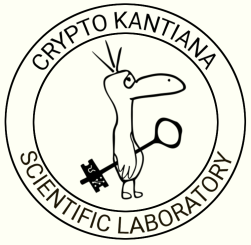
Алгоритмы редукции базиса линейного кода

Семён Новосёлов

БФУ им. И. Канта

SibeCrypt 2025



IKBFU

Введение

Цель: Адаптировать алгоритмы LLL и BKZ на коды.

- **2022** Debris-Alazard T., Ducas L., van Woerden W.P.J.
  An Algorithmic Reduction Theory for Binary Codes: LLL and More. IEEE IT
- **2024** Ghentiyala S., Stephens-Davidowitz N.
  More basis reduction for linear codes: backward reduction, BKZ, slide reduction, and more. APPROX/RANDOM

Обозначения

- $\mathcal{C}$ – линейный $[n, k]$-код, с порождающей матрицей

$$B = \{b_1, \ldots, b_k\}$$

- Λ – решётка в $\mathbb{R}^n$ с базисной матрицей $B = \{b_1, \ldots, b_n\}$ и ортогонализацей Грама–Шмидта (GSO)

$$B^* = \{b_1^*, \ldots, b_n^*\}$$

План

1. Алгоритмы LLL и BKZ для бинарных кодов
2. Приложения
    1. Задача поиска кодового слова малого веса
    2. Задача декодирования: алгоритм Ли-Брикелла-Бабая

Почему нельзя использовать LLL алгоритм как есть?

При замене евклидовой метрики на метрикой Хэмминга со скалярным произведением:

1. $\langle x, y \rangle = 0 \nRightarrow \|x + y\| = \|x\| + \|y\|$
2. нет аналога для инварианта $\det \Lambda = \prod_i \|b_i^*\|$, из которого следуют свойства:

   γSVP: при увеличении длины $b_2^*, \ldots, b_n^*$ уменьшается b_1

   γCVP (Бабай): при $\|b_1^*\| \approx \ldots \approx \|b_n^*\|$ достигается наилучшая редукция целевого вектора

Теория редукции базиса кода

Решётки

- $\mathbf{a} \perp \mathbf{b} \iff \langle \mathbf{a}, \mathbf{b} \rangle = 0$
- GSO
- ортогональные вектора

Коды

- $\mathbf{a} \perp \mathbf{b} \iff \mathrm{Supp}(\mathbf{a}) \cap \mathrm{Supp}(\mathbf{b})$
- эпиподальная матрица
- ортоподальные вектора

Проекции

Ортогональная проекция в решётках заменяется на:

$$\pi^{\perp}_{\{b_1,\ldots,b_i\}}(x)_j = \begin{cases} x_j, \text{ если } b_{1,j} \neq 0 \wedge \ldots \wedge b_{i,j} \neq 0, \\ 0, \text{ в противном случае.} \end{cases}$$

(в позициях x из $\mathrm{Supp}(b_1) \cup \ldots \cup \mathrm{Supp}(b_i)$ ставим 0)

Эпиподальная матрица

$$B^+ = \left\{ \pi^{\perp}_{\{b_1,\ldots,b_{i-1}\}}(b_i) \right\}_{i=1,\ldots,k}$$

Свойства:

- $\|b_1^+\| = \|b_1\| \implies$ уменьшение веса вектора b_1^+ ведёт к уменьшению веса b_1 (цель алгоритмов LLL/BKZ)
- большинство строк в B^+ имеют вес 1
- $k_1 = \{$число строк B^+ веса $\neq 1\}$ – сложность алгоритмов декодирования с редукцией зависит от данного параметра

Профиль

Обозначим $\ell_i = \|b_i^+\|$ для $i = 1, \ldots, k$. Тогда:

$$|\mathrm{Supp}(\mathcal{C})| = \sum_{i=1}^{k} \ell_i$$

$\Longrightarrow$ для уменьшения $\ell_1 = \|b_1\|$ надо увеличивать $\ell_2, \ldots, \ell_k$.

Профиль: $\ell_1, \ldots, \ell_k$.

Алгоритм BKZ для кодов

$$B_{[i,j]} = \left\{ \pi^{\perp}_{\{b_1,\ldots,b_{i-1}\}}(b_r) \right\}_{r=i,\ldots,j}$$

Идея: для каждого i-го подкода $\mathcal{C}(b_1, \ldots, b_i)$: подбираем наименьший по весу вариант для b_i^+ в коде $\mathcal{C}(B_{[i,i+\beta-1]})$ и ставим его на место b_i^+.

