Epistemology via Brute Force Computing

Roman V. Yampolskiy
Computer Engineering and Computer Science
University of Louisville, USA
roman.yampolskiy@louisville.edu

Abstract

In this work we show how disjoint theories of information, complexity, communication and computation may be theoretically unified via brute force computing. The paper suggests using improvement over the brute force algorithm as a common unifying factor necessary for the creation of a unified theory of information processing. By defining such diverse terms as knowledge, intelligence and information in terms of a common denominator we are able to show how computing can serve to add to the field of epistemology.

1 Introduction

The quest for a Unified Theory of Everything (UTE) is well known to be a central goal in natural sciences. In recent years a similar aspiration to find a Unified Theory of Information (UTI) has been observed in computational sciences (Braga, 1977; Burgin, 2003; Doucette, Bichler, Hofkirchner, & Raffl, 2007; Fleissner & Hofkirchner, 1996; Floridi, 2002; Fluckiger, July 1997; Hofkirchner, 1999, 2005, 2009; Holmstrom & Koli, 2002; Ji, 2003; Mizzaro, 2001; Zhong, 2005). Despite numerous attempts, no such theory has been discovered and the quest to unify Shannon's Information Theory (Shannon, July 1948), Kolmogorov-Chaitin Complexity theory (Chaitin, 1966; Kolmogorov, 1965), Solomonoff's Algorithmic Information Theory (Solomonoff, February 4, 1960) and Yao's Communication complexity (Yao, 1979b), as well as concepts of intelligence and knowledge continues. In this paper we present a novel set of definitions for information and computation related concepts and theories which is based on a common concept of efficiency. We show that a common thread exists and that future efforts could succeed in formalizing our intuitive notions[*].

2 Efficiency Theory

The proposed Efficiency Theory (EF) is derived with respect to the universal algorithm known as the "brute force" approach. Brute Force (BF) is an approach to solving difficult computational problems by considering every possible answer. BF is an extremely inefficient way of solving problems and is usually considered inapplicable in practice to instances of difficult problems of non-trivial size. It is an amazing and underappreciated fact that this simplest to discover, understand and implement algorithm also produces the most accurate (not approximate) solutions to the set of all difficult computational problems (NP-Hard, NP-Complete, etc.). In this paper we consider BF in an even broader context, namely, BF could be inefficient in other ways, for example representing otherwise compressible text strings by specifying every symbol.

[*] An extended version of this article has been submitted to the Journal of Discrete Mathematical Sciences & Cryptography.

Efficiency in general describes the extent to which resources such as time, space, energy, etc. are well used for the intended task or purpose. In complexity theory it is a property of algorithms for solving problems which require at most a number of steps (or memory locations) bounded from above by some polynomial function to be solved. The size of the problem instance is considered in determining the bounding function. Typically efficiency of an algorithm could be improved at the cost of solution quality. This often happens in cases where approximate solutions are acceptable. We also interpret efficiency to mean shorter representations of redundant data string. Essentially, EF measures how far can we get away from the BF in terms of finding quick algorithms for difficult problems studied in Complexity Theory (Levin (Levin, 1986), Cook (Cook & Reckhow, 1979), Karp (Karp, 1972), etc.) as well as towards discovering succinct string encodings (Shannon (Shannon, July 1948), Kolmogorov (Kolmogorov, 1965), Solomonoff (Solomonoff, February 4, 1960), Chaitin (Solomonoff, February 4, 1960)). Many fundamental notions related to information and computation could be naturally formalized in terms of their relevance to BF or efficiency.

3 Information and Knowledge

Information is a poorly understood concept and can be analyzed by different researchers from very different domain specific points of view (Mizzaro, 2001). Pervez assembled the following collection of definitions for the concept of information from over 20 different studies (Pervez, 2009): data that can be understood as a commodity or physical resource; signal, code, symbol, message or medium; formal or recovered knowledge; subjective or personal knowledge; thinking, cognition, and memory; technology; text; uncertainty reduction; linkage between living organisms and their environment; product of social interaction that has a structure capable of changing the image structure of a recipient; as a stimulus, information facilitates learning and acts as means for regulation and control in society.

