А. Ромащенко, А. Румянцев, А. Шень

Заметки по теории кодирования

Москва
Издательство МЦНМО, 2009

ББК 22.1
Р47

Ромащенко А., Румянцев А., Шень А.

Р47 Заметки по теории кодирования. — М.: МЦНМО, 2009. — 64 с.

ISBN 978-5-XXXXX-XXX-X

В этих заметках, написанных по материалам лекций М. Судана в Массачусестком технологическом институте (с его любезного разрешения), излагаются базовые результаты теории кодирования, а также некоторые более новые её достижения, представляющие интерес для computer science. Книга рассчитана на математиков и программистов (начиная со студентов младших курсов), впервые знакомящихся с теорией кодирования.

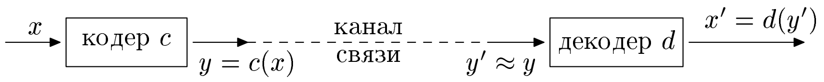
ББК 22.1

Оригинал-макет предоставлен авторами.

ISBN 978-5-XXXXX-XXX-X

Предисловие

Слово «кодирование» в названии этой книжки не означает шифрования (сохранения сообщения в секрете — этим занимается не теория кодирования, а криптография, которой мы не касаемся); не идёт речь также о теоретических и практических проблемах перехода от одной кодировки символов к другой. Основной вопрос теории кодирования — как записать сообщение в такой форме, чтобы искажение некоторой части записи (например, при передаче данных) не помешало восстановлению сообщения в исходной форме.



Имеется в виду, что исходное сообщение x (последовательность битов или других символов) преобразуется в некоторую другую (более длинную) последовательность y. Затем y передаётся по каналу связи с помехами (или хранится на ветхом носителе), превращаясь при этом в y'. Наконец, из y' извлекается сообщение x', которое должно совпадать с исходным (x).

Например, мы можем повторить каждую букву сообщения x три раза, получив втрое более длинное сообщение y. Если одна из букв в слове y испортится, это не помешает восстановить x. В этом примере алгоритм кодирования (кодер) c утраивает каждую букву, а алгоритм декодирования (декодер) d берёт наиболее частую букву в каждой тройке. Этот простейший код позволяет исправлять одну ошибку. (А для исправления двух ошибок достаточно повторять каждую букву пять раз.)

Придумывая код (способы кодирования и декодирования), мы хотим сразу многого:

- чтобы код позволял исправлять возможно большее число ошибок;
- чтобы избыточность кода (насколько y длиннее x) была бы поменьше;
- наконец, чтобы кодирование и декодирование выполнялись бы простыми алгоритмами.

Эти требования к коду во многом противоречат друг другу, и различные варианты компромисса между ними и составляют предмет теории кодирования. (Отметим сразу же, что повторение символов, о котором шла речь — далеко не наилучший способ!)

Вопрос о том, при каких сочетаниях параметров (избыточность и число исправляемых ошибок) такое возможно, до сих пор остаётся открытым, хотя придумано множество различных кодов, часто с использованием сложной математической техники (например, алгебраической геометрии). Вместе с тем многие важные результаты можно изложить, ограничиваясь лишь начальными сведениями из алгебры (многочлены и поля, векторные пространства над полем из двух элементов). Такое изложение и составляет предмет этой книжки, предназначенной для первого знакомства (будущих) программистов и математиков с основными результатами теории кодирования. В неё включены также некоторые более новые результаты (декодирование списком, теорема Голдрайха-Левина, связь с экспандерами).

* * *

Большая часть этих заметок была сделана участниками семинара кафедры математической логики и теории алгоритмов механико-математического факультета МГУ в 2004/5 учебном году, разбиравшими записи лекций Мадху Судана в Массачусетском технологическом институте (по выложенным в интернете материалам, см. `http://theory.csail.mit.edu/~madhu`). Рассказывали на семинаре в основном А. Румянцев и А. Ромащенко, текст подготовил к печати А. Шень.

Мы благодарны Мадху Судану за любезное разрешение использовать его материалы при подготовке текста, и предупреждаем, что возможные ошибки в этом пересказе целиком на нашей совести. Мы благодарим также Григория Кабатянского и Сергея Еханина, которые посмотрели этот текст и указали на некоторые (ныне исправленные) ошибки.

А. Ромащенко, А. Румянцев, А. Шень

1. Коды с исправлением ошибок: постановка задачи

Пусть Σ — конечный *алфавит* (множество *букв*). Через Σ^n обозначаем множество всех *слов* (конечных последовательностей букв) длины n в алфавите Σ. *Расстояние Хэмминга* между двумя словами $x_1 \ldots x_n$ и $y_1 \ldots y_n$ из Σ^n определяется как число позиций, в которых эти слова отличаются (число тех i, при которых $x_i \neq y_i$).

Рассмотрим отображение $F\colon \Sigma^k \to \Sigma^n$ при $k < n$ как способ кодирования информации при её передаче. Получив на вход слово $A \in \Sigma^k$, мы передаём по каналу связи слово $F(A)$. В процессе передачи некоторые символы слова $F(A)$ могут быть искажены. Несмотря на это, мы хотим, чтобы было возможно восстановить исходное слово A. Другими словами, мы хотим, чтобы $F(A)$ нельзя было спутать с $F(A')$ даже после того, как некоторые буквы в обоих словах искажены. Если разрешается искажать не более e позиций (в каждом слове), то для однозначного восстановления необходимо и достаточно, чтобы расстояние между $F(A)$ и $F(A')$ было $2e + 1$ или больше.

Более формально, *кодом* называется отображение $F\colon \Sigma^k \to \Sigma^n$. Значения этого отображения называются *кодовыми словами*. Минимальное из расстояний $d(F(A), F(A'))$ при $A \neq A'$ называется *минимальным расстоянием* кода F. Код с минимальным расстоянием d позволяет исправлять $e = \lfloor (d-1)/2 \rfloor$ ошибок, поскольку при таком e шары радиуса e с центрами в кодовых словах не пересекаются. (*Шар* радиуса e с центром в слове $x \in \Sigma^n$ определяется как множество всех слов, отличающихся от x не более чем в e позициях. Через $\lfloor t \rfloor$ обозначается целая часть числа t, то есть наибольшее целое число, не превосходящее t.)

Такая «помехоустойчивость» кода возникает, как говорят, за счёт «избыточности» при передаче информации: вместо k символов мы передаём n символов, при этом $n > k$. Чем больше e и чем ближе k к n, тем лучше код. Конечно, эти два требования противоречат друг другу и одно достигается за счёт другого.

Задачу построения оптимального кода при данных Σ, n и e можно интерпретировать геометрически: в метрическом пространстве Σ^n нужно упаковать без пересечений как можно больше шаров радиуса e. Другой вариант формулировки: требуется найти как можно больше точек с попарными расстояниями $2e + 1$ или больше. После того как шары (их центры) выбраны, можно взять k, равное целой части логарифма числа таких шаров (точек) по основанию $|\Sigma|$ (число букв во входном алфавите), и произвольно сопоставить слова длины k с центрами шаров. (Число k выбрано так, что шаров хватит; часть точек может остаться неиспользованной.)

Выбор этого соответствия (между кодируемыми словами и их кодами) безразличен, пока мы не интересуемся сложностью алгоритмов кодирования и декодирования. Алгоритм кодирования вычисляет функцию F, то есть по входному слову X даёт $F(X)$. Алгоритм декодирования с исправлением e ошибок получает на вход любое слово Y, отстоящее от некоторого кодового слова $F(X)$ на расстояние не больше e, и даёт X на выходе. Естественно, мы хотим, чтобы алгоритмы кодирования и декодирования были не очень сложными и работали быстро.

2. Базовые оценки

При каких значениях параметров $q = |\Sigma|, k, n, e$ код $F\colon \Sigma^k \to \Sigma^n$, исправляющий e ошибок, существует, а при каких нет? Две самые простые оценки (с той и другой стороны) таковы:

Граница Хэмминга

Если код с указанными параметрами существует, то

$$q^k V_q(e,n) \leqslant q^n$$

Здесь через $V_q(e,n)$ обозначается объём (число элементов) шара радиуса e в пространстве слов длины n в алфавите из q букв. (Очевидно, число элементов в шаре не зависит от того, какой у него центр.)

В самом деле, если код существует, то q^k шаров с центрами в кодовых словах помещаются в пространстве из q^n элементов без пересечений.

Логарифмируя, можно переписать это неравенство (*границу Хэмминга*) в таком виде:

$$\frac{k}{n} + \frac{\log_q V_q(e,n)}{n} \leqslant 1.$$

Первое слагаемое можно назвать «коэффициентом полезного действия» кода (в литературе его часто называют «скоростью кода »).

В англоязычной литературе границу Хэмминга часто называют *volume bound* (volume — объём).

Граница Гилберта

Если
$$(q^k-1)V_q(2e,n)<q^n,$$
то существует код с параметрами q, k, n, e.

В самом деле, будем выбирать кодовые слова одно за другим произвольным образом, следя лишь за тем, чтобы расстояния между ними были больше $2e$. Если нового слова выбрать нельзя, это значит, что всё пространство (в котором q^n элементов) полностью покрыто шарами, каждый из которых состоит из $V_q(2e,n)$ элементов. Указанная оценка гарантирует, что при этом имеется как минимум q^k шаров, что и требуется.

Переходя к логарифмам, эту оценку можно записать так:
$$\frac{k}{n}+\frac{\log_q V_q(2e,n)}{n}\leqslant 1.$$
(для простоты записи мы немного усилили условие и тем самым ослабили утверждение о существовании кода, отбросив вычитание единицы). Это неравенство называют *границей Гилберта*. Не следует путать автора этой оценки Эдгара Гилберта (Edgar N. Gilbert) со знаменитым математиком Давидом Гильбертом (David Hilbert).

Размер шара

Обе эти оценки включают в себя число элементов в шаре, поэтому полезно иметь для этого числа хотя бы приближённую формулу. Для случая $q=2$ число элементов в шаре радиуса s равно сумме биномиальных коэффициентов
$$C_n^0+C_n^1+\ldots+C_n^s$$
При $s\leqslant n/2$ (в основном нас интересует этот случай, так как шаров большего радиуса даже и два не поместится) слагаемые в этой

сумме возрастают слева направо и (с точностью до полиномиальных от n множителей) можно ограничиться последним слагаемым. С помощью формулы Стирлинга получаем, что оно (и вся сумма) есть

$$\mathrm{poly}(n)2^{nH(p)}$$

где $p = s/n$, а $H(p)$ — *энтропия Шеннона* распределения $(p, 1-p)$, которая определяется как

$$H(p) = p \log \frac{1}{p} + (1-p) \log \frac{1}{1-p}.$$

Для произвольного q формула имеет вид

$$V_q(s, n) = \mathrm{poly}(n) q^{nH_q(p)},$$

где $p = s/n$, а

$$H_q(p) = \left[p \log_q \frac{1}{p} + (1-p) \log_q \frac{1}{1-p} \right] + p \log_q (q-1)$$

Здесь квадратная скобка, как и раньше, соответствует выбору не более чем pn позиций, в которых есть отличия (от центра шара), но появляется ещё один член: в каждой из этих pn позиций может стоять любой из $q-1$ символов (не считая теперешнего).

Графики

Полезно изобразить границы Хэмминга и Гилберта на одном графике. По горизонтали будем откладывать кодовое расстояние d (примерно равное $2e$, удвоенному числу исправляемых ошибок) в долях n; по вертикали — коэффициент полезного действия кода (отношение k/n, где k — число кодируемых символов, а n — длина кода).

Ограничимся случаем $q = 2$ и будем использовать приближённую формулу для объёма шара.

Необходимое условие существование кода, исправляющего e ошибок (граница Хэмминга):

$$\frac{k}{n} + H(e/n) \leqslant 1 + O(\log n / n)$$

Достаточное условие существования кода с расстоянием не меньше d

$$\frac{k}{n} + H(d/n) \leqslant 1 - O(\log n/n)$$

Формулы эти похожи, но только d и e отличаются примерно в два раза. Поэтому график для границы Хэмминга получается двукратным растяжением графика для границы Гилберта по горизонтали (рис. 1).

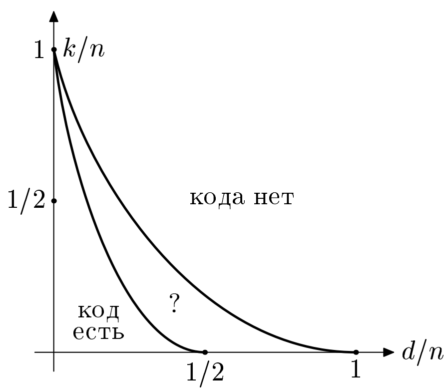


Рис. 1. Границы Хэмминга и Гилберта.

По этой картинке можно определить, скажем, что кода с коэффициентом полезного действия $1/2$ и расстоянием в $n/3$ не существует — соответствующая точка лежит выше границы Хэмминга. С другой стороны, код с коэффициентом полезного действия $1/8$ и расстоянием $n/4$ (и тем самым допускающим исправление $n/8$ ошибок) существует — соответствующая точка лежит ниже границы Гилберта. (Всё сказанное относится к достаточно большим n, так как мы пренебрегали членами порядка $O(\log n/n)$.)

Что происходит между этими границами — один из основных вопросов теории кодирования, до сих пор не решённый полностью (несмотря на множество частичных результатов).

3. Случайные коды

Границу Гилберта можно достичь и другими способами. В этом разделе мы используем случайный код, а в следующем разделе — линейный.

Пусть фиксировано некоторое число N. Будем выбирать N кодовых слов $\xi_1, \ldots, \xi_N$ случайно среди q^n равновероятных элементов Σ^n. При этом мы считаем все $\xi_1, \ldots, \xi_n$ независимыми. (В частности, вероятность совпадения ξ_i и ξ_j при $i \neq j$ отлична от нуля, хотя и мала.)

Для фиксированного $i \in \{1, \ldots, N\}$ рассмотрим вероятность того, что в шар радиуса $2e$ с центром ξ_i попадут другие кодовые слова. Вероятность попадания ξ_j с данным j равна $V_q(2e, n)/q^n$ (доле пространства, занимаемой шаром радиуса $2e$); вероятность того, что хотя бы одно ξ_j (при $j \neq i$) попадёт в этот шар, не больше $NV_q(2e, n)/q^n$. Будем называть те i, при которых в шар с центром в ξ_i попали другие ξ_j, «плохими». Тогда вероятность каждого i оказаться плохим не больше $NV_q(2e, n)/q^n$. Поэтому математическое ожидание числа плохих i не больше $N^2V_q(2e, n)/q^n$, а математическое ожидание *доли* плохих i (среди всех чисел $1, \ldots, N$) — не больше $NV_q(2e, n)/q^n$.

Предположим теперь, что

$$\frac{NV_q(2e, n)}{q^n} \leqslant 1/2$$

(вдвое меньшее число кодовых слов, чем допускает граница Гилберта). Тогда математическое ожидание доли плохих i не больше $1/2$, и, следовательно, существует вариант, при котором эта доля не больше $1/2$. Зафиксировав один из таких вариантов и выкинув плохие кодовые слова (их не более половины), мы получим код с расстоянием не менее $2e + 1$, который всего лишь в четыре раза хуже (по числу слов — мы начали с вдвое меньшего числа, да ещё отбросили половину), чем построенный ранее.

4. Линейные коды

Если q есть степень простого числа, то (как известно из алгебры) существует поле $\mathbb{F}_q$ из q элементов. В этом случае можно считать, что $\Sigma = \mathbb{F}_q$, а слова образуют векторное пространство (размерности n или k) над $\mathbb{F}_q$. В этой ситуации возникает понятие *линейного* кода. Такой код представляет собой линейное (над $\mathbb{F}_q$) отображение $\mathbb{F}_q^k \to \mathbb{F}_q^n$. Это отображение задаётся матрицей размера $n \times k$, элементы которой принадлежат полю $\mathbb{F}_q$.

Множество всех кодовых слов линейного кода (образ этого отображения) представляет собой подпространство в $\mathbb{F}_q^n$. В силу линейности окрестности всех кодовых слов устроены одинаково (отличаются сдвигом). Поэтому минимальное расстояние линейного кода можно измерять в любой точке, в частности, в нуле: оно равно минимально возможному числу ненулевых координат в ненулевом кодовом слове.

Таким образом, для построения линейного кода с расстоянием больше $2e$ достаточно указать k-мерное подпространство C в пространстве $\mathbb{F}_q^n$, все векторы которого содержат более $2e$ ненулевых координат (не попадают в шар радиуса $2e$ с центром в нуле). (Само кодовое отображение $\mathbb{F}_q^k \to C$ после этого можно выбрать любым, лишь бы оно было изоморфизмом линейных пространств.)

Чтобы построить искомый линейный код, будем добавлять в базис подпространства C вектор за вектором, следя за тем, чтобы кодовое расстояние оставалось больше $2e$. Пусть уже есть s базисных векторов, которые порождают некоторое подпространство. Новый (добавляемый) базисный вектор должен быть на расстоянии более $2e$ от всех векторов подпространства; это, как легко понять, гарантирует, что и после его добавления расстояние между любыми двумя векторами останется больше $2e$. Другими словами, новый вектор должен лежать вне объединения 2^s шаров радиуса $2e$.

Таким образом, для получения k-мерного подпространства достаточно выполнения неравенства

$$q^{k-1} V_q(2e, n) < q^n$$

Эту оценку называют границей *Варшамова–Гилберта.* Она чуть лучше оценки в границе Гилберта (там было $q^k - 1$, а теперь q^{k-1}), но это рассуждение годится лишь при некоторых q (степенях простых чисел). При стремлении n бесконечности оценки Гилберта и Варшамова–Гилберта дают одинаковые асимптотические оценки для отношения k/n.

Заметим, что в случае линейного кода кодирование выполняется легко (умножение на матрицу кода), но про декодирование по-прежнему ничего хорошего сказать нельзя (хотя для некоторых линейных кодов, как мы увидим, существуют быстрые алгоритмы декодирования).

Линейное подпространство размерности k в пространстве $\mathbb{F}_q^n$ можно задавать не только как образ линейного оператора $\mathbb{F}_q^k \to \mathbb{F}_q^n$, но и как ядро оператора $\mathbb{F}_q^n \to \mathbb{F}^{n-k}$, имеющего максимальный ранг $(n-k)$. Такой оператор задаётся матрицей T из $n-k$ строк и n столбцов. Вектор-столбец x высоты n является кодовым словом тогда и только тогда, когда $Tx = 0$ ($n-k$ уравнений с n переменными; можно сказать, что эти уравнения представляют собой «контрольные суммы», которые у кодовых слов должны равняться нулю). Эта матрица называется *проверочной*, поскольку умножением на неё проверяется принадлежность множеству кодовых слов. Код с проверочной матрицей T имеет расстояние d тогда и только тогда, когда любые $d-1$ столбцов матрицы T линейно независимы. В самом деле, кодовое слово, в котором лишь $d-1$ позиций отличны от нуля, как раз и выражает линейную зависимость между $d-1$ столбцами проверочной матрицы.

5. Код Хэмминга

Сейчас мы рассмотрим случай, когда удаётся указать простой код с наилучшими возможными параметрами (шары плотно заполняют всё пространство, достигая границы Хэмминга). Этот код называется *кодом Хэмминга* и позволяет исправлять одну ошибку, то есть имеет минимальное расстояние 3.

Рассмотрим алфавит из двух букв $\mathbb{B} = \{0, 1\}$. Мы будем рассматривать его как поле из двух элементов со сложением ($\oplus$) и умножением по модулю 2.

В пространстве $\mathbb{B}^n$ шар радиуса 1 состоит из $n+1$ элементов (центр и ещё n слов, отличающихся от центрального в одной из n позиций). Всего слов 2^n, поэтому шансы на укладку шаров без пробелов и перекрытий есть, лишь если 2^n кратно $n+1$, то есть если $n+1$ есть степень двойки: $n = 2^s - 1$ для некоторого целого s (при $n = 3, 7, 15, 31, \ldots$; случай $n = 1$ тривиален).

При $n = 3$ есть 8 вершин куба, которые разбиваются на два шара с центрами в противоположных вершинах. Каждый из шаров содержит 4 элемента, так что плотная упаковка действительно возможна.

Оказывается, что она возможна и при остальных значениях n из указанной серии. Чтобы убедиться в этом, рассмотрим произвольное $n = 2^s - 1$ и запишем все числа от 1 до n в двоичной системе по строкам матрицы. Например, при $n = 7$ получится матрица

$$\begin{matrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ 0 & 1 & 1 \\ 1 & 0 & 0 \\ 1 & 0 & 1 \\ 1 & 1 & 0 \\ 1 & 1 & 1 \end{matrix}$$

Столбцы этой матрицы будем рассматривать как способы вычисления контрольных сумм. А именно, пусть дано произвольное слово

$$x_1x_2x_3x_4x_5x_6x_7$$

из $\mathbb{B}^7$. Для него мы вычислим три контрольных суммы, складывая по модулю 2 те биты, против которых в матрице стоит единица: $x_4 \oplus x_5 \oplus x_6 \oplus x_7$ (для первого столбца), $x_2 \oplus x_3 \oplus x_6 \oplus x_7$ (для второго столбца) и $x_1 \oplus x_3 \oplus x_5 \oplus x_7$ (для третьего столбца).

