S©MR ISSN 1813-3304

СИБИРСКИЕ ЭЛЕКТРОННЫЕ
МАТЕМАТИЧЕСКИЕ ИЗВЕСТИЯ

Siberian Electronic Mathematical Reports
http://semr.math.nsc.ru

Том 11, стр. 144–144 (2014) УДК 510.652
MSC 11U99

ГЕНЕРИЧЕСКИ НЕРАЗРЕШИМЫЕ И ТРУДНОРАЗРЕШИМЫЕ ПРОБЛЕМЫ

А.Н. РЫБАЛОВ

Abstract. Generic-case approach to algorithmic problems studies behavior of an algorithm on typical (almost all) inputs and ignores the rest of inputs. The method of generic amplification was proposed by A.Myasnikov and A.Rybalov for constructing of generically undecidable problems. The main ingredient of this method is a technique of cloning, which unites inputs of the problem together in the large enough sets of equivalent inputs. Equivalence is understood in the sense that the problem is solved similarly for them. In this paper we present a generalization of this method. Also we construct a decidable in the classical sense problem, which is not generically decidable in polynomial time. We use another methods for this than the generic amplification method, because generic amplification, highly likely, is not applicable for this.

Keywords: generic amplification, undecidability, hardness.

1. Введение

Генерический подход [3] – это один из подходов к изучению алгоритмических проблем для «почти всех» входов. Исследования вычислительной сложности для «почти всех» входов началось в 1970-80-х годах, после того как был выделен огромный пласт трудноразрешимых алгоритмических проблем – NP-полных проблем, для которых не удалось найти эффективных алгоритмов, работающих за полиномиальное время для всех входов. Оказалось, что если немного ослабить требование эффективности – рассматривать не все входы, а «почти все» или случайные входы, то иногда можно быстро решать задачу для

Rybalov, A.N., Generically undecidable and hard problems.

Работа выполнена в рамках государственного задания ИМ СО РАН, проект FWNF-2022-0003.

Поступила 30 июня 2023 г., опубликована 1 сентября 2023 г.

таких типичных входов. Это имеет практический смысл, когда алгоритм должен решать быстро задачу для случайных входных данных: если вероятность «наткнуться» на «плохой» вход пренебрежимо мала, то алгоритм будет быстро работать практически всегда. В рамках генерического подхода изучается поведение алгоритмов на множестве «почти всех» входов (это множество называется генерическим), игнорируя поведение алгоритма на остальных входах, на которых алгоритм может работать медленно или вообще не останавливаться. Понятие «почти все» формализуется введением асимптотической плотности на множестве входных данных.

В исследованиях по генерической вычислимости и сложности вычислений можно выделить два основных направления. Первое связано с построением генерических (полиномиальных) алгоритмов для алгоритмических проблем, которые являются неразрешимыми или трудноразрешимыми в классическом смысле. Второе направление концентрируется на поиске алгоритмических проблем, которые остаются неразрешимыми или трудноразрешимыми и в генерическом смысле. Данная работа относится ко второму направлению исследований.

Первые генерически неразрешимые алгоритмические проблемы были найдены А. Г. Мясниковым и А.Н.Рыбаловым в [4]. Для доказательства генерической неразрешимости ими был предложен метод генерической амплификации, который позволяет по проблеме, неразрешимой в классическом смысле строить проблему, которая будет генерически неразрешимой. Данный метод был успешно применен к следующим алгоритмическим проблемам: проблема остановки для машин Тьюринга [5], проблема равенства для полугрупп [4], проблема разрешимости элементарных теорий [4, 6], десятая проблема Гильберта [7]. Однако формализация метода, предложенная в [4], оказалась не совсем удобной: напрямую ее удается применить только для построения конечно определенной полугруппы с генерически неразрешимой проблемой равенства, а в остальных случаях генерическая амплификация используется неформально. В данной работе предлагается формализация более общей схемы генерической амплификации, которая работает во всех случаях.