Ever since Shannon presented his information theory, different approaches to measuring information have been suggested: Langefors' infological equation (Langefors, 1966), Brookes' fundamental equation (Brookes, 1975), Semantic Information Theory (Hintikka, 1970), and many others. In the proposed Efficiency Theory, *information* (Shannon (Shannon, July 1948), Hartley (Hartley, July 1928), Kelly (Kelly, 1956)) measures how inefficiently knowledge (or specified information) is represented. (A special type of information sharing known as Communication Complexity (Yao, 1979a) deals with the efficiency of communication between multiple computational processes and could be a subject to similar analysis). Shannon himself defined the fundamental problem of communication as that of "… reproducing at one point either exactly or approximately a message selected at another point. (Shannon, July 1948)" The BF approach to this problem would be to simply send over the whole message, symbol after symbol, completely disregarding any knowledge we might have about the properties of the text string in question. However, a more efficient approach may be to incorporate any knowledge we might already have about the message (for example that a certain symbol always starts any message) and to only transmit symbols which would reduce uncertainty about the message and by doing so provide us with novel knowledge.

From the above, ET allows us to define k*nowledge* as efficiently represented specified information. We are free to interpret the word efficiency either as an effective symbolic encoding or as an effective computation. Perhaps a few examples would help to define what we mean. With respect to efficient symbolic representation, Hoffman coding is a well-known example of an entropy encoding algorithm which uses variable-length codes calculated based on probability of occurrence of each source symbol to represent the message in the most efficient way (Huffman, September 1952). The next example explains what we mean by efficient knowledge representation with respect to computation. If we want to share two numbers, we can do so in a number of ways. In particular, we can share the numbers in a direct and efficient, to retrieve, representation of knowledge:

"3980750864240649373971255005503864911990643623425267084063851895759463889572617685833317" and "4727721461074353025362230719730482246329146953020971164598521711305207112563635903975527" or we can share the same two numbers, but in the form of necessary information, not efficiently accessible knowledge, as in, find the two factors of ("The RSA Challenge Numbers," 2007):
"1881988129206079638386972394616504398071635633794173827007633564229888597152346654853190606065047430453173880113033967161996923212057340318795506569962213051687593076502570 59". Both approaches encode exactly the same two numbers, only in the second case the recipient would have to spend a significant amount of computational resources (time) converting inefficiently presented data (information) into efficiently stored data (knowledge). Mizzaro suggests that the two types of information be referred to as "actual" and "potential" (Mizzaro, 2001).

Another example aimed to illustrate information/knowledge distinction comes from an article by Aaronson (Aaronson, 2012): The largest known prime number, as reported by the Mersenne.org, is p := $2^{43112609} - 1$. But what does it mean to say that p is "known"? Does that mean that, if we desired, we could print out all 30 pages of its decimal digits? That doesn't seem right. All that should really matter is that the expression '$2^{43112609} - 1$' picks out a unique positive integer, and that integer has been proven to be prime. However, if those are the criteria, then why can't we immediately beat the largest-known-prime record by postulating that: *p′ = The first prime larger than $2^{43112609} - 1$*. Clearly p′ exists and it is uniquely defined, and is also a prime number by definition. "If we want, we can even write a program that is guaranteed to find p′ and output its decimal digits, using a number of steps that can be upper-bounded a priori. Yet our intuition stubbornly insists that $2^{43112609} - 1$ is a "known" prime in a sense that p′ is not. Is there any principled basis for such a distinction? The clearest basis that I can suggest is the following. We know an algorithm that takes as input a positive integer k, and that outputs the decimal digits of $p = 2^k - 1$ using a number of steps that is polynomial—indeed, linear—in the number of digits of p. But we do not know any similarly-efficient algorithm that provably outputs the first prime larger than $2^k - 1$ (Aaronson, 2012)."

Again, the only distinction between information and knowledge is how efficiently we can get access to the desired answer. In both cases we are dealing with pre-specified information since we know that the answer is going to represent a prime number, but knowledge is immediately available to us, while information may require an insurmountable amount of processing to deliver the same result. This leads us to an interesting observation: Information can't be created or destroyed, only made less efficiently accessible. For example prime numbers existed before the Big Bang and will continue to exist forever regardless of our best efforts to destroy them. At any point in time, one can simply start printing out a list of all integers and such a list will undoubtedly contain all prime numbers and as long as we are willing to extract specific numbers from such a list, our knowledge of particular prime numbers could be regained after paying some computational cost. Consequently that means that, knowledge could be created or destroyed by making it significantly less or more efficient to access or by providing or deleting associated specifications.

