Generic NP-incomplete problems

Alexander Rybalov
Omsk State Technical University, prospekt Mira 11, Omsk 644050, Russia.
E-mail: alexander.rybalov@gmail.com

Abstract. Ladner in 1975 proved that if $P \neq NP$ then there exist problems in NP, which are not decidable in polynomial time and not NP-complete. Kapovich, Myasnikov, Schupp and Shpilrain in 2003 developed a theory of generic-case complexity. Generic-case approach considers an algorithmic problem on “most” of inputs instead of all domain and ignores its behaviour on the rest of inputs. In this paper we prove a generic analog of the Ladner’s theorem: If $P \neq NP$ and $P = BPP$, then there exists a set $S \in NP$, such that $S$ is not strongly generically decidable in polynomial time and not generically NP-complete. Supported by Russian Science Foundation, grant 17-11-01117.

1. Introduction
Polynomial reducibility of algorithmic problems is one of the most important notion in classical computational complexity theory. It enables to compare the complexity of algorithmic problems and to develop rich theory of NP-completeness. Excellent book of Garey and Johnson [1] on NP-complete problems was recognized in 2006 as the most cited reference in computer science literature. It is amazingly, that the majority of algorithmic problems, interesting both from a practical and a theoretical point of view, are classified either as polynomial time decidable problems or as NP-complete problems. Ladner in [4] proved that if $P \neq NP$ then there exist problems in NP, which are not decidable in polynomial time and not NP-complete. Perhaps, famous graph isomorphism problem is the one such NP-intermediate problem.

In [3] a theory of generic-case complexity was developed. Generic-case approach considers an algorithmic problem on “most” of inputs instead of all domain and ignores its behaviour on the rest of inputs. This generic complexity approach has numerous applications, especially in cryptography, where cryptosystems must be based on problems which are hard on almost all (random) instances. Usually such behavior of problems is studied in terms of average-case complexity, but generic-case complexity and generic computability are much more convenient in applications, in particular it allow to study complexity of undecidable problems. Rybalov in [5] defined a notion of generic polynomial reducibility and proved that the Boolean satisfiability problem is generically NP-complete.

In this paper we prove a generic analog of the Ladner’s theorem: If $P \neq NP$ and $P = BPP$, then there exists a set $S \in NP$, such that $S$ is not strongly generically decidable in polynomial time and not generically NP-complete. Here class BPP consists of problems, decidable in polynomial time by probabilistic Turing machines. The equation $P = BPP$ means that every polynomial probabilistic algorithm can be effectively derandomized to an equivalent polynomial deterministic algorithm. There are significant results toward the proving of equation $P = BPP$ (see [2]), but this hypothesis is still unproven.
2. Preliminaries

Let $I$ be the set of all inputs and $I_n$ be the set of all inputs of size $n$ (sphere of radius $n$). For a subset $S \subseteq I$ define the following sequence

$$\rho_n(S) = \frac{|S \cap I_n|}{|I_n|}, \ n = 1, 2, 3, \ldots$$

The value $\rho_n(S)$ is probability to generate an input from $S$ during random and uniform generation of inputs from sphere $I_n$. The asymptotic density of $S$ is the following limit (if it exists)

$$\rho(S) = \lim_{n \to \infty} \rho_n(S).$$

$S$ is called generic if $\rho(S) = 1$ and negligible if $\rho(S) = 0$. Clearly, $S$ is generic if and only if its complement in $I$ is negligible.

Following [3] we call a set $S$ strongly negligible if sequence $\rho_n(S)$ exponentially fast converges to 0, i.e. there are constants $0 < \sigma < 1$ and $C > 0$ such that for every $n$

$$\rho_n(S) < C\sigma^n.$$

Now $S$ is called strongly generic if its complement is strongly negligible.

A set $S \subseteq I$ is generically decidable in polynomial time if there exists a set $G \subseteq I$ such that

(i) $G$ is generic,
(ii) $G$ is decidable in polynomial time,
(iii) $S \cap G$ is decidable in polynomial time.

If $G$ is strongly generic, then $S$ is called strongly generically decidable in polynomial time. A generic algorithm $A$ for $S$ works on an input $x \in I$ in the following way. At first $A$ decides whether $x \in G$. If $x \in G$ then $A$ can decide $S$ on $G$, else $A$ says "I don’t know". So $A$ correctly decides $S$ on "almost all" inputs (inputs from generic set).

