Об одном алгоритме ML-декодирования для низкоплотностных кодов

Д. Цымжитов

В работе представлен алгоритм мягкого декодирования для низкоплотностных двоичных кодов с графом Таннера без циклов. Показано, что этот алгоритм находит оптимальное решение задачи декодирования по максимальному правдоподобию и имеет асимптотическую сложность $O(n)$, где n — длина кодового слова.

1. Введение

Декодирование по максимальному правдоподобию, или, как еще говорят, ML-декодирование (*англ.* maximum-likelihood decoding), является важнейшей алгоритмической проблемой в теории кодирования. Это объясняется тем, что именно ML-декодирование позволяет свести к минимуму вероятность появления ошибок в принятой последовательности символов, если вероятности передачи для всех кодовых последовательностей одинаковы. Для случая двоичных линейных кодов и двоичного канала связи с аддитивным белым гауссовым шумом (гауссовского канала) эту задачу можно сформулировать так.

Пусть $\mathcal{C} \subset \mathbb{F}_2^n$ — двоичный линейный код длины n. Пусть на выходе канала связи принята некоторая последовательность $\mathbf{r} \in \mathbb{R}^n$. Требуется найти такой вектор $\mathbf{c} \in \mathcal{C}$, что вероятность $P(\mathbf{c} \,|\, \mathbf{r})$ (вероятность того, что было передано кодовое слово $\mathbf{c}$, при условии, что принята последовательность $\mathbf{r}$) максимальна. Если предположить, что вероятности передачи всех кодовых слов одинаковы, то задача максимизации условной вероятности $P(\mathbf{c} \,|\, \mathbf{r})$ сводится к задаче максимизации функции плотности условной вероятности $p(\mathbf{r} \,|\, \mathbf{c})$ (вероятности

того, что при передаче кодового слова $\mathbf{c}$ будет принята последовательность $\mathbf{r}$). Собственно нахождение вектора $\mathbf{c}$, максимизирующего величину $p(\mathbf{r} \mid \mathbf{c})$, и составляет суть задачи ML-декодирования.

В известной работе Берлекэмпа и др. [1] было показано, что задача ML-декодирования для произвольных линейных кодов, даже при использовании двоичного симметричного канала связи (более простого по сравнению с гауссовским каналом), является NP-трудной. Более того, эта задача остается NP-трудной даже в том случае, когда допускается сколь угодно длительная предобработка кода [2].

Настоящая работа посвящена решению задачи ML-декодирования в предположении, что для передачи данных используется двоичный гауссовский канал связи, а вместо произвольных двоичных линейных кодов используются так называемые низкоплотностные коды, или, как еще говорят, LDPC-коды (*англ.* low-density parity-check codes). Особенностью этого класса линейных кодов является то, что их проверочные матрицы содержат очень мало ненулевых элементов. Как известно, их основным преимуществом являются высокая исправляющая способность и простота декодирования. Еще Галлагером было показано [3], что у случайных LDPC-кодов с ростом длины кода возрастает и минимальное кодовое расстояние, если максимальное число ненулевых элементов в строках и столбцах проверочной матрицы остается ограниченным сверху (то есть если проверочная матрица становится все более разреженной). Кроме того, для случайных LDPC-кодов доказано [4], что они позволяют вплотную приблизиться к границе Шеннона.

В этой статье мы предложим алгоритм ML-декодирования для достаточно узкого класса LDPC-кодов, характеризуемых тем, что граф Таннера их проверочной матрицы не содержит циклов. В связи с этим, мы хотели бы обратить внимание на два обстоятельства. Во-первых, известно [5], что эти коды обладают очень слабой исправляющей способностью: если скорость такого кода не меньше $1/2$, то его минимальное расстояние не превосходит 2. Таким образом, данное исследование и его результаты носят в бóльшей степени теоретический, нежели практический, характер. Во-вторых, уже достаточно давно известны алгоритмы, имеющие линейную по длине кодового слова сложность и способные находить ML-решение для кодов с ацик-

лическим графом Таннера. Таковым, например, является алгоритм «min-sum» [6], с которым наш алгоритм декодирования имеет много общего.