Алгоритм BKZ для кодов. II

Вход: порождающая матрица B кода $\mathcal{C}$ в систем. форме.
Выход: β-BKZ редуцированная порождающая матрица

1. $i = 1$
2. **while** i < k **do**
3. $\quad a = \mathrm{MinCodeword}(B_{[i,i+\beta-1]})$
4. $\quad$ **if** $\|a\| < \|b_i^+\|$ **then**
5. $\quad\quad$ найти T т.ч. $(T \cdot B_{[i,i+\beta-1]})_1 = a$
6. $\quad\quad B = (I_{i-1} \oplus T \oplus I_{k-j})B$
7. $\quad\quad i = \max(1, i - \beta + 1)$
8. $\quad$ **else**
9. $\quad\quad i = i + 1$
10. **return** B

Алгоритм BKZ для кодов. III

- алгоритм LLL: $\beta = 2$
- сложность: открытая задача
- фактор аппроксимации: $\|b_1\| \leq 2^{k-\beta} d_{min}$
- увеличение размера блока редуцирует первую строку
  $\Longrightarrow$ нахождение кодовых слов малого веса

Приложение к задаче декодирования

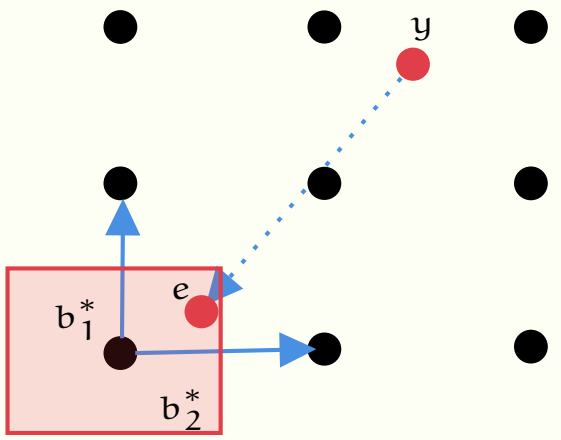
Алгоритм Ли-Брикелла-Бабая:

- алгоритм Ли-Брикелла в форме с порождающей матрицей
- Бабай (кодовая версия): редукция произвольного вектора в $y \in \mathbb{F}_2^n$ до фундаментальной области: $\mathcal{F}(B^+)$

Алгоритм Бабая

$$\mathcal{F}(B^*) = \left\{ \sum_i a_i b_i^* \mid a_i \in (-1/2, 1/2) \right\}$$

$$\Lambda + \mathcal{F}(B^*) = \mathbb{R}^n$$



алгоритм Бабая обрезает вектор y до фунд. области:

$$x = y - e \in \Lambda, e \in \mathcal{F}(B^*)$$

- $\|e\|^2 \leq \frac{1}{4} \sum_i \|b_i^*\|^2 \to \min$ при $\|b_1^*\| \approx \ldots \approx \|b_n^*\|$
- короткий базис $\Lambda \implies$ короткий e

Алгоритм Бабая для кодов

Фундаментальная область:

$$\mathcal{F}(B^+) = \left\{ y \in \mathbb{F}_2^n : \forall i = 1, \ldots, k; \|y \wedge b_i^+\| + TB_{b_i^+}(y) \leq \frac{\|b_i^+\|}{2} \right\},$$

где

$$TB_b(y) = \begin{cases} 0, \|b\| \text{ нечётное;} \\ 0, y_j = 0 \text{ для } j = \min \mathrm{Supp}(b); \\ 1/2, \text{в противном случае.} \end{cases}$$

Замощение пространства: $\mathcal{C} + \mathcal{F}(B^+) = \mathbb{F}_2^n$

Редукция: $\|y \wedge b_i^+\| \geq \frac{\|b_i^+\|}{2} \implies \|(y + b_i) \wedge b_i\| \leq \frac{\|b_i^+\|}{2} \implies$
замена $y \mapsto y + b_i$ переносит y в $\mathcal{F}(B^+)$.

Алгоритм Бабая для кодов. II

Идея: добавлением элементов базиса $\mathcal{C}$ добиваемся попадания целевого вектора y в фундаментальную область.