Let $I, J$ be sets of inputs. A set $A \subseteq I$ is generically polynomially reducible to a set $B \subseteq J$ (written $A \leq_{\text{GenP}} B$), if there are a polynomial probabilistic algorithm $R : I \times \mathbb{N} \to P(J) \cup \{?, !\}$ (here $P(J)$ is the set of all subsets of $J$), a polynomial $p(n)$, a polynomial $q(n)$ of degree greater than 2 and a constant $C > 0$ such that

(i) for all $x \in I$ either $(\forall n \ R(x, n) = ?)$, or for all $n \geq q(k)$, where $k = \text{size}(x)$, it holds:
   (a) $\forall y \in R(x, n) \ (y \neq ! \Rightarrow \text{size}(y) = n)$;
   (b) $R$ outputs all elements of $R(x, n) \setminus \{!\}$ uniformly;
   (c) the probability of ($R(x, n) = !)$ is not greater than $2^{-Ck}$;
   (d) $\frac{|R(x, n)|}{|J_n|} < \frac{1}{(p(n))^k}$;
   (e) $x \in A \Rightarrow R(x, n) \subseteq B$;
   (f) $x \notin A \Rightarrow R(x, n) \subseteq J \setminus B$;

(ii) the set $\{x \in I : \forall n \ (R(x, n) = ?)\}$ is strongly negligible.

Now define a generic analog of class NP. A set $S \subseteq I$ belongs to the class sgNP, if there exists a polynomial strongly generic set $G \subseteq I$ such that $S \cap G \in \text{NP}$. A set $S \in \text{sgNP}$ is generically NP-complete, if $A \leq_{\text{GenP}} S$ for all $A \in \text{sgNP}$. 

3. Representation of Boolean formulas

Classical representation of the Boolean formulas by truth tables is not practical because the size of this representation is exponential in the number of Boolean variables. We will represent Boolean formulas by labeled binary trees. This more compact way of representation is used in many real programs and it is convenient for calculations.

Let \( \varphi \) be a Boolean formula in the basis \{\( \lor, \land, \neg \)\}. Without loss of generality we will assume that all negations in \( \varphi \) are only above variables. One can naturally associate with the formula \( \varphi \) a binary tree \( T_\varphi \) that presents the construction of \( \varphi \) from Boolean variables and their negations by means of disjunctions and conjunctions. The internal vertices of \( T_\varphi \) are labeled by symbols \( \lor \) and \( \land \), and the leafs of \( T_\varphi \) are labeled by Boolean variables or their negations. Conversely, given a binary tree \( T \) as above one can uniquely reconstruct a Boolean formula \( \varphi_T \). This gives a one-to-one representation of the Boolean formulas \( \varphi \) by the binary trees \( T_\varphi \). If \( T_\varphi \) has \( n \) leaves then at most \( n \) variables may occur in \( T_\varphi \), so we may assume from the beginning that all variables in \( T_\varphi \) belong to the set \( x_1, \ldots, x_n \). The size of formula \( \varphi \) is the number of leafs in \( T_\varphi \).

From now on we identify Boolean formulas \( \varphi \) with their trees \( T_\varphi \). Denote by \( \mathcal{F} \) the set of all Boolean formulas, and by \( \mathcal{F}_n \) the set of all formulas in \( \mathcal{F} \) of size \( n \).

**Lemma 1.** \( |\mathcal{F}_n| = 2^{n-1}(2n)^n C_{n-1} \), where \( C_n = \frac{1}{n+1} \binom{2n}{n} \) is the \( n \)-th Catalan number.

**Proof.** Any formula from \( \mathcal{F}_n \) is a binary labeled tree with \( n \) leafs and \( n - 1 \) internal vertices. There are (see [?]) \( C_{n-1} \) non-labeled binary trees with \( n \) leafs. Every internal vertex of such tree can be labeled by \( \lor \) or \( \land \), so there are \( 2^{n-1} \) such labelings. Every leaf can be labeled by variables \( x_1, \ldots, x_n \) or its negations, so there are \( (2n)^n \) such labelings. So \( |\mathcal{F}_n| = 2^{n-1}(2n)^n C_{n-1} \). \( \square \)

For every Boolean formula \( \varphi \) define the set

\[
\text{Eq}(\varphi) = \{ \varphi \lor ((x_1 \land \neg x_1) \land \psi), \, \psi \text{ — arbitrary Boolean formula} \}.
\]

It is clear, that \( \varphi \) is satisfiable if and only if every formula from \( \text{Eq}(\varphi) \) is satisfiable.

