists y : f(x, y) = 1 \\ 1 & \text{if} \quad \exists y : f(x, y) = 1 \end{cases}$$

Note that no witness y is outputted by the algorithm.

For many NP problems it is possible to use an P-time decision algorithm to implement a P-time algorithm that outputs an witness if it exists,

$$A'(x) = \begin{cases} \text{FALSE} & \text{if } \neg\exists y : f(x, y) = 1 \\ y & \text{such that } f(x, y) = 1 \text{ otherwise} \end{cases}$$

For many NP problems, including SAT, Clique, Hamiltonian cycle, and many others, eventual P-time decision algorithms can be easily transformed in P-time algorithms "with result". However, for the factorization problem, the decision problem (is the integer composed?) does not seem to help much the "result" problem (if the integer if not a prime, output a non trivial factor). Quoting the Wikipedia,

> When discussing what complexity classes the integer factorization problem falls into, it's necessary to distinguish two slightly different versions of the problem:
> The "function" [or "result"] problem version: given an integer n, find an integer d with $1 < d < n$ that divides n (or conclude that n is prime). This problem is trivially in FNP [function problem extension of the decision problem class NP] and it's not known whether it lies in FP or not. This is the version solved by most practical implementations.
> The "decision" problem version: given an integer n and an integer m with $1 \leq m \leq n$, does n have a factor d with $1 < d < m$? This version is useful because most well-studied complexity classes are defined as classes of decision problems, not function problems. This is a natural decision version of the problem, analogous to those frequently used for optimization problems, because it can be combined with binary search to solve the function problem version in a logarithmic number of queries.
> It is not known exactly which complexity classes contain the decision version of the integer factorization problem. It is known to be in both NP and co-NP. This is because both YES and NO answers can be verified in polynomial time given the prime factors (we can verify their primality using the AKS primality test, and that their product is n by multiplication). The fundamental theorem of arithmetic guarantees

that there is only one possible string that will be accepted (providing the factors are required to be listed in order), which shows that the problem is in both UP [complexity class of decision problems solvable in polynomial time on a non-deterministic Turing machine with at most one accepting path for each input] and co-UP. [...] It is [...] a candidate for the NP-intermediate complexity class.
In contrast, the decision problem "is n a composite number?" [...] appears to be much easier than the problem of actually finding the factors of n. Specifically, the former can be solved in polynomial time (in the number of digits of n) with the AKS primality test. In addition, there are a number of probabilistic algorithms that can test primality very quickly in practice if one is willing to accept the vanishingly small possibility of error.

1.2 Potential associated with a number

The length of the representation of an integer n is denoted by $|n|$, thus $||n||$ denotes the length of the length of the representation of n. Using basis 2 to represent integers and as the basis of the logarithms, $|n| \approx \log n$, $||n|| \approx \log(\log n)$. Definition in [1]:

Definition 1 *For all $k \in \mathbb{N}$, for all $\sigma \in \{0,1\}^\star$ (and for all $\tau \in \{0,1\}^\star$), σ is* k-potent *(with respect to τ) if there is a program p of size less than or equal to $k||\sigma||$, which, with blanks (τ) as input, halts with output σ in less than or equal to $|\sigma|^k$ steps.*

Intuitively, σ is k-potent when there is a very short and efficient (more precisely, in time $|\sigma|^k$) program that prints σ. So, k-potent could perhaps be called k-easy.

The definition above can be expressed in terms of the time-bounded Kolmogorov complexity of σ:

$$\sigma \text{ is } k\text{-potent: } K^{|\sigma|^k}(\sigma|\tau) \leq k||\sigma|| \tag{1}$$

Definition 2 *(Given a string $\tau \in \{0,1\}^\star$), a string $\sigma \in \{0,1\}^\star$ is* not-k-potent *(with respect to τ) if for every $k \in \mathbb{N}$ there are infinitely many $\sigma \in \{0,1\}^\star$ such that σ) is not* k-potent *(with respect to τ).*

1.3 The existence of polynomial algorithms for non k-potent numbers

Let $n = |\sigma|$, $|n| = ||\sigma|| \approx \log n$. A k-potent string σ can be generated in polynomial time (polynomial in $n = |\sigma|$). This is immediate from (1) and the definition of K^t with $t = |\sigma|^k$.

Also,

Theorem 1 *For every k the set of strings not longer than σ that can be efficiently generated from σ, more precisely,*

$$f_k(\sigma) = \{\alpha : |\alpha| \leq n \ \textit{ and } \alpha \textit{ is } k\text{-potent } \textit{wrt } \sigma\}$$

(where $n = |\sigma|$) is computable in polynomial time.

Proof. Notice that, as α is k-potent wrt σ and $|\alpha| \leq |\sigma| = n$, the set $f_k(\sigma)$ is small, $\#f_k(\sigma) \leq 2^{k||\sigma||} = n^k$; there are at most $2^{k|n|} \approx n^k$ programs with length at most $k|n|$. The simulation of all these programs during the time $|\sigma|^k = n^k$ and the construction of the set $f_k(\sigma)$ takes polynomial time. □

1.4 Tuples of integers as integers: elementary codifications

Using an efficient encoding, a tuple of integers or strings can be seen as single integer with essentially the same potency.