Можно сказать, что мы умножаем строку $(x_1, \ldots, x_7)$ на эту матрицу, получая строку из трёх контрольных сумм.

Кодовые слова (центры шаров) — это те строки длины 7, для которых все три контрольные суммы равны нулю. Линейная алгебра говорит, что они образуют подпространство размерности 4 (три уравнения на семь переменных; легко проверить, что уравнения независимы, посмотрев на первую, вторую и четвёртую строки), состоящее из $2^4 = 16$ кодовых слов. Это в точности соответствует определению линейного кода с помощью проверочной матрицы, только матрица транспонирована.

Что случится с контрольными суммами, если мы отойдём от центра шара на единицу, то есть изменим какой-либо из битов x_i? Изменятся (станут равными единице) те контрольные суммы, в которые этот бит входит, то есть те, где в i-й строке матрицы стоит единица. При этом контрольные суммы образуют как раз двоичную запись числа i, поскольку так устроена матрица. Таким образом, по этим контрольным суммам мы восстановим, какой бит изменился, и сможем изменить его обратно.

Мы описали алгоритм декодирования, исправляющий ошибку в любом (но только одном) бите. Отсюда автоматически следует, что шары не пересекаются. Тем самым у нас есть 16 шаров, каждый из которых состоит из 8 элементов, и вместе они покрывают все 128 элементов семимерного булева куба $\mathbb{B}^7$.

Точно так же для произвольных s и $n = 2^s - 1$ строится матрица из n строк длины s (двоичные записи чисел $1, \ldots, n$). Её столбцы указывают s контрольных сумм для векторов из $\mathbb{B}^n$. Кодовых слов 2^{n-s}, в каждом шаре $n + 1 = 2^s$ элементов и они покрывают $\mathbb{B}^n$ полностью.

Заметим, что этот код не только *совершенный* (всё пространство занято шарами без пробелов), но и имеет простые алгоритмы кодирования и декодирования. Единственный его недостаток — что он исправляет только одну ошибку.

Замечания.

1. Можно вычеркнуть из матрицы часть строк. При этом мы получим код с тем же числом контрольных сумм, но с меньшим числом битов. Он уже не будет совершенным, но будет достаточно эффективным. (Больше половины строк выкидывать нет смысла, поскольку тогда уж лучше уменьшить ширину матрицы. А если мы выбрасываем меньше половины строк, то теряем не больше половины места.)

2. Аналогичное построение возможно и при $q > 2$, если существует поле $\mathbb{F}_q$ из q элементов (это бывает, когда q есть степень простого числа). Для этого напишем в строках матрицы ненулевые векторы из $\mathbb{F}_q^s$, причём из каждого класса пропорциональных друг другу векторов оставим только один. Всего ненулевых векторов $q^s - 1$, в каждом классе $q - 1$ пропорциональных друг другу векторов, так что классов (и строк матрицы) будет $n = (q^s - 1)/(q - 1)$. Ясно, что ранг полученной матрицы равен s.

Изменение одной позиции в кодовом слове изменит контрольные суммы на вектор, пропорциональный одной из строк матрицы, и по построению эта строка (и тем самым номер изменённого бита) определится однозначно.

Кодовых слов будет q^{n-s}, размер шара $1 + n(q - 1)$ (в каждой из n позиций можно заменить букву на $(q - 1)$ других, да ещё центр шара), что равно
$$1 + \frac{q^s - 1}{q - 1}(q - 1) = q^s.$$

Таким образом, шары заполняют всё пространство.

3. Для доказательства того, что код Хэмминга имеет расстояние не меньше 3, достаточно было бы сослаться на то, что любые две строки в нашей матрице (=любые два столбца в проверочной матрице из предыдущего раздела) линейно независимы. Но нам важен и простой алгоритм декодирования.

6. Неравенство Синглтона

Для любого кода $F\colon \Sigma^k \to \Sigma^n$ с расстоянием d выполняется *неравенство Синглтона*

$$d \leqslant n - k + 1$$

В самом деле, пусть имеется q^k кодовых слов в Σ^n. Выделим в этих словах первые $k-1$ позиций. Для некоторой пары кодовых слов эти $k-1$ позиций совпадают по принципу Дирихле (число слов q^k больше числа вариантов q^{k-1}). Расстояние между такими словами не больше $n-(k-1) = n-k+1$, что и требовалось доказать.

Для кодов в двухбуквенном алфавите эта оценка уступает оценке Хэмминга. А именно, для кода с расстоянием d она оценивает сверху коэффициент полезного действия (скорость) кода величиной $1-d/n$, что на графике рис. 1 соответствует прямой, соединяющей точки $(0,1)$ и $(1,0)$. Видно, что эта прямая соединяет концы кривой Хэмминга и лежит выше неё.

Для большего размера алфавита это уже не так. Дело в том, что оценка Хэмминга не даёт ничего хорошего при $d \approx n$. Рассмотрим, например, случай $q=3$ и шар радиуса $n/2$ или чуть меньше. Сколько элементов в этом шаре? Позиций несовпадения примерно $n/2$ среди n; вариантов выбора не более 2^n. После такого выбора останется выбрать в каждой позиции несовпадения один из двух несовпадающих символов, получится не более $2^{n/2}$. Таким образом, шар радиуса не более $n/2$ содержит не более $2^{3n/2} = (2\sqrt{2})^n$ элементов. А всего в пространстве 3^n точек, поэтому оценка на число шаров будет всего лишь $(3/2\sqrt{2})^n$ и кривая Хэмминга не стремится к нулю при $d \approx n$ (в отличие от прямой Синглтона).

7. Код Рида – Соломона

Рассмотрим конечное поле $\mathbb{F}_q$ из q элементов (из алгебры известно, что это возможно, когда q есть степень простого числа; для начала можно ограничиться случаем, когда само q простое и рассматривать поле вычетов по модулю q).

Кодируемое слово $a_0a_1\ldots a_{k-1}$ (где $a_i \in \Sigma = \mathbb{F}_q$) будем рассматривать как последовательность коэффициентов многочлена

$$A(x) = a_0 + a_1x + \ldots + a_{k-1}x^{k-1}$$

степени меньше k. Ему соответствует кодовое слово, состоящее из значений многочлена A в n заранее выбранных точках поля $\mathbb{F}_q$ (естественно, что для этого надо, чтобы $n \leqslant q$). Из курса алгебры известно, что два различных многочлена степени меньше k могут совпадать максимум в $k-1$ точках. Поэтому для различных многочленов имеется как минимум $n-k+1$ точек (среди выбранных нами n) несовпадения, то есть кодовое расстояние построенного *кода Рида – Соломона* не меньше $n-k+1$. Тем самым для этих кодов неравенство Синглтона обращается в равенство.

Кодирование выполняется просто. Некоторая проблема тут с декодированием: как восстановить многочлен по таблице значений, где некоторые значения испорчены? Оказывается, что это можно сделать за полиномиальное время (чего с первого взгляда не скажешь).

8. Декодирование кодов Рида – Соломона

Итак, пусть имеется таблица значений многочлена P степени меньше k в некоторых n точках, причём $e = \lfloor (n-k)/2 \rfloor$ значений в ней могут быть неверными. (Такое значение e соответствует числу исправляемых ошибок при кодовом расстоянии $d = n-k+1$.) Другими словами, нам дана искажённая функция $\tilde{P}$, определённая в n точках и отличающаяся от неизвестного нам многочлена P не более чем в e точках. Как найти P?

Поскольку число ошибок не больше e, существует (пока что неизвестный нам) многочлен $D(x)$ степени e, который обращается в нуль в местах ошибок. Тогда $P(x)D(x)$ равно $\tilde{P}(x)D(x)$ во всех n точках, поскольку разница между $P(x)$ и $\tilde{P}(x)$ приходится на те x, где $D(x) = 0$. Многочлен $Q(x) = P(x)D(x)$ имеет степень меньше $k + e$. Таким образом, верна

Лемма 1. Существуют многочлены $D(x)$ степени e и $Q(x)$ степени менее $k + e$, для которых $\tilde{P}(x)D(x) = Q(x)$ для всех n точек.

Удобно договориться, что старший коэффициент многочлена D (при x^e) равен единице. Тогда лемму 1 можно считать утверждением о разрешимости некоторой системы линейных уравнений. Неизвестными в ней являются остальные e коэффициентов многочлена D и $k+e$ коэффициентов многочлена Q (всего $k+2e$ штук), а уравнений в ней n (что не меньше $k + 2e$, хотя нам это и не важно).

Тем самым некоторые многочлены, удовлетворяющие условиям леммы 1, можно найти, решив эту систему. Правда, пока мы не знаем, как они связаны с «настоящими» D и Q. Об этом говорит

Лемма 2. Пусть $\tilde{D}(x)$ и $\tilde{Q}(x)$ — произвольные многочлены, удовлетворяющие условиям леммы 1 (это значит, что они имеют нужные степени и равенство $\tilde{P}(x)\tilde{D}(x) = \tilde{Q}(x)$ выполняется во всех n точках). Тогда

(**а**) $\tilde{Q}$ делится на $\tilde{D}$ в кольце многочленов $\mathbb{F}_q[x]$;

(**б**) частное $\tilde{Q}/\tilde{D}$ равно исходному (и искомому) многочлену P.

Доказательство. Многочлены $P\tilde{D}$ и $\tilde{Q}$ имеют степень меньше $k + e$. Они совпадают во всех тех точках, где $P = \tilde{P}$, а таких точек как минимум $n - e \geqslant k + e$, поэтому эти многочлены равны.

Применяя эти две леммы, мы получаем алгоритм декодирования кодов Рида-Соломона. Оба его этапа — и решение системы линейных уравнений, и деление многочленов, — выполняются за время, полиномиальное от n и $\log q$.

Замечание 1. Это утверждение требует некоторых пояснений. Дело в том, что поле $\mathbb{F}_q$ — вещь абстрактная. Оно единственно с точностью до изоморфизма, но не имеет какого-то «канонического» представления. Можно доказать, что его элементы можно представить числами $0 \ldots q - 1$ таким образом, что операции сложения, вычитания, умножения и деления выполняются за полиномиальное от $\log q$ время. При использовании такого представления алгоритм декодирования и будет полиномиальным от n и $\log q$.

Замечание 2. Можно рассматривать также задачу о декодировании с ошибками и пропусками (или, как обычно говорят. «стираниями»). Под *пропуском* мы понимаем позицию, в которой буква неизвестна. (Это лучше, чем неверная буква, так как мы хотя бы знаем, в какой позиции проблема.) Видно, что для декодирования кода Рида – Соломона нужно, чтобы число пропусков плюс удвоенное число ошибок не превосходило $n-k$ (поскольку пропуск одной позиции даёт снова код Рида – Соломона с заменой n на $n-1$). Так что пропуске считается за пол-ошибки («два переезда — как один пожар»).

Всем хороши эти коды, но только они требуют достаточно большого алфавита (размер алфавита должен быть не меньше длины кодовых слов, да ещё быть степенью простого числа). Уменьшить алфавит можно, используя конструкцию каскадных кодов, описываемую в следующем разделе.

9. Каскадные коды

Пусть имеется некоторый код с бо́льшим алфавитом Σ. Можно ли его переделать в код с меньшим (скажем, двоичным) алфавитом? Например, пусть Σ содержит 2^s букв. Тогда можно записать каждую букву из Σ с помощью блока из s битов. При этом код $\Sigma^k \to \Sigma^n$ превратится в код $\mathbb{B}^{sk} \to \mathbb{B}^{sn}$.

Кодовое расстояние при этом не уменьшится. В самом деле, если для исходного кода оно было d, то теперь два кодовых слова отличаются по крайней мере в d блоках, а отличие в блоке означает отличие хотя бы в одном бите этого блока.

Общая длина кодовых слов, однако, возрастёт (в s раз), так что допустимый процент ошибок в кодовом слове уменьшится в s раз.

Более надёжный двоичный код получится, если кодировать буквы алфавита Σ блоками не из s битов, а из большего числа битов, причём использовать при этом какой-либо код, исправляющий ошибки.

В общем случае такая конструкция «каскадного кода», или «конкатенации» двух кодов, применима к кодам

$$F_1\colon \Sigma_1^{k_1} \to \Sigma_1^{n_1} \quad \text{и} \quad F_2\colon \Sigma_2^{k_2} \to \Sigma_2^{n_2},$$

если $|\Sigma_1| = |\Sigma_2|^{k_2}$. Тогда каждую букву алфавита Σ_1 можно рассматривать как блок из k_2 символов алфавита Σ_2 и кодировать блоком из n_2 символов алфавита Σ_2.

Формально, определим *конкатенацию* «внешнего» кода F_1 и «внутреннего» кода F_2 как код

$$F\colon \Sigma_2^{k_1k_2} \to \Sigma_2^{n_1n_2}$$

Чтобы найти значение F на слове длины k_1k_2, составленном из букв алфавита Σ_2, это слово надо разбить на k_1 блоков длиной k_2. Если эти блоки считать буквами алфавита Σ_1, получится слово длиной k_1 в алфавите Σ_1. Применив к нему код F_1, получим слово из n_1 блоков длины k_2. Остаётся закодировать каждый из этих блоков по отдельности кодом F_2, получив n_1 блоков длиной n_2, или всего n_1n_2 букв алфавита Σ_2.

Теорема. Кодовое расстояние конкатенации кодов не меньше произведения их кодовых расстояний.

В самом деле, рассмотрим два различных кодовых слова в конкатенации. Каждое из них состоит из n_1 блоков длиной n_2. Условие на код F_1 гарантирует, что имеется по крайней мере d_1 различных блоков. Условие на код F_2 гарантирует, что в каждой паре различных блоков имеется (как минимум) d_2 различных букв, всего получается d_1d_2 различий.

Таким образом, каскадный код позволяет исправлять около $d_1d_2/2$ ошибок (половина кодового расстояния). Но как именно это сделать?

10. Декодирование каскадных кодов

Пусть код F является конкатенацией кодов F_1 и F_2. Имея алгоритмы декодирования для F_1 и F_2, можно предложить следующий естественный алгоритм декодирования для F: декодировать каждый блок с помощью F_2-декодирования, а к полученному слову применить F_1-декодирование. Чтобы этот алгоритм работал, в декодируемом слове должно быть не более e_1 блоков с более чем e_2 ошибками (где e_1 и e_2 — допустимые числа ошибок для кодов F_1 и F_2). Это заведомо будет так, если общее число ошибок (во всех блоках) не больше e_1e_2. Учитывая, что $e_i \approx d_i/2$, мы видим, что такой алгоритм позволяет исправлять примерно $d_1d_2/4$ ошибок, что составляет четверть кодового расстояния и вдвое меньше максимально допустимого числа ошибок (которое примерно равно половине кодового расстояния).

Более устойчивый к ошибкам (и тем не менее достаточно эффективный) алгоритм декодирования возможен, если в качестве внешнего кода (как часто бывает) применяется код Рида – Соломона. Или любой другой код, который позволяет эффективно декодировать слова с ошибками и пропусками, причём пропуск считается за полошибки. В этом случае удаётся исправлять до $d_1 e_2$ ошибок (что соответствует приведённой выше оценке для минимального расстояния каскадного кода).

Делается это так. Код F_2 позволяет исправлять $e_2 = \lfloor (d_2-1)/2 \rfloor$ ошибок. Пусть дан некоторый «порог» t в диапазоне $0 \ldots e_2$. (Как его выбирать, мы обсудим дальше.) Получив для декодирования n_1 блоков по n_2 символов в каждом, мы применяем алгоритм F_2-декодирования к каждому блоку в отдельности, а затем для проверки применяем алгоритм кодирования и смотрим, в скольких позициях есть отличие. Если полученное кодовое слов отличается от исходного блока более чем в e_2 позициях (а также если алгоритм декодирования вообще не дал результата), то этот блок объявляется «неизвестным». Более того, если число отличий в блоке больше порога t, то блок также объявляется неизвестным. Результаты декодирования остальных («известных») блоков вместе с информацией о том, какие блоки неизвестны, подаются на вход алгоритма декодирования кода F_1 — который, по предположению, умеет работать с u пропусками и v ошибками, если $u + 2v < d_1$.

Лемма. Если общее число ошибочных позиций (среди $n_1 n_2$ поданных на вход описанного алгоритма) меньше $d_1 e_2$, то описанный процесс позволит правильно декодировать исходное слово хотя бы при одном значении параметра t.

Прежде чем доказывать лемму, заметим, что она не говорит, как найти хорошее значение параметра t — но это и не нужно. Мы можем перепробовать все значения (их число заведомо не превосходит n, поэтому этот перебор не повредит полиномиальности алгоритма) и затем проверить полученные результаты с помощью кодирования (правильный должен давать менее $d_1 d_2/2$ отличий).

Осталось доказать лемму. Предположим, что некоторое t выбрано. Пусть ε_i — истинное число ошибок в i-м блоке (при $i = 1, 2, \ldots, n_1$). Числа ε_i нам неизвестны, но мы знаем, что:

• если $\varepsilon_i \leqslant t$, то i-й блок декодирован правильно (и сочтён известным);

• если $t < \varepsilon_i < d_2 - t$, то i-й блок объявлен неизвестным (мы не могли принять его за другой блок и счесть известным, так как если расстояние до другого центра шара не больше t, то расстояние до нашего центра не меньше $d_2 - t$ по неравенству треугольника);

• наконец, если $\varepsilon_i \geqslant d_2 - t$, то с i-м блоком может быть всякое (он может быть сочтён неизвестным или быть принятым за другой блок).

Таким образом, для данного t все индексы $i = 1, 2, \ldots, n_1$ делятся на три категории (задаваемые тремя указанными неравенствами на ε_i). Количество таких индексов обозначим через $\alpha(t)$, $\beta(t)$ и $\gamma(t)$ соответственно. Нам достаточно доказать, что при некотором t (из промежутка $0 \ldots e_2$) выполняется неравенство $\beta(t) + 2\gamma(t) < d_1$. (Реальная ситуация может быть даже немного лучше, если блок с большим числом ошибок не был принят за другой блок, а был сочтён неизвестным.)

Чтобы доказать существование такого значения t, достаточно установить, что *среднее* значение величины $\beta(t) + 2\gamma(t)$ меньше d_1. При усреднении мы считаем все значения t от 0 до e_2 равновероятными, то есть речь идёт просто о среднем арифметическом. А кодируемое слово, его код и внесённые в него ошибки (тем самым, и числа ε_i) мы считаем фиксированными.

В силу аддитивности математического ожидания (линейности среднего арифметического) можно вычислить средний вклад каждого отдельного i, а потом сложить их по всем i. Итак, пусть фиксировано некоторое i (и тем самым значение ε_i).

• Если $\varepsilon_i \leqslant e_2$, то в третью категорию мы попасть не можем, а во вторую попадём, когда $t < \varepsilon_i$. Вероятность этого события (она же — вклад i в интересующее нас среднее) есть $\varepsilon_i/(e_2 + 1)$.

• Если $\varepsilon_i > e_2$, то мы попадём либо во вторую категорию, либо в третью — последнее случится, если $t \geqslant d_2 - \varepsilon_i$. Вероятность этого события равна

$$\frac{(e_2 + 1) - (d_2 - \varepsilon_i)}{e_2 + 1},$$

поэтому матожидание вклада позиции i в $\beta + 2\gamma$ не превосходит

$$1 + \frac{e_2 + 1 - d_2 + \varepsilon_i}{e_2 + 1} = \frac{2e_2 + 2 - d_2 + \varepsilon_i}{e_2 + 1} \leqslant \frac{\varepsilon_i + 1}{e_2 + 1} < \frac{\varepsilon_i}{e_2}$$

(мы использовали при оценке, что $2e_2 + 1 \leqslant d_2$: таково соотношение между числом исправляемых ошибок и кодовым расстоянием для внутреннего кода).

Заметим, что полученная только что оценка верна (хотя и тривиальна) при $\varepsilon > d_2$.

Таким образом, среднее в обоих случаях не превосходит ε_i/e_2 для данного i и $(\sum \varepsilon_i)/e_2$ в сумме. Что по предположению леммы меньше d_1 (ведь $\sum \varepsilon_i$ есть общее число ошибок в кодовом слове каскадного кода).

Лемма доказана.

11. Теорема Форни

Будем рассматривать двоичные коды с возрастающей длиной кодового слова и требовать, чтобы доля исправляемых ошибок (e/n в наших обозначениях) и коэффициент полезного действия кода (k/n в наших обозначениях) были отделены от нуля.

Существование таких кодов очевидно следует из оценки Варшамова – Гилберта.

Но если мы хотим, чтобы алгоритмы кодирования и декодирования были бы достаточно эффективны (скажем, полиномиальны по n, длине кодового слова), требуется более сложная конструкция.

В двух словах эта конструкция (Форни, середина 1960-х годов) такова: рассмотрим конкатенацию двух кодов, причём внешним является код Рида – Соломона, а внутренний ищется перебором кодовых слов, как в доказательстве оценки Варшамова – Гилберта. Если внутренний код имеет длину кодового слова $O(\log n)$, то его поиск и алгоритмы кодирования/декодирования будут экспоненциальными от $\log n$, но полиномиальными по n.