**Lemma 2.** For any Boolean formula \( \varphi \) it holds

\[
\frac{|\text{Eq}(\varphi) \cap \mathcal{F}_n|}{|\mathcal{F}_n|} > \frac{1}{(16(n-2))^k}
\]

for any \( n > k + 2 \), where \( k \) is the size of formula \( \varphi \).

**Proof.** Let formula \( \varphi \) has size \( k \). Then for any formula \( \varphi \land \psi \) from \( \text{Eq}(\varphi) \cap \mathcal{F}_{n+2} \) formula \( \psi \) has size \( n - k \). Also \( \psi \) can contain any variable from \( x_1, \ldots, x_n \) As in the proof of Lemma 1 we can count

\[
|\text{Eq}(\varphi) \cap \mathcal{F}_{n+2}| = 2^{n-k-1}(2n)^{n-k} C_{n-k-1}.
\]

Therefore

\[
\frac{|\text{Eq}(\varphi) \cap \mathcal{F}_{n+2}|}{|\mathcal{F}_{n+2}|} = \frac{2^{n-k-1}(2n)^{n-k} C_{n-k-1}}{2^{n-1}(2n)^n C_{n-1}} = \frac{1}{(4n)^k} \cdot \frac{C_{n-k-1}}{C_{n-1}} = \frac{1}{(4n)^k} \cdot \frac{n}{n-k} \cdot \frac{(2n-k-1)}{(2n-1)} >
\]

\[
> \frac{1}{(4n)^k} \cdot \frac{(n-1)!}{(n-k-1)!} \cdot \frac{2(n-k-1) \ldots (n-k)}{2(n-1) \ldots n} = \frac{1}{(4n)^k} \cdot \frac{(n-1) \ldots (n-k)}{2(n-1) \ldots (2n-2k-1)} =
\]

\[
> \frac{1}{(4n)^k} \left( \frac{(n-1) \ldots (n-k)}{2(n-1) \ldots (2n-2k-1)} \right)^2 > \frac{1}{(4n)^k} \cdot \frac{1}{2^{2k}} = \frac{1}{(16n)^k}.
\]
So we can bound
\[
\frac{|Eq(\varphi)_{n+2}|}{|\mathcal{F}_{n+2}|} > \frac{1}{(16n)^k},
\]
and finally
\[
\frac{|Eq(\varphi)_n|}{|\mathcal{F}_n|} > \frac{1}{(16(n - 2))^k}.
\]

4. Generic Ladner’s theorem

**Theorem 1.** If \( P \neq NP \) and \( P = BPP \), then there exists a set \( S \) such that

(i) \( S \in sgNP \),
(ii) \( S \) is not strongly generically decidable in polynomial time,
(iii) \( S \) is not generically NP-complete.

**Proof.** Denote the set of all satisfiable Boolean formulas by SAT. It follows from the proof of original Ladner’s theorem (see [4]) that there exists a set \( A \subseteq SAT \) such that \( A \in NP, A \notin P \) and \( A \) is not NP-complete. Consider the following set
\[
Eq(A) = \bigcup_{\varphi \in A} Eq(\varphi).
\]

We prove that \( Eq(A) \) is a set we need to construct. At first note, that \( Eq(A) \in NP \), because \( A \in NP \). Therefore \( Eq(A) \in sgNP \).

Now prove that \( Eq(A) \) is not strongly generically decidable in polynomial time. Suppose, to the contrary, that there are a strongly generic polynomial set of formulas \( G \) and a polynomial algorithm \( A \), deciding \( Eq(A) \) for every formula \( \varphi \in G \). We construct a polynomial probabilistic algorithm \( B \), deciding \( A \) for every formula \( \varphi \). Since \( P = BPP \), then there is a deterministic polynomial algorithm deciding \( A \) – a contradiction with \( A \notin P \). Algorithm \( B \) works on a formula \( \varphi \) of size \( n \) in the following way.