In fact we can generalize our prime number list example to the list of all possible strings of increasingly larger size. The idea of such a list is not novel and has been previously considered by Jorge Luis Borges in *The Library of Babel* (Borges, 2000), by Hans Moravec in *Robot* (Moravec, 1999) and by Bruno Marchal in his PhD thesis (Marchal, 1998). Essentially, all the knowledge we will ever have is already available to us in the form of such string libraries. The only problem is that it is stored in an inefficient to access format, lacking specifications. The knowledge discovery process (computation) converts such inefficient information into easily accessible knowledge by providing descriptive pointers to optimally encoded strings to give them meaning. Specified information is a tuple (x,y) there f(x) has the same semantic meaning as y and function f is a specification. Given enough time we can compute any computable function so time is a necessary resource to obtain specified knowledge. Since multiple, in fact infinite, number of semantic pointers could refer to the

same string (Mizzaro, 2001) that means that a single string could contain an infinite amount of knowledge if taken in the proper semantic context, generating multiple levels of meaning. Essentially that means that obtained knowledge is relative to the receiver of information. It is mainly to avoid the resulting complications that Shannon has excluded semantics from his information theory (Shannon, July 1948).

Jurgen Schmidhuber has also considered the idea of string libraries and has gone so far as to develop algorithms for "Computing Everything" (Jurgen Schmidhuber, 1997, 2002). In particular, concerned with the efficiency of his algorithm Schmidshuber has modified a Levin Search algorithm (Levin, 1973) to produce a provably fastest way to compute every string (Jurgen Schmidhuber, 2002). Schmidshuber's work shows that computing all information is easier than computing any specific piece, or in his words: "… computing all universes with all possible types of physical laws tends to be much cheaper in terms of information requirements than computing just one particular, arbitrarily chosen one, because there is an extremely short algorithm that systematically enumerates and runs all computable universes, while most individual universes have very long shortest descriptions (Juergen Schmidhuber, November 30, 2000)."

4 Intelligence and Computation

Computation is the process of obtaining efficiently represented information (knowledge) by any algorithm (including inefficient ones). Intelligence in the context of EF could be defined as the ability to design algorithms which are more efficient compared to brute force. Same ability shown for a variety of problems is known as general intelligence or universal intelligence (Legg & Hutter, December 2007). An efficient algorithm could be said to exhibit intelligence in some narrow domain. In addressing specific instances of problems, an intelligent system can come up with a specific set of steps which don't constitute a general solution for all problems of such type, but are nonetheless efficient. Intelligence could also be defined as the process of obtaining knowledge by efficient means. If strict separation between different complexity classes (such as P VS NP) is proven, it would imply that no efficient algorithms for solving NP complete problems could be developed (Roman V Yampolskiy, 2011). Consequently, that would imply that intelligence has an upper limit, a non-trivial result which has only been hinted at from limitations in physical laws and constructible hardware (Lloyd, 2000).

Historically, the complexity of computational processes has been measured either in terms of required steps (time) or in terms of required memory (space). Some attempts have been made in correlating the compressed (Kolmogorov) length of the algorithm with its complexity (Trakhtenbrot, 1984), but such attempts didn't find much practical use. We suggest that there is a relationship between how complex a computational algorithm is and intelligence, in terms of how much intelligence is required to either design or comprehend a particular algorithm. Furthermore we believe that such an intelligence based complexity measure is independent from those used in the field of complexity theory.

To illustrate the idea with examples we again will begin with the brute force algorithm. BF is the easiest algorithm to design as it requires very little intelligence to understand how it works. On the other hand an algorithm such as AKS primality test (Agrawal, Kayal, & Saxena, 2004) is non-trivial to design or even to understand since it relies on a great deal of background knowledge. Essentially the intelligence based complexity of an algorithm is related to the minimum intelligence level required to design an algorithm or to understand it. This is a very important property in the field of education where only a certain subset of students will understand the more advanced material. We can speculate that a student with an "IQ" below a certain level can be shown to be incapable of understanding a particular algorithm. Likewise we can show that in order to solve a particular problem (P VS. NP)

someone with IQ of at least X will be required. With respect to computational systems it would be inefficient to use extraneous intelligence resources to solve a problem for which a lower intelligence level is sufficient.

Consequently, efficiency is at the heart of algorithm design and so efficiency theory can be used to provide a novel measure of algorithm complexity based on necessary intellectual resources. Certain algorithms while desirable could be shown to be outside of human ability to design them because they are just too complex from the available intelligence-resources point of view. Perhaps the invention of superintelligent machines will make discovery/design of such algorithms feasible (R.V. Yampolskiy, 2011). Also by sorting algorithms based on the perceived required IQ resources we might be able to predict the order in which algorithms will be discovered. Such an order of algorithm discovery would likely be consistent among multiple independently working scientific cultures, making it possible to make estimates of state-of-the-art in algorithm development. Such capability is particularly valuable in areas of research related to cryptography and integer factorization (Yampolskiy, 2010).