Более подробно. При построении каскадного кода коэффициенты полезного действия перемножаются, как и доли исправляемых ошибок (даже если не использовать улучшенный алгоритм декодирования из предыдущего раздела). Поэтому нам нужно всего лишь, чтобы для каждого из них коэффициент полезного действия и доля исправляемых ошибок были отделены от нуля.

Для наглядности возьмём какой-нибудь пример с конкретными значениями параметров. Фиксируем натуральное k и рассмотрим поле $\mathbb{F}$ из 2^k элементов. Кодируемое слово будет состоять из 2^{k-1} блоков (число блоков — половина размера поля). Каждый блок содержит k битов, то есть представляет собой элемент поля $\mathbb{F}$. Мы считаем блоки коэффициентами многочлена степени меньше 2^{k-1}.

Внешним кодом является код Рида–Соломона: от коэффициентов этого многочлена мы переходим к его значениям во всех 2^k точках поля. При этом число блоков удваивается.

Далее мы кодируем по отдельности каждый блок с помощью внутреннего кода. Будем считать, что при кодировании блок удлиняется вдвое (из k битов получается $2k$). Граница Варшамова–Гилберта (рис. 1 на с. 9) показывает, что при этом можно добиться кодового расстояния 10% (от длины кодового слова, в данном случае $2k$) и исправления 5% ошибок.

В итоге конкатенация кодов увеличивает число битов в 4 раза и имеет кодовое расстояние 5% (от длины слова). При этом тривиальный алгоритм декодирования позволяет исправлять 1,25% ошибок, а улучшенный — 2,5%.

Осталось оценить сложность алгоритмов кодирования и декодирования. Элементы поля из 2^k элементов можно представить k-битовыми строками. Вспомним построение этого поля, известное из курса алгебры. Оно есть кольцо многочленов $\mathbb{F}_2[t]$, профакторизованное по неприводимому многочлену степени k. (Здесь $\mathbb{F}_2 = \{0, 1\}$ — поле из двух элементов.) Как найти этот неприводимый многочлен? Сейчас у нас большой запас времени и можно просто испробовать все многочлены степени k, проверяя их неприводимость перебором возможных делителей. Это не лучший способ проверки неприводимости — для этого существует более эффективный *алгоритм Берлекэмпа*, но использовать его не обязательно, поскольку многочленов 2^k и возможных делителей ещё меньше, так что перебор будет полиномиальным по длине кодируемого слова, равной $k2^{k-1}$. После отыскания такого многочлена операции в поле выполняются за полиномиальное от k время (что даже лучше, чем нужно). Так что единственная сложная операция — это поиск кода, выполняемый при доказательстве оценки Варшамова–Гилберта. Но и он полиномиален по 2^k (по крайней мере в двух доказательствах из трёх приведённых нами — не годится доказательство со случайным кодом, так как там вероятностное пространство имеет размер двойной экспоненты).

Декодирование внутреннего кода при этом происходит с помощью перебора всех кодовых слов внутреннего кода (что допустимо, так как их не более 2^k).

Замечание. Вместо использования поля из 2^k элементов можно было бы (с чуть худшими параметрами) использовать поле вычетов по простому модулю p между 2^k и 2^{k+1}. Такое простое число существует («постулат Бертрана«, он же теорема Чебышёва), и найти его можно перебором (полиномиальное от 2^k число вариантов).

12. Код Форни – Возенкрафта – Юстесена

Интересная модификация конструкции Форни (см. предыдущий раздел) была предложена Юстесеном. Её идея в том, что мы не обязаны использовать один и тот же внутренний код для всех блоков. Более того, если для некоторой малой части блоков мы выберем неудачный внутренний код, это тоже не страшно (хотя и приведёт к некоторому уменьшению кодового расстояния конкатенации).

Оказывается, что можно (следуя Возенкрафту) предложить очень простое семейство кодов, большинство из которых хорошие. А именно, пусть $\mathbb{F}$ — поле из 2^k элементов, представленных k-битовыми строками. Для каждого $\alpha \in \mathbb{F}$ рассмотрим код

$$C_\alpha \colon x \mapsto (x, \alpha \cdot x)$$

(к произвольным k битам дописываются ещё k битов, получаемые из первых умножением на α в смысле поля $\mathbb{F}$). Мы будем считать, что сложение в поле $\mathbb{F}$ соответствует побитовому сложению строк по модулю 2. (Именно так и получается при построении поля как факторкольца.) Тогда код C_α является линейным (для любого элемента $\alpha \in \mathbb{F}$).

Не всегда код C_α хорош. Например, при $\alpha = 0$ мы дописываем нули, что совсем бессмысленно; при $\alpha = 1$ мы дублируем каждый бит слова x, что тоже ничего хорошего не даёт. Но оказывается, что при малых $\varepsilon > 0$ доля тех $\alpha \in \mathbb{F}$, при которых кодовое расстояние кода C_α больше εk, близка к единице. Это следует из такой леммы:

Лемма Возенкрафта. Доля тех $\alpha \in \mathbb{F}$, при которых код C_α имеет кодовое расстояние не более s, не превосходит

$$\frac{V_2(s,k)^2}{2^k}.$$

Поскольку $V_2(s,k) \approx 2^{kH(s/k)}$, при малых s/k это отношение, примерно равное $2^{(2H(s/k)-1)k}$, близко к нулю.

Доказательство леммы Возенкрафта совсем просто. Пусть кодовое расстояние для C_α не больше s. Поскольку код линеен, это значит, что найдётся такое ненулевое слово x, при котором общее количество единиц в битовых строках x и αx не больше s. В частности, слова x и αx находятся в шаре радиуса s с центром в нуле. Тогда коэффициент α представим в виде дроби, числитель и знаменатель которой лежат в указанном шаре, а таких дробей не больше, чем квадрат числа элементов шара, то есть $V_2(s,k)^2$. А всего в поле $\mathbb{F}$ имеется 2^k элементов, что и даёт указанную оценку.

Теперь можно построить каскадный *код Форни – Возенкрафта – Юстесена*, используя для внутреннего кодирования коды C_α. Конкретно это выглядит так. Как и в конструкции Форни, мы начинаем с 2^{k-1} блоков длины k, рассматривая блоки как элементы поля $\mathbb{F}$. Эти элементы мы считаем коэффициентами некоторого многочлена P степени менее 2^{k-1}. Затем мы берём значения многочлена P во всех точках $x \in \mathbb{F}$ и подвергаем их внутреннему кодированию. В данном случае к значению $P(x)$ мы применяем код C_x, то есть рассматриваем пару

$$(P(x), xP(x))$$

как слово длины $2k$. Все эти слова вместе и образуют кодовое слово длины $2k2^k$.

Этот код имеет коэффициент полезного действия 25% (удлиняет слова в четыре раза). Кодовое расстояние этого кода (делённое на длину кодового слова, то есть в расчёте на один символ) отделено от нуля, как и в конструкции Форни.

Это можно доказать примерно так: при некотором $\varepsilon > 0$ доля тех $x \in \mathbb{F}$, при которых кодовое расстояние кода C_x (в расчёте на символ) меньше ε, не превосходит 1%. А один процент блоков внешнего кода не может сильно повлиять на его кодовое расстояние.

Замечания.

1. При этой конструкции построение кода и кодирование не требуют перебора. Однако декодирование внутреннего кода (в тех блоках, где оно удаётся) по-прежнему требует перебора всех возможных кодовых слов.

2. Построенный код легко описать безо всякого упоминания о конкатенации. В самом деле, мы делаем всё как в кодах Рида – Соломона, только берём значения (во всех точках поля) не одного многочлена $P(x)$, а двух: многочлена $P(x)$ и многочлена $xP(x)$. Другими

словами, мы записываем данный нам набор элементов поля в коэффициенты многочленов дважды — начиная со свободного члена и начиная с первой степени, — и объединяем полученные значения.

13. Оценка Плоткина

Различие между границами Хэмминга и Гилберта особенно разительно в правой половине рис. 1: при $d > n/2$ граница Хэмминга не запрещает существования кодов, а граница Гилберта не гарантирует их существования. Что же происходит на самом деле? (Речь идёт о двоичных кодах, когда $\Sigma = \{0, 1\}$.)

Максимальное расстояние между двумя элементами $\mathbb{B}^n$ равно n. Точки, между которыми такое расстояние, отличаются во всех позициях, и потому ясно, что три точки с попарными расстояниями n найти нельзя. Более того, даже если ослабить требования и искать три точки с попарными расстояниями не меньше $0{,}99n$, это тоже не удастся: если B и C отличаются от A в 99% мест, то B совпадает с C по крайней мере в 98% мест.

Аналогичное утверждение, хотя и не по столь очевидным причинам, справедливо для любой доли $\beta > 1/2$.

Лемма. Пусть $\beta > 1/2$. Количество точек в $\mathbb{B}^n$, попарные расстояния между которыми не меньше βn, не превосходит

$$\frac{1}{2\beta - 1} + 1.$$

Здесь важно, что оценка на число точек не зависит от n (хотя и зависит от β: чем ближе β к $1/2$, тем слабее оценка).

Доказательство леммы. Удобно вместо последовательностей нулей и единиц рассматривать последовательности единиц и минус единиц, считая их векторами в $\mathbb{R}^n$. Определим скалярное произведение в $\mathbb{R}^n$, положив

$$(\langle x_1, \ldots, x_n \rangle, \langle y_1, \ldots, y_n \rangle) = \frac{\sum x_i y_i}{n}.$$

Разница с обычным скалярным произведением в $\mathbb{R}^n$ состоит лишь в дополнительном делении на n. Это технически удобно, поскольку при этом евклидова длина любого слова (которое, напомним, мы считаем вектором из плюс-минус единиц) равна 1.

Если расстояние Хэмминга между двумя кодовыми словами больше βn, то их скалярное произведение отрицательно и меньше $1-2\beta$. Теперь легко получить оценку на число векторов: если имеется N единичных векторов $e_1, \ldots, e_N$ в евклидовом пространстве V и при этом скалярное произведение любых двух из них меньше $1-2\beta$, то

$$0 \leqslant (e_1 + \ldots + e_N, e_1 + \ldots + e_N) \leqslant N + N(N-1)(1-2\beta),$$

(имеется N квадратов и $N(N-1)$ попарных произведений различных векторов), откуда

$$N - 1 \leqslant \frac{1}{2\beta - 1},$$

что и требовалось в лемме.

Таком образом, если мы хотим, чтобы количество передаваемой информации росло с ростом n, на исправление более чем 25% ошибок надеяться не приходится. В частности, в правой половине рис. 1 никаких интересных кодов нет, хотя оценка Хэмминга их и не запрещает.

Если требовать, чтобы доля ошибок была бы хоть чуть-чуть больше 50%, то все углы между кодовыми словами будут тупыми и число точек не больше $n+1$, как показывает следующее простое утверждение:

Если несколько векторов евклидова пространства образуют попарно тупые углы, то после удаления любого из векторов получается линейно независимая система.

В самом деле, предположим, что вектора $x_0, \ldots, x_k$ образуют попарно тупые углы. Покажем, что $x_1, \ldots, x_k$ линейно независимы. Пусть это не так и

$$\lambda_1 x_1 + \ldots + \lambda_k x_k = 0$$

Вычеркнем все нулевые коэффициенты. Если оставшиеся коэффициенты имеют один знак, то умножим скалярно на x_0 и получим противоречие: сумма нескольких отрицательных скалярных произведений с коэффициентами одного знака равна нулю.

Если оставшиеся коэффициенты имеют разные знаки, то разнесём их в разные части и получим равенство

$$\lambda_i x_i + \ldots = \lambda_j x_j + \ldots,$$

где все коэффициенты положительны. Если обе части равны нулю, то смотри выше. Если же нет, то скалярное произведение левой части на правую одновременно положительно (как скалярный квадрат вектора) и отрицательно (как сумма отрицательных скалярных произведений с положительными коэффициентами). Лемма доказана.

Если разрешить и прямые углы (наряду с тупыми) то максимальное число векторов в n-мерном евклидовом пространстве возрастёт до $2n$. В самом деле, можно взять плюс-минус вектора базиса, их как раз $2n$. Больше векторов взять не удастся:

Если векторы в n-мерном евклидовом пространстве образуют попарно прямые или тупые углы, то их число не превосходит $2n$.

В самом деле, можно считать, что линейная оболочка векторов совпадает со всем пространством (иначе можно свести дело к меньшей размерности, рассуждая по индукции) и векторы $x_1, \ldots, x_n$ образуют базис. Выразим остальные x_k (при $k > n$) через $x_1, \ldots, x_n$.

Шаг 1. Заметим, что в таких выражениях все ненулевые коэффициенты (а в каждом из выражений они есть, поскольку $x_k \neq 0$) отрицательны. В самом деле, предположим, что это не так и в каком-то из выражений

$$x_k = \lambda_1 x_1 + \ldots + \lambda_n x_n.$$

есть положительные λ_i. Перенесём все отрицательные коэффициенты в левую часть, получим равенство

$$x_k + (-\lambda_s)x_s + \ldots = \lambda_t x_t + \ldots,$$

где все коэффициенты при x_i положительны и в правой части что-то осталось (ведь были положительные коэффициенты). Тогда правая часть этого равенства (а значит, и левая) отлична от нуля в силу линейной независимости векторов базиса. Произведение левой части на правую должно быть одновременно положительно (как скалярный квадрат ненулевого вектора) и неположительно (поскольку после раскрытия скобок становится суммой неположительных скалярных произведений с положительными коэффициентами). Противоречие.

Шаг 2. Если общее число векторов больше $2n$, то вектора x_k при $k > n$ (их не меньше $n + 1$ штук в n-мерном пространстве) линейно зависимы. Коэффициенты их нулевой линейной комбинации не могут быть одного знака, ведь в выражениях через базисные вектора

$x_1, \ldots, x_n$ коэффициенты отрицательны и сократиться друг с другом не могут. Значит, есть разные знаки и можно снова разнести их по разным частям:

$$\mu_s x_s + \ldots = \mu_t x_t + \ldots,$$

где все коэффициенты μ_i положительны. Обе части содержат хотя бы один ненулевой член и потому не равны нулю (как мы уже знаем про линейные комбинации с коэффициентами одного знака). Поэтому скалярное произведение левой части и правой одновременно положительно (как скалярный квадрат ненулевого вектора) и неположительно (как сумма неположительных скалярных произведений с положительными коэффициентами). Противоречие.

Таким образом, мы приходим к *оценке Плоткина*: число кодовых слов длины n с попарными расстояниями не меньше $n/2$ не превосходит $2n$.

14. Улучшение оценки Синглтона

Имея оценку Плоткина, вернёмся к рассуждению, использованному для доказательства оценки Синглтона, и получим более сильную оценку. Будем рассматривать случай кодов в двухбуквенном алфавите.

Пусть код имеет длину кодовых слов n, длину кодируемого слова k и расстояние d (как говорят, $[n, k, d]_2$-код; индекс 2 — размер алфавита). Принцип Дирихле показывает, что для любого t от 1 до $k-1$ среди 2^k кодовых слов можно указать 2^{k-t} кодовых слов, которые совпадают в первых t битах. Они образуют код с параметрами $[n-t, k-t, d]_2$ (поскольку первые t битов совпадают, d различий приходятся на остальные $n-t$ битов).

Доказывая оценку Синглтона, мы брали $t = k-1$ и замечали, что для полученного кода (в котором не менее двух кодовых слов) расстояние не больше длины кодового слова:

$$n - (k-1) \geqslant d$$

Теперь мы можем взять меньшее t, при котором у полученного кода длина кодового слова будет равна удвоенному расстоянию:

$$n - t = 2d, \quad \text{или} \quad t = n - 2d.$$

Согласно оценке Плоткина, число кодовых слов не превосходит удвоенной длины:

$$2^{k-t} \leqslant 4d, \quad \text{или} \quad k - n + 2d \leqslant \log 4d.$$

Замечая, что $d \leqslant n$ и деля на n получаем оценку

$$\frac{k}{n} + 2\frac{d}{n} \leqslant 1 + 4\log n/n.$$

При бо́льших n последним слагаемым можно пренебречь и на рис. 1 появляется новая граница (пунктир на рис. 2).

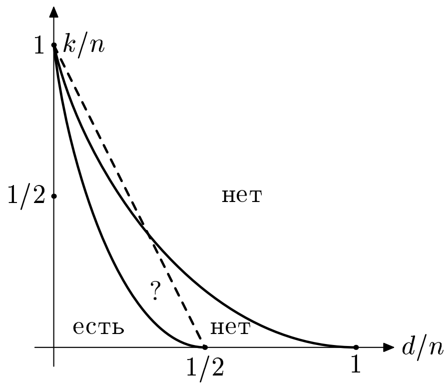


Рис. 2. Улучшение границы Синглтона.

Видно, что она кое-где сильнее, а кое-где слабее границы Хэмминга.

15. Код Адамара

Оказывается, что оценка Плоткина является точной, если n есть степень двойки. Соответствующий пример доставляют коды Адамара.

Код Адамара состоит из $2n$ кодовых слов длины n, любые два из которых отличаются либо ровно в половине позиций (так что соответствующие векторы ортогональны), либо во всех позициях.

Код Адамара строится при $n = 2^s$. Для этого позиции в слове будем представлять себе как вершины s-мерного булева куба $\mathbb{B}^s$, а слова — как функции $\mathbb{B}^s \to \mathbb{B}$. Такая функция задаётся своими значениями в $2^s = n$ вершинах, и эти значения образуют слово длины n, надо только фиксировать какой-либо порядок на вершинах куба.

В качестве кодовых слов возьмём аффинные функции, то есть функции вида

$$\langle x_1, \ldots, x_s \rangle \mapsto a_0 + a_1 x_1 + \ldots + a_s x_s$$

при $a_0, \ldots, a_s \in \mathbb{B}$ (умножение и сложение — как в поле из двух элементов). Такая функция задаётся набором своих коэффициентов, и потому имеется $2^{s+1} = 2n$ аффинных функций.

Две такие функции либо различаются во всех точках (если отличаются лишь коэффициентом a_0), либо ровно в половине точек (образующих аффинное подпространство коразмерности 1 над полем $\mathbb{B}$). Код Адамара построен.

С точки зрения структуры евклидова пространства $\mathbb{R}^n$, о котором мы говорили в предыдущем разделе, кодовые слова кода Адамара являются плюс-минус базисными векторами ортонормированного базиса (замена знака соответствует добавлению единицы к аффинной функции, то есть изменению a_0).

Итак, к настоящему времени у нас есть три явные конструкции кодов для двухбуквенного алфавита: код Хэмминга, код Форни – Юстесена – Возенкрафта и код Адамара. Код Хэмминга имеет коэффициент полезного действия, близкий к 100%, но исправляет только одну ошибку. Напротив, код Адамара исправляет почти что максимально возможное число ошибок (до 25%), но зато имеет очень малый коэффициент полезного действия. Код Форни занимает промежуточное положение: у него и доля исправляемых ошибок, и коэффициент полезного действия отделены от нуля, хотя и невелики.

Для бо́льших алфавитов ситуация лучше: там код Рида – Соломона при подходящем выборе параметров позволяет достичь любого заданного коэффициента полезного действия и исправляет при этом максимально возможное (по неравенству Синглтона) число ошибок.

Замечание. Если мы хотим явно указать код, приближающийся к границе Плоткина (50%), то можно использовать каскадный код, в котором внутренним кодом является код Адамара, а внешним — код Рида – Соломона.

Рассмотрим код Адамара с 2^{m+1} кодовыми словами размера 2^m. Используя его как внутренний код, мы получаем для внешнего кода алфавит из 2^{m+1} символов. Для простоты записи формул пожертвуем половиной и будем использовать лишь 2^m из них. Зафиксируем некоторое $\varepsilon > 0$. Внешний код Рида – Соломона будет кодировать $\varepsilon 2^m$ таких символов (блоков из m битов) с помощью 2^m символов (блоков), каждый из которых при внутреннем кодировании изображается 2^m битами.

В итоге параметры каскадного кода таковы: длина входного слова $\varepsilon m 2^m$; длина выходного слова $(2^m)^2$; кодовое расстояние (в расчёте на символ) не меньше произведения кодовых расстояний, то есть $\frac{1}{2}(1-\varepsilon)$.

Таким образом, мы можем достичь кодового расстояния, сколь угодно близкого к 50% (при малом фиксированном ε). При этом длина кодового слова ограничивается примерно квадратом длины кодируемого слова.

(Такая граница представляет интерес лишь потому, что наш код описан явно — граница Варшамова – Гилберта позволяет получить линейные коды с лучшими параметрами, которые можно использовать либо сами по себе, либо как внутренний код для кода Рида – Соломона, как в конструкции Форни.)

16. Вероятностное декодирование кодов Адамара

Пусть фиксировано некоторое значение $\alpha < 1/4$. Тогда, как мы знаем, возможно однозначное декодирование кода Адамара с исправлением до α ошибок (на символ).

Можно ли это делать за полиномиальное время? Постановка этого вопроса требует уточнения: полиномиальное время от чего? От длины кодируемого слова или длины кодового слова? Поскольку мы кодируем $m+1$ битов с помощью 2^m битов, разница весьма существенна.

