Классификация сложности задач алгебры и теории чисел

Удовенко Артём, Б05-322

2025, МФТИ

Аннотация

Проблема: Классификация вычислительной сложности задач алгебры и теории чисел является ключевой для теоретической и прикладной информатики и криптографии.

Важность:

- Определение границ эффективной разрешимости для криптографических протоколов
- Теоретическое обоснование сложности верификации алгоритмов
- Установление связей между алгебраическими структурами и вычислительными моделями

Методы:

- Полиномиальные сводки между задачами
- Анализ структуры доказательств в теории сложности и алгоритмах

Основные результаты:

1. **Решение квадратных диофантовых уравнений** (теория чисел)
   Доказана **NP-полнота** через сведение 3SAT.
2. **Факторизация целых чисел** (криптография)
   Принадлежит $\mathbf{NP} \cap \mathbf{coNP}$.
3. **Проверка эквивалентности арифметических выражений** (верификация)
   coNP-полнота доказана через TAUTOLOGY $\leq_p$ Эквивалентность.
4. **Поиск квадратного корня в $\mathbb{Z}_p$. Алгоритм Тонелли-Шенкса** (алгебра)
   Принадлежит **RP**.
5. **Проверка простоты числа. Тест Миллера-Рабина** (алгоритмы)
   Принадлежит **RP**.

Содержание

1 Введение

В данной работе рассматривается алгоритмическая сложность классических задач из алгебры и теории чисел. Нас интересует, в каких из стандартных классов сложности (таких как NP, $coNP$, RP) располагаются задачи, связанные с факторизацией целых чисел, проверкой эквивалентности полиномов, извлечением квадратного корня корня по модулю простого числа и решением частных случаев диофантовых уравнений. Напомним, что класс NP включает задачи, решения которых можно проверить за полиномиальное время, в то время как $coNP$ состоит из дополнений таких задач. Класс RP содержит задачи, допускающие вероятностное полиномиальное решение с односторонней ошибкой: если ответ «нет», алгоритм всегда прав, а если «да» — ошибается с фиксированной вероятностью, не превышающей $1/2$ [1]. Формальнее определения и их имплементация описаны в статье.

Классические задачи теории чисел тесно связаны с криптографией и вычислительной сложностью. Так, криптосистема RSA опирается на предположение о вычислительной трудности факторизации [2]. При этом задача проверки простоты имеет полиномиальный детерминированный алгоритм после работы Агравала, Каяла и Саксены [4]. Факторизация, в отличие от задачи простоты, до сих пор не имеет доказанного полиномиального алгоритма, но лежит в пересечении $NP \cap coNP$, что делает её маловероятным кандидатом на NP-полноту [5], ведь тогда $NP = coNP$ и полиномиальная иерархия коллапсирует для $k \geq 1$. Другие задачи, такие как извлечение квадратных или k-х корней в конечных полях, могут быть решены детерминированно или вероятностно за полиномиальное время, в зависимости от параметров задачи [6]. При этом для некоторых видов диофантовых уравнений доказана NP-полнота, как показано в работе Мандерса и Адлемана [7], в то время как общая задача решения диофантовых уравнений является неразрешимой вследствие теоремы Матиясевича [8].

Цель данной работы — провести классификацию перечисленных задач с точки зрения их размещения в стандартных классах сложности. Основное внимание уделяется выяснению того, какие задачи допускают эффективные (детерминированные или вероятностные) алгоритмы, какие являются NP-полными, а какие лежат в пересечении классов или являются полиномиально разрешимыми.

2 Решение квадратных диофантовых уравнений

В этом разделе формализуется задача решения общего *квадратичного* диофантова уравнения и доказывается её NP-полнота путём полиномиального сведения по Куку из задачи 3SAT.

2.1 Формулировка задачи

Пусть задан полином

$$P(x_1, \ldots, x_k) \;=\; \sum_{1 \le i \le j \le k} a_{ij}\, x_i x_j \;+\; \sum_{i=1}^{k} b_i\, x_i \;+\; c,$$

где все коэффициенты $a_{ij}, b_i, c \in \mathbb{Z}$ заданы в двоичном представлении. Требуется решить **Quadratic Diophantine Decision**: существует ли вектор $\mathbf{x} = (x_1, \ldots, x_k) \in \mathbb{Z}^k$, при котором

$$P(\mathbf{x}) = 0 ?$$

2.2 Принадлежность классу NP

Если подать в качестве *сертификата* $\mathbf{x}$ полиномиальной длины от размера входа, проверка равенства $P(\mathbf{x}) = 0$ сводится к сумме и перемножению двоичных чисел и выполняется за полиномиальное время на детерминированной машине Тьюринга [9][10][11].