Given current state of the art in understanding of human and machine intelligence the proposed measure is not computable. However different proxy measures could be used to approximate the intellectual resources to solve a particular problem. For example the number of scientific papers published on the topic may serve as a quick heuristic to measure the problem's difficulty. Supposedly, in order to solve the problem one would have to be an expert in all of the relevant literature. As our understanding of human and machine intelligence increases a more direct correlation between available intellectual resources such as memory and difficulty of the problem will be derived.

5 Conclusions and Future Directions

All of the concepts defined above have a common factor, namely "efficiency," and could be mapped onto each other. First the constituent terms of pairs of opposites could be trivially defined as opposite ends of the same spectra. Next, some interesting observations could be made with respect to the relationships observed on less obviously related terms. For example, problems could be considered information, while answers to them are knowledge. Efficiency (or at least rudimentary efficient algorithms) could be produced by brute force approaches simply by trying all possible algorithms up to a certain length until a more efficient one is found. A universal efficiency measure could be constructed by contrasting the resulting solution with the pure brute force approach. So depending on the domain of analysis the ratio of symbols, computational steps, memory cells or communication bits to the number required by the brute force algorithm could be calculated as the normalized efficiency of the algorithm. Since the number of possible brute force algorithms is also infinite and they can greatly differ in their efficiency we can perform our analysis with respect to the most efficient brute force algorithm which works by considering all possible solutions, but not impossible ones.

Some problems in NP are solvable in practice, while some problems in P are not. For example an algorithm with running time of 1.00000001^n is preferred over the one with a running time of n^{10000} (Aaronson, 2012). This is a well-known issue and a limitation of a binary tractable/intractable separation of problems into classes. In our definition, efficiency is not a binary state but rather a degree ranging from perfectly inefficient (brute force required) to perfectly efficient (constant time solvable). Consequently the efficiency theory is designed to study the degree and limits of efficiency in all relevant domains of data processing.

References

Aaronson, S. (2012). Why Philosophers Should Care About Computational Complexity *Computability: Godel, Turing, Church, and beyond*: MIT Press.

Agrawal, M., Kayal, N., & Saxena, N. (2004). PRIMES is in P. *Annals of Mathematics, 160(2)*, 781-793.

Borges, J. L. (2000). *The Library of Babel* Jeffrey, New Hemshire: David R. Godine Publisher

Braga, G. M. (1977). Semantic Theories of Information. *Information Sciences (Ciência da Informação), 6(2)*, 69-73.

Brookes, B. C. (1975). The fundamental equation of information science. *Problems of Information Science, 530*, 115-130.

Burgin, M. (2003). Information: Paradoxes, Contradictions, and Solutions. *Triple C - Cognition, Communication, Co-operation, 1(1)*, 53-70.

Chaitin, G. J. (1966). On the Length of Programs for Computing Finite Binary Sequences. *Journal of the ACM (JACM), 13(4)*, 547-569.

Cook, S. A., & Reckhow, R. A. (1979). The Relative Efficiency of Propositional Proof Systems. *The Journal of Symbolic Logic, 44(1)*, 36-50.

Doucette, D., Bichler, R., Hofkirchner, W., & Raffl, C. (2007). Toward a New Science of Information. *Data Sciences Journal, 6(7)*, 198-205.

Fleissner, P., & Hofkirchner, W. (1996). Emergent Information: Towards a Unified Information Theory. *BioSystems, 38(2-3)*, 243-248.

Floridi, L. (2002). What Is The Philosophy of Information? *Metaphilosophy, 33(1-2)*, 123-145.

Fluckiger, F. (July 1997). Towards a Unified Concept of Information: Presentation of a New Approach. *World Futures: The Journal of General Evolution, 49/50 (3-4)*, 309.

Hartley, R. V. L. (July 1928). Transmission of Information *Bell System Technical Journal 7(3)*, 535-563.

Hintikka, J. (1970). *On Semantic Information. In: Physics, Logic, and History*: Plenum Press.

Hofkirchner, W. (1999). *Cognitive Sciences In the Perspective of a Unified Theory of Information*. Paper presented at the 43rd Annual Meeting of the International Society for the Systems Sciences (ISSS), Pacific Grove, CA.

Hofkirchner, W. (2005). *A Unified Theory of Information As Transdisciplinary Framework*. Paper presented at the ICT&S Center for Advanced Studies and Research, http://www.idt.mdh.se/ECAP-2005/articles/BIOSEMANTICS/WolfgangHofkirchner/WolfgangHofkirchner.pdf.

Hofkirchner, W. (2009). How To Achieve a Unified Theory of Information. *Triple C - Cognition, Communication, Co-operation, 7(2)*, 357-368.