Тривиальный ответ таков: за полиномиальное от 2^m время мы можем перебрать все 2^{m+1} возможных кодируемых слов и найти нужное, а за полиномиальное от m время мы сможем прочесть лишь пренебрежимо малую часть кодового слова, и вся эта часть может попасть в зону ошибок, так что декодирование невозможно.

(Более аккуратное рассуждение: проведём декодирование для какого-то одного случая, скажем, нулевого слова, потом посмотрим, какие позиции в кодовом слове мы прочли, и во всех кодовых словах эти позиции сделаем нулевыми; доля ошибок будет невелика, а декодирование разрушено.)

Другое возможное уточнение: говоря о декодировании слова длины 2^m за полиномиальное от m время, будем предполагать, что это слово задано нам «как оракул»: по номеру бита (этот номер имеет длину m) нам сообщают значение соответствующего бита. (Это уточнение необходимо, поскольку, скажем, давать на вход слово длины 2^m на ленте бессмысленно, поскольку будет доступно лишь его начало полиномиальной длины.)

Оказывается, что в данной постановке задачи декодирование за полиномиальное время может быть выполнено вероятностным алгоритмом со сколь угодно малой вероятностью ошибки. Точная формулировка:

Теорема. Для всякого $\alpha < 1/4$ и для всякого $\varepsilon > 0$ существует вероятностный алгоритм, который, получив число m и (в виде оракула) кодовое слово длины 2^m с не более чем $\alpha 2^m$ ошибок, работает полиномиальное от m время и правильно декодирует слово с вероятностью не менее $1-\varepsilon$.

(Заметим, что полином, ограничивающий время работы, зависит от α и от ε: чем ближе α к $1/4$ и чем меньше ε, тем дольше работает алгоритм.)

Доказательство. По условию мы можем найти значение неизвестной аффинной функции $f\colon \mathbb{B}^n \to \mathbb{B}$, имеющей вид

$$f(x_1,\ldots,x_n) = f_0 + f_1x_1 + f_2x_2 + \ldots + f_nx_n$$

в случайной точке с вероятностью ошибки не больше α. Коэффициент f_1 можно записать как разность значений в двух точках, отличающихся по первой координате:

$$f_1 = f(x_1+1, x_2,\ldots,x_n) - f(x_1,\ldots,x_n).$$

Если в качестве $(x_1,\ldots,x_n)$ взять случайную равномерно распределённую точку, то сдвинутая точка $(x_1+1,\ldots,x_n)$ также будет равномерно распределена (хотя, конечно, эти две точки не будут независимы). Каждое из двух значений f нам известно с вероятностью ошибки не больше 2α, поэтому правильное значение для f_1 будет получено с вероятностью не меньше $1-2\alpha > 50\%$.

Повторяя это несколько раз (для независимых точек) и затем определяя ответ большинством, можно (как известно из теории вероятности) быстро уменьшать вероятность ошибки и тем самым найти коэффициент f_1 со сколь угодно малой вероятностью ошибки. Аналогично найдём и все остальные f_i, после чего уже можно находить f_0 (тоже голосованием по нескольким точкам).

Теорема доказана.

Замечание. Как известно из теории вероятностей, при таком повторении (и голосовании) вероятность ошибки убывает быстро, поэтому можно сделать её, скажем, меньше 2^{-m}, оставив алгоритм полиномиальным по m.

17. Коды Рида – Маллера

Коды Рида – Соломона и коды Адамара являются двумя крайними случаями в семействе кодов, называемых *кодами Рида – Маллера*: в кодах Рида – Соломона мы рассматривали многочлены от одной переменной, но большой степени, а в кодах Адамара – от многих переменных, но первой степени.

В общем случае мы рассматриваем многочлены от k переменных степени не выше d над полем $\mathbb{F}_q$. При этом степень понимается как суммарная степень по всем переменным. (При $k=1$ получаются коды Рида – Соломона, при $d=1$ — коды Адамара.) Алфавит состоит из q символов (элементов поля); коэффициенты многочлена степени не выше d образуют кодируемое слово, а значения этого многочлена во всех точках $\mathbb{F}^k$ — кодовое слово.

Длина кодового слова равна q^k, а длина кодируемого слова равна числу решений неравенства

$$m_1 + \ldots + m_k \leqslant d$$

в неотрицательных целых числах, то есть C^k_{d+k} (решение такого неравенства задаётся разбиением ряда из d объектов на $k+1$ групп с помощью k перегородок; числа $m_1, \ldots, m_k$ — числа объектов между перегородками; объекты справа от последней перегородки не входят ни в одно m_i и составляют разницу между частями неравенства.)

Оценка кодового расстояния основана на следующем простом алгебраическом утверждении:

Лемма. Пусть $\mathbb{F}_q$ — поле из q элементов, а $P(x_1, \ldots, x_k)$ многочлен от k переменных над этим полем, причём его степень (суммарная по всем переменным) равна d. Тогда доля точек $(x_1, \ldots, x_k) \in \mathbb{F}^k$, для которых

$$P(x_1, \ldots, x_k) = 0,$$

не превосходит d/q.

Доказательство. Лемма очевидна, если многочлен содержит член степени d, в который входит только одна переменная. Тогда при любых значениях остальных переменных получается ненулевой многочлен степени d, имеющий не более d корней, так что на каждой прямой (когда все остальные переменные фиксированы) доля нулей не больше d/q, и остаётся их усреднить.

К этому случаю можно пытаться сводить произвольный многочлен линейной невырожденной заменой переменных (которая не меняет суммарную степень); проще, однако, доказать утверждение леммы индукцией по числу переменных. Для $k = 1$ утверждение леммы общеизвестно.

Для произвольного k рассуждаем так. Выделим какую-нибудь переменную, например, x_1, и рассмотрим моном с максимальной степенью по x_1. Пусть эта степень равна s; очевидно, $s \leqslant d$. Запишем P как многочлен по x_1, коэффициенты которого суть многочлены от $x_2, \ldots, x_k$. Коэффициент при x_1^s представляет собой некоторый многочлен $Q(x_2, \ldots, x_k)$ степени не больше $d - s$. Случай $d - s = 0$ мы уже обсуждали. В общем случае для равномерно распределённых случайных $x_1, \ldots, x_k$ вероятность события

$$Q(x_2, \ldots, x_k) = 0$$

по предположению индукции не превосходит $(d-s)/q$, а вероятность события

$$P(x_1, \ldots, x_k) = 0, \quad \text{но} \quad Q(x_2, \ldots, x_k) \neq 0$$

не превосходит s/q, поскольку для кажого набора значений $x_2, \ldots, x_k$, при которых $Q(x_2, \ldots, x_k) \neq 0$, существует не более s значений x_1, при которых $P(x_1, \ldots, x_k) = 0$. (Так что даже и условная вероятность

$$P(x_1, \ldots, x_k) = 0 \quad \text{при условии} \quad Q(x_2, \ldots, x_k) \neq 0$$

не превосходит s/q.) Складывая вероятности, получаем искомую оценку d/q. Лемма доказана.

Таким образом, кодовое расстояние не меньше $q^k(1 - d/q)$, то есть $(1-d/q)$ в расчёте на один символ. Например, при $d = 1$ и $q = 2$ получается расстояние 50%, как мы уже видели для кодов Адамара.

Для примера рассмотрим случай $k = 2$. Кодовое слово состоит из q^2 символов, а кодируемое слово из $C^2_{d+2} = (d+1)(d+2)/2 \approx d^2/2$ символов. Таким образом, коэффициент полезного действия кода примерно равен $(1/2)d^2/q^2$, что значительно хуже, чем у кодов Рида–Соломона при том же удельном кодовом расстоянии $1-d/q$. Зато размер алфавита теперь уже равен корню из длины кодового слова (а не самой этой длине).

18. Коды БЧХ

Эти коды названы БЧХ (в английском варианте — BCH) по именам своих изобретателей: Боуза (Bose), Чоудхури (Ray-Chaudhuri) и Хоквингема (Hocquenghem). Они позволяют исправлять любое фиксированное число ошибок над двухбуквенным алфавитом и примыкают к коду Хэмминга (который является их частным случаем).

Рассмотрим поле $\mathbb{F}$ из $n = 2^k$ элементов и многочлены степени меньше n с коэффициентами в этом поле. Такой многочлен

$$A(x) = a_0 + a_1x + \ldots + a_{n-1}x^{n-1}$$

определяется n коэффициентами $a_0, \ldots, a_{n-1}$. С другой стороны, он однозначно определяется своими значениями в n точках поля $\mathbb{F}$, которые можно выбрать произвольно, так что соответствие

$$\text{коэффициенты} \leftrightarrow \text{значения}$$

является изоморфизмом n-мерных пространств над полем $\mathbb{F}$. (Этот изоморфизм является дискретным вариантом преобразования Фурье.)

Наложим теперь на многочлен A некоторые ограничения. Во-первых, потребуем, чтобы его степень была меньше $n - s$ (для некоторого фиксированного s). Другими словами, потребуем, чтобы старшие s коэффициентов равнялись нулю. (Таблицы значений таких многочленов образуют, как мы помним, код Рида–Соломона с расстоянием $s + 1$.)

Во-вторых, потребуем ещё, чтобы значения многочлена A во всех точках поля равнялись нулю или единице (поля $\mathbb{F}$). В этом случае таблица значений будет двоичным словом длины n.

Все такие таблицы и являются кодовыми словами кода БЧХ.

Переформулировка: слово из n нулей и единиц является кодовым словом кода БЧХ, если соответствующий ему многочлен (принимающий указанные значения в n точках поля $\mathbb{F}$ имеет степень меньше $n - s$. (Здесь n и s — параметры кода.)

Кодовое расстояние этого кода не меньше $s + 1$ (поскольку он является частью кода Рида – Соломона с таким кодовым расстоянием). Сложнее понять, сколько при этом образуется кодовых слов. Хотя оба ограничения на многочлен A просты, но одно из них формулируется в терминах его коэффициентов, а другое — в терминах значений, и не так просто понять, как они взаимодействуют друг с другом.

Для этого нам понадобятся некоторые сведения из алгебры. В поле $\mathbb{F}$ выделим подполе $\mathbb{F}_2 \subset \mathbb{F}$, состоящее из нуля и единицы. [Чтобы убедиться, что это действительно подполе, достаточно проверить, что $1 + 1 = 0$ в $\mathbb{F}$, то есть что характеристика поля $\mathbb{F}$ равна 2. Это доказывается так: если характеристика равна $p > 2$, то в $\mathbb{F}$ есть подполе из p элементов, и $\mathbb{F}$ есть векторное пространство над этим подполем, так что размер $\mathbb{F}$ есть степень числа p.]

Как мы уже говорили, многочлены степени меньше n образуют векторное пространство над $\mathbb{F}$; условие обращения s старших коэффициентов в нуль задаёт подпространство этого пространства. Но второе наше условие — что значения во всех точках нули и единицы — не является линейным над $\mathbb{F}$. Однако оно является линейным над $\mathbb{F}_2$ (по тривиальным причинам). Поэтому построенное нами множество кодовых слов кода БЧХ (обозначим его C) является векторным пространством над полем $\mathbb{F}_2$, и надо только определить его $\mathbb{F}_2$-размерность.

Если бы не требование обращения в нуль s старших коэффициентов, то $\mathbb{F}_2$-размерность C была бы равна n. Обращение в нуль каждого из s коэффициентов задаётся k уравнениями: коэффициент принадлежит полю $\mathbb{F}$, которое имеет размерность k над $\mathbb{F}_2$.

Следствие: $\mathbb{F}_2$-размерность C не меньше $n - sk$, то есть $n - s \log n$.

Это уже кое-что: мы получаем код длины n с $s \log n$ проверочными символами (и $n - s \log n$ значащими) и расстоянием $s + 1$. Например, при $s = 2$ получается код с расстоянием 3 (как и у кода Хэмминга) и $2\log n$ проверочными символами (что вдвое больше, чем у Хэмминга).

Однако на самом деле ситуация оказывается ещё немного лучше: ks условий, соответствующих обращению в нуль старших s коэффициентов многочлена A, линейно зависимы над полем $\mathbb{F}_2$. Чтобы понять это, нам понадобится вспомнить ещё кое-что из алгебры.

Для начала заметим, что условие «все значения многочлена A — нули или единицы» можно переформулировать так: $A^2(x) = A(x)$ для всех $x \in \mathbb{F}$. Другими словами, многочлен $A^2 - A$ должен обращаться в нуль во всех точках поля $\mathbb{F}$. Это не значит, что все его коэффициенты нулевые (ведь степень многочлена $A^2 - A$ может быть почти в два раза больше числа элементов в поле), а означает лишь, что A делится на многочлен

$$P(x) = \prod_{\alpha \in \mathbb{F}} (x - \alpha)$$

Последний многочлен равен $x^n - x$. Действительно, по теореме Лагранжа порядок каждого элемента в мультипликативной группе поля делит порядок группы, равный $n - 1$ (на самом деле верно более сильное свойство: мультипликативная группа поля обязана быть циклической; но это сейчас не важно). Поэтому $x^{n-1} = 1$ для любого ненулевого элемента поля, и $x^n = x$ для любого элемента поля. Значит, многочлен $x^n - x$ обращается в нуль во всех точках поля, поэтому он делится на P; остаётся заметить, что у этих двух многочленов совпадают степени и старшие коэффициенты.

Итак, условие «все значения A — нули или единицы» можно переформулировать так: $A^2 - A$ делится на многочлен $x^n - x$ без остатка. Как записать это условие в терминах коэффициентов многочлена A?

Заметим, что делить с остатком на $x^n - x$ очень просто. Надо понизить степени по правилам

$$\begin{array}{rcl} x^n & \to & x \\ x^{n+1} & \to & x^2 \\ x^{n+2} & \to & x^3 \\ & \ldots & \\ x^{2n-2} & \to & x^{n-1} \end{array}$$

(Дальше должна идти строка $x^{2n-1} \to x^n \to x$, но до таких степеней в нашем случае дело не дойдёт.) Затем надо привести подобные члены. (Эта процедура соответствует обычному делению многочленов «уголком».)

Сделаем это для многочлена $A^2 - A$, если

$$A(x) = a_{n-1}x^{n-1} + a_{n-2}x^{n-2} + \ldots + a_1 x + a_0.$$

Вычислить A^2 легко, если вспомнить, что в поле характеристики 2, где $1 + 1 = 0$, имеет место равенство $(u+v)^2 = u^2 + v^2$, поскольку член с $2uv$ обращается в нуль. Поэтому квадрат суммы есть сумма квадратов, и

$$A^2(x) = a_{n-1}x^{2n-2} + a_{n-2}x^{2n-4} + \ldots + a_1 x^2 + a_0.$$

Делимость $A^2 - A$ на $x^n - x$ означает, что после замен одночленов в соответствии с приведённой выше таблицей из A^2 должно получиться A, то есть

$$\begin{aligned} a_{n-1}^2 &= a_{n-1} \\ a_{n-2}^2 &= a_{n-3} \\ a_{n-3}^2 &= a_{n-5} \\ &\ldots \\ a_{n/2}^2 &= a_1 \end{aligned}$$

(последнюю строку можно получить, заметив, что n чётно и что индекс в левой части есть полусумма $n-1$ и индекса в правой части). Заметим, что несмотря на свой квадратичный вид, каждое из этих условий является $\mathbb{F}_2$-линейным (поскольку возведение в квадрат, как мы уже говорили, линейно над полем характеристики 2).

Итак, мы записали условие «все значения многочлена A — нули или единицы» в виде $\mathbb{F}_2$-линейных уравнений на его коэффициенты. Как мы знаем, пространство решений этой системы линейных уравнений имеет размерность n. А хотим мы выяснить, напомним, насколько уменьшится размерность, если наложить дополнительные ограничения $a_{n-1} = 0, \ldots, a_{n-s} = 0$.

Про a_{n-1} мы и так уже знаем, что $a_{n-1}^2 = a_{n-1}$, то есть что a_{n-1} равно нулю или единице. Поэтому условие $a_{n-1} = 0$ уменьшает размерность пространства решений на 1 (а не на k, как было в нашей прежней оценке). Дальнейшие уравнения системы выражают a_{n-3}, a_{n-5},..., a_1 через a_i с большими индексами. Поэтому условия обращения в нуль достаточно записывать лишь для a_{n-2}, a_{n-4}, ... вплоть до a_{n-s} (для чётного s) и a_{n-s+1} (для нечётного s).

Ограничимся для простоты случаем чётного s (при этом кодовое расстояние будет не меньше $s+1$, и можно исправлять $s/2$ ошибок). Тогда получится $s/2$ условий коразмерности k плюс ещё одно условие коразмерности 1 (на a_{n-1}). Отсюда получаем, что *$\mathbb{F}_2$-размерность пространства кодов БЧХ длины n с расстоянием не меньше $s+1$ для чётного s равна $n-1-ks/2 = n-1-(s/2)\log n$.*

Словами: на каждую исправляемую ошибку нужно $\log n$ контрольных битов, плюс ещё один контрольный бит на всех.

Интересно сравнить параметры БЧХ-кодов с границей Хэмминга. Шар радиуса $e = s/2$ содержит примерно $C_n^e \approx n^e/e!$ элементов (мы считаем, что $e \ll n$ и потому заменяем $n(n-1)\ldots(n-e+1)$ на n^e). Граница Хэмминга говорит, что логарифм числа кодовых слов не превосходит примерно

$$n - \log(n^e/e!) = n - e\log n + \log e!;$$

первые два слагаемых как раз соответствуют БЧХ-кодам, так что разница лишь на $\log e!$.

19. БЧХ и Хэмминг

Покажем, что код Хэмминга может быть получен как частный случай кода БЧХ.

Пусть, например, $k = 3$, $n = 8$ и мы рассматриваем поле $\mathbb{F}_8$. Многочлен A однозначно определяется своими значениями в 8 точках этого поля, а также 8 коэффициентами $a_7, a_6, \ldots, a_0$. Условие «A принимает лишь значения 0 и 1» запишется как система уравнений

$$a_7^2 = a_7,\ a_6^2 = a_5,\ a_5^2 = a_3,\ a_4^2 = a_1.$$

Наложив ограничение $a_7 = 0$, мы получим код размерности 7 (одно уравнение на 8 переменных) с расстоянием 2. Легко понять, что это за код: поскольку расстояние равно 2, то в уравнение обязаны входить все переменные, и это уравнение есть проверка на чётность (сумма по модулю 2 равна нулю).

Следующее ограничение $a_6 = 0$. Оно соответствует трём уравнениям и уменьшает размерность кода до 4. Из уравнения $a_6^2 = a_5$ видно, что в этом случае и $a_5 = 0$, так что кодовое расстояние будет не меньше 4. Выбросив любой бит, мы не изменим размерность кода (поскольку по условию чётности он определялся остальными) и уменьшим расстояние не более чем на 1. Получится код длины 7 и размерности 4 с кодовым расстоянием не меньше 3 — всё как у кода Хэмминга.

Чтобы убедиться, что это и есть код Хэмминга, надо выяснить, как от значений многочлена A в точках поля перейти к его коэффициентам. Это делается по интерполяционной формуле Лагранжа:

$$A(x) = \sum_{\alpha\in\mathbb{F}} A(\alpha)P_\alpha(x),$$

где P_α — многочлен (степени меньше n), равный 1 в точке α и 0 в остальных точках поля. В данном случае в качестве такого многочлена можно взять

$$P_\alpha(x) = (x-\alpha)^{n-1} + 1.$$

В самом деле, при $x \neq \alpha$ первое слагаемое равно 1 (порядок мультипликативной группы поля, равный $n-1$, делится на порядок любого её элемента), и многочлен обращается в нуль (наше поле характеристики 2).

Разложение $(x-\alpha)^{n-1}$ по биному Ньютона даёт (в поле характеристики 2) многочлен

$$x^{n-1} + x^{n-2}\alpha + x^{n-3}\alpha^2 + \ldots + x\alpha^{n-2} + \alpha^{n-1}.$$

В этом можно убедиться, вспомнив, что в строке треугольника Паскаля, номер которой на единицу меньше степени двойки, все члены нечётные. А можно домножить записанную сумму на $(x-\alpha)$, получится $x^n - \alpha^n$, что равно $(x-\alpha)^n$ (поскольку возведение в квадрат, а также в любую степень двойки, перестановочно со сложением).

Так или иначе, мы можем теперь записать более конкретно условия обращения в нуль коэффициентов a_7 и a_6 в терминах значений многочлена A. Как видно из формулы Лагранжа и формулы для P_α,

$$a_7 = \sum_{\alpha\in\mathbb{F}} A(\alpha), \quad a_6 = \sum_{\alpha\in\mathbb{F}} \alpha A(\alpha).$$

Поэтому условие на a_7 представляет собой проверку на чётность (это мы и так знаем из общих соображений). Условие на a_6 представляет собой равенство

$$\sum_{\alpha} \alpha A(\alpha) = 0$$

в поле $\mathbb{F}$, которое можно рассматривать как векторное пространство размерности 3 над $\mathbb{F}_2$. При этом семь ненулевых элементов α превращаются в семь ненулевых векторов из $\mathbb{F}_2^3$, то есть в векторы

$$(0,0,1),(0,1,0),(0,1,1),(1,0,0),(1,0,1),(1,1,0),(1,1,1),$$

и условия на коэффициенты $A(\alpha)$ при $\alpha \neq 0$ будут те самые, что были в коде Хэмминга. Поэтому из кода БЧХ получается код Хэмминга, если отбросить $A(0)$, как мы и говорили.