Алгоритм ($SizeReduction$): Редукция вектора y.

Вход: Базис B кода $\mathcal{C}$ и целевой вектор y.
Выход: $e \in \mathcal{F}(B^+)$ т.ч. $y + e \in \mathcal{C}$.

1. $e = y$
2. **for** i = k **downto** 1
3. **if** $\|e \wedge b_i^+\| + TB_{b_i^+}(e) > \frac{\|b_i^+\|}{2}$ **then**
4. $e = e + b_i$
5. **return** e

Т.к. прибавляли только элементы из кода, то $y + e \in \mathcal{C}$.

Эффективность редукции

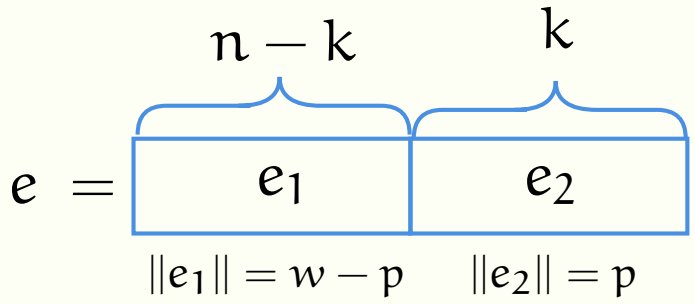
Предположим, что e случайно распределено в фунд. области. Тогда

$$\mathbb{E}[\|e\|] \leq 2^{k-1} - \frac{1}{\sqrt{\pi}} \sum_{i=1}^{k} \sqrt{\lceil \frac{\ell_i}{2} \rceil \left(1 - \frac{1}{\lceil \frac{\ell_i}{2} \rceil + 1}\right)}$$

$\to \min$ при $\ell_1 \approx \ldots \approx \ell_k$.

Алгоритм Ли-Брикелла

Перебор эквивалентных кодов, пока не получим ошибку в виде:



- алгоритм Пранжа: $p = 0$
- как правило $p = 3$, e_2 перебирается простым перебором (Ли-Брикелл), либо методами на основе парадокса дней рождений (Думер/Штерн, MMT, BJMM)

Алгоритм Ли-Брикелла-Бабая (LBB)

Вход: Базис B кода $\mathcal{C}$ в полусистематической форме, целевой вектор y, вес ошибки w и параметр p.

Выход: $\mathrm{c} \in \mathcal{C}$ т.ч. $\|\mathrm{y} \oplus \mathrm{c}\| \leq w$ или FAIL.

1. **foreach** $\mathrm{J} \subset [\mathrm{k}_1 + 1, \mathrm{k}]$, $\#\mathrm{J} \leq \mathrm{p}$:
2. $\quad \mathrm{e} = \mathrm{SizeRed}((\mathrm{b}_1, \ldots, \mathrm{b}_{\mathrm{k}_1}), \mathrm{y} \oplus \sum_{\mathrm{j} \in \mathrm{J}} \mathrm{b}_\mathrm{j})$
3. $\quad$ **if** $\|\mathrm{e}\| \leq w$ **then**
4. $\quad\quad$ **return** $\mathrm{y} \oplus \mathrm{e}$.
5. **return** FAIL

(при получении FAIL алгоритм перезапускается с LLL/BKZ-редуцированным базисом случайного экв. кода)

Сложность алгоритма

$$C_{LBB} = \frac{C_{LB}}{2^{\lambda}},$$

где

$$\lambda = \log_2\left(\frac{1-R}{1-R-a}\right) \cdot k_1 - \log_2(k_1).$$

- $R = \frac{k}{n}$, скорость кода
- $a = \frac{w}{n}$, плотность ошибки
- LLL ($\beta = 2$): $\log n \leq k_1 \leq 2\log n$
- BKZ ($\beta \geq 16$): $k_1 \geq 2\log n$

Случайные линейные коды

- $R = 0.5$, $a \approx 0.11$ (граница Г-В)

$$\lambda = 0.358 \cdot k_1 - \log_2(k_1)$$

- $R = 0.8$, $a \approx 0.0155$ (граница Г-В), `Classic McEliece`

$$\lambda = 0.116 \cdot k_1 - \log_2(k_1)$$

Итог: при $k_1 = 2\log_2 n$ получается полиномиальное ускорение алгоритма LB в $\widetilde{O}(n^{0.716})$ раз для $R = 0.5$ и $\widetilde{O}(n^{0.232})$ раз для $R = 0.8$.

