Poznań University of Technology

On SAT information content, its polynomial-time solvability and fixed code algorithms

M.Drozdowski

Research Report RA-01/23

2023

Institute of Computing Science, Piotrowo 2, 60-965 Poznań, Poland

On SAT information content, its polynomial-time solvability and fixed code algorithms

M.Drozdowski
Institute of Computing Science, Poznań University of Technology,
Piotrowo 2, 60-965 Poznań, Poland

Abstract

Amount of information in SAT is estimated and compared with the amount of information in the fixed code algorithms. A remark on SAT Kolmogorov complexity is made. It is argued that SAT can be polynomial-time solvable, or not, depending on the solving algorithm information content.

Keywords: computational complexity, information theory.

1 Introduction

A number of observations can be made in relation to the performance of algorithms solving combinatorial problems and the amount of information they hold:

• In [3] a connection between entropy of the Markov chains representing behavior of simulated annealing algorithms and the convergence of the expected objective function value has been made for maximum 3-SAT problem.
• In [8] it is argued that there is a link between the fraction of problem instances achieving certain histogram of values and the entropy of the histogram.
• In evolutionary optimization it is widely accepted rule of thumb that with growing population diversity and size, the chances of producing high quality solutions improve. Intuitively, such populations have more information.
• There is a notion of a graph *hard-to-color* for a certain algorithm in graph node coloring [5, 6]. A graph that is hard-to-color is colored by the considered algorithm with more colors than the optimum. There are examples of graphs hard-to-color for many deterministic algorithms. Random sequential algorithm visits graph nodes in a random sequence and assigns to a node the

lowest feasible color. For random sequential algorithm no hard-to-color graph exists, and hence, this algorithm cannot deterministically fail. Ominously, this algorithm is connected to a source of randomness, that is, a source of unlimited amount of information.

The notations used in this paper are summarized in Tab.1. Search version of SAT problem is defined as follows:

SAT – SEARCH VERSION
INPUT: sums $k_j, j = 1, \dots, m$, of binary variables, or their negations, chosen over a set of n binary variables $x_1, \dots, x_n$. The input data is SAT instance I. Let $|I|$ denote instance I size, i.e., length of the string encoding I according to some reasonable rule.

REQUEST: Find the assignment of values 0/1 to binary variables $x_1, \dots, x_n$, i.e. vector $\overline{x}$ of n 0/1 values, such that the conjunction of the clauses $F(I, \overline{x}) = \prod_{j=1}^{m} k_j$ is 1. If such a vector does not exist then signal $\emptyset$.

If the binary vector $\overline{x}$ such that $F(I, \overline{x}) = 1$ exists then we will be saying that I is a "yes" instance. Otherwise I is a "no" instance. The input sums k_j will be alternatively referred to as clauses. If clauses k_j comprise exactly three variables we will say that it is a 3-SAT problem instance.

Definition 1 *Fixed code algorithm is an algorithm which is encoded in limited number of immutable bits.*

Thus, a fixed code algorithm does not change its code during the runtime. Furthermore, it has no access to a source of randomness and is deterministic. Let $|A|$ denote the size of algorithm A code in bits.

Postulate 2 *Information is not created* ex nihilo *by fixed code algorithms.*

Postulate 3 *An algorithm to solve a problem must be capable of representing at least the same amount of information as the amount of the information in the problem.*

Definition 4 Truly random bit sequence *(TRBS) is a sequence of bits, that no bit can be computed on the basis of the other bits.*

Thus, for a TRBS of length N it is not possible to compute the ith bit on the basis of bits $\{1, \dots, N\} \setminus \{i\}$. In effect, a TRBS cannot be compressed, and the only way to represent it is to store it in its whole entirety on N bits.

Postulate 5 *Truly random bit sequences exist.*

2 SAT Polynomial-time Solvability

SAT can be solved in $O(|I|)$ time by referring to precomputed solutions. In more detail, the search for a precomputed solution of I can be conducted in a binary tree with $2^{|I|}$ leaves and $2^{|I|}-1$ internal nodes using pointers (addresses) of length $|I|+1$ to arrive at the leaves. An internal node holds two pointers to its successors. A leaf holds an answer to a SAT solution ($\overline{x}$ or $\emptyset$). The data-structure has size $O(2^{|I|}|I|)$ because it has $2^{|I|+1}-1$ nodes each holding at most $2(|I|+1)$ bits.

The tree can be traversed top-down in $O(|I|)$ time. Thus, *SAT can be solved in polynomial time*, at least in principle, provided that an algorithm for SAT has unlimited (precisely, exponential in $|I|$) amount of information.