В этом примере мы считали, что $k = 3$ ($n = 8$), но всё сказанное очевидно переносится и на случай произвольного k.

20. Декодирование списком

До сих пор мы требовали, чтобы исправление ошибок происходило однозначно. Посмотрим, что будет, если ослабить это требование и разрешить до L вариантов декодирования (при некотором L). Более формально, множество кодовых слов должно быть таким, чтобы любой шар радиуса e содержал не более L кодовых слов. В этом случае мы говорим, что код *допускает декодирование списком длины* L (с исправлением e ошибок).

Как и раньше, нас интересует возможность построения такого множества при заданных параметрах q (размер алфавита), k (длина кодируемого слова), n (длина кодовых слов), e (число возможных ошибок) и, наконец, L. (Случай $L = 1$ соответствует прежней постановке задачи.)

Аналог границы Хэмминга теперь выглядит так:

$$q^k V_q(e,n) \leqslant L q^n$$

В самом деле, шары радиуса e в количестве q^k штук содержат по $V_q(e,n)$ элементов каждый и покрывают множество из q^n элементов не более чем в L слоёв.

В логарифмической шкале:

$$\frac{k}{n}+\frac{\log_q V_q(e,n)}{n}\leqslant 1+\frac{\log_q L}{n}$$

Таким образом, если рассматривать не слишком большие L (скажем, полиномиальные от n), то граница почти не сдвигается.

Зато сдвигается другая граница: указанное только что необходимое условие существование кода оказывается почти что достаточным. А именно, верно такое утверждение (теорема Элайеса):

Теорема. Пусть выполнено неравенство Хэмминга (в прежней форме, для $L=1$):

$$q^k V_q(e,n)\leqslant q^n$$

Тогда существует код, допускающий декодирование списком длины L, где $L=O(n\log q)$.

Точнее говоря, нам годится любое L, при котором $L! > q^n$; поскольку $L!\approx (L/2{,}718\ldots)^L$, это даёт оценку $L=O(n\log q)$.

Доказательство. Покажем, что случайно выбранный код допускает декодирование L-списком с положительной вероятностью (или, другими словами, что число кодов, не допускающих такого декодирования, меньше общего числа кодов).

Всего кодов (точнее, отображений $\Sigma^k\to\Sigma^n$) имеется $(q^n)^{(q^k)}$. Подсчитаем число плохих — тех, где некоторое слово покрывается L (или более) шарами. Плохое отображение можно задать, указав

• плохое слово (покрытое L или более шарами) [q^n вариантов];

• индексы покрывающих шаров [подмножество размера L в множестве Σ^k, всего $C^L_{q^k}$ вариантов];

• центры покрывающих шаров (=кодовые слова, попадающие в шар радиуса e с центром в точке, выбранной на первом шаге) [L точек в шаре, всего $V_q(e,n)^L$ вариантов];

• остальные кодовые слова [q^k-L точек, всего $(q^n)^{q^k-L}$ вариантов].

Таким образом, общее число плохих отображений не превосходит

$$q^n C^L_{q^k}(V_q(e,n))^L(q^n)^{q^k-L}\leqslant q^n\frac{(q^k)^L}{L!}(V_q(e,n))^L(q^n)^{q^k-L}$$

(на последнем шаге мы использовали оценку $C^u_v\leqslant (v^u)/u!$, которая получается из формулы для числа сочетаний, если все множители в числителе заменить на наибольший). Таким образом, всё будет хорошо, если

$$q^n \frac{(q^k)^L}{L!}(V_q(e,n))^L (q^n)^{q^k-L} < (q^n)^{q^k},$$

что после сокращений даёт

$$\frac{1}{L!}(q^k V_q(e,n))^L < (q^n)^{L-1}.$$

По предположению теоремы $q^k V_q(e,n) \leqslant q^n$ (граница Хэмминга), поэтому достаточно неравенства $q^n < L!$, как мы и говорили. Теорема доказана.

Замечание. Если немного уменьшить пропускную способность конструируемого кода, предположив, что

$$\frac{k}{n} + \frac{\log_q V_q(e,n)}{n} \leqslant c < 1$$

то в левой части оценок добавится множитель $((q^{c-1})^n)^L$, меньший единицы, поскольку тогда

$$q^k V_q(e,n) \leqslant q^{cn} = q^n (q^{c-1})^n,$$

и останется неравенство

$$q^n((q^{c-1})^n)^L < L!$$

которое выполняется при $L = O(1)$ (величина L зависит от выбора c и q, но не растёт с ростом n). В самом деле, достаточно взять L, при котором

$$q \cdot (q^{c-1})^L \leqslant 1.$$

Мы доказали существование кодов, декодируемых списком, вблизи границы Хэмминга, но не указали явно этих кодов. Сложность описания построенных кодов экспоненциальна, не говоря уже о времени кодирования и декодирования.

21. Кодовое расстояние и декодирование списком

Пусть имеется код $F\colon \Sigma^k \to \Sigma^n$ с минимальным расстоянием d между кодовыми словами. Мы уже знаем, что он позволяет декодировать e ошибок при $e < d/2$. Переходя от различающихся позиций к совпадающим, можно переформулировать это так. Пусть известно, что любые два кодовых слова совпадают максимум в u ($=n-d$) позициях. Тогда для любого слова $y \in \Sigma^n$ не более одного кодового слова может совпасть с y более чем в $(n+u)/2$ ($=n-d/2$) позициях.

В этих же обозначениях удобно формулировать утверждение о декодировании списком, заменив среднее арифметическое на среднее геометрическое:

Теорема. Пусть любые два кодовых слова (из Σ^n) совпадают максимум в u позициях. Тогда для любого слова $y \in \Sigma^n$ не более чем $(n+1)$ слов могут совпадать с ним более чем $\sqrt{nu}$ позициях.

(Заметим, что среднее геометрическое больше среднего арифметического, так что всё согласовано.)

Доказательство. Пусть слова $y_1, \ldots, y_N$ совпадают со словом y более чем в $\sqrt{nu}$ позициях (каждое в своих). Пусть $Y_i \subset \{1, 2, \ldots, n\}$ — номера позиций, где y_i совпадает с y. Тогда каждое из множеств Y_i содержит более $\sqrt{nu}$ элементов, а пересечение $Y_i \cap Y_j$ при $i \neq j$ содержит не более u элементов.

Рассмотрим характеристические функции множеств Y_i как случайные величины на вероятностном пространстве $\{1, \ldots, n\}$ (с равномерным распределением). Напомним, что корреляция двух случайных величин ξ и η определяется как

$$\langle \xi, \eta \rangle = \mathsf{E}[(\xi - \mathsf{E}\xi)(\eta - \mathsf{E}\eta)],$$

где E — математическое ожидание (в данном случае — среднее арифметическое n чисел). Другими словами, корреляция ξ и η есть скалярное произведение $\xi - \mathsf{E}\xi$ и $\eta - \mathsf{E}\eta$.

Раскрывая скобки и пользуясь линейностью математического ожидания, получаем, что

$$\langle \xi, \eta \rangle = \mathsf{E}(\xi\eta) - (\mathsf{E}\xi)(\mathsf{E}\eta).$$

Для случая, когда случайные величины — индикаторы событий (равны единице, если событие произошло, и нулю, если не произошло), получаем формулу

$$\langle \xi_A, \xi_B \rangle = \Pr[A \text{ и } B] - \Pr[A]\Pr[B];$$

корреляция равна нулю, когда события независимы.

В данном случае вероятность каждого события больше $\sqrt{nu}/n = \sqrt{u/n}$, а вероятность пересечения любых двух не больше u/n, так что корреляции отрицательны.

Теперь вспомним, что корреляции можно записать как скалярные произведения. А мы уже видели, что в n-мерном пространстве может быть не более $(n+1)$ векторов, образующих друг с другом тупые углы.

Теорема доказана.

Заметим, что если разрешить каждому Y_i содержать ровно $\sqrt{un}$ элементов, то углы из тупых станут неострыми, и вместо $n+1$ получится $2n$ векторов.

Возвращаясь к исходным параметрам: если минимальное расстояние кода $F\colon \Sigma^k \to \Sigma^n$ равно d, то код допускает декодирование $(n+1)$-списком с исправлением менее чем $n - \sqrt{n(n-d)}$ ошибок.

Доказанную теорему можно прочесть и так: при $n-e > \sqrt{n(n-d)}$ в шаре радиуса e может содержаться не более чем $n+1$ элементов с попарными расстояниями d или более. (Имеется в виду расстояние Хэмминга в Σ^n.)

Для кода с кодовым расстоянием 50% (в частности, для кода Адамара) эта теорема устанавливает возможность декодирования списком при доле ошибок $1 - \sqrt{1/2} \approx 29\%$, что лишь немного больше границы в 25% для однозначного декодирования. Однако (в отличие от случая однозначного декодирования) даваемая этой теоремой оценка отнюдь не является точной. Например, для кодов Адамара возможно декодирование списком при любой фиксированной доле ошибок, меньшей $1/2$. (Более точную оценку Джонсона мы приведём дальше.)

22. Декодирование списком кодов Адамара

Пусть $\varepsilon > 0$ — произвольное положительное число.

Теорема. Число кодовых слов кода Адамара, для которых доля несовпадений с данным словом не больше $\frac{1}{2}(1-\varepsilon)$, не превосходит $1/\varepsilon^2$.

Доказательство. Вспомним, что кодовые слова кода Адамара были плюс-минус базисными векторами ортонормированного базиса в евклидовом пространстве, если рассматривать кодовые слова как последовательности единиц и минус единиц и определять скалярное произведение по формуле

$$\langle (x_1,\ldots,x_n),(y_1,\ldots,y_n)\rangle = \frac{1}{n}\sum_i x_i y_i$$

(при этом n есть степень двойки, но сейчас это не важно).

Пусть x — произвольное слово (рассматриваемое как последовательность единиц и минус единиц). Если слово y отличается от него не более чем в $\frac{1}{2}(1-\varepsilon)n$ позициях, то

$$\langle x,y\rangle > \varepsilon$$

Теперь надо в качестве y взять все базисные векторы (с плюсом и минусом), и оценить, для скольких из них такое возможно. Из пары векторов y и $-y$ лишь один может удовлетворять этому неравенству; заменяя знаки у базисных векторов, можно считать, что это вектор с плюсом.

По теореме Пифагора (которую в данном случае называют равенством Парсеваля) квадрат гипотенузы $\langle x,x\rangle$ равен сумме квадратов сторон $\langle x,y_i\rangle$, взятой по всем базисным векторам y_i. В нашем случае $\langle x,x\rangle = 1$, поэтому количество y_i, при которых $\langle x,y_i\rangle \geqslant \varepsilon$, не превосходит $1/\varepsilon^2$, что и требовалось доказать.

23. Оценка Джонсона

Похожее утверждение можно доказать для любых кодов (над двухбуквенным алфавитом) с расстоянием около 50%. Вот точная формулировка:

Теорема. Пусть код имеет длину кодовых слов n и расстояние не меньше $\frac{1}{2}(1-\varepsilon)n$ для некоторого $\varepsilon > 0$. Тогда он допускает декодирование списком при доле ошибок $\frac{1}{2}(1-\sqrt{\varepsilon})$ и размере списка $2n$.

Мы сформулировали это утверждение в терминах кодов, но по существу речь идёт о точках в шаре: теорема утверждает, что в $\mathbb{B}^n$ шар радиуса $\frac{1}{2}(1-\sqrt{\varepsilon})n$ может содержать не более $2n$ точек с попарными расстояниями не меньше $\frac{1}{2}(1-\varepsilon)n$.

Доказательство. Как и в доказательстве оценки Плоткина, будем рассматривать последовательности единиц и минус единиц, являющиеся векторами единичной длины в евклидовом пространстве $\mathbb{R}^n$.

Пусть шар с центром u имеет радиус $\frac{1}{2}(1-\sqrt{\varepsilon})$ (в метрике Хэмминга) и содержит точки $v_1, \ldots, v_N$, причём расстояние (Хэмминга) между v_i и v_j не меньше $\frac{1}{2}(1-\varepsilon)n$ при любых $i \neq j$. В терминах скалярного произведения это означает, что

$$(u, v_i) \geqslant \sqrt{\varepsilon} \quad \text{и} \quad (v_i, v_j) \leqslant \varepsilon.$$

Другими словами, угол между u и v_i острый и не слишком близок к прямому, в то время как угол между v_i и v_j достаточно близок к прямому или даже тупой.

Мы хотим доказать, что $N \leqslant 2n$. Это было бы так, если бы углы между v_i и v_j были тупыми или прямыми (см. доказательство оценки Плоткина). Чтобы свести дело к этому случаю, вычтем немного u из каждого v_i. Для этого найдём λ, при котором

$$(v_i - \lambda u, v_j - \lambda u) = (v_i, v_j) - \lambda((v_i, u) + (v_j, u)) + \lambda^2(u, u) \leqslant \\ \leqslant \varepsilon - 2\lambda\sqrt{\varepsilon} + \lambda^2 = (\lambda - \sqrt{\varepsilon})^2 \leqslant 0,$$

Видно, что всё подобрано так, что годится $\lambda = \sqrt{\varepsilon}$. Оценка Джонсона доказана.

Аналогичная (но более сложно доказываемая) оценка имеет место и для кодов над q-буквенным алфавитом, только надо заменить $1/2$ на $(q-1)/q$ и увеличить константу при n в оценке для размера списка.

24. Оценка Элайеса – Бассалыго

Оценку Джонсона можно использовать для улучшения границы Хэмминга. Напомним, что она получалась так: если код имеет расстояние d, то шары радиуса $d/2$ с центрами в кодовых словах не пересекаются, и потому их суммарный объём не больше объёма всего пространства.

Теперь мы возьмём шары большего радиуса, чем $d/2$. Про них уже нельзя утверждать, что пересечений нет. Однако — при подходящем радиусе — оценка Джонсона позволяет утверждать, что любая точка z покрыта небольшим числом шаров (поскольку шар с центром z содержит небольшое число кодовых слов). Поэтому суммарный объём шаров превосходит объём всего пространства в небольшое число раз.

Конкретно: пусть код со словами длины n в двухбуквенном алфавите имеет расстояние $d = \frac{1}{2}(1-\varepsilon)n$. Шары радиуса $\frac{1}{2}(1-\sqrt{\varepsilon})$ могут пересекаться, но кратность пересечений не превосходит $2n$. Получаем неравенство

$$2^k V_2(\tfrac{1}{2}(1-\sqrt{\varepsilon})n) \leqslant 2n \cdot 2^n,$$

для кодов с расстоянием

$$d = \tfrac{1}{2}(1-\varepsilon)n$$

Переходя к логарифмам, используя приближённую формулу для V_2 (с шенноновской энтропией), деля на n и пренебрегая множителем $2n$ (малым по сравнению с 2^n при больши́х n), получаем асимптотическую границу, показанную на рис. 3 пунктирной линией. Эта граница называется *границей Элайеса – Бассалыго*.

25. Декодирование списком кодов Рида – Соломона

Пусть $\Sigma = \mathbb{F}_q$ — конечное поле из q элементов. Код Рида – Соломона кодирует многочлен степени меньше k (то есть набор из k его коэффициентов) значениями этого многочлена в n точках поля. Кодовое расстояние для такого кода равно $n-k+1$, поэтому для восстановления многочлена по (испорченному) набору значений достаточно иметь примерно $(n+k)/2$ правильных значений в этом наборе.

Как мы видели, для декодирования списком (полиномиального размера) достаточно иметь примерно $\sqrt{nk}$ правильных значений.

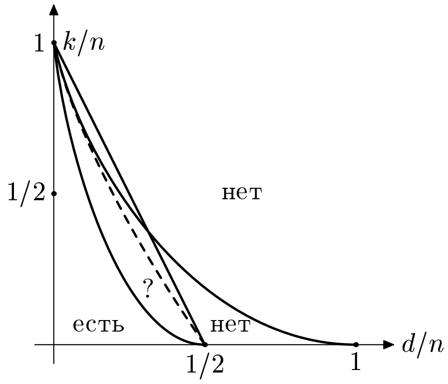


Рис. 3. Граница Элайеса – Бассалыго.

Сейчас мы дадим новое доказательство этого факта, из которого можно извлечь алгоритм декодирования.

Идея доказательства: при декодировании кода Рида – Соломона мы искали многочлены $D(x)$ и $Q(x)$, для которых пары (x_i, y_i) (точки и значения в этих точках) ложатся на кривую $D(x)y - Q(x) = 0$. Другими словами, мы искали многочлен $R(x, y)$ степени 1 по переменной y, который обращается в нуль в этих точках, а затем доказывали, что $Q(x)$ делится на $D(x)$ и многочлен $R(x, y)$ представляется в виде $D(x)(y - P(x))$, где $P(x)$ — частное от деления Q на D — искомый ответ.

Сейчас мы будем делать то же самое, но с двумя отличиями:

• многочлен R уже не обязательно первой степени по y;

• требуется не просто обращение в нуль выражения $R(x_i, y_i)$, а обращение в нуль с некоторой кратностью a_i.

Говорят, что многочлен $R(x, y)$ *обращается в нуль в точке* (a, b) *с кратностью* m, если $R(a+u, b+v)$ как многочлен от u и v не имеет членов степени меньше m. Кратность 1 означает просто $R(a, b) = 0$, кратность 2 означает, что и производные по x и y равны нулю, кратность 3 — обращение в нуль вместе со вторыми производными и так далее.

Заметим, что обращение в нуль в данной точке есть линейное условие на коэффициенты многочлена R; обращение в нуль с кратностью m представляет собой набор из $1+2+\ldots+m=m(m+1)/2$ условий. Тем самым мы можем подобрать искомый многочлен R методом неопределённых коэффициентов, если только этих коэффициентов больше, чем условий во всех точках. Если при этом степень многочлена R будет не слишком велика, то его обращение в нуль на многих точках кривой $y=P(x)$ (где P — исходный многочлен) гарантирует, что он обращается в нуль на всей этой кривой.

Чтобы сформулировать точное утверждение, определим *k-взвешенную степень* многочленов в $\mathbb{F}_q[x,y]$: моном $x^u y^v$ имеет степень $u+kv$; степень многочлена — наибольшая из степеней мономов. (Переменная y считается имеющей вес k, поскольку мы собираемся на её место подставлять многочлен степени k.)

Следующие леммы носят чисто алгебраический характер.

Лемма 1. Пусть $x_1,\ldots,x_n$ и $y_1,\ldots,y_n$ — произвольные элементы поля $\mathbb{F}_q$. Пусть $a_1,\ldots,a_n$ — произвольные натуральные числа. Если a — натуральное число, для которого

$$a^2 > k\sum_{i=1}^{n} a_i(a_i+1),$$

то существует ненулевой многочлен $R(x,y)$, который имеет k-взвешенную степень меньше a и который обращается в нуль в точке (x_i,y_i) с кратностью a_i (при всех $i=1,\ldots,n$).

Доказательство. Размерность пространства многочленов, имеющих k-взвешенную степень меньше a, равна числу решений неравенства

$$u+kv<a$$

в целых числах $u,v\geqslant 0$. А это число не меньше площади треугольника, ограниченного прямыми $u+kv\leqslant a$, $u\geqslant 0$, $v\geqslant 0$, которая равна $a^2/2k$ (половина произведения катетов a и a/k). В самом деле, если нарисовать этот треугольник на клетчатой бумаге и взять любую точку (u,v) внутри него (не на границе), то эта точка попадает в клетку с левым нижним углом $(\lfloor u\rfloor,\lfloor v\rfloor)$. Координаты этого левого нижнего угла целые и удовлетворяют неравенству, поэтому число целых решений неравенства не меньше площади треугольника (ведь он покрыт клетками).

Поэтому размерность пространства многочленов, у которых k-взвешенная степень меньше a, не меньше $a^2/2k$. С другой стороны, число линейных условий, выражающих обращение в нуль многочлена R в точках (x_i, y_i) с кратностью a_i, равно

$$\sum_{i=1}^{n} \frac{a_i(a_i+1)}{2},$$

что по условию леммы меньше размерности пространства многочленов.

Лемма 2. Пусть $x_1, \ldots, x_n$ и $y_1, \ldots, y_n$ — произвольные элементы поля $\mathbb{F}_q$. Пусть $a_1, \ldots, a_n$ — произвольные натуральные числа. Пусть многочлен $R(x,y)$ имеет k-взвешенную степень меньше a и обращается в нуль в точке (x_i, y_i) с кратностью a_i (при всех $i = 1, \ldots, n$). (Пока всё как в лемме 1.) Пусть, наконец, многочлен $P(x)$ имеет степень не больше k и

$$\sum \{a_i \mid P(x_i) = y_i\} \geqslant a$$

(сумма весов точек (x_i, y_i), через которые проходит его график, не меньше a). Тогда $R(x, P(x))$ является нулевым многочленом (равен нулю в кольце многочленов от одной переменной).