Редукция и более продвинутые алгоритмы

- теория применима к любому алгоритму ISD
- анализа модификаций алгоритмов Думера/Штерна, MMT, BJMM – нет в открытом доступе

Алгоритмы редукции на практике

Low Weight Codeword Problem

Details on record 17

Seed	2
Weight	211
Vector	10000110000001100000010010000000100001010010000100 00000000000000101010100000000100010000000000000001 .....
Authors	Leo Ducas, Marc Stevens
Algorithm	Hybrid Wagner-Babai using Code Reduction [eprint:2020/869]
Hardware	2x 20-core Xeon Gold 6248 @ 2.5Ghz
Runtime	4 days, 19 hours, 4 minutes.
Comments	Implemented in MCCL, running 40 individual threads that only share best known shortest vector. Wagner is 2 levels with p=3. Babai can use typically 11 basis vectors, basis profile sample: [213 130 81 57 36 27 18 15 11 10 8]. Start from challenge file, found 219 after 38min, 218 after 2h16m, 216 after 17h37m, 214 after 22h37m, 213 after 2d6h52m, 211 after 4d19h4m.
Date	2024-02-09

- $n = 1280$, $R = 0.5$, $w = 144$ (В-Г), decodingchallenge.org

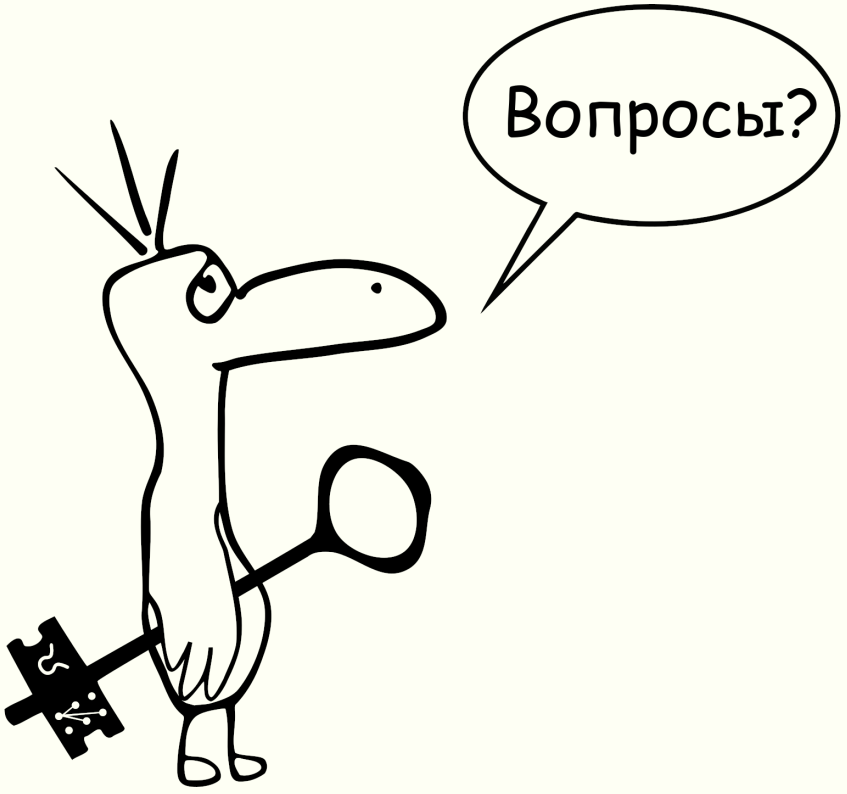
Алгоритмы редукции на практике II

Реализация алгоритма BKZ и результаты экспериментов для $\beta = 24$:

- Колесников Н.С., Новоселов С.А. “Оценка сложности задачи декодирования и нахождение кодовых слов малого веса с помощью редукции базиса кода”. SibeCrypt 2025

Открытые вопросы

1. сложность алгоритма BKZ для кодов
2. полезный фактор-аппроксимации алгоритмов LLL/BKZ для кодов
3. теория для больших q (при $q << n$)
4. анализ модификаций алгоритмов Думера/Штерна, MMT, BJMM
5. другие приложения: системы полиномиальных уравнений над $\mathbb{F}_2$



Контакты

snovoselov@kantiana.ru
crypto-kantiana.com/semyon.novoselov