МАТЕМАТИЧЕСКИЕ ОСНОВЫ ИНФОРМАТИКИ И ПРОГРАММИРОВАНИЯ

УДК 510.52

О ГЕНЕРИЧЕСКОЙ NP-ПОЛНОТЕ ПРОБЛЕМЫ ВЫПОЛНИМОСТИ БУЛЕВЫХ СХЕМ[1]

А. Н. Рыбалов

Институт математики им. С. Л. Соболева СО РАН, г. Омск, Россия

Генерический подход к алгоритмическим проблемам предложен Каповичем, Мясниковым, Шуппом и Шпильрайном в 2003 г. В рамках этого подхода алгоритмическая проблема рассматривается не на всём множестве входов, а на некотором подмножестве «почти всех» входов. Понятие «почти все» формализуется введением естественной меры на множестве входных данных. В 2017 г. А. Н. Рыбалов ввёл понятие полиномиальной генерической сводимости алгоритмических проблем, которое сохраняет свойство разрешимости проблемы для почти всех входов и обладает свойством транзитивности, и доказал, что классическая проблема выполнимости булевых формул является полной относительно этой сводимости в генерическом аналоге класса NP. При этом булевы формулы представлялись в виде двоичных размеченных деревьев. В данной работе доказывается генерическая NP-полнота проблемы выполнимости для булевых схем.

Ключевые слова: *булева схема, генерическая сложность, проблема выполнимости, NP-полнота.*

DOI 10.17223/20710410/47/8

ON GENERIC NP-COMPLETENESS OF THE PROBLEM OF BOOLEAN CIRCUITS SATISFIABILITY

A. N. Rybalov

Sobolev Institute of Mathematics, Omsk, Russia

E-mail: alexander.rybalov@gmail.com

Generic-case approach to algorithmic problems was suggested by Miasnikov, Kapovich, Schupp and Shpilrain in 2003. This approach studies behavior of an algorithm on typical (almost all) inputs and ignores the rest of inputs. In 2017 A. Rybalov introduced a concept of polynomial generic reducibility of algorithmic problem that preserves the decidability property problems for almost all inputs and has the property of transitivity, and proved that the classical problem of the satisfiability of Boolean formulas is complete with respect to this reducibility in the generic analogue of class NP. Then the Boolean formulas were represented by binary labeled trees. In this paper, we prove the generic NP-completeness of the satisfiability problem for the so-called Boolean circuits. Boolean circuit is a way to represent Boolean functions, which show how the

[1]Работа поддержана грантом РНФ № 18-71-10028.

value of a Boolean function is obtained from values of variables using logical connectives. Boolean circuits are convenient models for the development of microprocessors, and are also the most important object of studying in computational complexity theory. Boolean circuit contains a finite number of variables $x_1, \ldots, x_n$. Every variable x_i can be either input, or defined through other variables by assigning one of the following types: $x_i = x_j \vee x_k$ or $x_j \wedge x_k$, where $j, k < i$; $x_i = \neg x_j$ or x_j, where $j < i$. The last variable x_n of the circuit is called output. By the size of a Boolean circuit we mean the number of variables in it. The number of Boolean circuits of size n is $\prod_{m=1}^{n} (1 + 2(m-1)^2 + 2(m-1))$.

Keywords: *Boolean circuit, generic complexity, Boolean satisfiability problem, NP-completeness.*

Введение

Генерический подход к алгоритмическим проблемам предложен в [1]. В рамках этого подхода алгоритмическая проблема рассматривается не на всём множестве входов, а на некотором подмножестве «почти всех» входов. Такие входы образуют так называемое генерическое множество. Понятие «почти все» формализуется введением естественной меры на множестве входных данных. С точки зрения практики алгоритмы, решающие быстро проблему на генерическом множестве, так же хороши, как и быстрые алгоритмы для всех входов. Классическим примером такого алгоритма является симплекс-метод — он за полиномиальное время решает задачу линейного программирования для большинства входных данных, но имеет экспоненциальную сложность в худшем случае. Более того, может так оказаться, что проблема трудноразрешима или вообще неразрешима в классическом смысле, но легкоразрешима на генерическом множестве.