Применение генерической амплификации для построения генерически трудноразрешимых проблем сталкивается с трудностями, которые связаны с необходимостью контролировать скорость сходимости последовательности частот множества «плохих» входов. Поэтому тут удается получить лишь результаты об отсутствии сильно генерических полиномиальных алгоритмов, которые решают проблему быстро на множестве входов, относительные частоты которых экспоненциально быстро стремяться к 1. Например, в таком виде генерическая амплификация применима для арифметики Пресбургера [8]. В данной работе, с помощью других методов, строится пример разрешимой в классическом смысле проблемы, для которой не существует полиномиального генерического алгоритма.

2. Предварительные сведения

Пусть I – некоторое множество входов. Для подмножества $S \subseteq I$ определим последовательность

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

где I_n – множество входов размера n, а $S_n = S \cap I_n$ – множество входов из S размера n. Здесь для конечного множества A через $|A|$ обозначено число его элементов. *Асимптотической плотностью* S назовем предел (если он существует)

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

Множество S называется *генерическим*, если $\rho(S) = 1$ и *пренебрежимым*, если $\rho(S) = 0$. Назовем множество S *сильно пренебрежимым*, если последовательность $\rho_n(S)$ экспоненциально быстро сходится к 0, 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 *сильно генерическим*, если его дополнение $I \setminus S$ сильно пренебрежимо.

Алгоритм $\mathcal{A}$ с множеством входов I называется *(сильно) генерическим*, если множество $\{x \in I : \mathcal{A}(x)\downarrow\}$ (сильно) генерическое. Здесь $\mathcal{A}(x)\downarrow$ означает, что алгоритм $\mathcal{A}$ останавливается на входе x. Генерический алгоритм $\mathcal{A}$ вычисляет функцию $f : I \to J$, если

$$\forall x \in I\ \mathcal{A}(x)\downarrow \Rightarrow\ f(x) = \mathcal{A}(x).$$

Генерический алгоритм $\mathcal{A}$ работает за полиномиальное время, если существует полином $p(n)$ такой, что

$$\forall x \in I\ \mathcal{A}(x)\downarrow \Rightarrow t_{\mathcal{A}}(x) < p(size(x)),$$

где $size(x)$ – размер входа x, а $t_{\mathcal{A}}(x)$ – время работы алгоритма $\mathcal{A}$ на входе x. Еще такие алгоритмы мы будем еще называть полиномиальными генерическими.

С практической точки зрения, когда требуется построить алгоритм, решающий конкретную алгоритмическую проблему для почти всех входов, удобнее рассматривать алгоритмы следующего типа [1]. Каждый такой алгоритм останавливается на всех входах, на входах из некоторого генерического множества выдает правильный ответ, а на пренебрежимом множестве остальных входов выдает специальный ответ «?» – «Не знаю».

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

(1) $\forall x \in I\ \mathcal{A}(x)\downarrow$,
(2) множество $\{x \in I : \mathcal{A}(x) =?\}$ (сильно) пренебрежимо.

Эффективно генерический алгоритм $\mathcal{A}$ вычисляет функцию $f : I \to J$, если

$$\forall x \in I\ \mathcal{A}(x) \neq ?\ \Rightarrow\ f(x) = \mathcal{A}(x).$$

Множество $S \subseteq I$ и соответствующая проблема распознавания (S, I) *(эффективно) (сильно) генерически разрешимы (за полиномиальное время)*, если существует (эффективно) (сильно) генерический (полиномиальный) алгоритм, вычисляющий характеристическую функцию S.

Легко видеть, что из эффективной генерической разрешимости следует генерическая разрешимость. Действительно, любой эффективный генерический алгоритм можно легко переделать в генерический заменив выдачу ответа «?» на бесконечное зацикливание. В обратную сторону это неверно – см., например, теорему 2.22 и следствие 2.24 в [2]. Однако для полиномиальной сложности верно и обратное: из полиномиальной генерической разрешимости следует

полиномиальная эффективная разрешимость. Действительно, если имеется полиномиальная оценка $p(n)$ на время работы генерического алгоритма в случае, когда он останавливается, то можно завести счетчик T числа шагов, и, в случае, если $T > p(n)$, можно обрывать вычисление и выдавать ответ «?», – в этом случае генерический алгоритм уже не остановится. Таким образом получается эффективно генерический полиномиальный алгоритм, решающий ту же проблему.