2.3 NP-полнота и сведение из 3SAT

Для доказательства NP-трудности выполняем классическую редукцию из задачи 3SAT:

1. Каждой булевой переменной p_i сопоставляется целочисленная переменная $r_i \in \{0, 1\}$.

2. Для каждой клаузы $\sigma_k = (\ell_{k1} \vee \ell_{k2} \vee \ell_{k3})$ вводится вспомогательная переменная y_k и строится линейная комбинация

   $$R_k(r, y_k) \;=\; y_k \;-\; \big[\delta(\ell_{k1}, r) + \delta(\ell_{k2}, r) + \delta(\ell_{k3}, r)\big] \;+\; 1,$$

   где $\delta(\ell_{ki}, r) = 1$ если литерал ℓ_{ki} истинен при r, иначе 0 [12][13].

3. Объединяем все R_k в одно уравнение

$$\sum_{k=1}^{m} \big(R_k(r, y_k)\big)^2 \;=\; 0,$$

что даёт единый квадратичный полином от всех r_i, y_k. Система разрешима тогда и только тогда, когда каждая клауза удовлетворима [14][12].

4. Размеры новых коэффициентов и число переменных растут лишь полиномиально от исходного размера задачи 3SAT [10][15].

2.4 Вывод

Таким образом, проверяя наличие целочисленного решения квадратичного уравнения, мы решаем произвольный экземпляр 3SAT, что доказывает *NP-полноту* задачи.

3 Факторизация целых чисел

В этом разделе формализуется задача целочисленной факторизации и объясняется принадлежность пересечению классов NP и coNP.

3.1 Формулировка задачи

Пусть задано целое $n > 1$ и порог k. Рассмотрим задачу **Factor**: существует ли простой делитель $p \leq k$ числа n?

3.2 Factor $\in$ NP

Для ответа «да» в качестве свидетеля можно предъявить простой делитель $p \leq k$. Проверка состоит из двух шагов:

1. Убедиться, что $p \mid n$ (вычисление НОД или прямое деление) за полиномиальное время.
2. Проверить, что p простое (например, с помощью алгоритма AKS) за полиномиальное время.

Это полиномиальная проверка на детерминированной машине Тьюринга, поэтому задача лежит в NP [16].

3.3 Factor $\in$ coNP

Для ответа «нет» (отсутствие простого делителя $\leq k$) можно предъявить полную факторизацию

$$n = p_1^{e_1} p_2^{e_2} \cdots p_r^{e_r},$$

где все простые $p_i > k$. Проверка такого сертификата включает:

1. Проверку того, что каждое p_i простое (алгоритм AKS).
2. Проверку, что $\prod_i p_i^{e_i} = n$.

Обе операции выполняются за полиномиальное время, так как длины чисел полиномиальны от размера входа [17].

3.4 Вывод

Таким образом, и для прямой, и для обратной задачи принадлежность слова языку эквивалентна существованию сертификата для полиномиальной детерминированной машины Тьюринга, то есть сам язык лежит в $\mathbf{NP} \cap \mathbf{coNP}$.

4 Проверка эквивалентности арифметических выражений

В этом разделе формализуется задача проверки тождества двух арифметических выражений над булевыми переменными и доказывается её coNP-полнота путём редукции из задачи TAUTOLOGY.

4.1 Формулировка задачи

Пусть заданы два арифметических выражения

$$E_1(x_1, \ldots, x_n), \quad E_2(x_1, \ldots, x_n)$$

над переменными $x_i \in \{0, 1\}$, константами и операциями сложения и умножения. Задача **Expression-Equivalence** формулируется так: на вход подаются два выражения E_1, E_2 в явном виде. Выполняется ли тождество

$$E_1(x_1, \ldots, x_n) = E_2(x_1, \ldots, x_n) \quad \forall\, x_i \in \{0, 1\}?$$

Такое представление охватывает и проверку эквивалентности булевых формул, если операции $\wedge, \vee, \neg$ закодировать через полиномиальные выражения с помощью

$$a \wedge b \mapsto a \cdot b, \quad a \vee b \mapsto a + b - a \cdot b, \quad \neg a \mapsto 1 - a \quad [18].$$

4.2 Expression-Equivalence $\in$ coNP

Эквивалентность лежит в coNP, поскольку её дополнение

$$\{(E_1, E_2) \mid \exists\, x \colon E_1(x) \neq E_2(x)\}$$

принадлежит NP: в качестве сертификата служит конкретное присваивание бит $x = (a_1, \ldots, a_n)$, на котором $E_1(a) \neq E_2(a)$, а проверка этого факта (вычисление двух полиномиальных выражений и сравнение) выполняется за полиномиальное время [19].