Важнейшим понятием классической теории сложности вычислений является понятие полиномиальной сводимости алгоритмических проблем. С его помощью можно сравнивать проблемы по вычислительной сложности и развивать богатую теорию NP-полноты [2]. Л. Левин ввёл понятие полиномиальной сводимости и NP-полноты в среднем [3]. Ю. Гуревич привёл примеры алгоритмических проблем, которые являются NP-полными в среднем [4]. При анализе вычислительной сложности проблемы в среднем изучается математическое ожидание времени работы алгоритма для всех входов данного размера. А. Н. Рыбалов ввёл понятие полиномиальной генерической сводимости алгоритмических проблем, которое сохраняет свойство разрешимости проблемы для почти всех входов и обладает свойством транзитивности, и доказал, что классическая проблема выполнимости булевых формул является полной относительно этой сводимости в генерическом аналоге класса NP [5]. При этом булевы формулы представлялись в виде двоичных размеченных деревьев. Отметим, что А. Мясников и А. Ушаков в [6] дали другое определение генерической полиномиальной сводимости и в рамках этого определения доказали генерическую NP-полноту рандомизированной ограниченной версии проблемы остановки для машин Тьюринга.

В данной работе доказывается генерическая NP-полнота проблемы выполнимости для так называемых булевых схем. Булева схема — это способ представления булевых функций, в котором указывается, как значение булевой функции получается из значений переменных с помощью логических связок. Булевы схемы являются удобными моделями для разработки микропроцессоров, а также важнейшим объектом изучения в теории сложности вычислений [7].

1. Генерические алгоритмы

Пусть I — некоторое множество. Функция $\mathrm{size} : I \to \mathbb{N}$ называется *функцией размера*, если для любого $n \in \mathbb{N}$ множество $I_n = \{x \in I : \mathrm{size}(x) = n\}$ конечно. Например, если $I = \Sigma^*$ — множество слов над конечным алфавитом Σ, то функцией размера будет функция, определённая для любого слова w как его длина $|w|$. Для множества натуральных чисел $\mathbb{N}$ функция размера сопоставляет любому натуральному числу длину его двоичной записи. Как обычно делается в теории вычислимости, будем под алгоритмическими проблемами понимать проблемы распознавания подмножеств из некоторого множества входов с определённой на нем функцией длины. Для подмножества $S \subseteq I$ определим последовательность

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

где $S_n = S \cap I_n$ — множество входов из S размера n. Заметим, что $\rho_n(S)$ — это вероятность попасть в S при случайной и равновероятной генерации входов из I_n. *Асимптотической плотностью* S назовём предел

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

Множество S называется *генерическим*, если $\rho(S) = 1$, и *пренебрежимым*, если $\rho(S) = 0$. Очевидно, что S генерическое тогда и только тогда, когда его дополнение $I \setminus S$ пренебрежимо.

Следуя [1], назовём множество S *строго пренебрежимым*, если последовательность $\rho_n(S)$ экспоненциально быстро сходится к нулю, т. е. существуют константы σ, $0 < \sigma < 1$, и $C > 0$, такие, что для любого n

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

Теперь S называется *строго генерическим*, если его дополнение $I \setminus S$ строго пренебрежимо.

Алгоритм $\mathcal{A}$ с множеством входов I и множеством выходов $J \cup \{?\}$ ($? \notin J$) называется (*строго*) *генерическим*, если

1) $\mathcal{A}$ останавливается на всех входах из I;
2) множество $\{x \in I : \mathcal{A}(x) \neq ?\}$ является (строго) генерическим.

Аналогично определяется понятие вероятностного генерического алгоритма. Генерический алгоритм $\mathcal{A}$ вычисляет функцию $f : I \to J$, если $(\mathcal{A}(x) = y \in J) \Rightarrow (f(x) = y)$ для всех $x \in I$. Ситуация $\mathcal{A}(x) = ?$ означает, что $\mathcal{A}$ не может вычислить функцию f на аргументе x. Но условие 2 гарантирует, что $\mathcal{A}$ корректно вычисляет f на почти всех входах (входах из генерического множества). Множество $S \subseteq I$ называется (*строго*) *генерически разрешимым за полиномиальное время*, если существует (строго) генерический полиномиальный алгоритм, вычисляющий его характеристическую функцию.