3. Генерическая амплификация

Опишем сначала схему, обобщающую конструкцию генерической амплификации, предложенную в [4].

Пусть I – множество входов. *Клонирование* множества I – это функция $C : I \to P(I)$, где $P(I)$ есть множество всех подмножеств множества I. Будем называть клонирование $C : I \to P(I)$ *эффективным*, если существует всюду определенная вычислимая функция $E : I \times \mathbb{N} \to I$ такая, что для любого $x \in I$

$$C(x) = \{E(x, 0), E(x, 1), \dots, \}.$$

Таким образом, с помощью алгоритма E можно эффективно перечислять все элементы каждого клона $C(x)$. Клонирование $C : I \to P(I)$ называется *непренебрежимым* (соответственно, *не сильно пренебрежимым*), если для любого $x \in I$ множество $C(x)$ не является пренебрежимым (соответственно, сильно пренебрежимым).

Пусть $S \subseteq I$. Будем говорить, что клонирование $C : I \to P(I)$ *сохраняет* множество S, если

- $\forall x \in S\ C(x) \subseteq S$,
- $\forall x \notin S\ C(x) \subseteq I \setminus S$.

Теорема 1. *Пусть I – множество входов, $S \subseteq I$ и $C : I \to P(I)$ – эффективное клонирование, сохраняющее S. Тогда имеет место следующее:*

1. *Если C непренебрежимое клонирование и проблема распознавания (S, I) генерически разрешима, то проблема распознавания (S, I) разрешима.*
2. *Если C не сильно пренебрежимое клонирование и проблема распознавания (S, I) сильно генерически разрешима, то проблема распознавания (S, I) разрешима.*

Доказательство. Докажем пункт 1. Пусть $\mathcal{A}$ – генерический алгоритм, распознающий S такой, что множество

$$G(\mathcal{A}) = \{x \in I : \mathcal{A}(x)\downarrow\}$$

генерическое. Построим алгоритм $\mathcal{B}$, который будет решать проблему распознавания S для всех входов. На входе $x \in I$ алгоритм $\mathcal{B}$ работает следующим образом.

1. Установить $i = 0$.
2. Сделать i шагов вычисления алгоритма $\mathcal{A}$ на $E(x, 0), E(x, 1), \dots, E(x, i)$.
3. Если в результате этого алгоритм $\mathcal{A}$ остановился на каком-то $E(x, k)$, $k \leq i$, и выдал ответ, остановиться и выдать этот ответ.
4. Иначе, положить $i = i + 1$ и вернуться на шаг 2.

Так как $C(x)$ непренебрежимо, то $C(x)$ имеет непустое пересечение с множеством $G(\mathcal{A})$. Поэтому, параллельно запуская алгоритм $\mathcal{A}$ на элементах $E(x, 0)$,

$E(x,1)$, ..., мы найдем эффективно зависящее от x, число i_x такое, что $x' = E(x,i_x) \in G(\mathcal{A})$. Очевидно, $x \in S$ тогда и только тогда, когда $x' \in S$ и тогда и только тогда, когда $\mathcal{A}$ выдает ответ ДА для x'. Поэтому мы можем эффективно решать, выполнено ли $x \in S$, и пункт 1 доказан. Доказательство пункта 2 аналогично. □

Доказанную теорему можно переформулировать в терминах генерической неразрешимости.