А именно, пусть $\mathbf{r} \in \mathbb{R}^n$ — мягкая последовательность, принятая на выходе канала связи, и $\hat{\mathbf{r}} \in \mathbb{F}_2^n$ — соответствующая ей жесткая последовательность битов. Предлагаемый нами алгоритм декодирования получает на вход вектор начального приближения $\mathbf{e}^{(0)} \in \hat{\mathbf{r}}+\mathcal{C}$ и строит в $\hat{\mathbf{r}}+\mathcal{C}$ последовательность векторов $\mathbf{e}^{(1)}, \mathbf{e}^{(2)}, \ldots$, сходящуюся к оптимальному вектору ошибки $\mathbf{e} \in \hat{\mathbf{r}} + \mathcal{C}$, так что кодовый вектор $\mathbf{c} = \hat{\mathbf{r}} - \mathbf{e}$ является ML-решением. Каждая итерация алгоритма строит очередной вектор $\mathbf{e}^{(t)}$, основываясь на предыдущем векторе $\mathbf{e}^{(t-1)}$, и выполняется в два этапа. Первый этап, являющийся подготовительным, вычисляет значения специальных переменных, приписанных ребрам графа Таннера, используя технику обмена сообщениями (*англ.* message-passing) между его вершинами. Этот этап практически идентичен аналогичной процедуре алгоритма «min-sum». Второй этап использует технику последовательного изменения символов вектора $\mathbf{e}^{(t-1)}$, которая позволяет получить новый вектор $\mathbf{e}^{(t)}$, расположенный еще ближе к оптимальному значению вектора ошибки.

Алгоритм «min-sum», напротив, обращается к процедуре обмена сообщениями только один раз и на основании вычисленных данных сразу получает оптимальное решение задачи ML-декодирования. Таким образом, представленный нами алгоритм декодирования отличается от алгоритма «min-sum» тем, что в его основе лежит иная стратегия поиска оптимального решения, которая, как мы думаем, может представлять определенный теоретический интерес.

2. Постановка задачи и формулировка результата

Более детальное изложение понятий и фактов, составляющих содержание настоящего раздела, а также приложения 8, приведенного в конце данной статьи, можно найти в [7, гл. 1 и 11].

Пусть $\mathcal{C} \subset \mathbb{F}_2^n$ — двоичный линейный код размерности $k = n - m$ с проверочной матрицей $\mathbf{H} = (h_{ij})$ размера $m \times n$. Скорость кода $\mathcal{C}$ рав-

на $R = k/n < 1$. Будем считать, что код $\mathcal{C}$ используется при передаче по каналу связи с аддитивным белым гауссовым шумом, имеющим распределение $\mathcal{N}(0, \sigma^2)$. Если для передачи одного информационного бита тратится энергия E_b, то для передачи одного кодированного бита тратится энергия $E_c = RE_b$. Тогда амплитуда сигнала, представляющего один кодированный бит, равна $a = \sqrt{E_c}$, а надежность канала связи, по определению, равна $L_c = 2a/\sigma^2$.

Для передачи по каналу связи кодовое слово $\mathbf{c} \in \mathcal{C}$ преобразуется в вектор $\tilde{\mathbf{c}} \in \mathbb{R}^n$ посредством отображения модуляции $\mu\colon \mathbb{F}_2^n \to \mathbb{R}^n$, так что

$$\mu\colon (c_j) \mapsto (\tilde{c}_j), \quad \tilde{c}_j = \begin{cases} -a, & \text{если } c_j = 0, \\ a, & \text{если } c_j = 1. \end{cases}$$

Всюду далее мы будем писать $\tilde{\mathbf{c}}$, имея ввиду образ $\mu(\mathbf{c})$ вектора $\mathbf{c}$ при отображении модуляции.

