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

Maciej Drozdowski

Institute of Computing Science, Poznań University of Technology, Poland

This talk is about:

- information content of combinatorial problems
- information content of algorithms
- exponential amount of information in satisfiability problem (SAT)
- polynomial-time solvability of SAT
- Kolmogorov complexity of SAT

Related work

There is a connection between the combinatorial optimization **algorithm performance** and the **amount of information**:

- between the fraction of problem instances achieving certain histogram of values and the entropy of the histogram[1]
- between entropy of the Markov chains representing behavior of simulated annealing algorithms and the convergence of the expected objective function value for maximum 3-SAT[2].
- A graph coloring *random sequential algorithm* visiting nodes in a random sequence and assigning the lowest feasible color cannot deterministically fail because it is connected to a source of unlimited amount of information[3].

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

[2] 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.

[3] M.Kubale (ed.), Graph Colorings, American Mathematical Society, Providence, Rhode Island, 2004

SAT

Definition

INPUT I: sums $k_j, j = 1, \dots, m$, of binary variables, or their negations, chosen over a set of n binary variables $x_1, \dots, x_n$.

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

$|I|$ – instance I size, i.e., length of the string encoding I

3-SAT – when clauses k_j comprise exactly three variables

"yes" instance – if for instance I : $\exists \overline{x} : F(I, \overline{x}) = 1$

"no" instance – otherwise

Fixed code algorithm

Definition

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,
- has no access to a source of randomness,
- is deterministic.

$|A|$ – the size of fixed code algorithm A code in bits.

Truly random bit sequence

Definition

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

Thus:

- a TRBS cannot be compressed,
- the only way to represent it is to store it in its whole entirety,
- an N-bit TRBS has information content N bits.

Postulate

Truly random bit sequences exist.

Information conservation

Postulate

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.

Postulate

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

Proposition

SAT can be solved in $O(|I|)$ time, *at least in principle, provided the algorithm for SAT has exponential amount of information.*

SAT can be solved in $O(|I|)$ time by referring to precomputed solutions, e.g.:

- use a binary tree with $2^{|I|}$ leaves and $2^{|I|}-1$ internal nodes,
- with pointers (addresses) of length $|I|+1$,
- an internal node holds two pointers to its successors,
- a leaf holds a SAT solution ($\overline{x}$ or $\emptyset$),
- the tree can be traversed top-down in $O(|I|)$ time,
- such a data-structure has size $O(2^{|I|}|I|)$ because it has $2^{|I|+1}-1$ nodes holding at most $2(|I|+1)$ bits.

SAT as a string relation

Let:
Σ – an alphabet
e – some reasonable encoding scheme over Σ,
Σ^+ – a set of strings encoding instances of SAT using scheme e over alphabet Σ.

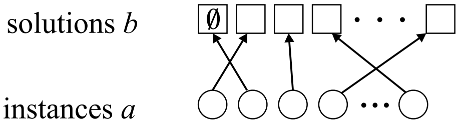
SAT-search is an example of a string relation:

Definition

Search problem Π is a string relation

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

SAT as a string relation



Thus, SAT can be thought of as:

- a mapping from strings $a =$ instances to strings $b =$ solutions,
- set of arcs from instances a to solutions b,
- an arc requires $|I| + n$ bits of information,
- there are $2^{|I|}$ strings of size $|I|$,
- at least $\Omega(2^{|I|})$ bits of information seem necessary to encode SAT as string relation $R[SAT, e]$.
- However, SAT as a string relation can be *compressed*.

SAT and Fixed Code Algorithm

- An algorithm solving SAT must provide an answer **for each** input instance.
- $\Rightarrow$ An algorithm for SAT 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 I and the algorithm A because information is not created *ex nihilo*.
- The amount of information in a fixed code algorithm A and in the input instance I is $|I| + |A|$ bits.
- For sufficiently large $|I|$, the instance and the algorithm have less information than $\Omega(2^{|I|})$ bits necessary to represent SAT as a string relation.
- However, can SAT be compressed to fewer than $\Omega(2^{|I|})$ bits?