3 Amount of Information in SAT

Let Σ^+ be a set of strings encoding instances of SAT using some reasonable encoding scheme e over alphabet Σ. An empty string $\epsilon \notin \Sigma^+$. SAT-search is an example of a search problem, while search problems are string relations [4]:

Definition 6 *A search problem* Π *is a string relation*

$$R[\Pi, e] = \left\{ (a,b) : \begin{array}{l} a \in \Sigma^+ \textit{ is the encoding of an instance } I \in D_\Pi \textit{ and} \\ b \in \Sigma^+ \textit{ is the encoding of a solution } s \in S_\Pi(I) \\ \textit{under coding scheme } e \end{array} \right\},$$

where $S_\Pi(I)$ *is a set of solutions for instance* I *of* Π.

Thus, SAT can be thought of as a mapping from strings a representing instances to strings b representing solutions. Each string a is either encoding a "yes" instance, or not. In the former case an n-bit solution $\overline{x}$ must be provided by the mapping. In the latter case a string representing $S_{SAT}(I)=\emptyset$ must be provided. If the a string is not encoding any SAT instance, then such a case can be represented in the same was as a negative answer $\emptyset$. In order to encode each (a,b) pair of the relation representing SAT it is necessary to have an equivalent of a graph arc from string a to its solution b. Such an arc requires $|I|+n$ bits of information which is at least $\Omega(|I|)$ bits. There are $2^{|I|}$ strings of some size $|I|$. Since it is necessary to at least distinguish whether

b strings represent $\emptyset$ or $\overline{x}$, at least $\Omega(2^{|I|})$ bits of information are necessary to encode SAT as string relation $R[SAT, e]$.

An algorithm solving problem Π must represent a mapping from the instances to the solutions. The mapping requires a certain number of bits to be represented. This information must be provided in the input instance and the algorithm to solve the *problem* because information is not created ex nihilo in fixed code algorithms. Note that an algorithm solves a problem if it provides an answer *for each* input instance [4]. The amount of information in a fixed code algorithm A solving SAT and in the input instance I is $|I| + |A|$ bits. Since $|A|$ is constant and for sufficiently large $|I|$, the instance and the algorithm together have less information than $\Omega(2^{|I|})$ bits necessary to represent SAT as a string relation. However, it is still possible that SAT is encoded in fewer than $\Omega(2^{|I|})$ bits. Hence, the above consideration does not preclude existence of some more compact representation of SAT. In other words, SAT could possibly be expressed in a size smaller than $|I| + |A|$ bits, for some fixed code algorithm A.

Theorem 7 *The amount of information in SAT grows exponentially with instance size.*

Proof. Assume there are n variables and $4n$ clauses in 3-SAT. Let there be 4 clauses $k_{i1} = x_a + x_b + \widetilde{x_i}, k_{i2} = \overline{x_a} + x_b + \widetilde{x_i}, k_{i3} = x_a + \overline{x_b} + \widetilde{x_i}, k_{i4} = \overline{x_a} + \overline{x_b} + \widetilde{x_i}$ for each $i = 1, \dots, n$. $\widetilde{x_i}$ denotes that variable x_i is with or without negation. No valuing of x_a, x_b makes the four clauses simultaneously equal 1. The four clauses may simultaneously become equal 1 only if $\widetilde{x_i} = 1$. Satisfying formula $F = k_{11}k_{12}k_{13}k_{14} \dots k_{n4}$ depends on valuing of variables $\widetilde{x_i}$ for $i = 1, \dots, n$. Depending on whether x_i is negated or not there can be 2^n different ways of constructing formula F, thus leading to 2^n different "yes" instances with 2^n different solutions. Variables x_a, x_b are chosen such that $a \neq b$ and $a, b \neq i$. Since there are $(n-1)(n-2)/2$ possible pairs a, b for each i, it is possible to generate pairs a, b satisfying the above conditions for $n \geq 3$.

We are now going to calculate the number of different "yes" instances as a function of instance size $|I|$. Suppose the uniform cost criterion [1] is assumed, then each number has value limited from above by constant K. The length of the encoding of the instance data is $|I| = 4n \times 3 \log K + \log K = 12n \log K + \log K$ because it is necessary to record the number of variables in $\log K$ bits, each binary variable induces 4 clauses of length $3 \log K$. Negation of a variable, or lack thereof, is encoded on one bit within $\log K$. Consequently, the number of possible unique solutions is $2^n = 2^{(|I|-\log K)/(12\log K)} = 2^{|I|/(12\log K)}2^{-1/12}$, which is $\Omega(2^{d_1|I|})$, where $d_1 = 1/(12\log K) > 0$ is constant.