При прохождении вектора $\tilde{\mathbf{c}} = (\tilde{c}_j)$ по каналу связи, к нему прибавляется случайный вектор шума, все компоненты которого независимы и имеют распределение $\mathcal{N}(0, \sigma^2)$. Следовательно, на приемном конце будет принят некоторый вектор $\mathbf{r} \in \mathbb{R}^n$, $\mathbf{r} = (r_j)$, компоненты которого также имеют нормальное распределение условных вероятностей

$$p(r_j \,|\, \tilde{c}_j) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{1}{2\sigma^2}(r_j - \tilde{c}_j)^2\right).$$

Таким образом, плотность условной вероятности того, что при передаче кодового слова $\mathbf{c}$ будет принят вектор $\mathbf{r}$, равна

$$p(\mathbf{r} \,|\, \mathbf{c}) = \prod_{j=1}^{n} p(r_j \,|\, \tilde{c}_j) = \frac{1}{(\sigma\sqrt{2\pi})^n} \exp\left(-\frac{1}{2\sigma^2}||\mathbf{r} - \tilde{\mathbf{c}}||^2\right), \qquad (1)$$

где $||\mathbf{r} - \tilde{\mathbf{c}}||$ — евклидово расстояние между векторами $\mathbf{r}$ и $\tilde{\mathbf{c}}$. Более того, имеет место равенство

$$p(\mathbf{r} \,|\, \mathbf{c}) = \alpha(\mathbf{r}) \exp\left(-L_c \sum_{j\colon r_j\tilde{c}_j<0} |r_j|\right), \qquad (2)$$

где $\alpha(\mathbf{r}) > 0$ (см. доказательство в приложении 8).

Пусть на выходе канала связи принят некоторый вектор $\mathbf{r} \in \mathbb{R}^n$. Нетрудно убедиться (см. [7, п. 1.5.2]), что способ декодирования, минимизирующий вероятность ошибки декодирования, состоит в поиске кодового вектора $\mathbf{c} \in \mathcal{C}$, максимизирующего условную вероятность $P(\mathbf{c} \,|\, \mathbf{r})$. Задачу поиска такого вектора $\mathbf{c}$, называют задачей декодирования *по максимуму апостериорной вероятности*, или задачей *MAP-декодирования* (*англ.* maximum a posteriori).

Легко видеть (см. приложение 8), что если вероятности передачи всех кодовых слов одинаковы, то задача MAP-декодирования сводится к задаче ML-декодирования, которая состоит в нахождении такого вектора $\mathbf{c} \in \mathcal{C}$, на котором функция плотности условной вероятности $p(\mathbf{r} \,|\, \mathbf{c})$ достигает максимума:

$$p(\mathbf{r} \,|\, \mathbf{c}) \to \max, \quad \mathbf{c} \in \mathcal{C}. \tag{3}$$

Преобразуем вектор $\mathbf{r} \in \mathbb{R}^n$ в двоичный вектор $\hat{\mathbf{r}} \in \mathbb{F}_2^n$ посредством отображения демодуляции $\mu^* \colon \mathbb{R}^n \to \mathbb{F}_2^n$, так что

$$\mu^* \colon (r_j) \mapsto (\hat{r}_j), \quad \hat{r}_j = \begin{cases} 0, & \text{если } r_j \leqslant 0, \\ 1, & \text{если } r_j > 0. \end{cases}$$

Всюду далее мы будем писать $\hat{\mathbf{r}}$, имея ввиду образ $\mu^*(\mathbf{r})$ вектора $\mathbf{r}$ при отображении демодуляции.

При решении задачи (3) мы будем пользоваться понятиями, являющимися прямым обобщением понятий расстояния и веса Хэмминга. Дадим их определения. Положим $\lambda \in \mathbb{R}^n$, $\lambda = (\lambda_j)$, где $\lambda_j = |r_j| \geqslant 0$ для всех $j = 1, \ldots, n$.

Пусть $\mathbf{a}, \mathbf{b} \in \mathbb{F}_2^n$, $\mathbf{a} = (a_j)$ и $\mathbf{b} = (b_j)$. *Обобщенным расстоянием* между $\mathbf{a}$ и $\mathbf{b}$ относительно λ назовем величину

$$d_\lambda(\mathbf{a}, \mathbf{b}) = \sum_{j \colon a_j \neq b_j} \lambda_j.$$

Пусть $\mathbf{a} \in \mathbb{F}_2^n$, $\mathbf{a} = (a_j)$. *Обобщенным весом* вектора $\mathbf{a}$ относительно λ назовем величину

$$w_\lambda(\mathbf{a}) = \sum_{j \colon a_j \neq 0} \lambda_j.$$

Для любых $\mathbf{a}, \mathbf{b} \in \mathbb{F}_2^n$ имеем $d_\lambda(\mathbf{a}, \mathbf{b}) = w_\lambda(\mathbf{a} - \mathbf{b})$ и $w_\lambda(\mathbf{a}) = d_\lambda(0, \mathbf{a})$.