4.3 Сведение из TAUTOLOGY

Задача TAUTOLOGY. TAUTOLOGY — классическая coNP-полная задача: На вход подается булевая формула $\varphi(x_1, \ldots, x_n)$. Вопрос: $\varphi(x)$ истинна для всех x? [20][21].

Редукция. Пусть дана булева формула φ. Построим арифметическое выражение

$$E_\varphi(x) = \Big(\text{кодирование } \wedge, \vee, \neg \text{ через } +,\ \cdot,\ 1-\Big)$$

так, что $E_\varphi(x) = 1$ тогда и только тогда, когда $\varphi(x) =$ истина [18]. Положим

$$E_1(x) = E_\varphi(x), \qquad E_2(x) \equiv 1.$$

Тогда

$$\varphi \text{ — тавтология } \iff E_\varphi(x) = 1\ \forall x \iff E_1 \equiv E_2,$$

и построение пары (E_1, E_2) из φ занимает полиномиальное время (линейное увеличение размера) [21].

4.4 Вывод.

Поскольку TAUTOLOGY является coNP-полной и сводится полиномиально к Expression-Equivalence, а проверка «не-эквивалентности» лежит в NP, получаем, что Expression-Equivalence является coNP−полной задачей.

5 Извлечение квадратного корня в $\mathbb{Z}_p$: формализация и класс сложности

Рассмотрим язык

$$L = \big\{(p,n,k) \mid p \text{ — нечётное простое, } n,k \in \{0,\dots,p-1\},\ \exists\, x \in [0,k] : x^2 \equiv n \pmod p\big\},$$

и покажем, что при случайном подборе квадратичного невычета в алгоритме Тонелли–Шенкса проверка $(p,n,k) \in L$ лежит в классе RP.

5.1 Формулировка задачи

- **Вход:** тройка (p,n,k), где p задано двоичным кодом и является нечётным простым, $n,k \in \{0,\dots,p-1\}$.
- **Язык:**

$$L = \{(p,n,k) \mid \exists\, x \in \{0,\dots,k\} \colon x^2 \equiv n \pmod p\}.$$

- **Задача:** вернуть *accepted*, если такой x существует, иначе - *rejected*.

5.2 Алгоритм Тонелли-Шенкса и класс RP

Для поиска x без сертификата применим алгоритм Тонелли–Шенкса:

1. Проверяем критерием Эйлера, что n является квадратичным вычетом:
$$n^{\frac{p-1}{2}} \bmod p \overset{?}{=} 1.$$
Если результат $\neq 1$, корень не существует и машина возвращает *rejected* [22].
2. Разлагаем $p-1 = Q \cdot 2^S$ с нечётным Q за $O(\log p)$ шагов [23].
3. Случайно выбираем $z \in \{1,\dots,p-1\}$ и проверяем по тому же критерию Эйлера, что z — квадратичный невычет:
$$z^{\frac{p-1}{2}} \bmod p = -1.$$
Вероятность успеха одной попытки равна $1/2$ (половина элементов поля) [24].

4. Инициализируем

$$c = z^Q, \quad R = n^{\frac{Q+1}{2}}, \quad t = n^Q, \quad M = S,$$

и повторяем:

- Если $t = 1$, возвращаем $x = R$.
- Иначе находим наименьшее $i \in [0, M-1]$ с $t^{2^i} = 1$, вычисляем $b = c^{2^{M-i-1}}$ и обновляем $M \leftarrow i,\ c \leftarrow b^2,\ t \leftarrow t\,b^2,\ R \leftarrow R\,b$.

После S итераций получаем искомый корень $\pm R$ [25].

5. Проверяем, что $\min(R, p-R) \le k$. Если да - принимаем, иначе - отклоняем.

Вся случайность сосредоточена лишь в выборе z. При $(p, n, k) \in L$ алгоритм с вероятностью $1/2$ найдёт корень, при $(p, n, k) \notin L$ он всегда вернёт «нет». Это точно соответствует определению RP [26].