Assume logarithmic cost criterion [1], then the number of bits necessary to record n is $\lfloor \log n \rfloor + 1$, and $\lfloor \log n \rfloor + 2$ bits are needed to encode the index of a variable and its negation, or lack thereof. Length of the encoding string is $|I| = 12n(\lfloor \log n \rfloor + 2) + \lfloor \log n \rfloor + 1 \leq 15n \log n = dn \ln n$, for $n > 2^{24}$ and $d = 15/\ln 2 \approx 21.6404$. An inverse function of $(cx \ln x)$, for some constant $c > 0$, is $\frac{x}{c}/W(\frac{x}{c})$, where W is Lambert W-function [7]. Lambert W function for big x can be approximated by $W(x) = \ln x - \ln\ln x + O(1)$. Given instance size $|I|$, we have $n \geq \frac{|I|}{d}/W(\frac{|I|}{d}) \approx \frac{|I|}{d}/(\ln\frac{|I|}{d} - \ln\ln\frac{|I|}{d} + O(1)) \geq \frac{|I|}{d}/(2\ln\frac{|I|}{d}) \geq \frac{|I|}{d}/(2\ln|I| - 2\ln d) \geq |I|/(2d\ln|I|)$, for sufficiently large $|I|$. Note that $|I|, dn\ln n, \frac{x}{c}/W(\frac{x}{c})$ are increasing in n, x. Thus, by approximating $|I|$ from above we get a lower bound of n after calculating an inverse of the upper bound of $|I|$. The number of possible unique solutions is $2^n \geq 2^{|I|/(d_2\ln|I|)}$ where $d_2 = 2d$. Observe that $2^{|I|/(d_2\ln|I|)}$ exceeds any polynomial function of $|I|$ for sufficiently large $|I|$, because for a polynomial function $O(|I|^k)$, $\ln(|I|^k) < |I|/(d_2\ln|I|)$ with $|I|$ tending to infinity.

Consider a truly random bit sequence (TRBS) of length 2^n. Assume that $j = 1, \ldots, 2^n$ is one of the instances of 3-SAT constructed in the above way. Let $j[i]$ for $i = 1, \ldots, n$ be the i-th bit of j binary encoding. If bit j of the TRBS is equal to 1 we set variables $\widetilde{x_i}$ such that $\widetilde{x_i} = j[i]$ satisfies clauses $k_{1i}, \ldots, k_{4i}$. For example, if $j[i] = 1$, then $\widetilde{x_i}$ is written as x_i. Thus, if bit j of the TRBS is equal to 1 then a "yes" instance is constructed. Conversely, if $j = 0$ then at least one variable x_i in the corresponding clauses $k_{1i}, \ldots, k_{4i}$ is set inconsistently, i.e., $\widetilde{x_i}$ appears in $k_{1i}, \ldots, k_{4i}$ both with negation and without. Hence, if bit j of the TRBS is equal to 0, the j-th instance constructed in the above way becomes a "no" instance. Note that in this way the TRBS of length 2^n was encoded in 3-SAT search problem. The amount of information in 3-SAT grows at least in the order of $\Omega(2^{d_1|I|})$ for uniform ($\Omega(2^{|I|/(d_2\ln|I|)})$, for logarithmic) cost criterion. □

4 On Consequences

Note that by Theorem 7, SAT problem has amount of information that grows at least exponentially with the size of the input instances. This information cannot be reduced because a TRBS of size exponentially growing with instance size is put in SAT.

Proposition 8 *Fixed code algorithm is not capable of representing SAT in polynomial time.*

Proof. The amount of information that can be produced by the polynomial-time algorithm running in time $p(|I|)$, where p is a polynomial, is $|I|+|A|+p(|I|)\log(p(|I|))$, where $|I|+|A|$ is the instance and algorithm information, while $p(|I|)\log(p(|I|))$ bits of information come from the progress of time because $p(|I|)$ values of time can be recorded on $\log(p(|I|))$ bits. It is still less information than comprised in SAT. □

Observation 9 *Problems in class* **NP** *have more information than their Kolmogorov complexity.*

Proof. Note that by Theorem 7 information content of SAT grows exponentially with the size of the instance because sufficiently long TRBS can be injected in SAT. For some given number of variables n a TRBS of length 2^n can be placed in SAT. However, the minimum amount of information required to represent SAT is at most $\log n+|E|+|V|$, where E is an algorithm that enumerates all input instances and all solutions according to some SAT instance encoding scheme, while V is the fixed code algorithm capable of verifying if a given solution for I is correct. Given n, it is possible to enumerate all strings encoding SAT. For example, SAT instance may be encoded as a sequence of values: $(n, m, k_1, \dots, k_m)$. Since each variable can be used with or without negation in clauses k_i, the number of possible clauses is $m \leq 2^{2n}$ which can be encoded in $2n$ bits. Each clause can be encoded as a sequence of n bit pairs representing at position $i=1,\dots,n$: 00_i or 01_i – variable x_i is absent in the current clause, 10_i – variable x_i is present in the current clause as x_i, 11_i – variable x_i is present in the current clause as $\overline{x_i}$. Thus, each clause can be encoded in $2n$ bits. All clauses of the instance can be encoded in $2nm \leq 2n\times 2^{2n}$ bits. The whole SAT instance can be encoded as a binary number of length $\log n+2n+2n\times 2^{2n}$ bits. All possible values of this number can be enumerated by a constant information size Turing machine adding 1 to a binary-encoded number recorded on the tape. Similarly it is possible to enumerate all 2^n potential solutions of a SAT instance with n variables. Thus, a fixed code algorithm E enumerating all input instances and all solutions exists. Algorithm V exists because SAT$\in$**NP**, and all problems in **NP**