Замечание. Обобщенное расстояние, вообще говоря, не всегда является метрикой в строгом смысле. Несмотря на то, что расстояние d_λ симметрично и удовлетворяет неравенству треугольника, оно может оказаться вырожденным, если хотя бы одна из компонент вектора λ равна нулю.

С учетом данных определений и обозначений выражение (2) принимает вид

$$p(\mathbf{r} \mid \mathbf{c}) = \alpha(\mathbf{r}) \exp\left(-L_c d_\lambda(\hat{\mathbf{r}}, \mathbf{c})\right).$$

Следовательно, задача (3) принимает вид

$$d_\lambda(\hat{\mathbf{r}}, \mathbf{c}) \to \min, \quad \mathbf{c} \in \mathcal{C}. \tag{4}$$

В терминах вектора ошибки $\mathbf{e} = \hat{\mathbf{r}} - \mathbf{c}$ задачу (4) можно переписать в виде

$$w_\lambda(\mathbf{e}) \to \min, \quad \mathbf{e} \in \hat{\mathbf{r}} + \mathcal{C}. \tag{5}$$

При этом, если $\mathbf{s} = (s_i) = \mathbf{H}\hat{\mathbf{r}}$ — синдром вектора $\hat{\mathbf{r}}$, то условие $\mathbf{e} \in \hat{\mathbf{r}} + \mathcal{C}$ можно представить в виде равенства $\mathbf{He} = \mathbf{s}$ или системы

$$\sum_{j=1}^{n} h_{ij} e_j = s_i, \quad i = 1, \ldots, m.$$

Все дальнейшие наши рассуждения будут проводиться в контексте задачи (5). Поэтому далее мы будем считать, что вектор $\mathbf{r} \in \mathbb{R}^n$ фиксирован, равно как и производные от него векторы $\hat{\mathbf{r}}$, $\mathbf{s}$ и λ.

Как уже говорилось выше, предлагаемый нами общий подход к решению задачи (5) состоит в том, чтобы, выбрав некоторое начальное приближение $\mathbf{e}^{(0)} \in \hat{\mathbf{r}} + \mathcal{C}$, построить в $\hat{\mathbf{r}} + \mathcal{C}$ последовательность векторов $\mathbf{e}^{(1)}, \ldots, \mathbf{e}^{(T)}$, так что

$$w_\lambda(\mathbf{e}^{(t-1)}) > w_\lambda(\mathbf{e}^{(t)}) \quad \text{при всех} \quad t = 1, \ldots, T, \tag{6}$$

и вектор $\mathbf{e}^{(T)}$ — оптимальное решение задачи.

Графом Таннера матрицы $\mathbf{H} = (h_{ij})$ размера $m \times n$ является двудольный граф, содержащий две доли по m и n вершин, соответствующих соответственно строкам и столбцам матрицы, причем вершина, соответствующая i-й строке, соединяется ребро с вершиной, соответствующей j-му столбцу, точно тогда, когда $h_{ij} = 1$.

Теорема 1. *Если граф Таннера* $\mathfrak{T}$ *матрицы* $\mathbf{H}$ *размера* $m \times n$ *не содержит циклов, то существует итеративный алгоритм ML-декодирования, который строит последовательность приближений (6), такой, что построенная последовательность (6) сходится к оптимальному решению задачи (5) вне зависимости от начального приближения* $\mathbf{e}^{(0)} \in \hat{\mathbf{r}}+\mathcal{C}$. *При этом если максимальная степень вершин графа* $\mathfrak{T}$ *лежит в классе* $O(1)$ *при* $n \to \infty$, *то сложность данного алгоритма лежит в классе* $O(n)$ *при* $n \to \infty$.