2. Генерическая полиномиальная сводимость

Напомним определения генерической полиномиальной сводимости и генерической NP-полноты из [5].

Пусть I, J — некоторые множества входов с определёнными на них функциями размера. Множество $A \subseteq I$ *генерически полиномиально сводится* к множеству $B \subseteq J$ (обозначается $A \leqslant_{\mathrm{GenP}} B$), если существуют вероятностный полиномиальный алгоритм $\mathcal{R} : I \times \mathbb{N} \to P(J) \cup \{?, \perp\}$ ($P(J)$ — это множество всех подмножеств J), полином $p(n)$, полином $q(n)$ степени больше 2 и константа $C > 0$, такие, что

1. Для всех $x \in I$ либо $(\forall n\,(\mathcal{R}(x,n) = ?))$, либо для всех $n \geqslant q(k)$, где $k = \mathrm{size}(x)$, выполнены следующие свойства:
   а) $\forall y \in \mathcal{R}(x,n)\ (y \neq \perp \ \Rightarrow\ \mathrm{size}(y) = n)$;
   б) все элементы в $\mathcal{R}(x,n) \setminus \{\perp\}$ выдаются алгоритмом $\mathcal{R}$ равновероятно;
   в) вероятность получить ответ $\perp$ в $\mathcal{R}(x,n)$ не больше 2^{-Ck};
   г) $\dfrac{|\mathcal{R}(x,n)|}{|J_n|} > \dfrac{1}{(p(n))^k}$;
   д) $x \in A \Rightarrow \mathcal{R}(x,n) \subseteq B$;
   е) $x \notin A \Rightarrow \mathcal{R}(x,n) \subseteq J \setminus B$.
2. Множество $\{x \in I : \forall n\ (\mathcal{R}(x,n) = ?)\}$ строго пренебрежимо.

В [5] доказано, что определённая таким образом сводимость сохраняет свойство «быть строго генерически разрешимым за полиномиальное время» для алгоритмических проблем, а также обладает свойством транзитивности.

Определим генерический аналог класса NP. Множество $S \subseteq I$ принадлежит классу $\mathrm{sg\,NP}$, если существует полиномиальное строго генерическое множество $G \subseteq I$, такое, что $S \cap G \in \mathrm{NP}$. Множество $S \in \mathrm{sg\,NP}$ называется *генерически NP-полным*, если для любого $A \in \mathrm{sg\,NP}$ имеет место $A \leqslant_{\mathrm{GenP}} S$.

3. Булевы схемы

Булева схема содержит конечное число переменных $x_1, \ldots, x_n$. Переменная x_i может быть либо *входной*, либо определяться через другие переменные с помощью присваивания одного из следующих типов:

1) $x_i = x_j \vee x_k$, где $j, k < i$;
2) $x_i = x_j \wedge x_k$, где $j, k < i$;
3) $x_i = \neg x_j$, где $j < i$;
4) $x_i = x_j$, где $j < i$.

Заметим, что первая переменная любой булевой схемы обязана быть входной. Последняя переменная x_n схемы называется *выходной*. Естественным образом любая булева схема C задаёт булеву функцию от входных переменных схемы следующим образом. Входным переменным схемы можно присвоить любые булевы значения. Затем значения всех остальных переменных схемы вычисляются в порядке возрастания их индексов через значения входных и предыдущих переменных схемы согласно соответствующим присваиваниям. Значение выходной переменной и будет значением соответствующей функции. С другой стороны, для любой булевой схемы можно легко построить булеву схему (и не одну), задающую эту функцию. Под *размером булевой схемы* будем понимать число переменных в ней. Обозначим через $\mathcal{BC}$ множество всех булевых схем.