Holmstrom, J., & Koli, T. (2002). Making the Concept of Information Operational. *Department of Information Technology and Media, Mid Sweden University*(Sweden), Master Thesis in Informatics, http://www.palmius.com/joel/lic/infoop.pdf.

Huffman, D. A. (September 1952). *A Method for the Construction of Minimum-Redundancy Codes*. Paper presented at the Institute of Radio Engineers (I.R.E.).

Ji, S. (2003, February 5-11, 2003). *Computing With Numbers, Words, or Molecules: Integrating Mathematics, Linguistics and Biology Through Semiotics.* Paper presented at the Reports of the Research Group on Mathematical Linguistics, Taqrragona, Spain.

Karp, R. M. (1972). Reducibility Among Combinatorial Problems. In R. E. Miller & J. W. Thatcher (Eds.), *Complexity of Computer Computations* (pp. 85-103). New York: Plenum.

Kelly, J. L. (1956). A New Interpretation of Information Rate. *Bell System Technical Journal, 35*, 917-926.

Kolmogorov, A. N. (1965). Three Approaches to the Quantitative Definition of Information. *Problems Inform. Transmission, 1(1)*, 1-7.

Langefors, B. (1966). *Theoretical analysis of information systems*. Lund: Studentlitteratur.

Legg, S., & Hutter, M. (December 2007). Universal Intelligence: A Definition of Machine Intelligence. *Minds and Machines, 17(4)*, 391-444.

Levin, L. (1973). Universal Search Problems. *Problems of Information Transmission, 9(3)*, 265--266.

Levin, L. (1986). Average-case complete problems. *SIAM J. Comput, 15*, 285-286.

Lloyd, S. (2000). Ultimate Physical Limits to Computation. *Nature, 406*, 1047-1054.

Marchal, B. (1998). *Calculabilite, Physique et Cognition*. Paper presented at the PhD thesis, L'Universite des Sciences et Technologies De Lilles.

Mizzaro, S. (2001). Towards a Theory of Epistemic Information. *Information Modelling and Knowledge Bases, 12*, 1-20.

Moravec, H. (1999). *Robot*: Wiley Interscience.

Pervez, A. (2009). Information As Form. *Triple C - Cognition, Communication, Co-operation, 7(1)*, 1-11.

The RSA Challenge Numbers. (2007). *RSA Laboratories*, *http://www.rsa.com/rsalabs/node.asp?id=2093*

Schmidhuber, J. (1997). A Computer Scientist's View of Life, the Universe, and Everything. *LNCS, Springer*, 201-288.

Schmidhuber, J. (2002). *The Speed Prior: A New Simplicity Measure Yielding Near-Optimal Computable Predictions*. Paper presented at the 15th Annual Conference on Computational Learning Theory (COLT 2002), Sydney, Australia.

Schmidhuber, J. (November 30, 2000). *Algorithmic Theories of Everything*. Available at: http://arxiv.org/pdf/quant-ph/0011122v2.

Shannon, C. E. (July 1948). A Mathematical Theory of Communication. *Bell Systems Technical Journal, 27(3)*, 379-423.

Solomonoff, R. (February 4, 1960). *A Preliminary Report on a General Theory of Inductive Inference* Paper presented at the Report V-131, Zator Co., Cambridge, Ma.

Trakhtenbrot, B. A. (1984). A Survey of Russian Approaches to Perebor (Brute-Force Searches) Algorithms. *IEEE Annals of the History of Computing, 6(4)*, 384-400.

Yampolskiy, R. V. (2010). Application of Bio-Inspired Algorithm to the Problem of Integer Factorisation. *International Journal of Bio-Inspired Computation (IJBIC), 2(2)*, 115-123.

Yampolskiy, R. V. (2011). AI-Complete CAPTCHAs as Zero Knowledge Proofs of Access to an Artificially Intelligent System. *ISRN Artificial Intelligence, 271878*.

Yampolskiy, R. V. (2011). Construction of an NP Problem with an Exponential Lower Bound. *Arxiv preprint arXiv:1111.0305*.

Yao, A. C. (1979a). *Some Complexity Questions Related to Distributed Computing*. Paper presented at the Proc. of 11th STOC.

Yao, A. C. (1979b). *Some Complexity Questions Related to Distributed Computing*. Paper presented at the 11th Symposium on Theory of Computing (STOC).

Zhong, Y. X. (2005). *Mechanism Approach to a Unified Theory of Artificial Intelligence*. Paper presented at the IEEE International Conference on Granular Computing, Beijing.