Доказательство. Многочлен $N(x) = R(x, P(x))$ (как многочлен от одной переменной x) имеет степень меньше a. Если (x_i, y_i) является корнем $R(x,y)$ кратности m, а $P(x_i) = y_i$, то многочлен $N(x)$ имеет корень x_i кратности m. Поэтому у многочлена N число корней с учётом кратности не меньше a, а степень меньше a. Следовательно, $N = 0$, что и утверждает лемма.

Лемма 3. Пусть $R(x,y)$ — многочлен от двух переменных, а $P(x)$ — от одной переменной, причём $R(x, P(x)) = 0$. Тогда $R(x,y)$ делится на многочлен $y - P(x)$ в кольце многочленов от двух переменных.

(Равенство $R(x, P(x)) = 0$ понимается как равенство в кольце многочленов, то есть как равенство нулю всех коэффициентов; это, вообще говоря, более сильное утверждение, чем равенство во всех точках, если элементов в основном поле мало.)

Доказательство. Рассмотрим $R(x,y)$ как многочлен от y, коэффициенты которого являются многочленами от x, и разделим его уголком на $y - P(x)$. Поскольку старший коэффициент делителя равен единице, при этом не появится никаких дробей, частное будет многочленом от x и y, а остаток — многочленом от x (поскольку делитель имеет степень 1 по y, остаток имеет степень 0 по y). Таким образом,

$$R(x,y) = (y - P(x))Q(x,y) + S(x).$$

Подставив теперь $P(x)$ вместо y, получим, что $S(x)$ — нулевой многочлен, что и требовалось доказать.

Теперь всё готово, чтобы вернуться к декодированию списком. Пусть имеется n точек (x_i, y_i). Нас интересует, сколько различных многочленов степени не больше k могут проходить через s из них (для данных n, k и s).

Чтобы оценить это число, рассмотрим целое положительное N (пока произвольное) и положим $a_1 = a_2 = \ldots = N$. (Кажется странным, зачем вообще нужны кратности, если они все равны. Тем не менее, как мы увидим, это полезно.)

Чтобы применить лемму 1, нужно выбрать натуральное a, для которого

$$a^2 > k \sum_{i=1}^{n} a_i(a_i + 1) = knN(N+1),$$

то есть

$$a > \sqrt{knN(N+1)}$$

Применение леммы 1 даёт многочлен $R(x,y)$, имеющий k-взвешенную степень меньше a и обращающийся в нуль с кратностью N во всех точках (x_i, y_i).

Если многочлен P имеет степень не выше k и проходит через s точек из числа (x_i, y_i), причём

$$sN \geqslant a,$$

то $R(x, P(x)) = 0$ по лемме 2 и $R(x,y)$ делится на $y - P(x)$ по лемме 3.

Из курса алгебры известно, что в кольце $\mathbb{F}_q[x,y]$ разложение на множители однозначно, так что число делителей вида $y - P(x)$ у многочлена $R(x,y)$ меньше a/k (иначе суммарная k-взвешенная степень произведения достигнет a, ведь y имеет взвешенную степень k).

Собирая всё вместе (считая a равным sN и поделив обе части неравенства на N) получаем такое утверждение:

Теорема. Если

$$s > \sqrt{kn(1 + 1/N)}\,,$$

то количество различных многочленов P степени не выше k, проходящих через s или более точек среди $(x_1, y_1), \ldots, (x_n, y_n)$, меньше sN/k.

Видно, что эта теорема применима при $s > \sqrt{nk}$; чем ближе это неравенство к равенству, тем большее N придётся брать (и тем слабее будет оценка на число различных многочленов). Ещё из неё видно, например, что при $s = (1+\varepsilon)\sqrt{nk}$ и $n = O(k)$ число различных многочленов P степени меньше k, проходящих через s точек из n, есть $O(1)$.

В следующем разделе нам потребуется аналогичное утверждение для случая произвольных весов:

Теорема. Пусть $x_1, \ldots, x_n$ и $y_1, \ldots, y_n$ — произвольные элементы поля $\mathbb{F}_q$. Пусть $a_1, \ldots, a_n$ — произвольные натуральные числа. Пусть натуральные числа k и a таковы, что

$$a^2 > k \sum_{i=1}^{n} a_i(a_i + 1).$$

Тогда количество многочленов $P(x)$ степени не выше k, для которых набираемый ими вес

$$\sum \{a_i \mid P(x_i) = y_i)\}$$

не меньше a, меньше a/k.

В самом деле, по лемме 1 можно найти многочлен $R(x, y)$, обращающийся в нуль в (x_i, y_i) с кратностью a_i и имеющий k-взвешенную степень меньше a; по лемме 2 выполнено равенство $R(x, P(x)) = 0$ для любого многочлена степени не выше k, набирающего вес не меньше a; по лемме 3 $R(x, y)$ делится на $y - P(x)$ для всех этих многочленов и потому их число меньше a/k.

Приведённое доказательство после некоторого дополнительного анализа даёт полиномиальный алгоритм декодирования списком кодов Рида – Соломона. Надо только научиться раскладывать на множители многочлены в $\mathbb{F}_q[x, y]$ или хотя бы выделять из них все множители вида $y - P(x)$.

Вообще-то известны полиномиальные алгоритмы для полного разложения на множители (произвольного вида) многочленов от нескольких переменных над конечным полем. Но нам достаточно уметь выделять множители специального вида, поэтому мы приведём простое рассуждение специально для этого случая.

Пусть нам дан многочлен $R(x, y)$ и пусть (a, b) — некоторая точка, где он обращается в нуль: $R(a, b) = 0$. Посмотрим на производную по второму аргументу R_y. Если в точке (a, b) она не обращается в нуль, то можно действовать как в курсе анализа, где к уравнению $R(x, y) = 0$ в окрестности точки (a, b) применяют теорему о неявной функции и находят функцию φ, для которой $\varphi(a) = b$ и $R(x, \varphi(x)) = 0$; при этом высшие производные функции φ можно найти, зная соответствующие производные функции R.

Поскольку у нас поле конечной характеристики, то лучше действовать непосредственно в терминах коэффициентов многочлена. Сдвигом по x и y можно свести дело к случаю $a = 0$, $b = 0$. Запишем многочлен $R(x, y)$ по степеням y:

$$R(x, y) = r_0(x) + r_1(x)y + r_2(x)y^2 + r_3(y)y^3 + \ldots$$

Равенство $R(x, P(x)) = 0$ даёт нам уравнения на коэффициенты многочлена $P(x) = p_1x + p_2x^2 + p_3x^3 + \ldots$ (свободного члена у $P(x)$ нет, так как по предположению $P(0) = 0$). Приравнивая свободные члены в

$$0 = R(x, P(x)) = r_0(x) + r_1(x)P(x) + r_2(x)P^2(x) + r_3(x)P^3(x) + \ldots,$$

получаем равенство $R(0, 0) = 0$, то есть $r_0(0) = 0$. Приравнивая коэффициенты при x, получаем линейное уравнение на p_1, при этом коэффициент равен $r_1(0) = R'_y(0, 0)$, что не равно нулю по нашему предположению. Приравнивая члены с x^2, получаем линейное уравнение на p_2 с тем же коэффициентом $r_1(0)$ и так далее.

Так мы можем определять коэффициенты P в порядке возрастания степеней и вопрос только в том, получится ли многочлен или ряд. Поскольку мы заранее знаем верхнюю оценку на степень многочлена $P(x)$, при которой $y - P(x)$ только и может быть делителем $R(x, y)$, то достаточно получить коэффициенты P до этой границы и затем сделать проверку (подстановкой в R).

Переберём все пары (a, b), для которых $R(a, b) = 0$ и $R'_y(a, b) \neq 0$, и выполним для каждой такой пары указанную процедуру. В результате мы найдём все делители вида $y - P(x)$, кроме тех P, которые целиком проходят через точки с нулевой производной R'_y. Но эти P

являются решениями уравнения $R'_y(x, P(x)) = 0$, и мы можем считать (рассуждая по индукции), что они нам уже известны. (При этом степень уменьшается на единицу, так что алгоритм остаётся полиномиальным.)

26. Рид – Соломон плюс Адамар: декодирование списком

Рассматривая код Адамара в разделе 15, мы уже упоминали о возможности использовать его в качестве внутреннего кода, когда внешним является код Рида – Соломона. Сейчас мы используем методы предыдущего раздела (декодирование кода Рида – Соломона с весами), чтобы получить алгоритм декодирования списком для такого каскадного кода.

Напомним параметры рассматриваемого кода. Мы используем поле $\mathbb{F}_{2^m}$ из 2^m элементов, обозначаемых $x_1, \ldots, x_{2^m}$. Внешний код Рида – Соломона кодирует $\varepsilon 2^m$ элементов поля, рассматривая их как коэффициенты многочлена P степени меньше $\varepsilon 2^m$ и беря значения $P(x_1), \ldots, P(x_{2^m})$. Внутренний код Адамара кодирует каждое из этих значений (то есть слово из m битов) словом из 2^m битов.

Результирующий каскадный код кодирует слово из $\varepsilon m 2^m$ битов, разбивая его на $\varepsilon 2^m$ блоков по m битов, и затем заменяя каждый блок на его код Адамара. Получается 2^m блоков по 2^m битов в каждом, всего 2^{2m} битов.

Кодовое расстояние (удельное, в расчёте на бит кодового слова) для внешнего кода равно $1-\varepsilon$, а для внутреннего равно $1/2$, поэтому для конкатенации оно равно $\frac{1}{2}(1-\varepsilon)$.

Оценка Джонсона говорит теперь, что декодирование списком возможно с долей ошибок $\frac{1}{2}(1-\sqrt{\varepsilon})$ и длиной списка $2 \cdot 2^{2m}$ (удвоенная длина кодового слова). Покажем, что такой список можно получить с помощью описанной в предыдущем разделе техники (набором весов вдоль алгебраической кривой).

Итак, пусть имеется некоторое слово z длины 2^{2m}, для которого мы хотим составить список близких к нему кодовых слов. Разобьём слово z на 2^m блоков $z_1, \ldots, z_{2^m}$ длины 2^m каждый. Как и раньше, декодирование каскадного кода начинается с декодирования каждого блока по отдельности. Но в данном случае мы не будем выбирать какого-то одного варианта декодирования, а рассмотрим все возможные варианты. А именно, для каждого блока z_i и для любого элемента x_j поля $\mathbb{F}_{2^m}$ посмотрим, насколько близко слово z_i к коду Адамара элемента x_j. Оба сравниваемых слова имеют длину 2^m, и мы вычтем число несовпадений из числа совпадений; разность мы будем называть весом и обозначать a_{ij}. Если $a_{ij} < 0$ (несовпадений больше, чем совпадений) то такой вариант декодирования (z_i получилось искажением кода элемента x_j) мы сразу же отвергнем. Для всех оставшихся вариантов веса a_{ij} неотрицательны. Получается максимум 2^{2m} неотрицательных весов, поскольку каждый из индексов i, j пробегает 2^m значений (соответствующих 2^m элементам поля).

Эти веса будут использованы при декодировании внешнего кода. Мы будем проводить алгебраическую кривую через все точки $(x_i, x_j) \in \mathbb{F}^2_{2^m}$, у которых веса $a_{i,j}$ неотрицательны. (Заметим, что теперь у нас есть много точек с одинаковой первой координатой, чего раньше не было.) В обозначениях последней теоремы предыдущего раздела:

$$q = 2^m, \quad n \leqslant 2^{2m}, \quad k = \varepsilon 2^m.$$

Остаётся понять, каким должно быть значение a. Чем оно больше, нем утверждение теоремы слабее, поэтому его надо взять минимально возможным, для которого

$$a^2 > k \sum_{i=1}^{n} a_i(a_i+1) = \varepsilon 2^m \sum_{i,j} a_{ij}(a_{ij}+1).$$

(в левой части равенства используются обозначения предыдущего раздела, а справа подставлены их текущие значения). Правую часть можно записать как

$$\varepsilon 2^m [\sum_{i,j} a_{ij}^2 + \sum_{i,j} a_{ij}].$$

Второе (меньшее) слагаемое оценим сверху, заменив все a_{ij} на их максимально возможное значение 2^m, получится $\varepsilon 2^{4m}$. Первое слагаемое будем оценивать более деликатным способом. Сгруппируем слагаемые с одним и тем же i (соответствующие одному блоку в конкатенации, но разным вариантам его декодирования), получим

$$\varepsilon 2^m \sum_i [\sum_j a_{ij}^2].$$

Теперь оценим отдельно каждую квадратную скобку (при фиксированном j). Вес a_{ij} представляет собой скалярное произведение кода Адамара элемента x_j и слова z_i, взятое с множителем 2^m (скажем, если скалярное произведение равно 1, то есть имеет место полное совпадение, вес равен 2^m). Остаётся вспомнить про неравенство Бесселя, согласно которому сумма квадратов скалярных произведений единичного вектора с векторами ортонормированного набора не превосходит 1. После учёта коэффициентов это даёт

$$\varepsilon 2^m \sum_i [\sum_j a_{ij}^2] \leqslant \varepsilon 2^m \sum_i 2^{2m} = \varepsilon 2^m \cdot 2^m \cdot 2^{2m} = \varepsilon 2^{4m}$$

Таким образом, для применения теоремы из предыдущего раздела достаточно неравенства

$$a^2 > \varepsilon 2 \cdot 2^{4m},$$

то есть

$$a > \sqrt{2\varepsilon}\, 2^{2m}.$$

Таким образом, количество многочленов степени не выше $\varepsilon 2^m$, набирающих вес a или больше, невелико. Остаётся понять, что суммарный вес, набираемый многочленом, есть разница между суммарным числом совпадений и несовпадений в соответствующем кодовом слове, и тем самым можно выловить все кодовые слова, для которых эта разница между числом совпадений и несовпадений больше

$$\sqrt{2\varepsilon} 2^{2m},$$

то есть в расчёте на бит больше $\sqrt{2\varepsilon}$. Поскольку эта разница есть $1 - 2d$, где d — доля ошибок, условие на d будет таким:

$$d < \frac{1}{2}(1 - \sqrt{2\varepsilon}.$$

При этом число кодовых слов в списке оценивается величиной

$$a/k = \sqrt{2\varepsilon} 2^{2m}/(\varepsilon 2^m),$$

что по порядку величины равно 2^m, то есть корню из длины кодового слова (немного лучше, чем в оценке Джонсона).

Зато у нас вместо $\sqrt{\varepsilon}$, как в оценке Джонсона, появляется $\sqrt{2\varepsilon}$. В основном это из-за того, что мы очень грубо оценивали сумму первых степеней. Сделаем это точнее: при каждом i у нас имеется сумма 2^m чисел, сумма квадратов которых не превосходит 2^{2m}, так что их среднее квадратичное не превосходит $2^{m/2}$. Среднее арифметическое не больше среднего квадратичного, поэтому сумма $\sum_j a_{ij}$ не больше $2^{3m/2}$ (а раньше мы её оценивали как 2^{2m}, так что есть заметное улучшение).

Отсюда ясно, что константу 2 в $\sqrt{2\varepsilon}$ можно (для достаточно бóльших m) заменить на сколь угодно близкую к единице.

27. Вероятностное декодирование списком для кодов Адамара

Ещё один результат, связанный с кодами Адамара — теорема Голдрайха и Левина о вероятностном декодировании списком (первоначальная формулировка была в терминах вычислительной криптографии, но по существу она именно об этом).

Теорема. Пусть $\alpha < 1/2$ и $\varepsilon > 0$. Существует вероятностный полиномиальный алгоритм, который по данному m и по данному слову g длины 2^m указывает список полиномиальной длины, который с вероятностью не менее $1-\varepsilon$ содержит все кодовые слова кода Адамара, отличающиеся от g не более чем в $\alpha 2^m$ позиций.

Эта формулировка требует трёх уточнений. Во-первых, полиномиальность означает, что время работы алгоритма (во всех случаях) ограничено некоторым полиномом от m. Во-вторых, слово g (имеющее длину 2^m) подаётся на вход алгоритма в виде «оракула» (алгоритм может запросить значение любого бита, указав его номер, то есть слово рассматривается как функция $g\colon \mathbb{B}^m \to \mathbb{B}$, и алгоритм может её «вызывать»). В третьих, алгоритм указывает не сами кодовые слова кода Адамара (они имеют экспоненциальную длину, и за полиномиальное от m время их выписать нельзя), а их прообразы при кодировании (то есть слова из $m+1$ битов, которые являются коэффициентами соответствующей аффинной функции).

Доказательство. Будем считать, что $\alpha = \frac{1}{2} - \delta$ при некотором $\delta > 0$.

Для доказательства теоремы мы построим алгоритм $\mathcal{B}$, получающий на вход рациональные числа $\delta, \varepsilon > 0$, натуральное число m и (в виде оракула) функцию $g\colon \mathbb{B}^m \to \mathbb{B}$, и выдающий список $\mathcal{B}(\delta, \varepsilon, m, g)$ аффинных функций (то есть их коэффициентов). При этом алгоритм будет полиномиальным от m, $1/\delta$ и $1/\varepsilon$, если δ и ε достаточно просты, так что длины их записей ограничены полиномом от $1/\delta$ и $1/\varepsilon$ (это заведомо так, если они обратны к целым числам, и в дальнейшем мы рассматриваем только такие ε и δ).

Основное свойство этого алгоритма:

Лемма. Пусть $f\colon \mathbb{B}^m \to \mathbb{B}$ — аффинная функция, а $g\colon \mathbb{B}^m \to \mathbb{B}$ — произвольная функция, отличающаяся от f не более чем в $(\frac{1}{2} - \delta)$-доле аргументов. Тогда вероятность того, что f войдёт в список $\mathcal{B}(\delta, \varepsilon, m, g)$, не меньше $1 - \varepsilon$.

Прежде чем доказывать лемму, поясним её связь с исходной формулировкой теоремы. Лемма говорит о вероятности покрыть списком *одну* аффинную функцию, близкую к g; в теореме мы говорили о покрытии *всех* таких функций. Но поскольку таких функций мало (что следует из свойств кода Адамара — при данном δ их не больше некоторой константы), то вероятность при переходе от леммы к теореме возрастёт не сильно. (Кстати, тот факт, что функций немного, следует из леммы — если бы их было много, то список полиномиального размера, выдаваемый алгоритмом, не мог с высокой вероятностью покрывать любую из них.)

Доказательство леммы. Итак, в роли алгоритма $\mathcal{B}$ мы имеем доступ к g — «искажённому варианту» неизвестной аффинной функции

$$f(x_1, \ldots, x_m) = f_0 + f_1 x_1 + \ldots + f_m x_m,$$

и должны отгадать коэффициенты $f_0, \ldots, f_m$. Эти коэффициенты определяются значениями f в $m+1$ точках

$$(0, 0, \ldots, 0), (1, 0, \ldots, 0), (0, 1, \ldots, 0), \ldots, (0, 0, \ldots, 1)$$

(и наоборот), и нам будет удобнее говорить о отгадывании этих значений. (На самом деле конкретный вид точек нам не важен, мы можем отгадывать значения в произвольных $m+1$ точках — и даже в произвольном полиномиальном числе точек.)

Поскольку функция f является аффинной, то для любых x и r из $\mathbb{B}^m$ выполняется равенство

$$f(x) = f(x + r) - \bar{f}(r),$$

где $\bar{f}$ — линейная часть f (без f_0). Пусть x фиксировано, а r пробегает $\mathbb{B}^m$. Поскольку f и g совпадают по крайней мере в $(\frac{1}{2}+\delta)$-доле позиций (что больше половины), то

$$f(x) = \text{большинство из } \left[g(x+r) - \bar{f}(r)\right],$$

и это большинство можно определить методом Монте-Карло, выбрав несколько случайных r и проведя голосование среди выбранных значений.

План этот на первый взгляд не имеет смысла, так как в правой части стоит $\bar{f}(r)$, которого мы всё равно не знаем (ведь $\bar{f}$ отличается от f лишь отсутствием коэффициента f_0).

Заметим, однако, что для нахождения большинства по методу Монте-Карло не обязательно, чтобы испытания были по-настоящему независимы. Достаточно (хотя и с худшей оценкой на вероятность ошибки), чтобы они были *попарно* независимы. Именно, имеет место

Закон больших чисел для попарно независимых событий. Пусть события $A_1, \ldots, A_N$ попарно независимы и каждое из них имеет вероятность меньше $\frac{1}{2} - \delta$. Тогда вероятность того, что произойдёт половина или больше событий, не превосходит

$$\frac{1}{\delta^2} \cdot \frac{1}{N}$$

(и потому мала при больши́х N).

В самом деле, рассмотрим индикаторы этих событий (случайные величины a_i, равные единице, если A_i случилось, и нулю в противном случае). Математическое ожидание каждого из индикаторов не больше $\frac{1}{2} - \delta$. Поэтому математическое ожидание суммы не больше $N(\frac{1}{2} - \delta)$. Теперь воспользуемся попарной независимостью: она гарантирует, что дисперсия суммы $\sum a_i$ равна сумме дисперсий; дисперсия a_i не превосходит 1 (на самом деле даже 1/4), поэтому дисперсия суммы не превосходит N. Дисперсия есть математическое ожидание квадрата отклонения от математического ожидания. Если произойдёт больше половины событий, то отклонение будет по крайней мере $N\delta$, а его квадрат — $N^2\delta^2$. Поэтому по неравенству Чебышёва вероятность такого события не больше

$$\frac{N}{N^2\delta^2} = \frac{1}{\delta^2} \cdot \frac{1}{N},$$

что и требовалось доказать.

