МАТЕМАТИЧЕСКИЕ ОСНОВЫ ИНФОРМАТИКИ И ПРОГРАММИРОВАНИЯ

УДК 510.52

РЕЛЯТИВИЗОВАННЫЕ ГЕНЕРИЧЕСКИЕ КЛАССЫ P И NP[1]

А. Н. Рыбалов

Омский государственный технический университет, г. Омск, Россия

Бейкер, Гилл и Соловей в 1975 г. построили такие два оракула A и B, что $\mathrm{P}^A = \mathrm{NP}^A$, но $\mathrm{P}^B \neq \mathrm{NP}^B$. Тем самым они показали, что неравенство $\mathrm{P} \neq \mathrm{NP}$ не может быть доказано с использованием метода диагонализации. В рамках генерического 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. В работе определяются генерические аналоги genP и genNP классов вычислительной сложности P и NP, а также их релятивизованные версии. Доказывается генерический аналог теоремы Бейкера — Гилла — Соловея: существуют такие оракулы A и B, что $\mathrm{genP}^A = \mathrm{genNP}^A$, но $\mathrm{genP}^B \neq \mathrm{genNP}^B$. Таким образом, для решения генерического аналога проблемы совпадения классов P и NP метод диагонализации также неприменим.

Ключевые слова: *генерическая сложность, проблема $P = NP$, оракулы.*

DOI 10.17223/20710410/40/8

RELATIVIZED GENERIC CLASSES P AND NP

A. N. Rybalov

Omsk State Technical University, Omsk, Russia

E-mail: alexander.rybalov@gmail.com

Classical theorem of Baker, Gill and Solovay states that there exist two oracles A and B such that $\mathrm{P}^A = \mathrm{NP}^A$, but $\mathrm{P}^B \neq \mathrm{NP}^B$. This result indicates that the classical tools of computability theory (such as diagonalization) are inapplicable to prove the inequality $\mathrm{P} \neq \mathrm{NP}$. 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. Many classical undecidable or hard algorithmic problems become feasible in the generic case. In this paper we introduce generic analogs genP and genNP of the classical computational complexity classes P and NP. We prove a generic analog of the Baker — Gill — Solovay theorem: there exist two oracles A and B such that $\mathrm{genP}^A = \mathrm{genNP}^A$, but $\mathrm{genP}^B \neq \mathrm{genNP}^B$. Therefore the diagonalization arguments cannot be applied to prove the inequality $\mathrm{P} \neq \mathrm{NP}$ in the generic case too.

[1]Работа поддержана грантом РНФ № 17-11-01117.

Keywords: *generic complexity, P vs NP problem, oracles.*

Введение

Наиболее сильные результаты в теории вычислительной сложности, такие, как теоремы о временной и пространственной иерархии классов сложности, получены с применением метода так называемой диагонализации. Этот метод, пришедший из классической теории алгоритмов и берущий свое начало ещё в работах Геделя и Тьюринга, активно использовался в 1960–70-х гг. в попытках решения многих центральных проблем теории сложности вычислений, прежде всего, проблемы равенства классов P и NP. Однако в 1975 г. Бейкер, Гилл и Соловей в [1] доказали теорему, которая говорит о том, что с помощью метода диагонализации нельзя доказать неравенство $\mathrm{P} \neq \mathrm{NP}$. Оказывается, что любое доказательство неравенства каких-либо классов проблем $C_1 \neq C_2$, полученное с помощью диагонализации, релятивизуемо. Это означает, что та же схема проходит и для доказательства неравенства $C_1^A \neq C_2^A$, где A — любой счётный оракул. Оракул A является просто некоторым подмножеством входов. Релятивизованные классы определяются при помощи машин, которые имеют команды обращения к оракулу вида $x \in A?$. Такие машины в процессе вычислений могут делать запросы к оракулу за одну единицу времени и проверять принадлежность различных элементов множеству A. При надлежащем выборе оракула вычислительная мощность релятивизованных машин увеличивается: они могут быстро решать труднорешаемые и даже неразрешимые проблемы. Формальные определения релятивизованных машин и классов можно найти в [2, 3]. Бейкер, Гилл и Соловей построили такие два оракула A и B, что $\mathrm{P}^A = \mathrm{NP}^A$, но $\mathrm{P}^B \neq \mathrm{NP}^B$. Тем самым они показали, что неравенство $\mathrm{P} \neq \mathrm{NP}$ не может быть доказано с использованием метода диагонализации.

В рамках генерического подхода [4] 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.

В данной работе определяются генерические аналоги genP и genNP классов вычислительной сложности P и NP, а также их релятивизованные версии. Доказывается генерический аналог теоремы Бейкера — Гилла — Соловея: существуют такие оракулы A и B, что $\mathrm{genP}^A = \mathrm{genNP}^A$, но $\mathrm{genP}^B \neq \mathrm{genNP}^B$. Таким образом, для решения генерического аналога проблемы совпадения классов P и NP метод диагонализации также неприменим.