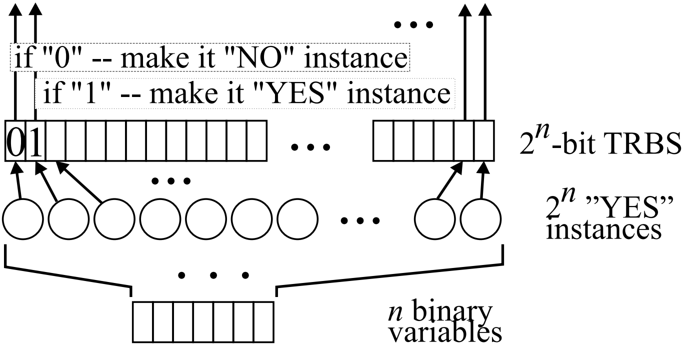
Exponential Information Content of SAT

Theorem

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

Exponential Information Content of SAT

Proof: the idea



Exponential Information Content of SAT

Proof: Exponential number of "Yes" instances

- $\widetilde{x_i}$ – a variable x_i with or without negation
- n – number of variables, $4n$ – number of clauses
- Let there be 4 clauses for $i = 1, \ldots, n$:
  $k_{i1} = x_a + x_b + \widetilde{x_i}$, $k_{i2} = \overline{x_a} + x_b + \widetilde{x_i}$,
  $k_{i3} = x_a + \overline{x_b} + \widetilde{x_i}$, $k_{i4} = \overline{x_a} + \overline{x_b} + \widetilde{x_i}$.
- No valuing of x_a, x_b alone makes the four clauses simultaneously equal 1.
- The four clauses may simultaneously become equal 1 only if $\widetilde{x_i} = 1$.
- Satisfying formula $F = k_{11}k_{12}k_{13}k_{14}\ldots k_{n4}$ depends on valuing of variables $\widetilde{x_i}$ for $i = 1, \ldots, n$.
- There are 2^n different ways of constructing formula F, 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 $a, b \neq i$.

Exponential Information Content of SAT

Proof: From the number of variables n to instance size $|I|$
Uniform cost criterion:

- uniform criterion $\Rightarrow$ each number has value limited from above by some constant K,
- 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 $\log K$,
- $\Rightarrow$ the length of the encoding of the instance data is $|I| = 4n \times 3 \log K + \log K = 12n \log K + \log K$
- $\Rightarrow$ 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.

Exponential Information Content of SAT

Proof: From the number of variables n to instance size $|I|$
Logarithmic cost criterion:

- the number of bits necessary to record n is $\lfloor \log n \rfloor + 1$
- $\lfloor \log n \rfloor + 2$ bits are needed to encode the index of a variable and its negation, or lack thereof.
- $|I| = 12n(\lfloor \log n \rfloor + 2) + \lfloor \log n \rfloor + 1 \leq 15n \log n = dn \ln n$, 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[4].
- Lambert W function for big x can be approximated by $W(x) = \ln x - \ln \ln x + O(1)$.

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

Exponential Information Content of SAT

Proof: From the number of variables n to instance size $|I|$
Logarithmic cost criterion, cont.:

- given instance size $|I|$, we have $n \geq \frac{|I|}{d} / W(\frac{|I|}{d}) \approx \frac{|I|}{d} / (\ln \frac{|I|}{d} - \ln \ln \frac{|I|}{d} + O(1)) \geq \frac{|I|}{d} / (2 \ln \frac{|I|}{d}) \geq \frac{|I|}{d} / (2 \ln |I| - 2 \ln d) \geq |I| / (2d \ln |I|)$,
- the number of possible unique solutions is $2^n \geq 2^{|I|/(d_2 \ln |I|)}$ where $d_2 = 2d = 30/\ln 2$.

Observe that $2^{|I|/(d_2 \ln |I|)}$ exceeds any polynomial function of $|I|$ for sufficiently large $|I|$, because for a polynomial function $O(|I|^k)$, $\ln(|I|^k) < |I|/(d_2 \ln |I|)$.