have algorithms verifying given solutions in polynomial time. Hence, SAT as a string relation can be reconstructed by enumerating all input instances and choosing the correct answer by use of algorithm V.

On the one hand, Kolmogorov complexity of SAT is at most $\log n + |E| + |V|$. On the other hand, SAT has $\Omega(2^n)$ incompressible bits. By the example given in Theorem 7 SAT has $\Omega(2^{d_1|I|})$ (uniform criterion) or $\Omega(2^{|I|/(d_2 \ln |I|)})$ (logarithmic criterion) incompressible bits, while Kolmogorov complexity of SAT is $\log(d_1|I|) + |E| + |V|$ (uniform) or $\log(|I|/(d_2 \ln |I|)) + |E| + |V|$ (logarithmic criterion). Since by Cook's theorem SAT is a foundation of all **NP**-complete problems, the above observations can be extended to all problems in class **NP**. □

The discrepancy between these two numbers can be explained by the existence of algorithm V and the fact that information is created by the progress of time, the process of solution enumeration and filtering by algorithm V. Informally, SAT has exponential compression efficiency and SAT is information-inflated by solutions enumeration and verification.

Acknowledgments

I thank Joanna Berlińska, Małgorzta Sterna and Piotr Formanowicz for discussing with me in the earlier stages of this consideration [2].

References

[1] A.Aho, J.E.Hopcroft, J.D.Ullman, The Design and Analysis of Computer Algorithms, Addison-Wesley Publishing Company, Reading MA, 1974

[2] M.Drozdowski, On polynomial-time solvability and fixed code size algorithms, Research Report RA-06/16, Institute of Computing Science, Poznań University of Technology, 2016 http://www.cs.put.poznan.pl/mdrozdowski/rapIIn/MD-RA-6-16.pdf

[3] M.Fleischer, S.H.Jacobson, Information Theory and the Finite-Time Behavior of the Simulated Annealing Algorithm: Experimental Results, INFORMS Journal on Computing 11(1), Winter 1999.

[4] M.R.Garey, D.S.Johnson, *Computers and Intractability: A guide to the theory of NP-completeness*, Freeman, San Francisco, 1979.

[5] M.Kubale (ed.), Graph Colorings, American Mathematical Society, Providence, Rhode Island, 2004. (I used Polish version: Optymalizacja dyskretna. Modele i metody kolorowania grafów, WNT, Warszawa, 2002).

[6] K.Manuszewski, Grafy algorytmicznie trudne do kolorowania, Ph.D. Thesis, Gdańsk University of Technology, 1997.

[7] Eric W. Weisstein, Lambert W-Function, MathWorld–A Wolfram Web Resource. [accessed in September 2015]. `http://mathworld.wolfram.com/LambertW-Function.html`

[8] D.H.Wolpert, W.G. Macready, No Free Lunch Theorems for Optimization, IEEE Trans. on Evolutionary Computation 1(1), April 1997.

Version 3. January 1, 2024. Previous version is here.

Table 1: Summary of notations.

$\|A\|$	Size of algorithm A in bits according to some reasonable encoding rule
D_Π	set of instances for problem Π
F	$F = \prod_{j=1}^{m} k_j$ conjunction of clauses k_j
$F(I, \overline{x})$	value of F for instance I and bit assignment $\overline{x}$
I	instance of a problem
$\|I\|$	instance size, i.e., length of the string encoding instance I according to some reasonable encoding rule (e.g. numbers encoded at base greater or equal 2)
k_j	jth clause of SAT instance, for $j = 1, \ldots, m$
n	number of variables in the SAT problem
m	number of clauses in the SAT problem
$S_\Pi(I)$	set of solutions for instance I of search problem Π
x_i	ith variable in SAT problem, for $i = 1, \ldots, n$
$\widetilde{x}_i$	ith variable x_i with or without negation
$\overline{x}$	vector of n binary values, alternatively n-bit unsigned integer