(i) \( B \) checks does \( \varphi \) belong to \( G \). If \( \varphi \in G \), then \( B \) determines whether \( \varphi \in A \) by algorithm \( A \) and stops. If \( \varphi \notin G \), then go to step 2.

(ii) \( B \) generates a random formula \( \psi \in Eq(\varphi) \) of size \( n^2 \).

(iii) \( B \) checks does \( \psi \in G \)? If \( \psi \in G \) \( B \) determines whether \( \varphi \in A \) by algorithm \( A \) and stops.

(iv) If \( \psi \notin G \), then \( B \) outputs NO.

Note that polynomial algorithm \( B \) outputs correct answer on steps 1 and 3, but can output incorrect answer on step 4. We need to prove that the probability of step 4 is less than \( 1/2 \). The probability that a random formula \( \psi \in Eq(\varphi) \) of size \( n^2 \) does not belong to \( G \) is at most
\[
\frac{|(\mathcal{F} \setminus G)_{n^2}|}{|Eq(\varphi)_{n^2}|} = \frac{|(\mathcal{F} \setminus G)_{n^2}|}{|\mathcal{F}_{n^2}|} \times \frac{|\mathcal{F}_{n^2}|}{|Eq(\varphi)_{n^2}|}.
\]

Since \( G \) is a strongly generic set, then there exists a constant \( \alpha > 0 \) such that
\[
\frac{|(\mathcal{F} \setminus G)_{n^2}|}{|\mathcal{F}_{n^2}|} < \frac{1}{2\alpha n^2}
\]
for all \( n \). By Lemma 2
\[
\frac{|\mathcal{F}_{n^2}|}{|Eq(\varphi)_{n^2}|} < (16(n^2 - 2))^n.
\]
Therefore the probability we need is less than

$$\frac{(16(n^2 - 2))^n}{2^\alpha n^2} < \frac{(32n^2)^n}{2^\alpha n^2} = \frac{2^{5n + 2n \log n}}{2^\alpha n^2} < \frac{1}{4}$$

for large enough $n$. So the probability of step 3 is less than $1/2$.

Now prove that $Eq(A)$ is not generically NP-complete. Assume, to the contrary, that $Eq(A)$ is generically NP-complete. Then for any $S \in \text{NP}$, $S \leq_{\text{GenP}} Eq(A)$ and there is a polynomial probabilistic algorithm $R : I \times \mathbb{N} \rightarrow P(\mathcal{F}) \cup \{?,!\}$ such that $x \in S \Rightarrow R(x,n) \subseteq Eq(A)$ and $x \notin A \Rightarrow R(x,n) \subseteq \mathcal{F} \setminus Eq(A)$ for large enough $n$. Since $P = \text{BPP}$, there is a deterministic polynomial algorithm $A$ such that $x \in S \Leftrightarrow A(x) \in Eq(A)$, i.e. $S$ is polynomially reducible to $Eq(A)$. But obviously, $Eq(A)$ is polynomially reducible to $A$, so $S$ is polynomially reducible to $A$. A contradiction. \hfill \Box

5. Summary

We proved a generic analog of the Ladner’s theorem: If $P \neq \text{NP}$ and $P = \text{BPP}$, then there exists a set $S \in \text{NP}$, such that $S$ is not strongly generically decidable in polynomial time and not generically NP-complete. This result can be applied in cryptography for generic and average-case complexity analysis of public-key cryptosystems, widely used in Internet security and blockchain technology.

[1] Garey M and Johnson D 1979 Computers and intractability: a guide to the theory of NP-completeness. W. H. Freeman & Co. New York, NY, USA
[2] Impagliazzo R and Wigderson A 1997 P=\text{BPP} unless E has subexponential circuits: derandomizing the XOR lemma Proceedings of the 29th STOC pp 220–229
[3] Kapovich I Myasnikov A Schupp P and Shpilrain V 2003 Generic-case complexity, decision problems in group theory and random walks J. Algebra 264(2) pp 665–694
[4] Ladner R 1975 On the structure of polynomial time reducibility Journal of the ACM 22 pp 155–171
[5] Rybalov A 2017 On generic NP-completeness of the Boolean satisfiability problem Prikladnaya Diskretnaya Matematika 36 pp 106–112 (in Russian)