3. Граф Таннера

Основной конструкцией, с которой мы будем иметь дело, является *граф Таннера* $\mathfrak{T}$ матрицы $\mathbf{H}$. Напомним, что это двудольный граф, который содержит вершины двух видов — проверочные и символьные. *Проверочные вершины*, соответствующие строкам матрицы $\mathbf{H}$, будем обозначать z_i, $i = 1, \ldots, m$. *Символьные вершины*, соответствующие столбцам матрицы $\mathbf{H}$, будем обозначать x_j, $j = 1, \ldots, n$. Ребра соединяют только те пары вершин, в которых одна является проверочной, а другая — символьной. Две вершины z_i и x_j соединяются ребром тогда и только тогда, когда $h_{ij} = 1$.

Введем некоторые обозначения для индексных множеств, которые нам понадобятся. Положим

$$I = \{i \in \mathbb{Z} \mid 1 \leqslant i \leqslant m\}, \quad J = \{j \in \mathbb{Z} \mid 1 \leqslant j \leqslant n\}.$$

Для всех $i \in I$, $j \in J$ положим

$$I_j = \{i \in I \mid h_{ij} = 1\}, \quad J_i = \{j \in J \mid h_{ij} = 1\},$$
$$I_{ji} = I_j \backslash \{i\}, \quad J_{ij} = J_i \backslash \{j\}.$$

Наконец, для произвольного подграфа $\mathfrak{S}$ в графе $\mathfrak{T}$ положим

$$I(\mathfrak{S}) = \{i \in I \mid z_i \in \mathfrak{S}\} \quad \text{и} \quad J(\mathfrak{S}) = \{j \in J \mid x_j \in \mathfrak{S}\}.$$

Снабдим граф $\mathfrak{T}$ дополнительной структурой. Припишем к каждой символьной вершине x_j число λ_j, которое будем называть *весом символьной вершины* x_j. Будем считать, кроме этого, что каждая

проверочная и символьная вершина может принимать *значения* из поля $\mathbb{F}_2$. Значения вершин z_i и x_j будем обозначать соответственно $\operatorname{val} z_i$ и $\operatorname{val} x_j$. Будем говорить, что проверочная вершина z_i *удовлетворена*, если ее значение равно по модулю 2 сумме значений соседних с нею символьных вершин, то есть если

$$\operatorname{val} z_i = \sum_{j \in J_i} \operatorname{val} x_j.$$

В этих терминах задачу (5) можно сформулировать следующим образом: *«найти такой вектор* $\mathbf{e} \in \mathbb{F}_2^n$*, что при*

$$\operatorname{val} z_i = s_i \quad u \quad \operatorname{val} x_j = e_j \quad \text{для всех} \quad i \in I,\ j \in J$$

сумма весов символьных вершин со значением 1 минимальна и все проверочные вершины удовлетворены».

С этого места и далее мы будем предполагать, что граф Таннера $\mathfrak{T}$ матрицы $\mathbf{H}$ *не* содержит циклов. Также будем считать, что матрица $\mathbf{H}$ не содержит строк, состоящих только из одной единицы. Это значит, что граф $\mathfrak{T}$ не содержит проверочных вершин, имеющих единичную степень. Путь, соединяющий вершины u и v графа $\mathfrak{T}$, обозначим через $[u, v]$. Вместо $I([u, v])$ и $J([u, v])$ будем писать коротко $I[u, v]$ и $J[u, v]$.

4. Символьная связность и алгоритм домино

Для произвольного вектора $\mathbf{a} \in \mathbb{F}_2^n$, $\mathbf{a} = (a_j)$, определим множество

$$\operatorname{supp} \mathbf{a} = \{j \in J \mid a_j = 1\}.$$

Определение 1. Ненулевой вектор $\mathbf{a} \in \mathbb{F}_2^n$ будем называть $\mathbf{H}$-*связным* (или просто *связным*), если для всех $j_1, j_2 \in \operatorname{supp} \mathbf{a}$ имеет место включение $J[x_{j_1}, x_{j_2}] \subset \operatorname{supp} \mathbf{a}$.

Понятие связного вектора будет играть ключевую роль во всех наших дальнейших рассуждениях. Но прежде, чем к ним перейти, приведем формулировку алгоритма, при помощи которого можно построить произвольный связный кодовый вектор $\mathbf{c} \in \mathcal{C}$, а также изучим некоторые его свойства.