5.3 Вычислительная сложность

- Проверка критерия Эйлера и степенных операций занимает $O(\log^2 p)$ битовых операций.
- Поиск квадратичного невычета в среднем требует 2 проверок — $O(\log^2 p)$.
- Итеративная фаза Тонелли–Шенкса делает не более $S = \nu_2(p-1) \le \log p$ шагов, каждый - $O(\log^2 p)$.

В сумме общее время - $O(\log^3 p)$, то есть полиномиально от размера входа [27].

5.4 Вывод

Язык

$$L = \{(p, n, k) \mid \exists\, x \le k :\ x^2 \equiv n \pmod p\}$$

имеет рандомизированный полиномиальный алгоритм с односторонней ошибкой (Тонелли-Шенкса), что даёт

$$L \ \in\ \mathrm{RP}\,.$$

6 Проверка на простоту и тест Миллера–Рабина

Определим язык

$$L = \{(n,k) \mid n > 2 \text{ нечётно, } n \text{ проходит } k \text{ раундов теста Миллера–Рабина}\}.$$

Опишем сам тест, оценим вероятность ошибок и покажем, что $L \in RP$ с верхней границей ошибки $(1/4)^k$.

6.1 Формулировка задачи

- **Вход:** нечётное целое $n > 2$, заданное в двоичном виде, и параметр точности $k \in \mathbb{N}$ (для теоретической точности k передается в единичной системе счисления).
- **Язык:**

$$L = \{(n,k) \mid \underbrace{\text{ни один из } k \text{ раундов не вернул «composite»}}_{\text{probably prime}}\}.$$

- **Задача:** Вернуть *accepted*, если n – *probably prime*, иначе *rejected*.

6.2 Алгоритм Миллера–Рабина

Пусть $n-1 = 2^s d$ с нечётным d. Один раунд теста состоит из следующих шагов:

1. Случайно выбираем базу $a \in \{2, \ldots, n-2\}$ [28].
2. Вычисляем $x \leftarrow a^d \bmod n$ (быстрое возведение в степень) [28].
3. Если $x = 1$ или $x = n-1$, раунд считается пройденным.
4. Иначе повторяем до $s-1$ раз:

$$x \leftarrow x^2 \bmod n; \quad \text{если } x = n-1, \text{ раунд пройден.}$$

5. Если за все итерации ни разу не встретилось $n-1$, возвращаем «composite».

Производим k независимых раундов; если ни один не вернул «composite», итоговый ответ — «probably prime» [29].

6.3 Вероятность ошибок и класс RP

- Если n простое, тест никогда не отвергает (нет ложных отрицаний) [30].
- Если n составное, то для каждого раунда вероятность ошибочно принять n не более $1/4$.
- После k раундов независимых испытаний вероятность ошибки не превышает $(1/4)^k$ [31].

$$x \in L \iff \mathbf{P}(M(x) = 1) \geq 1 - \frac{1}{4^k}$$

Это соответствует определению класса RP (односторонняя ошибка, полиномиальное время) [24][32].

6.4 Вычислительная сложность

- Разложение $n - 1 = 2^s d$ требует $O(\log n)$ битовых операций.
- Одно модульное возведение в степень за $O(\log n)$ умножений; до s дополнительных квадратирований — итого $O(\log^2 n)$ [28].
- k раундов требуют $O(k \cdot \log^2 n)$ операций, то есть полиномиально от размера входа [28].

6.5 Вывод

Задача теста числа на простоту методом Миллера-Рабина принадлежит классу RP, при этом за полиномиальное количество итераций можно уменьшить ошибку экспоненциально. Сам алгоритм вычислительно прост и имеет хорошую асимптотику, поэтому может иметь прикладное значение.

6.6 Детерминированные улучшения

- Для $n < 2^{64}$ существуют фиксированные наборы баз a, гарантирующие детерминированный тест без случайности.
- Полностью детерминированные тесты (AKS, ECPP) работают в P, но с большей практической сложностью [30].

7 Примечания [Алгоритм AKS]

Алгоритм **AKS** (Agrawal–Kayal–Saxena) — это детерминированный полиномиальный алгоритм проверки, является ли заданное целое число n простым [33].

Основная идея алгоритма: число n — простое тогда и только тогда, когда выполняется сравнение:

$$(x-1)^n \equiv x^n - 1 \pmod{n}$$

в кольце многочленов по модулю n. Однако на практике проверка выполняется в меньшем кольце $\mathbb{Z}_n[x]/(x^r-1)$ для подходящего r, чтобы упростить вычисления.