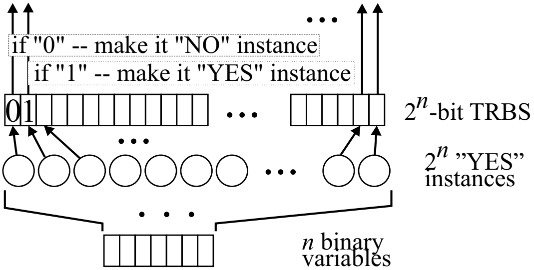
Exponential Information Content of SAT

Proof: Length 2^n TRBS injection into SAT by wrapping TRBS around the 2^n instances

- Consider a truly random bit sequence (TRBS) of length 2^n.
- Consider $j = 1, \dots, 2^n$ instances with clauses $k_{1i}, \dots, k_{4i}$ and variables $\widetilde{x}_i$ as constructed above.
- If TRBS bit j is 1, then the jth instance is constructed as "yes" instance by setting $\widetilde{x}_i$ in $k_{1i}, \dots, k_{4i}$ consistently with j binary encoding.
  Example: $j = 5, (j)_2 = 101$ set $\widetilde{x}_0 = x_0$ in $k_{10}, \dots, k_{40}$; $\widetilde{x}_1 = \overline{x_1}$, in $k_{11}, \dots, k_{41}$; $\widetilde{x}_2 = x_1$ in $k_{12}, \dots, k_{42}$.
- If TRBS bit j is 0, then the jth instance is constructed as "no" instance by setting some $\widetilde{x}_i$ variable in $k_{1i}, \dots, k_{4i}$ inconsistently with j binary encoding.
  Example: set $\widetilde{x}_0 = x_0$ in k_{10} and $\widetilde{x}_0 = \overline{x_0}$ in k_{20}.

Exponential Information Content of SAT

Proof: finishing



A 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 criterion,
- $\Omega(2^{|I|/(d_2 \ln |I|)})$, for logarithmic cost criterion.

□

Fixed code algorithm capability

Proposition

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

Proof.

- assume the runtime is $p(|I|)$, where p is a polynomial,
- assuming limited random number acquisition speed v, $v \times p(|I|)$ bits of information come from the progress of time which is $O(p(|I|))$ bits of information,
- the total amount of information in instance I, the polynomial-time algorithm A, running in time $p(|I|)$ is $|I| + |A| + O(p(|I|))$,
- which is less than $\Omega(2^{d_1|I|})$ bits of information in SAT ($\Omega(2^{|I|/(d_2 \ln |I|)})$, for logarithmic cost criterion). $\square$

Strange case of SAT Kolmogorov complexity

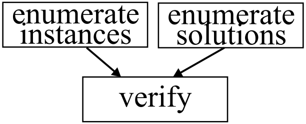
Kolmogorov complexity of some object[5] (informally):
"*the length of a shortest computer program ... that produces the object as output.*"

Observation

SAT has constant Kolmogorov complexity.

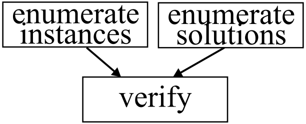
[5] Kolmogorov complexity, https://en.wikipedia.org/wiki/Kolmogorov_complexity visited Feb. 13, 2024.

SAT Kolmogorov complexity



- We show that fixed code algorithms can recreate SAT as a set of instance-solution links of a string relation.
- SAT instances can be encoded as strings $(n, m, k_1, \ldots, k_m)$, where: n – number of variables, $m < 2^{2n}$ – number of clauses, $k_1, \ldots, k_m$ – the clauses.
- $\Rightarrow$ SAT instance can be perceived as a $(\log n + 2n + 2n \times 2^{2n})$-bit binary number
- SAT solutions can be encoded on n bits

SAT Kolmogorov complexity



- $\Rightarrow$ all SAT instance-solution pairs can be enumerated by fixed code Turing machine E adding 1 to a binary number on the tape.
- Since SAT$\in$**NP**, an algorithm V exists verifying given solutions in polynomial time.
- Hence, the amount of information required to reconstruct SAT is $O(|E| + |V|)$. $\square$

Strange case of SAT Kolmogorov complexity

- SAT has constant Kolmogorov complexity and at least exponential information content ($\Omega(2^{d_1|I|})$ for uniform or $\Omega(2^{|I|/(d_2 \ln |I|)})$ for logarithmic criterion).
- Can the discrepancy between these two numbers be explained by the the existence of algorithm V and information extraction from the progress of time ??.
- Informally, SAT has exponential compression efficiency and SAT is information-inflated by solutions enumeration and verification.
- Since by Cook's theorem SAT is a foundation of all **NP**-complete problems, can the above observations can be extended to all problems in class **NP**?

Finish

That's all.
Thank you for listening.

Full text available from
https://arxiv.org/abs/2401.00947