Определение 2. Пусть семейство $\Phi = \{\phi_i \mid i \in I\}$ отображений $\phi_i \colon J_i \to 2^{J_i}$ таково, что для всех $i \in I$ и $j \in J_i$ множество $\phi_i(j)$ состоит из нечетного числа элементов и $\phi_i(j) \subset J_{ij}$. Тогда будем называть отображения ϕ_i *направляющими*, а Φ — *семейством направляющих*.

Алгоритм 1 (общий алгоритм домино). Пусть дано семейство направляющих Φ. Алгоритм принимает на вход и возвращает следующие параметры и результаты.

- **Параметры:** индекс $j_0 \in J$.
- **Результат:** вектор $\mathbf{a} \in \mathbb{F}_2^n$.

Будем считать, что корень дерева $\mathfrak{T}$ расположен в вершине x_{j_0}. Тогда все проверочные вершины будут располагаться на нечетных ярусах.

1) **Инициализация:** установить значения всех проверочных и символьных вершин дерева $\mathfrak{T}$ равными нулю, кроме корня x_{j_0}, значение которого установить равным единице.
2) **Последовательно обработать** все нечетные ярусы дерева $\mathfrak{T}$, начиная с первого, следующим образом. Для каждой неудовлетворенной проверочной вершины z_i на ярусе:
   а) определить индекс $j \in J_i$ родительской для z_i символьной вершины (очевидно, это единственная соседняя с z_i вершина с ненулевым значением).
   б) для всех $j' \in \phi_i(j) \subset J_{ij}$ установить $\operatorname{val} x_{j'} = 1$ (все измененные символьные вершины являются дочерними для z_i).
3) **Вернуть** результат $\mathbf{a} = (\operatorname{val} x_j)$.

Для алгоритма 1 с семейством направляющих Φ, мы будем применять обозначение DOMINO_Φ, а для результата выполнения этого алгоритма с параметром j_0 — обозначение $\mathsf{DOMINO}_\Phi(j_0)$.

Предложение 2. *При любом $j_0 \in J$ результат $\mathbf{c} = \mathsf{DOMINO}_\Phi(j_0)$ является связным кодовым вектором, таким что $j_0 \in \operatorname{supp} \mathbf{c}$.*

Доказательство. Поскольку предполагается, что в дереве $\mathfrak{T}$ нет проверочных вершин единичной степени, то на шаге (2.b) $J_{ij} \neq \varnothing$, и выполнение этого шага приводит к удовлетворению соответствующей проверочной вершины. Следовательно, по завершении алгоритма все проверочные вершины графа $\mathfrak{T}$ будут удовлетворены. То, что $j_0 \in \operatorname{supp}\mathbf{c}$, очевидно. Далее, если $j \in \operatorname{supp}\mathbf{c}$, то $J[x_{j_0}, x_j] \subset \operatorname{supp}\mathbf{c}$, а значит, то же верно и для любой пары индексов из множества $\operatorname{supp}\mathbf{c}$. Таким образом, $\mathbf{c}$ — связный кодовый вектор.

Также имеет место обратное утверждение.

Предложение 3. *Пусть имеется связный кодовый вектор* $\mathbf{c} \in \mathcal{C}$. *Тогда найдется такое семейство направляющих* Φ, *что* $\mathbf{c} = \mathsf{DOMINO}_\Phi(j_0)$ *для любого индекса* $j_0 \in \operatorname{supp}\mathbf{c}$.

Доказательство. Рассмотрим произвольную проверочную вершину z_i. Если $J_i \cap \operatorname{supp}\mathbf{c} \neq \varnothing$, то, поскольку $\mathbf{c} \in \mathcal{C}$, количество элементов в $J_i \cap \operatorname{supp}\mathbf{c}$ четно. Пусть $j_1 \in J_i \cap \operatorname{supp}\mathbf{c}$. Определим отображение ϕ_i следующим образом. Для всех $j \in J_i \setminus \operatorname{supp}\mathbf{c}$ положим $\phi_i(j) = \{j_1\}$, а для всех $j \in J_i \cap \operatorname{supp}\mathbf{c}$ положим $\phi_i(j) = (J_i \cap \operatorname{supp}\mathbf{c}) \setminus \{j\}$.