1. Генерические классы P и NP

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, под алгоритмическими проблемами будем 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 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 $\rho(S) = 0$.

В дальнейшем будем рассматривать в качестве множества входов $\{0,1\}^*$ — множество всех слов над двоичным алфавитом $\{0,1\}$. Тем не менее все последующие определения и результаты без особого труда могут быть перенесены на любое конструктивное множество входов.

Определим генерические аналоги классических классов вычислительной сложности P и NP. Множество $S \subseteq \{0,1\}^*$ принадлежит *классу* genP, если существует разрешимое за полиномиальное время генерическое множество $G \subseteq \{0,1\}^*$, такое, что $G \cap S$ разрешимо за полиномиальное время. Проблема из класса genP — это проблема, которая решается эффективно (за полиномиальное время), но не для всех входов, а для почти всех входов (входов из генерического множества G). Множество $S \subseteq \{0,1\}^*$ принадлежит *классу* genNP, если существует разрешимое за полиномиальное время генерическое множество $G \subseteq \{0,1\}^*$, такое, что $S \cap G \in \mathrm{NP}$. Напомним определение класса NP. Проблема $S \in \mathrm{NP}$, если существует такое множество $S' \in \mathrm{P}$ и такой полином $p(n)$, что $x \in S \Leftrightarrow \exists y \in \{0,1\}^*\,(|y| < p(|x|)$ и $(x,y) \in S')$. Здесь строка y называется подсказкой (или сертификатом, или решением), а множество S' — проверятелем. Проблема из класса genNP — это проблема, решение которой проверяется эффективно (за полиномиальное время) для почти всех входов (входов из генерического множества G).

Теперь определим релятивизованные версии этих классов. Пусть A есть некоторое подмножество I. *Машина с оракулом* A в своей программе может содержать команды обращения к оракулу вида «if $x \in A$ goto m», причём выполнение этой команды происходит за один шаг работы вычислительного устройства. Более формально машины с оракулом определены, например, в [3]. Множество $S \subseteq \{0,1\}^*$ принадлежит *классу* genP^A, если существует разрешимое за полиномиальное время на машинах с оракулом A генерическое множество $G \subseteq \{0,1\}^*$, такое, что $G \cap S$ разрешимо за полиномиальное время на машине с оракулом A. Множество $S \subseteq \{0,1\}^*$ принадлежит *классу* genNP^A, если существует разрешимое за полиномиальное время на машинах с оракулом A генерическое множество $G \subseteq \{0,1\}^*$, такое, что $S \cap G \in \mathrm{NP}^A$. Релятивизованный класс NP^A определяется аналогично обычному классу NP: проблема $S \in \mathrm{NP}^A$, если существует такое разрешимое за полиномиальное время на машинах с оракулом A множество S' и такой полином $p(n)$, что $x \in S \Leftrightarrow \exists y \in \{0,1\}^*\,(|y| < p(|x|)$ и $(x,y) \in S')$.

2. Генерическая теорема Бейкера — Гилла — Соловея

Теорема 1. Существует такой оракул A, что $\mathrm{genP}^A = \mathrm{genNP}^A$.

Доказательство. В качестве оракула A можно взять оракул из теоремы Бейкера — Гилла — Соловея, такой, что $\mathrm{P}^A = \mathrm{NP}^A$. Действительно, пусть $S \in \mathrm{genNP}^A$. Тогда существует A-разрешимое за полиномиальное время генерическое множество G, такое, что $S \cap G \in \mathrm{NP}^A$. Но $\mathrm{NP}^A = \mathrm{P}^A$, поэтому $S \cap G \in \mathrm{P}^A$. Это означает, что $S \in \mathrm{genP}^A$. Таким образом, $\mathrm{genP}^A = \mathrm{genNP}^A$. ■

Теорема 2. Существует такой оракул B, что $\mathrm{genP}^B \neq \mathrm{genNP}^B$.

Доказательство. Идея построения оракула B такова: построить связанное с ним множество из NP^B так, чтобы любая полиномиальная машина с оракулом B не распознавала это множество на непренебрежимом множестве. Этот множество следующее:

$$\Omega_B = \{\omega \in \{0,1\}^* : \text{ в } B \text{ существует строка длины } 2|\omega|\}.$$

Заметим, что $\Omega_B \in \mathrm{NP}^B$ для любого оракула B. Действительно, для входа ω сертификатом является строка x длины $2|\omega|$, такая, что $x \in B$. Если $\omega \in \Omega_B$, то такой сертификат найдётся, иначе нет. Процесс проверки сертификата выполняется за полиномиальное время на машине с оракулом B.