Лемма 1. Число булевых схем размера n равно

$$|\mathcal{BC}_n| = \prod_{m=1}^{n} \big(1 + 2(m-1)^2 + 2(m-1)\big).$$

Доказательство. Для переменной x_m имеются следующие варианты:

1) переменная x_m является входной — один вариант;
2) $x_m = x_j \vee x_k$, где $j, k < m$, — $(m-1)^2$ вариантов;
3) $x_m = x_j \wedge x_k$, где $j, k < m$, — $(m-1)^2$ вариантов;
4) $x_m = \neg x_j$, где $j < m$, — $(m-1)$ вариантов;
5) $x_m = x_j$, где $j < m$, — $(m-1)$ вариант.

Итого получается $\big(1+2(m-1)^2+2(m-1)\big)$ вариантов выбора для переменной x_m; для всех n переменных булевой схемы получаем $\prod\limits_{m=1}^{n}\big(1+2(m-1)^2+2(m-1)\big)$ вариантов выбора. ■

Выясним порядок роста функции $|\mathcal{BC}_n|$. Оценим каждый множитель в произведении $|\mathcal{BC}_n|$ снизу:

$$2m(m-1)=2(m-1)^2+2(m-1)<1+2(m-1)^2+2(m-1).$$

Отсюда $2^n n!(n-1)!<|\mathcal{BC}_n|$. Оценим каждый множитель в произведении $|\mathcal{BC}_n|$ сверху:

$$1+2(m-1)^2+2(m-1)=2m(m-1)+1<2m(m-1)+2m=2m^2.$$

Отсюда $|\mathcal{BC}_n|<2^n(n!)^2$. Итого получаем

$$\frac{2^n(n!)^2}{n}<|\mathcal{BC}_n|<2^n(n!)^2.$$

Для любой булевой схемы $C(x_1,\ldots,x_n)$ определим множество $Eq(C)$ всевозможных булевых схем от переменных $x_1,\ldots,x_n,x_{n+1},\ldots,x_{n+m}$, в которых переменные $x_1,\ldots,x_n$ определяются так же, как в схеме C, переменные $x_{n+1},\ldots,x_{n+m-1}$ определяются произвольным образом, а выходная переменная x_{n+m} определяется так: $x_{n+m}=x_n$. Очевидно, что любая схема из $Eq(C)$ вычисляет ту же булеву функцию, что и схема C.

Лемма 2. Для любой булевой схемы C имеет место

$$\frac{|Eq(C)\cap\mathcal{BC}_n|}{|\mathcal{BC}_n|}>\frac{1}{5n^{2k}}$$

для любого $n>k$, где k — размер схемы C.

Доказательство. Пусть булева схема C имеет размер k и $n>k$. В любой булевой схеме из множества $Eq(C)\cap\mathcal{BC}_n$ присваивания для первых k переменных и для последней переменной фиксированы. Переменные $x_{n+1},\ldots,x_{n+m-1}$ определяются произвольным образом. Поэтому, аналогично тому, как это делалось в доказательстве леммы 1, можно подсчитать

$$|Eq(C)\cap\mathcal{BC}_n|=\prod_{m=k+1}^{n-1}\big(1+2(m-1)^2+2(m-1)\big).$$

Отсюда

$$\frac{|Eq(C)\cap\mathcal{BC}_n|}{|\mathcal{BC}_n|}=\frac{\prod\limits_{m=k+1}^{n-1}\big(1+2(m-1)^2+2(m-1)\big)}{\prod\limits_{m=1}^{n}\big(1+2(m-1)^2+2(m-1)\big)}=\frac{1}{\prod\limits_{m=1}^{k}\big(1+2(m-1)^2+2(m-1)\big)}.$$

Так как $\dfrac{1}{\big(1+2(m-1)^2+2(m-1)\big)}>\dfrac{1}{5n^2}$ для любого $m<n$, то имеем

$$\frac{|Eq(C)\cap\mathcal{BC}_n|}{|\mathcal{BC}_n|}>\frac{1}{5n^{2k}}.$$

Лемма доказана. ■

4. Основной результат

Под *проблемой выполнимости булевых схем* будем понимать следующую алгоритмическую проблему. Для любой булевой схемы нужно определить, существуют ли булевы значения её входных переменных, при которых значение выходной переменной равно 1. Известно [7], что существует полиномиальный алгоритм, который по любой булевой формуле строит булеву схему, которая вычисляет ту же функцию. Поэтому проблема выполнимости булевых схем является NP-полной в классическом смысле. Следующее утверждение показывает, что она является генерически NP-полной.