Откуда мы возьмём попарно независимые величины? Если взять s (полностью) независимых векторов в $\mathbb{B}^m$, то их частичные суммы (всего 2^s-1 векторов) будут попарно независимы. Например, если u_1, u_2 и u_3 — (полностью) независимые векторы, то семь векторов

$$u_1, u_2, u_3, u_1+u_2, u_1+u_3, u_2+u_3, u_1+u_2+u_3$$

будут попарно независимы. (Скажем, при данном значении u_1+u_2 все значения u_1+u_3 равновероятны, поскольку u_3 не зависит от пары (u_1,u_2).)

Геометрически: на s случайных независимых равномерно распределённых векторов мы натягиваем параллелепипед, и 2^s-1 его вершин (не считая нулевой) будут попарно независимы.

Теперь уже можно объяснить идею доказательства: мы положим

$$f(x) = \text{большинство из } \left[g(x+r_i)-\bar f(r_i)\right],$$

где r_i — попарно независимые векторы (2^s-1 штук), а именно, ненулевые вершины параллелепипеда, натянутого на (полностью) независимые векторы $u_1,\ldots,u_s$. Чтобы получить малую вероятность ошибки, достаточно взять полиномиальное число попарно независимых векторов, то есть $2^s-1=\operatorname{poly}(m)$, а тогда $s=O(\log m)$. (Это — для фиксированных ε и δ, оценку для общего случая см. ниже.)

Теперь главное: все значения $\bar f(r_i)$ в силу линейности $\bar f$ определяются значениями $\bar f(u_1),\ldots,\bar f(u_s)$, то есть s битами. Возможных вариантов этих значений 2^s, то есть полиномиальное от m количество. Вспомним, что нам разрешалось отгадывать с нескольких попыток: алгоритм $\mathcal{B}$ выдаёт не один набор коэффициентов, а полиномиальный (от m) список вариантов. Так что нам позволено перепробовать эти варианты, и всё сходится.

Оценим возникающие вероятности более подробно. Нам нужно восстановить значения f в $m+1$ точках. Мы начинаем с того, что выбираем независимые векторы $u_1,\ldots,u_s$ и получаем из них попарно независимые векторы $r_1,\ldots,r_{2^s-1}$. Затем для каждой из $m+1$ точек мы проводим голосование с 2^s-1 участниками и получаем набор из $m+1$ предполагаемых значений. Эта процедура предполагает известными значения $\bar f(u_1),\ldots,\bar f(u_s)$ и проводится 2^s раз, для всех возможных вариантов. Подчеркнём, что выбор случайных векторов $u_1,\ldots,u_s$ производится только один раз: одни и те же векторы используются для всех $m+1$ точек и во всех 2^s попытках.

Для данной точки x события $f(x + r_i) \neq g(x + r_i)$ (всего $2^s - 1$ событий при $i = 1, 2, \ldots, 2^s - 1$) являются попарно независимыми. Вероятность каждого из них не больше $\frac{1}{2} - \delta$. Поэтому вероятность того, что выборка окажется «плохой» (не меньше половины попаданий в ошибки, где $f \neq g$), не превосходит

$$\frac{1}{\delta^2} \cdot \frac{1}{2^s - 1}.$$

Это — для данной точки x; вероятность того, что выборка окажется плохой хотя бы для одной из $m + 1$ точек, не больше

$$\frac{1}{\delta^2} \cdot \frac{1}{2^s - 1} \cdot (m + 1)$$

и потому меньше ε, если

$$2^s > \frac{m+1}{\varepsilon\delta^2} + 1.$$

А если выборка хороша для всех $m + 1$ точек, то в список, выдаваемый алгоритмом $\mathcal{B}$ (и содержащий 2^s вариантов), войдёт и правильный ответ (коэффициенты аффинной функции f). Длина списка и время работы алгоритма $\mathcal{B}$ при этом полиномиальны от $(m + 1)/\varepsilon\delta^2 + 1 = \mathrm{poly}(m, 1/\varepsilon, 1/\delta)$.

28. Линейные коды низкой плотности и экспандеры

В этом разделе мы вернёмся к линейным кодам и покажем, как можно построить линейный код (над полем $\mathbb{F}_2$), для которого не только кодирование, но и декодирование выполняется быстро. Идея построения таких кодов (LDPC, low density parity-check codes) была предложена Галлагером ещё в 1963 году, но потом эти коды были почти что забыты. Сипсер и Спилман вернулись к ним в середине 1990-х годов и обнаружили их связь с некоторым классом графов — экспандерами.

Линейные коды и графы

Линейный код над полем из двух элементов состоит из двоичных слов $x_1 x_2 \ldots x_n$, биты которых удовлетворяют некоторым проверочным уравнениям. Поскольку в поле только два элемента, достаточно указать, какие биты входят в уравнение, и код задаётся двудольным графом (рис. 4). В левой доле n вершин $x_1, \ldots, x_n$. Каждая вершина правой доли задаёт уравнение: сумма тех битов, с которыми она соединена, равна нулю (каждое ребро соответствует вхождению бита в уравнение; кратные рёбра не разрешены). Низкая плотность означает, что рёбер мало. Мы будем предполагать, что все вершины слева имеют одинаковую степень D (каждый бит входит в D контрольных сумм), и что уравнений меньше, чем переменных (и тем самым есть более одного кодового слова).

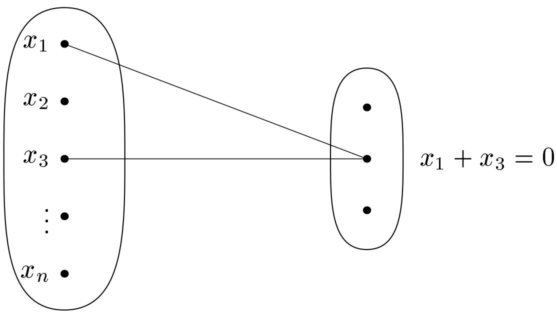


Рис. 4. Проверочная матрица как граф.

Пусть A — некоторое множество вершин левой доли (переменных). Из них выходят $D|A|$ рёбер, и поэтому множество соседей $N(A)$ содержит не более $D|A|$ элементов. Если эта оценка близка к точной для множеств A небольшого размера, граф называют *экспандером*. Точнее, мы фиксируем некоторое $\varepsilon > 0$ и некоторое $\alpha > 0$ и требуем, чтобы

$$|N(A)| \geqslant (1-\varepsilon)D|A| \quad \text{при } |A| \leqslant \alpha n$$

Это определение имеет смысл при достаточно малых ε; в дальнейшем нам понадобятся значения $\varepsilon < 1/4$. Оно говорит, что соседи левых вершин сравнительно мало пересекаются друг с другом, и если взять не слишком много таких вершин, то за счёт пересечений множество соседей уменьшится не более чем в $(1-\varepsilon)$ раз. Слова «не слишком много» важны, так как для больших множеств A в правой доле просто не хватит вершин. (Отметим, что при малом ε необходимо брать достаточно большое D, иначе две вершины слева, имеющие общего соседа справа, не будут удовлетворять условию — а такие вершины должны быть, так как справа меньше вершин, чем слева.)

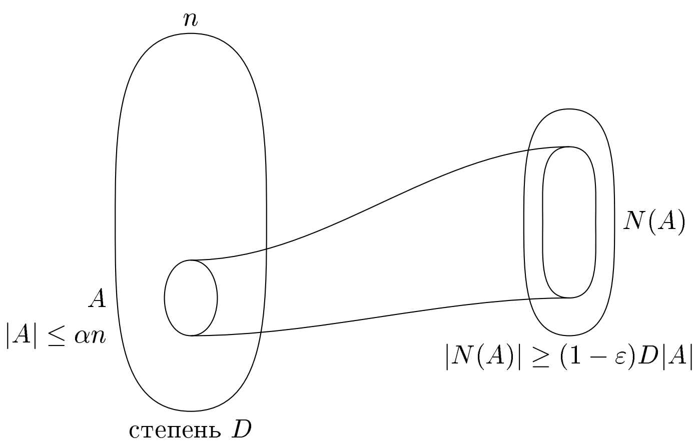


Рис. 5. К определению экспандера.

Исправление ошибок и экспандеры

Мы отложим вопрос о существовании экспандеров с данными параметрами (число вершин слева n, их степень D, число вершин справа m, границы α и ε) и сначала посмотрим на параметры получающегося кода. Прежде всего отметим, что имеется 2^{n-m} кодовых слов (или даже больше, если m уравнений оказались линейно зависимыми).

Мы уже говорили, что из $|A|$ вершин левой доли исходит $D|A|$ рёбер; они приходят в $(1-\varepsilon)D|A|$ вершин. Значит, имеется не более $\varepsilon D|A|$ вершин, в которые входит более одного ребра (каждая такая вершина уменьшает число соседей по крайней мере на 1), и остаётся не менее $(1-2\varepsilon)D|A|$ вершин, в которые входит только одно ребро.

Отсюда следует, что *минимальное расстояние кода больше* αn. В самом деле, пусть два кодовых слова различны, и A — те позиции (переменные), в которых они отличаются. Поскольку код линейный, будем считать, что есть кодовое слово, которое содержит единицы в позициях из A (а второе слово нулевое). Мы видим, что если единиц не более αn, то есть не менее $(1-2\varepsilon)D|A|$ «критических» контрольных сумм, которые содержат ровно одну единицу (и остальные нули). Если $\varepsilon < 1/2$, получаем противоречие: для кодовых слов все контрольные суммы должны быть равны нулю. (Заметим, что некритические контрольные суммы тоже могут быть ненулевыми, если содержат нечётное число переменных из A.)

Раз минимальное расстояние больше αn, то можно корректировать до $(\alpha/2)n$ ошибок. Покажем, что при $\varepsilon < 1/4$ это можно делать с помощью простого алгоритма: *пока есть ненулевые контрольные суммы, найдите переменную, изменение которой уменьшит их количество, и измените её.* Нужно только убедиться, что такая переменная всегда найдётся (то есть что мы не окажемся в «локальном минимуме»): после этого ясно, что число ненулевых сумм убывает, пока не достигнет нуля.

Итак, пусть нам дано некоторое слово z, которое отличается от кодового слова x в некоторых позициях. (Как и раньше, в силу линейности можно считать, что слово x нулевое.) Пусть A — соответствующее множество позиций. Мы уже видели, что из A выходит $D|A|$ рёбер, из которых в критические вершины ведёт не менее $(1-2\varepsilon)D|A|$, то есть больше половины, если $\varepsilon < 1/4$. Значит, в A найдётся переменная, из которой большинство рёбер ведут в критические контрольные суммы, и потому её изменение (которое исправит по крайней мере эти суммы за счёт возможной порчи других соседей) уменьшит число плохих сумм.

Всё? Не совсем. Дело в том, что и другие переменные, в том числе и вне A, могут обладать тем же свойством (их изменение уменьшает число нарушений), и изменение этих переменных удалит текущее слово от искомого, увеличив множество ошибок A. (Предотвратить это, не зная, где ошибки, мы не можем.) При этом число ненулевых контрольных сумм всё равно уменьшается, но если при таких изменениях множество A превысит размер αn, то с этого момента все наши рассуждения потеряют силу (может не найтись переменной, изменение которой уменьшает число плохих сумм; кроме того, при $|A| > \alpha n$ все контрольные суммы могут оказаться хорошими, хотя A и непусто).

Почему размер A не превысит αn? Вспомним, что в начале у нас было не более $(\alpha/2)n$ ошибок. Почему их число в ходе работы алгоритма увеличивается не более чем вдвое? Дело в том, что размер A связан с числом неверных сумм. Это число, как мы видели, не меньше $(1-2\varepsilon)D|A| > \frac{1}{2}D|A|$ и не больше $D|A|$ (общего числа соседей). Раз число неверных сумм убывает, а люфт в этом соотношении менее чем двукратный, размер A может возрасти не более чем вдвое. (Строго говоря, оценка $(1-2\varepsilon)D|A|$ для числа ненулевых контрольных сумм верна, лишь если доля ошибок не превзошла α. Но если до некоторого момента число ошибок оставалось меньше α, то и на следующем шаге это неравенство не нарушится. По индукции получаем, что число ошибок никогда не достигнет αn.)

Ясно, что алгоритм этот полиномиальный (число шагов не больше числа уравнений и тем самым числа переменных, и каждый шаг тоже полиномиальный). При дополнительном предположении о том, что степень всех вершин справа не превосходит $O(1)$ (см. ниже о том, почему такие экспандеры существуют) и достаточно умелой реализации алгоритм декодирования получается почти линейным. Действительно, можно хранить для каждой переменной число содержащих её ненулевых контрольных сумм; на каждом шаге надо выбирать наибольшее из этих чисел и затем корректировать соседей его соседей, что удобно делать с помощью очереди с приоритетами.

Известны также быстрые параллельные алгоритмы коррекции ошибок, но мы ограничимся уже сказанным.

Существование экспандеров

Итак, мы доказали, что на основе графа с n вершинами в левой доле и m вершинами в правой доле, в котором степень каждой вершины слева равна D и выполнено свойство экспандера с параметрами α и $\varepsilon < 1/4$, можно кодировать слова из $n-m$ битов словами из n битов и быстро исправлять до $(\alpha/2)n$ ошибок.

Остаётся доказать существование такого графа (при подходящем соотношении параметров). Зафиксируем произвольное рациональное $q < 1$ (отношение числа вершин в правой и левой долях) и произвольное $\varepsilon > 0$. Мы подберём подходящие D и α, при которых существует экспандер любого размера с такими параметрами (для любых целых m и n с отношением q). Пусть фиксированы количества вершин в левой и правой долях: числа n и m, для которых $m/n = q$. Будем выбирать двудольный граф с n вершинами слева и m вершинами справа случайно: из каждой левой вершины выпускаем D рёбер в правую долю (для каждой вершины случайно выбирается D разных соседей; при этом для разных вершин выбор делается независимо).

Мы хотим доказать, что свойство расширения выполнено с положительной вероятностью. Если оно не выполнено, то в левой доле графа найдётся такое множество A из $k \leqslant \alpha n$ вершин, что все соседи его вершин (в правой доле) лежат в некотором множестве B размера $(1-\varepsilon)Dk$. Для одного ребра вероятность попасть в такое B равна $|B|/m$, для D рёбер она не превосходит $(|B|/m)^D$ (даже немного меньше за счёт того, что все D соседей должны быть различными), и для разных вершин выборы независимы, так что надо возвести ещё в степень k. Полученный результат просуммируем по всем возможным A и B и получим оценку

$$\sum_{k=0}^{\alpha n} \left(C_n^k \cdot C_m^{(1-\varepsilon)Dk} \cdot \left(\frac{(1-\varepsilon)Dk}{m} \right)^{Dk} \right)$$

Нам нужно, чтобы это выражение было меньше 1. Воспользуемся оценкой для числа сочетаний: C_u^v не превосходит $(ue/v)^v$. (В самом деле, $C_u^v \leqslant u^v/v!$, и остаётся заметить, что $v! \geqslant (v/e)^v$, поскольку интеграл $\int_1^v \ln t\, dt$, равный $(t \ln t - t)|_{t=1}^{t=v}$, не превосходит своей верхней интегральной суммы $\ln 1 + \ln 2 + \ldots + \ln v = \ln(v!)$.)

Оценивая сверху биномиальные коэффициенты, получаем

$$\sum_{k=0}^{\alpha n} \left((ne/k)^k \cdot \left(\frac{me}{(1-\varepsilon)Dk} \right)^{(1-\varepsilon)Dk} \cdot \left(\frac{(1-\varepsilon)Dk}{m} \right)^{Dk} \right)$$

Если из второй скобки вынести наружу e, то она станет обратной к третьей, и можно привести подобные члены, получится

$$\sum_{k=0}^{\alpha n} \left[(ne/k) \cdot e^{(1-\varepsilon)D} \cdot \left(\frac{(1-\varepsilon)Dk}{m} \right)^{\varepsilon D} \right]^k$$

(мы заодно вынесли общую степень k). Заменяя $(1-\varepsilon)$ на единицу, мы лишь увеличим сумму:

$$\sum_{k=0}^{\alpha n} \left[(ne/k) \cdot e^{D} \cdot \left(\frac{Dk}{m} \right)^{\varepsilon D} \right]^k$$

Мы хотим оценить эту сумму как геометрическую прогрессию, но пока в основание входит k. Его мы заменим на верхнюю оценку αn.

Но для этого нужно, чтобы выражение в квадратных скобках росло с ростом k, то есть чтобы εD было больше 1. В этом предположении мы получаем (напомним, что $m/n = q$):

$$\sum_{k=0}^{\alpha n} \left[(e/\alpha) \cdot e^D \cdot \left(\frac{D\alpha}{q} \right)^{\varepsilon D} \right]^k$$

Теперь видно, как надо действовать: прежде всего мы выбираем D так, чтобы εD было больше 1. При таком D знаменатель геометрической прогрессии стремится к нулю при $\alpha \to 0$, и при достаточно малых α он будет меньше $1/2$, что и требуется.

Однородные справа экспандеры

Удобно иметь дело с экспандером, который регулярен не только слева, но и справа (все вершины справа имеют одинаковую степень). Это можно использовать при оценке времени работы алгоритма.

Снова зафиксируем произвольное рациональное $q < 1$ (отношение числа вершин в правой и левой долях) и произвольное $\varepsilon > 0$. Далее мы подберём такое целые D и вещественное $\alpha > 0$, что для всех достаточно больших целых m и n с отношением q существует двудольный граф с n вершинами слева (степени D) и m вершинами справа (степени $D' = D/q$), для которого выполнено нужное нам свойство расширения (с параметрами ε и α).

Итак, пусть фиксированы количества вершин в левой и правой долях: числа n и m, для которых $m/n = q$. Будем выбирать двудольный граф с n вершинами слева и m вершинами справа случайно по следующему распределению. Для каждой вершины слева возьмём D копий, а для каждой вершины справа — D' копий (это будут концы рёбер). Тогда слева и справа будет одно и то же число $N = Dn = D'm$ копий, и мы возьмём случайное взаимно однозначное соответствие между правыми и левыми концами (каждое из паросочетаний между левыми и правыми концами выбирается с вероятностью $1/N!$). Это ещё не окончательная конструкция, т.к. построенный нами граф может содержать кратные (параллельные) рёбра. Временно мы разрешим кратные рёбра и покажем, что в определённом выше графе свойство расширения выполнено с положительной (и даже большей $1/2$) вероятностью. Действительно, если нужное нам свойство не выполнено, то в левой доле графа найдётся такое множество A из $k \leqslant \alpha n$ вершин, что все соседи его вершин (в правой доле) лежат в некотором множестве B размера $(1-\varepsilon)Dk$. Оценим вероятность этого события для данных A и B.

На наше распределение вероятностей можно смотреть следующим образом. Можно считать, что сначала мы случайно выбираем Dk правых концов, куда будут «отправлены» рёбра из A (всего имеется C_N^{Dk} вариантов), затем распределяем все ребра, выходящие из A, между выбранными Dk позициями (для этого имеется $(Dk)!$ вариантов), после чего дополняем выбранные Dk рёбер до полного паросочетания (для этого есть $(N-Dk)!$ способов). Всего, как и положено, получается

$$C_N^{Dk} \cdot (Dk)! \cdot (N-Dk)! = N!$$

способов. (Хотя это представление зависит от множества A, получающееся в результате распределение одно и то же для всех A.)

Теперь ограничим свой выбор на первом шаге: будем выбирать множество соседей A не среди всех N потенциальных правых концов рёбер, а лишь среди (экземпляров) вершин множества B (таких экземпляров всего $D'|B| = (1-\varepsilon)DD'k$). При этом число вариантов сокращается с C_N^{Dk} до $C_{D'|B|}^{Dk} = C_{(1-\varepsilon)DD'k}^{Dk}$. Теперь понятно, что вероятность события «все соседи A лежат в множестве B» равна отношению биномиальных коэффициентов:

$$C_{(1-\varepsilon)DD'k}^{Dk} / C_N^{Dk}$$

Просуммируем эту оценку по всем возможным A и B. Получается, что вероятность того, что выбранный нами граф не является экспандером, не превосходит

$$\sum_{k=0}^{\alpha n} C_n^k \cdot C_m^{(1-\varepsilon)Dk} \cdot \left(C_{(1-\varepsilon)DD'k}^{Dk} / C_N^{Dk} \right)$$

Дальше рассуждаем почти также, как мы действовали при независимом выборе соседей каждой вершины. Мы хотим подобрать такие значения параметров, чтобы данная сумма была меньше $1/2$. Для начала воспользуемся оценкой для числа сочетаний: C_u^v не превосходит $(ue/v)^v$ и не меньше $(u/v)^v$. (Нижняя оценка получается, если в произведении $C_u^v = \frac{u}{v} \cdot \frac{u-1}{v-1} \cdot \ldots \cdot \frac{u-v+1}{1}$ все дроби заменить на u/v.)