Теперь нужно построить B так, чтобы $\Omega_B \notin \mathrm{genP}^B$. Пусть есть эффективная нумерация полиномиальных машин с оракулами $P_1, P_2, P_3, \ldots$ 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 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 оракула B проходит по шагам, стартуя с множества $B = \{0,1\}^*$, на i-м шаге вычеркиваем строки из B так, чтобы машина M_i не смогла распознать Ω_B на большинстве входов некоторого фиксированного размера. Опишем сначала шаг 1.

Ш а г 1. Выберем число k таким, что $p(k) < 2^k$, где $p()$ — это полином, ограничивающий время работы машины M_1. Запускаем M_1 на каждой строке размера k (их 2^k штук). Во время своей работы на каждом входе, как только она делает запрос к оракулу вида $a \in B?$, вычёркиваем эту строку a из B (вынуждая оракул всегда отвечать НЕТ). В процессе работы M_1 на всех 2^k входах мы вычеркнем некоторые $a_1, \ldots, a_m$, где $m < p(k)2^k < 2^{2k}$. Далее смотрим: если M_1 выдаёт на 2^k входах длины k выход 1 не менее чем в половине случаев (т. е. $\geqslant 2^{k-1}$ раз), то вычеркиваем все строки длины $2k$ из B, тем самым гарантируя, что $\omega \notin \Omega_B$ для любой строки ω длины k. Это означает, что машина M_1 ошибается на входах длины k не менее чем в половине случаев. Если же M_1 выдает в большинстве случаев 0, мы выбираем некоторую строку длины $2k$ из оставшихся в B и помечаем её (такая строка найдётся, так как мы вычеркнули $< 2^{2k}$ строк). Это также гарантирует, что $\omega \in \Omega_B$ для любой строки ω длины k. Таким образом, опять машина M_1 ошибается на входах длины k не менее чем в половине случаев.

Ш а г i. Итак, машины $M_1, \ldots, M_{i-1}$ не распознают Ω_B, ошибаясь на входах с длинами $k_1, \ldots, k_{i-1}$ не менее чем в половине случаев. Берем $k > k_{i-1}$, такое, что $p(k) < 2^k$, где $p()$ — это полином, ограничивающий время работы машины M_i. Кроме

того, k должно быть больше размеров всех строк, которые были помечены или вычеркнуты на предыдущих шагах. Опять запускаем машину M_i на каждом входе длины k и вычеркиваем строчки-запросы (делая ответы на запросы к оракулу НЕТ), но только те, которые не помечены. Если строка в запросе $a \in B$? помечена на предыдущих шагах, она уже точно будет в окончательном оракуле. Это гарантирует, что работа машины M_i на любом входе ω длины k не будет меняться в дальнейшем (то есть ответы на её запросы $a \in B$ будут теми же и на последующих шагах). Как и на шаге 1, если M_i выдаёт 1 на большинстве входов длины k, вычеркиваем все строки длины $2k$ из оракула B. Если она для большинства выдаёт 0, то помечаем одну из оставшихся строк длины $2k$ и переходим к следующему шагу. Опять M_i не может распознать Ω_B не менее чем для половины входов длины k.

Предельное множество в этом процессе и есть искомый оракул B. Действительно, $\Omega_B \in \mathrm{NP}^B$, а тем более $\Omega_B \in \mathrm{genNP}^B$. Чтобы убедиться, что $\Omega_B \notin \mathrm{genP}^B$, заметим, что для любой полиномиальной машины M с оракулом B множество входов, на которых M даёт неправильный ответ, имеет вид

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

где

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

причём $|A_i| \geqslant 2^{m_i}$ для любого i. Если теперь рассмотреть последовательность

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

то легко видеть, что $\rho_n(E(M)) \geqslant 1/2$ для бесконечно большого числа значений n. Поэтому множество $E(M)$ непренебрежимо. ■

ЛИТЕРАТУРА

1. *Baker T., Gill J., and Solovay R.* Relativizations of the P=?NP question // SIAM J. Computing. 1975. V. 4. P. 431–442.
2. *Вялый М., Китаев А., Шень А.* Классические и квантовые вычисления. М.: МЦНМО, ЧеРо, 1999. 192 с.
3. *Гэри М., Джонсон Д.* Вычислительные машины и труднорешаемые задачи. М.: Мир, 1982. 419 с.
4. *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.

REFERENCES

1. *Baker T., Gill J., and Solovay R.* Relativizations of the P=?NP question. SIAM J. Computing, 1975, vol. 4, pp. 431–442.
2. *Vyalyi M., Kitaev A., and Shen A.* Classical and Quantum Computation (Graduate Studies in Mathematics). AMS, 2002. 272 p.
3. *Garey M. and Johnson D.* Computers and Intractability. N. Y., Freeman & Co, 1979. 340 p.
4. *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.