Теорема 2. *Пусть I – множество входов, $S \subseteq I$ и $C : I \to P(I)$ – эффективное клонирование, сохраняющее S. Тогда если проблема распознавания (S,I) неразрешима, то имеет место следующее:*

(1) *Если C – непренебрежимое клонирование, то проблема распознавания (S,I) не является генерически разрешимой.*
(2) *Если C – не сильно пренебрежимое клонирование, то проблема распознавания (S,I) не является сильно генерически разрешимой.*

Теперь перейдем к описанию метода генерической амплификации, предложенного в [4]. Назовем клонирование $C : I \to P(I)$ *разделяющим*, если

$$\forall x, y \in I \ (x \neq y) \Rightarrow C(x) \cap C(y) = \varnothing.$$

Для множества $S \subseteq I$ определим *клон* $C(S)$ как объединение всех клонов элементов из S:

$$C(S) = \bigcup_{x \in S} C(x).$$

Легко видеть, что для любого $S \subseteq I$ разделяющее клонирование $C : I \to P(I)$ сохраняет множество $C(S)$. Поэтому, непосредственным следствием из теоремы 2 является следующее утверждение из [4].

Теорема 3. *Пусть I – множество входов, $S \subseteq I$ и $C : I \to P(I)$ – эффективное разделяющее клонирование. Тогда если проблема распознавания (S,I) неразрешима, то имеет место следующее:*

(1) *Если C – непренебрежимое клонирование, то проблема распознавания $(C(S),I)$ не является генерически разрешимой.*
(2) *Если C – не сильно пренебрежимое клонирование, то проблема распознавания $(C(S),I)$ не является сильно генерически разрешимой.*

Отметим, что в конкретных ситуациях клонирование редко получается разделяющим. Например, в доказательствах генерической неразрешимости проблемы остановки для нормализованных машин Тьюринга [5] и теорий первого порядка для нормализованных формул [6] клоны для различных элементов пересекаются. Однако соответствующие клонирования сохраняют рассматриваемые множества, а потому, по теореме 2, эти проблемы не являются генерически разрешимыми.

4. Генерически трудноразрешимые проблемы

Будем рассматривать машины Тьюринга, которые распознают подмножества двоичных строк $\{0,1\}^*$. Такие машины имеют два завершающих состояния q_a – допускающее состояние и q_r – отвергающее состояние. Заканчивать

работу они могут только в одном из этих состояний. Таким образом, эти машины могут выдавать только ответы ДА или НЕТ. Под размером строки w понимается ее длина $|w|$, поэтому число входов размера n есть 2^n.

Под *эффективной нумерацией всех полиномиальных машин Тьюринга* будем подразумевать эффективную нумерацию всех пар $\{(M_i, p_k(n))\}_{i\in\mathbb{N},k\in\mathbb{N}}$, где M_i – машина Тьюринга с номером i, а $p_k(n) = n^k + k$. Такая пара на входе x моделирует работу некоторой полиномиальной машины Тьюринга следующим образом:

$$(M_i, p_k(n))(x) = \begin{cases} M_i(x), \text{ если } M_i(x)\downarrow \text{ за } \leq p_k(|x|) \text{ шагов,} \\ \text{НЕТ, иначе.} \end{cases}$$

Легко видеть, что любая полиномиальная машина Тьюринга встретится в этой последовательности.

Теорема 4. *Существует рекурсивное множество, не являющееся генерически разрешимым за полиномиальное время.*

Доказательство. Пусть есть эффективная нумерация полиномиальных машин Тьюринга $P_1, P_2, P_3, \ldots$. Образуем из них следующую последовательность

$$\{M_i, i = 1, 2, 3, \ldots\} = \{P_1, P_1, P_2, P_1, P_2, P_3, P_1, P_2, P_3, P_4, P_1, P_2, \ldots\}.$$

В ней каждый раз, после того, как были выписаны машины $P_1, \ldots, P_k$, выписываются машины $P_1, \ldots, P_{k+1}$. Таким образом, каждая полиномиальная машина P_i выписывается бесконечно много раз.