Оценивая биномиальные коэффициенты и вынося общую степень k, получаем:

$$\sum_{k=0}^{\alpha n}\left(\left(\frac{ne}{k}\right)\cdot\left(\frac{me}{(1-\varepsilon)Dk}\right)^{(1-\varepsilon)D}\cdot\left(\frac{(1-\varepsilon)DD'ke}{N}\right)^{D}\right)^{k}$$

Вспоминаем, что $N = D'm$, мы видим, что по сравнению с предыдущим вычислением у нас появился множитель e в последней скобке, в результате чего $(1-\varepsilon)$ в показателе экспоненты заменяется на $(2-\varepsilon)$:

$$\sum_{k=0}^{\alpha n}\left((ne/k)\cdot e^{(2-\varepsilon)D}\cdot\left(\frac{(1-\varepsilon)Dk}{m}\right)^{\varepsilon D}\right)^{k}$$

Дальше всё происходит по прежней схеме. Мы хотим, чтобы выражение в скобках росло с ростом k. Для этого нужно, чтобы εD было больше 1. Выбрав (при заданном ε) такое D, можно заменить k внутри скобок на αn, от чего сумма только увеличится. Также огрубим $(2-\varepsilon)$ и $(1-\varepsilon)$ до 2 и 1 соответственно:

$$\sum_{k=0}^{\alpha n}\left((e/\alpha)\cdot e^{2D}\cdot\left(\frac{D\alpha}{q}\right)^{\varepsilon D}\right)^{k}$$

Теперь ряд превратился в геометрическую прогрессию, причём её знаменатель стремится к нулю при $\alpha\to 0$. Следовательно, при достаточно малых α он будет меньше $1/4$, и тогда сумма ряда не превосходит $1/2$. Подведём промежуточный итог: с вероятностью не меньше $1/2$ случайно выбранное паросочетание даёт нам регулярный двудольный граф, возможно, с кратными рёбрами, обладающий свойством расширения.

Устранение кратных рёбер потребует некоторого запаса в свойстве расширения. Для этого мы заменяем в приведённом выше вероятностном рассуждении параметр ε на $\varepsilon' = \varepsilon/2$. (Также запомним, что в этом случае степень D мы выбираем не меньше $1/\varepsilon' = 2/\varepsilon$.)

Как устранить кратные рёбра? Из каждой вершины в левой доле графа выходит по D рёбер. Будем считать, что рёбра занумерованы по их левым концам (от 1 до $N = Dn$). Нетрудно проверить, что для любых двух различных i, j (в частности, соответствующих одной левой вершине — в этом случае возможно образование кратных рёбер) вероятность того, что их правые концы попадут в одну и ту же вершину справа, не превосходит $1/m$. Суммируя эту вероятность по всем вершинам левой доли и по всем парам рёбер, выходящим из одной вершины, получаем, что математическое ожидание числа пар кратных рёбер в графе не превосходит $C_D^2 \cdot (n/m) = O(D^2/q)$. Вероятность того, что положительная величина более чем в два раза превышает своё математическое ожидание, должна быть меньше $1/2$ (неравенство Чебышёва). Следовательно, с вероятностью больше $1/2$ в случайно выбранном графе будет не больше $O(D^2/q)$ пар кратных рёбер.

Итак, с вероятностью не меньше $1/2$ случайное паросочетание даёт нам экспандер (с нужными параметрами), и в то же время с вероятностью больше $1/2$ в случайно выбранном графе число кратных рёбер ограничено $O(D^2/q)$. Это значит, что с положительной вероятностью для случайного графа выполняются оба свойства. Таким образом, существует регулярный (слева и справа) экспандер, в котором не больше $O(D^2/q)$ пар кратных рёбер. Заметим, что величина $O(D^2/q)$ не зависит от числа вершин в графе. Если число вершин n достаточно велико по сравнению с D, в таком графе можно «распараллелить» кратные рёбра, лишь незначительно ухудшив свойство расширения.

Опишем более подробно процедуру «распараллеливания». Назовём «запрещёнными» сами кратные рёбра, а также рёбра, имеющие с ними общую вершину (слева или справа). Пока в графе остаются кратные рёбра, будем производить операцию переклейки: берём одно из имеющихся кратных рёбер и обмениваем его правый конец с правым концом какого-нибудь незапрещенного ребра. При этом мы позаботимся, чтобы все незапрещённые рёбра, использованные в переклейках, не имели общих концов (это можно обеспечить, если n много больше числа кратных рёбер).

Понятно, что после этой процедуры граф остался регулярным, и в нём не осталось кратных рёбер. Посмотрим, насколько могло измениться свойство расширения графа. Раньше для всякого множества левых вершин A множество соседей состояло не менее чем из $(1-\varepsilon')D|A|$ правых вершин. После переклейки рёбер множество $N(A)$ могло стать меньше за счёт незапрещённых рёбер (выходящих из A), использованных в процедуре распараллеливания. Таких рёбер заведомо не больше $|A|$ (левые концы всех этих рёбер различны). Это значит, что в новом графе число соседей A не меньше

$$(1-\varepsilon')D|A| - |A| \geqslant (1-\varepsilon' - \frac{1}{D})D|A|$$

Таким образом, для нового графа свойство расширения выполняется со слегка ухудшенными параметрами: ε' заменяется на $(\varepsilon' + \frac{1}{D})$. За счёт предусмотренного запаса ($\varepsilon' = \varepsilon/2$) мы получаем экспандер с параметром расширения ε.

Таким образом, мы доказали, что при определённых значениях параметров (при любом $\varepsilon > 0$, при фиксированном отношении n/m, достаточно больших D и достаточно малых α) экспандеры существуют (и, более того, существуют экспандеры с одинаковой степенью всех вершин в правой доле). Однако наше доказательство неявное — оно не даёт возможности построить экспандер с заданными параметрами достаточно быстро (а это необходимо, чтобы организовать быстрое кодирование и декодирование для соответствующего кода равномерно по длинам кодовых слов). В 2002 году Вадхан, Вигдерсон, Капалбо и Рейнголд [M. Capalbo, O. Reingold, S. Vadhan, A. Wigderson, Randomness conductors and constant-degree lossless expanders, *Proceedings on 34th Annual ACM Symposium on Theory of Computing*, pp. 659–668, 2002] придумали явную конструкцию, которая для сколь угодно малого параметра расширения $\varepsilon > 0$ позволяет строить экспандеры с нужным свойством за полиномиальное (от числа вершин) время.

Параметры экспандерных кодов

Мы доказали существование экспандерных кодов и объяснили, что для них есть быстрые алгоритмы декодирования. Однако до сих пор мы не обсуждали качество таких кодов — как соотносятся их коэффициент полезного действия (скорость) и кодовое расстояние (а значит и число исправляемых ошибок). Сейчас мы оценим скорость экспандерного кода, исправлящего заданную долю ошибок.

Прежде всего нужно понять, как связаны параметры экспандера и характеристики соответствующего ему кода. Напомним, какие параметры есть у экспандера:

- число вершин слева и справа (n и m соответственно; мы считаем, что $m < n$, и обозначаем отношение этих чисел $q = m/n$);

- степень вершин слева (обозначется D);
- параметры расширения α и ε: всякое множество A, состоящее из не более чем αn вершин левой доли, должно иметь не менее $(1-\varepsilon)D|A|$ соседей (в правой доле). При построении кодов мы предполагали, что $\varepsilon < 1/4$.

Вспомним, какой код соответствует такому графу. Поскольку длина кодового слова равна n, а число контрольных сумм равно m, то число кодовых слов не меньше 2^{n-m}. Таким образом, коэффициент полезного действия кода заведомо не меньше $\frac{n-m}{n} = 1-q$. Далее, мы доказали, что расстояние экспандерного кода будет не меньше αn. Мы хотим, чтобы кодовое расстояние кода (при данной скорости) было как можно больше. Это значит, что при каждом значении q мы хотим максимизировать соответствующее ему α.

Теперь можно точно сформулировать задачу в терминах параметров экспандера. Мы фиксируем ε (некоторое число меньше $1/4$) и q (некоторое число меньше 1). Требуется найти максимальное $\alpha = \alpha(q, \varepsilon)$ (а также подходящее значение $D = D(q, \varepsilon)$), для которого существует экспандер с указанными параметрами.

Ранее мы доказали, что для существования экспандеров достаточно, чтобы выполнялись условия $\varepsilon D > 1$ и

$$(e/\alpha) \cdot e^D \cdot \left(\frac{D\alpha}{q}\right)^{\varepsilon D} < 1/2$$

Это значит, что нам нужно найти максимальное $\alpha = \alpha(q, \varepsilon)$ такое, что для некоторого $D = D(q, \varepsilon)$ эти неравенства верны. Точное решение этой задачи на поиск максимума довольно громоздко. Однако нетрудно подобрать значение D, которое будет достаточно близко к макисмуму (для нашего исследования экспандерного кода этого приближения окажется вполне достаточно). Достаточно взять, например, $\alpha = \frac{C(\varepsilon)q}{\log(3/q)}$ (для некоторого достаточно малого коэффициента $C = C(\varepsilon)$). Прямая подстановка показывает, что при данном α и при $D = \frac{1}{\varepsilon}\log(3/q)$ оба нужные нам неравенства выполняются. В самом деле, при выбранном D неравенство $\varepsilon D > 1$ очевидно. Чтобы убедиться, что и второе неравенство верное, его удобно прологарифмировать (по основанию 2). При выбранных α и D получаем

$$\log\left(e/\frac{C(\varepsilon)q}{\log(3/q)}\right) + \frac{1}{\varepsilon}\log(3/q)\log e + \log\left(C(\varepsilon)/\varepsilon\right) \cdot \log(3/q) < -1$$

что можно переписать в виде

$$\log\left(e/C(\varepsilon)\right) - \log\frac{1}{q} + \log\log\frac{3}{q} + \log\frac{3}{q}\left(\frac{1}{\varepsilon}\log e + \log\frac{C(\varepsilon)}{\varepsilon}\right) < -1$$

После сделанных преобразований видно, что нужное нам неравенство выполнено при достаточно малом $C(\varepsilon)$.

Для дальнейшего нам нужно запомнить только асимптотику полученного ответа: при фиксированном ε существует экспандер с параметром расширения $\alpha = O(q/\log(1/q))$.

Таким образом, мы выразили достижимое кодовое расстояние через q: для произвольного $q > 0$ можно построить экспандерный код со скоростью не менее $(1-q)$, исправляющий долю ошибок $\delta(q) = \Omega(q/\log 1/q)$. Полезно переписать это соотношение, выразив наоборот q через δ. Получается, что мы можем исправлять долю ошибок δ с помощью кода, скорость которого меньше $(1-q)$, где $q(\delta) = O(\delta\log 1/\delta)$.

Сравним полученную оценку на скорость экспандерного кода с оценкой Хэмминга. Напомним границу Хэмминга: всякий двоичный код, который позволяет исправлять ошибки в доле позиций δ, имеет скорость не больше $(1 - H(\delta))$, где $H(\delta) = \delta\log 1/\delta + (1-\delta)\log 1/(1-\delta)$. При $\delta \to 0$ мы имеем $H(\delta) = \delta\log 1/\delta + O(\delta)$. Таким образом, для малых δ экспандерный код требует не более чем в константу раз большей потери скорости по сравнению с (теоретически возможными) оптимальными кодами. Отметим, что экспандерные коды дают один из лучших известных компромисов между скоростью кода, кодовым расстоянием и быстротой декодирования.

29. Сложность декодирования

В предыдущем разделе был указан полиномиальный алгоритм декодирования для линейных кодов специального вида (LDPC). Возникает вопрос: а возможно ли полиномиальный алгоритм декодирования (работающий полиномиальное время от длины кодового слова) для *произвольных* линейных кодов? (Мы ограничиваемся линейными кодами, так как в общем случае уже задание кода как множества кодовых слов имеет экспоненциальный размер.)

В наиболее естественной постановке этот вопрос остаётся открытым, но имеется следующий частичный отрицательный результат: *задача о нахождении ближайшего кодового слова является* NP-*трудной.* (В этом разделе мы предполагаем знакомство читателя с начальными сведениями о классе NP.)

Множество кодовых слов линейного кода над полем из двух элементов (подпространство в $\mathbb{B}^n$) можно задавать как базисом в нём, так и уравнениями, выделяющими его из всего пространства, и от одного задания можно перейти к другому за полиномиальное время (метод Гаусса). Поэтому, говоря о подпространстве, можно не уточнять, каким из двух способов оно задано.

Рассмотрим такую задачу:

Даны: *n-битовое слово $X = x_1 \ldots x_n$, подпространство $L \subset \mathbb{B}^n$ и число k.*

Надо: *узнать, можно ли так изменить не более k битов в слове x, чтобы оно после этого попало в L.*

Эта задача очевидно принадлежит классу NP: если ответ положительный, то для доказательства этого достаточно указать изменяемые биты.

Теорема. Эта задача является NP-полной.

Из этой теоремы (её доказали Берлекэмп, МакЭлиес и ван Тилборг в 1978 году, вскоре после появления понятия NP-полноты) следует, что и более общая задача нахождения ближайшего элемента подпространства (кодового слова) является NP-трудной и вряд ли можно ожидать появления полиномиального алгоритма, её решающего. (Впрочем, как мы уже говорили, с точки зрения теории кодирования это мало что значит: нас интересуют не все подпространства, а только с достаточно большим расстоянием между точками, и декодировать мы хотим лишь слова, близкие к одной из точек.)

Для доказательства заметим, что к этой задаче сводится *задача о максимальном разрезе* (MAX-CUT): разбить вершины данного графа на два класса, чтобы число рёбер между вершинами разных классов было максимально. Точнее, поскольку мы говорим о задачах разрешения, надо сказать так: *дан граф и число m*; *выяснить, можно ли разбить вершины графа на два класса так, чтобы рёбер с концами из разных классов было не меньше m.*

Сведение совсем простое. Сопоставим графу такой код: биты кодового слова сопоставляются рёбрам графа; всякой раскраске вершин графа в два цвета (одни вершины красятся в чёрный цвет, другие — в белый) соответствует такое кодовое слово: пишем единицы на тех рёбрах, концы которых покрашены в разные цвета и нули на рёбрах с одноцветными концами. Нетрудно проверить, что получился линейный код. При этом наибольший разрез в графе соответствет кодовому слову, которое ближе всего к слову из одних единиц. (Более формально, разрез с m или более разноцветными рёбрами существует тогда и только тогда, когда в слове из одних единиц можно изменить не более $E - m$ битов, получив кодовое слово; здесь E — число рёбер графа.)

Остаётся установить, что задача MAX-CUT (в описанном варианте с ответом «да/нет») является NP-полной. Покажем, как к ней свести задачу 3-SAT (стандартную при доказательствах NP-полноты; SAT означает SATisfiablility, выполнимость).

В задаче 3-SAT требуется найти набор булевских значений $x_1, \ldots, x_n$, удовлетворяющий некоторым условиям. Каждое условие относится к каким-то трём переменных и запрещает одну комбинацию их значений. Это обычно записывают в виде «дизъюнкта»: запрет комбинации (к примеру) $x_5 = 1$, $x_7 = 0$, $x_{11} = 1$ записывается как $\neg x_5 \vee x_7 \vee \neg x_{11}$. Задача в целом тогда становится конъюнкцией (логическим И) этих дизъюнктов. Будем предполагать NP-полноту этой задачи известной.

В качестве леммы докажем NP-полноту частного случая этой задачи, в котором запрещения должны быть симметричными: запрещая какую-то тройку значений (скажем, $1, 0, 1$), мы должны запретить и противоположную тройку $(0, 1, 0)$ для тех же переменных. Если ввести обозначение $\mathrm{NotAllEqual}(a, b, c)$, означающее, что не все три бита a, b, c одинаковы, то можно переформулировать этот частный случай так: вместо дизъюнкций $a \vee b \vee c$ трёх литералов (переменных или их отрицаний) мы пишем $\mathrm{NotAllEqual}(a, b, c)$. При этом запрещается не только случай, когда они все ложны (как в дизъюнкции), но и случай, когда они все истинны. Этот частный случай иногда называют NAESAT (Not All Equal SATisfiablity).

Лемма. Общая задача 3-SAT сводится к NAESAT.

Прежде всего заметим, что решения задачи NAESAT выдерживают симметрию (замену всех значений на противоположные). Поэтому можно выбрать какую-то одну «спецпеременную» и наложить дополнительное условие: мы рассматриваем только наборы значений, где спецпеременная ложна (равна нулю).

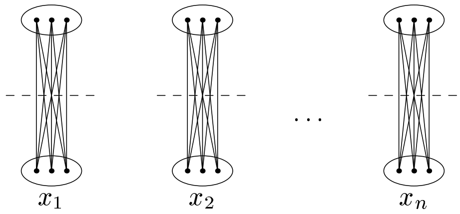
Заметим, что $\mathrm{NotAllEqual}(0, a, b) = a \vee b$, так что дизъюнкцию двух (обычных) переменных u и v можно выразить как $\mathrm{NotAllEqual}(\alpha, u, v)$, где α — спецпеременная. Условие $a \vee b \vee c$ (запрещение всем литералам a, b, c быть ложными) можно заменить двумя условиями $a \vee b = t$ и $t \vee c$, где t — вспомогательная новая переменная (своя для каждой дизъюнкции): новая система условий выполнима тогда и только тогда, когда выполнима старая. Остаётся выразить условие $a \vee b = t$, то есть запретить вариант $a = 1, t = 0$, вариант $b = 1, t = 0$ и вариант $a = 0, b = 0, t = 1$. Первые два соответствуют дизъюнкциям $\neg a \vee t$ и $\neg b \vee t$, в которых только две переменные (и это мы выражать умеем). В последнем случае беды не будет, если мы запретим и противоположный вариант $a = 1, b = 1, t = 0$ (он тоже не подходит и уже запрещён), написав $\mathrm{NotAllEqual}(a, b, \neg t)$. Лемма доказана.

Осталось свести задачу NAESAT к MAX-CUT. Для этого нам надо представить значения переменных как варианты разбиений вершин графа. Для начала рассмотрим граф из двух групп по N вершин и всех рёбер между группами. Максимальный разрез тут пересекает все N^2 рёбер. Любой другой разрез проигрывает ему по крайней мере N рёбер.

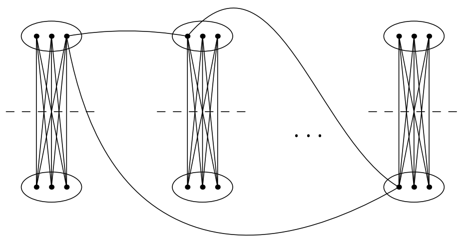
(Проще всего в этом убедиться с помощью такого трюка. Разрез — это разделение всех врешин на «белые» и «чёрные», величина разреза — это число рёбер с концами разных цветов. Рассмотрим произведение $(a_1 + \ldots + a_N)(b_1 + \ldots + b_N)$, в котором все a_i и b_i равны ± 1. Значения a_i соответствуют цветам вершин в одной группе, значения b_i — в другой. При этом для a_i знак «+»соответствует вершинвм белогоцвета, а для b_i — наоборот. Раскрывая скобки, убеждаемся, что это произведение равно числу разноцветных рёбер минус число одноцветных. Когда все a_i и b_j равны 1, это произведение максимально и состоит из N^2 членов $a_i b_j$, равных единице и соответствующих N^2 рёбрам между долями. Замена единицы на минус единицу соответствует перекрашиванию вершины, при этом пропавшее ребро даёт уменьшение суммы на 2. Поскольку все варианты, кроме $a_i = b_j = 1$ и $a_i = b_j = -1$, дают произведение не больше $(N-2)N$, видно, что пропадают по крайней мере N рёбер.)

Теперь соединим n копий такого графа. В этом графе есть 2^n максимальных разрезов: в каждой группе надо разрезать все рёбра, но какую из половин поместить по одну сторону разреза, а какую по другую, можно выбирать. Выбор одного из таких разрезов естественно кодируется последовательностью из n битов. Пока что все 2^n вариантов разреза одинаково хороши (разрезают все рёбра).

Это уже не так, если мы добавим дополнительные рёбра. Пусть, скажем, добавлен треугольник, соединяющий три вершины из раз-



ных копий. Из них разрезаны либо 0 (если все три вершины треугольника оказались с одной стороны разреза), либо 2 ребра. Таким образом, такой треугольник даёт премию в +2 ребра, если не все вершины с одной стороны (что как раз и соответствует соответствует NotAllEqual для битов из кода разреза).



Таким образом, вопрос о выполнимости набора NAESAT-ограничений сводится к вопросу о наличии разреза из $N^2 + 2m$ рёбер, где m — количество ограничений. Надо только выбрать N больше $3m$, чтобы никакой выигрыш за счёт раз дополнительных рёбер не перевесил проигрыш, если разрез не разделит группы в одной из копий. (На самом деле достаточно потребовать $N > m$, поскольку в каждом треугольнике мы можем выиграть только один разрез, получив 3 вместо 2.)

Оглавление