In this way we can talk about the k-potency of a tuple of integers or strings.

For instance, the factorization of an integer (2) can be seen as a function $\mathbb{N} \to \mathbb{N}$, instead of a function $\mathbb{N} \to \mathbb{N} \times \mathbb{N}$.

1.5 Inflating functions

A function $f(x)$ is inflating if, infinitely often, it is inefficient to obtain $f(x)$ from x. The definition of [1] is

Definition 3 *A function $f : \{0,1\}^\star \to \{0,1\}^\star$, f is an* inflating function *if, for all $k \in \mathbb{N}$ there are infinitely many $\sigma \in \{0,1\}^\star$ such that $f(\sigma)$ is not k-potent with respect to σ.*

It is as if, infinitely often, some difficult to obtain (deep) information about x is easy to obtain from (is shallow in) $f(x)$. Figure 1 (page 5) represents an inflating (factorize), deflating (forget) and (almost) "iso-flating" (multiply) functions.

1.6 Inflating theorem

Theorem 2 *Let* FAC *denote the integer factoring function. Then* FAC $\notin P$ *if and only if* FAC *is inflating.*

Proof. Let $n = |\alpha|$ and $m = |\mathrm{FAC}(\alpha)|$. That is, $n = |pq| \approx \log(pq)$ and $m = |\mathrm{FAC}(\langle p, q\rangle|) \leq 2\log(pq)$, using a simple pair codification.
1) FAC $\in P \Rightarrow$ FAC is not inflating.
Assume there an algorithm of size S factors in time n^k for some k. Let α be any input of size at least $2^{S/k}$, that is, $n = |\alpha| \geq 2^{S/k}$. We get $S \leq k \log n = k||\alpha||$.

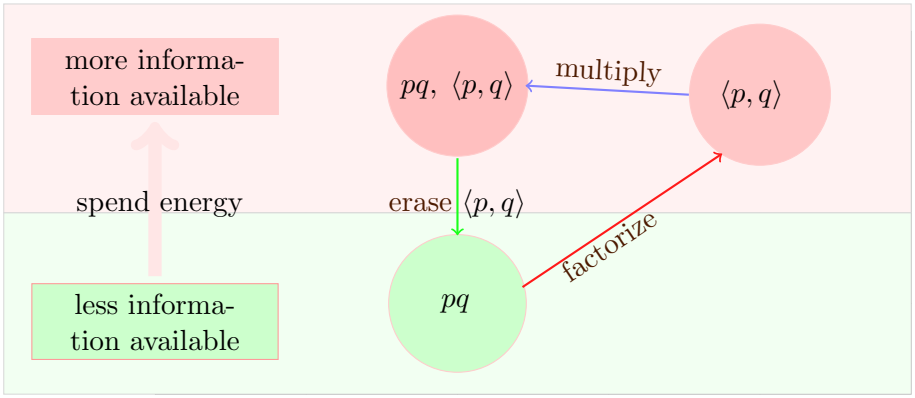

Figure 1: A representation of algorithms for factorization and product. An algorithm is inflating (FAC in the figure) if the corresponding arrow goes from bottom to top. A similar diagram would relate all the *instances* of NP-complete problems at the bottom, and the corresponding pairs ⟨instance, witness⟩ the top.

Thus, FAC is k-potent with respect to α (or "given α).

2) FAC is not inflating $\Rightarrow$ FAC $\in P$.
There is a k such that for almost all α, FAC(α) is k-potent with respect to α. But $|\text{FAC}(\alpha)| \leq 2|\alpha|$. By Theorem 1, $f_k(\alpha, 1^{2|\alpha|})$ is computable in polynomial time. Since all prime factors of α are in $f_k(\alpha, 1^{2|\alpha|})$, we can in polynomial time arrive at FAC(α) by taking GCDs. □

1.7 Inflating theorem for the SAT problem

We now concentrate on the SAT problem. Let ϕ be an instance of SAT and let $T(\phi)$ be a truth assignment satisfying ϕ (if the instance is positive). The following result is from [1].

Theorem 3 *Then* SAT $\notin P$ *if and only if for every* $k \in \mathbb{N}$ *there is an (a positive?) instance* $\phi \in$ SAT *such that every* $T(\phi)$ *is not* k-potent *with respect to* ϕ.

Proof. (i) If SAT $\in P$ there is a polynomial-time function (by binary searching truth assignments)

$$f_{\text{SAT}} = \begin{cases} 0 & \text{if } \phi \notin \text{SAT} \\ \text{the least } T(\phi) & \text{if } \phi \in \text{SAT} \end{cases}$$

Then f_{SAT} is non inflating. So, there is a k such that, for every $\phi \in$ SAT, there is a $T(\phi)$ that is k-potent with respect to ϕ.