Построение нужного множества S будет проходить по шагам. Стартуя с множества всех двоичных строк $\{0,1\}^*$ на нулевом шаге, мы будем на шаге i вычеркивать или оставлять некоторые числа в завсисимости от поведения машины M_i. Опишем подробно шаг $i > 0$. Запускаем машину M_i на каждом входе размера i и считаем количество ответов ДА и количество ответов НЕТ. Если ответов ДА получилось больше половины, то вычеркиваем все входы размера i, иначе все их оставляем. Предельное множество в этом процессе и есть искомое множество S.

Действительно, заметим, что для любой полиномиальной машины M множество входов, на которых M дает неправильный ответ, имеет вид

$$E(M) = \bigcup_{i=1}^{\infty} A_i,$$

где

$$A_i = \{w \in \{0,1\}^* : |w| = m_i\}, m_i > m_{i-1},$$

причем $|A_i| \geq 2^{m_i}/2$ для любого i. Если теперь рассмотреть последовательность

$$\rho_n(E(M)) = \frac{|E(M)_n|}{2^n},\ n = 1, 2, 3, \ldots,$$

то, легко видеть, что $\rho_n(E(M)) \geq \frac{1}{2}$ для бесконечно большого числа значений n. Поэтому множество $E(M)$ непренебрежимо.

Теперь, допустим существует генерический полиномиальный алгоритм $\mathcal{A}$, распознающий множество S. Без ограничения общности можно считать, что $\mathcal{A}$ – полиномиальный эффективно генерический алгоритм. По нему легко получить полиномиальную машину M, которая на любом входе x работает следующим образом:

(1) Вычисляет $\mathcal{A}(x)$.
(2) Если $\mathcal{A}(x) = 1$, выдает 1.
(3) Если $\mathcal{A}(x) = 0$, выдает 0.
(4) Если $\mathcal{A}(x) =?$, выдает 0.

Очевидно, что M, распознавая элементы S, ошибается на пренебрежимом множестве. Но это противоречит построению множества S.

Рекурсивность множества S следует из алгоритмической природы процедуры его построения. □

References

[1] D. Hirschfeldt. *Some Questions in Computable Mathematics*, Computability and Complexity, (2017), 22–55. *https://math.uchicago.edu/ drh/Papers/Papers/open.pdf*

[2] C. Jockusch, P. Schupp. *Generic computability, Turing degrees, and asymptotic density*, Journal of the London Mathematical Society, **85:2** (2012), 472–490. *https://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.225.1953&rep=rep1&type=pdf*

[3] I. Kapovich, A. Myasnikov, P. Schupp, V. Shpilrain. *Generic-case complexity and decision problems in group theory*, Journal of Algebra, **264** (2003), 665–694. *https://arxiv.org/pdf/math/0203239*

[4] A. Myasnikov, A. Rybalov. *Generic complexity of undecidable problems*, Journal of Symbolic Logic, **73:2** (2008), 656–673. *https://www.jstor.org/stable/27588653*

[5] A. Rybalov. *On the Generic undecidability of the Halting Problem for normalized Turing machines*, Theory of Computing Systems, **60:4** (2017), 671–676. *https://link.springer.com/article/10.1007/s00224-016-9698-9*

[6] A. Rybalov. *On generic complexity of elementary theories*, Herald of Omsk State University, **4** (2015), 14–17. (in Russian) *https://cyberleninka.ru/article/n/o-genericheskoy-slozhnosti-elementarnyh-teoriy*

[7] A. Rybalov. *Generic complexity of the Diophantine problem*, Groups Complexity Cryptology, **5:1** (2013), 25–30. *https://www.degruyter.com/document/doi/10.1515/gcc-2013-0004/html?lang=en*

[8] A. Rybalov. *Generic Complexity of Presburger Arithmetic*, Theory of Computing Systems, **46:1** (2010), 2–8. *https://link.springer.com/article/10.1007/s00224-008-9120-3*

Alexander Nikolaevich Rybalov
Sobolev Institute of Mathematics,
prospekt Koptyuga 4,
Novosibirsk, 630090, Russia.
Pevtsova 13,
Omsk, 644099, Russia.
Email address: alexander.rybalov@gmail.com