Теорема 1. Проблема выполнимости булевых схем генерически NP-полна.

Доказательство. Пусть $A \in \operatorname{sg}\mathrm{NP}$. Тогда существует полиномиальное строго генерическое множество G, такое, что $A \cap G \in \mathrm{NP}$. Так как проблема выполнимости булевых схем NP-полна, существует классическая полиномиальная сводимость f множества $A \cap G$ к ней. Полиномиальный вероятностный алгоритм $\mathcal{R}$, генерически сводящий проблему A к проблеме выполнимости булевых схем, работает на входе (x, n) следующим образом:

1) проверяет, принадлежит ли x множеству G;
2) если $x \notin G$, выдаёт ?;
3) если $x \in G$, то строит булеву схему $C = f(x)$, такую, что C выполнима тогда и только тогда, когда $x \in A$;
4) случайно и равновероятно генерирует булеву схему из множества $Eq(C) \cap \mathcal{BC}_n$.

Проверим свойства полиномиальной сводимости. Свойство 2 следует из того, что множество G строго генерическое. Свойства 1а и 1б следуют из описания шага 4. Свойство 1в очевидно выполняется: алгоритм вообще не выдаёт ответ $\perp$. Свойства 1д и 1е также выполняются по построению схемы C. Наконец, свойство 1г следует из леммы 2. ■

Автор выражает благодарность рецензенту за полезные замечания и предложения по улучшению текста статьи.

ЛИТЕРАТУРА

1. *Kapovich I., Miasnikov A., Schupp P., and Shpilrain V.* Generic-case complexity, decision problems in group theory and random walks // J. Algebra. 2003. V. 264. No. 2. P. 665–694.
2. *Гэри М., Джонсон Д.* Вычислительные машины и труднорешаемые задачи. М.: Мир, 1982. 419 с.
3. *Levin L.* Average case complete problems // SIAM J. Computing. 1987. V. 15. P. 285–286.
4. *Gurevich Y.* Average case completeness // J. Computer System Sciences. 1991. V. 42. P. 346–398.
5. *Рыбалов А.* О генерической NP-полноте проблемы выполнимости булевых формул // Прикладная дискретная математика. 2017. № 36. С. 106–112.
6. *Miasnikov A. and Ushakov A.* Generic case completeness // J. Computer System Sciences. 2016. V. 82. No. 8. P. 1268–1282.
7. *Сэвидж Д.* Сложность вычислений. М.: Факториал, 1998. 368 с.

REFERENCES

1. *Kapovich I., Miasnikov A., Schupp P., and Shpilrain V.* Generic-case complexity, decision problems in group theory and random walks. J. Algebra, 2003, vol. 264, no. 2, pp. 665–694.
2. *Garey M. and Johnson D.* Computers and Intractability. N. Y., Freeman & Co, 1979. 340 p.
3. *Levin L.* Average case complete problems. SIAM J. Computing, 1987, vol. 15, pp. 285–286.

4. *Gurevich Y.* Average case completeness. J. Computer System Sciences, 1991, vol. 42, pp. 346–398.
5. *Rybalov A.* O genericheskoy NP-polnote problemy vypolnimosti bulevyh formul [On generic NP-completeness of the Boolean satisfiability problem]. Prikladnaya Diskretnaya Matematika, 2017, no. 36, pp. 106–112. (in Russian)
6. *Miasnikov A. and Ushakov A.* Generic case completeness. J. Computer System Sciences, 2016, vol. 82, no. 8, pp. 1268–1282.
7. *Savage J.* The Complexity of Computing. John Wiley and Sons Inc., 1977. 391 p.