(ii) $(\forall k)\ (\forall \phi \in \text{SAT})\ (\exists T(\phi)\ \ k\text{-potent with respect to } \phi)$.
Then SAT $\in P$, since using the technique described in the proof of Theorem 2, we could enumerate all of the $T(\phi)$ which are k-potent in ϕ and confirm if one satisfies ϕ in polynomial time. □

2 Comments

2.1 Inflating theorem for other NP problems

We can apply the concept of k-potency to other classes of problems. The two levels represented in Figure 1 (page 5) can be generalized for "P/NPC with witnesses".

Two strings polynomially reducible to each other are considered to be equally difficult and placed on the same level (k-potent for some k, or not k-potent for any k).

The two levels P and NPC (NP-complete) problems are not directly amenable to k-potency or inflating analysis because in these classes we are dealing with decision problems, not with outputs of functions that are words or integers.

P and NPC with witnesses. Here we consider result problems (see definition in Section 1.1, page 2). Also notice that witnesses are often polynomially obtained with an algorithm with access to an oracle for the decision problem, see page 2.

If the Berman–Hartmanis Conjecture is true, problems in NPC are not only polynomially reducible to each other but also polynomially isomorphic.

2.2 Kolmogorov complexity in restricted algebras

One possible problem that relates KC with DC is to find (in some fixed logic L) a shortest proposition that is equivalent to a given proposition ϕ. The corresponding minimum length could be denoted by $K_L(\phi)$.
The shortest representations of integers as expressions of some fixed algebras were studied in [5, 4].

2.3 Why is the factorization problem used to define the "potential" of a number?

The potential of a number is not related to a particular problem. Many other "result" problems are possible. We could for instance, define the problem

Instance: Integer n.
Answer:

$$\begin{cases} \langle p_1, p_2 \rangle \text{ such that } n = p_1 + p_2 p_2 \\ \text{FALSE if there are no such pair } \langle p_1, p_2 \rangle \end{cases}$$

Then, we have the following important problem:

> There are several inflating functions, each producing not-k-potent strings $f(\sigma)$ relatively to the same input σ; these results, say, $f_1(\sigma)$, $f_2(\sigma)$,... are not necessarily at the same level of k-potency. Can we relate them in terms of k-potency? Is there a natural and interesting hierarchy of non k-potent strings?

3 Energy of a computation: the physical point of view

Instead of the "potential" associated with an integer, we can possibly use the "energy" it takes to move from one representation to another, say from pq (product of primes) to $\langle p, q\rangle$ (pair of primes). This energy can be defined precisely as the minimum physical energy dissipated (according to Landauer principle) by an algorithm that implements the transformation, for instance that factorizes pq.

It is well know that irreversible computations necessarily dissipate heat, see [2]. Landauer's principle states that (using the Wikipedia entry) "for a computational operation in which 1 bit of logical information is lost, the amount of entropy generated is at least $k \ln 2$, and so, the energy that must eventually be emitted to the environment is $E \geq kT \ln 2$", where k is Boltzman constant, about $1.38 \cdot 10^{-23}$ J/K. Currently, this lower bound is not longer negligible! For example, take the heavy but realistic computation

$$\begin{cases} 300 & \text{temperature, about 18 degrees Celsius} \\ 10^{10} & \text{elementary operations per second, 10GHz} \\ 10^{6} & \text{bits destroyed in each "elementary operation"} \\ 10000 & \text{processors} \\ 10000 & \text{total computation time in seconds} \end{cases}$$

We get

$$E \geq 1.38 \cdot 10^{-23} \times 300 \times 10^{10} \times 10^{6} \times 10\,000 \times 10\,000 \approx 4\,140 J$$

So, the idea is that there is a minimum physical energy needed to move from one representation to another. For instance, the transformation

$$pq \rightarrow \langle p, q\rangle \tag{2}$$

necessarily – by the Landauer principle and assuming that P≠NP – dissipates, infinitely often, a large energy.

The situation is asymmetric relatively to the reverse transformation $\langle p, q\rangle \to pq$; common computers are irreversible and do not produce energy.

3.1 Can potential theory and minimum energy be applied to QM computers?

In terms of factorization algorithms, the answer seems to be no. Because (i) there is a polynomial time QM algorithm, and (ii) QM is reversible so that Landauer principle is not interesting here!

References

[1] Leonard M. Adleman. Time, space and randomness. Technical Report MIT/LCS/TM-131, Massachusetts Institute of Technology. Laboratory, March 1979.

[2] Ralf Landauer. Irreversibility and heat generation in the computing process. *IBM Journal of Research and Development*, 5(3):183–191, 1961.

[3] Ming Li and Paul Vitányi. *An Introduction to Kolmogorov Complexity and Its Applications*. Springer, third edition, 2008.

[4] Armando B. Matos. Exponentiation: normal forms and properties, 2005 (revised in 2011).

[5] Armando B. Matos. Kolmogorov complexity in multiplicative arithmetic, 2005 (revised in 2011).