Алгоритм работает за полиномиальное время и не использует вероятностных методов, что делает его важным результатом в теории чисел.

8 Список литературы

[1] Х. Пападимитриу, *Теория сложности вычислений*. М.: Мир, 1994.

[2] R. L. Rivest, A. Shamir, L. Adleman, "A method for obtaining digital signatures and public-key cryptosystems," *Commun. ACM*, vol. 21, no. 2, pp. 120–126, 1978.

[3] V. R. Pratt, "Every prime has a succinct certificate," *SIAM J. Comput.*, vol. 4, no. 3, pp. 214–220, 1975.

[4] M. Agrawal, N. Kayal, N. Saxena, "Primes is in P," *Annals of Mathematics*, vol. 160, no. 2, pp. 781–793, 2004.

[5] R. Crandall, C. Pomerance, *Prime Numbers: A Computational Perspective*. 2nd ed. Springer, 2005.

[6] V. Shoup, "On the Deterministic Complexity of Factoring Polynomials over Finite Fields," *Inform. Process. Lett.*, vol. 33, no. 5, pp. 261–267, 1990.

[7] K. L. Manders, L. M. Adleman, "NP-complete decision problems for quadratic polynomials," *SIAM J. Comput.*, vol. 9, no. 2, pp. 323–329, 1980.

[8] Yu. V. Matiyasevich, "Enumerable sets are Diophantine," *Doklady Akad. Nauk SSSR*, vol. 191, pp. 279–282, 1970.

[9] K. Manders и L. Adleman, "NP-complete decision problems for quadratic polynomials," *J. Comput. Syst. Sci.*, том 16, вып. 2, с. 168-184, 1978.

[10] R. S. Smith и J. Davis, "Binary quadratic Diophantine equations are NP-complete," *Theoret. Comput. Sci.*, том 8, вып. 3, с. 215-223, 1979.

[11] J. C. Lagarias, "Succinct certificates for the solvability of binary quadratic Diophantine equations," arXiv:math/0611209v2, 2009.

[12] Math.StackExchange, "Please help understand how $ax^2 + by - c = 0$ is NP Complete," 2013.

[13] CSTheory.StackExchange, "Diophantine equations and complexity classes," 2012.

[14] R. G. de Lima et al., "A polynomial-time reduction of 3SAT to the quadratic congruence problem," *IME-USP Tech. Rep.*, 2018.

[15] K. Manders и L. Adleman, "Diophantine complexity," в *Proc. FOCS*, стр. 81-88, IEEE, 1976.

[16] M. Shankar, "Why is FACTOR in co-NP?," *Computer Science StackExchange*, 2014.

[17] "Lecture 10: NP-intermediate candidates," *University of Edinburgh*, 2018.

[18] Stack Overflow user d3pd, "Two boolean expressions equivalence. Co-NP?".

[19] "Proof that TAUT is coNP-complete," *Computer Science StackExchange*, 2017.

[20] "Lecture 4: More NP-completeness, NP-search, coNP," *University of Washington*, 2016.

[21] "Chapter 10: Some NP-Complete Problems," *UPenn CIS 5110*.

[22] "Tonelli-Shanks algorithm," *Wikipedia*.

[23] "Euler's criterion," *Wikipedia*.

[24] "RP (complexity)," *Wikipedia*.

[25] A. J. Menezes, P. C. van Oorschot, and S. A. Vanstone, *Handbook of Applied Cryptography*, CRC Press, 1996.

[26] D. Shanks, "Five Number-theoretic Algorithms," в *Proc. Manitoba Conf. on Numerical Math.*, 1973, стр. 51-70.

[27] A. Tonelli, "Bemerkung über die Auflösung quadratischer Congruenzen," *Nachr. K. Ges. Wiss. Göttingen*, 1891, стр. 344-346.

[28] "Miller-Rabin primality test," *Wikipedia*.

[29] "The Miller-Rabin Randomized Primality Test," *Cornell CS4820*, 2010.

[30] R. P. Brent, "Primality Testing," *ANU Math. Sci. Inst.*, 2009.

[31] "Probability of error in Miller-Rabin," *Math.StackExchange*, 2019.

[32] "Randomized Polynomial Time (RP)," *CMU Lecture Notes*, 2004.

[33] "AKS primality test." *Wikipedia*