Если $J_i \cap \operatorname{supp}\mathbf{c} = \varnothing$, то пусть $j_1, j_2 \in J_i$. Положим $\phi_i(j_1) = \{j_2\}$, а для всех $j \in J_i \setminus \{j_1\}$ положим $\phi_i(j) = \{j_1\}$. Легко видеть, что определенные таким образом отображения ϕ_i являются направляющими, и для любого индекса $j_0 \in \operatorname{supp}\mathbf{c}$ имеем $\mathbf{c} = \mathsf{DOMINO}_\Phi(j_0)$, где $\Phi = \{\phi_i \mid i \in I\}$.

5. Двухэтапный алгоритм домино

Для произвольных вектора $\mathbf{a} \in \mathbb{F}_2^n$, $\mathbf{a} = (a_j)$, и индекса $j_0 \in J$ рассмотрим задачу

$$w_\lambda(\mathbf{a} - \mathbf{c}) \to \min, \quad \begin{cases} \mathbf{c} \in \mathcal{C} \text{ — связный кодовый вектор,} \\ j_0 \in \operatorname{supp}\mathbf{c}. \end{cases} \tag{7}$$

Будем строить искомый кодовый вектор $\mathbf{c}$ при помощи алгоритма домино. Для этого нам нужно построить семейство направляющих Φ, которое обеспечит минимальность веса $w_\lambda(\mathbf{a} - \mathbf{c})$. С этой целью рассмотрим следующий алгоритм.

Алгоритм 2. Алгоритм принимает на вход и возвращает следующие параметры и результаты.

- **Параметры:** векторы $\lambda \in \mathbb{R}^n$ и $\mathbf{a} \in \mathbb{F}_2^n$.
- **Результат:** вектор $\gamma \in \mathbb{R}^n$, $\gamma = (\gamma_j)$, и системы значений

$$\Delta = \{\delta_{ij} \in \mathbb{R} \mid (i,j) \in I \times J,\ h_{ij} = 1\},$$
$$E = \{\epsilon_{ij} \in \mathbb{R} \mid (i,j) \in I \times J,\ h_{ij} = 1\}.$$

1) **Инициализация.** Для всех концевых вершин x_j дерева $\mathfrak{T}$ установить значение переменной δ_{ij}, такой что $I_j = \{i\}$, по формуле

$$\delta_{ij} = (-1)^{a_j} \lambda_j.$$

Значения всех других переменных δ_{ij} и ϵ_{ij} объявить как неопределенные.

2) **Последовательно вычислить** значения переменных δ_{ij} и ϵ_{ij} для всех $(i,j) \in I \times J$, $h_{ij} = 1$, по следующим правилам:

   а) **Вычисление ϵ_{ij}.** Если для всех $j' \in J_{ij}$ значения переменных $\delta_{ij'}$ определены, то установить значение переменной ϵ_{ij} по формуле

$$\epsilon_{ij} = \min_{J' \in 2^{J_{ij}}_{\text{odd}}} \left\{ \sum_{j' \in J'} \delta_{ij'} \right\}, \tag{8}$$

   где $2^{J_{ij}}_{\text{odd}}$ — множество всех подмножеств в J_{ij}, состоящих из нечетного числа элементов.

   б) **Вычисление δ_{ij}.** Если для всех $i' \in I_{ji}$ значения переменных $\epsilon_{i'j}$ определены, то установить значение переменной δ_{ij} по формуле

$$\delta_{ij} = (-1)^{a_j} \lambda_j + \sum_{i' \in I_{ji}} \epsilon_{i'j}. \tag{9}$$

3) **Вычисление γ_j.** Для всех $j \in J$, установить значение переменной γ_j по формуле

$$\gamma_j = (-1)^{a_j} \lambda_j + \sum_{i \in I_j} \epsilon_{ij}. \tag{10}$$

4) **Вернуть** результат (γ, Δ, E).

Легко видеть, что в результате выполнения алгоритма 2 все переменные δ_{ij} и ϵ_{ij} будут вычислены, и их значения будут соотноситься между собой по формулам (8) и (9). Выполнение этого алгоритма можно представлять себе в виде процесса последовательной передачи сообщений между вершинами дерева $\mathfrak{T}$ по его ребрам. Сообщение, передаваемое от вершины z_i к вершине x_j представлено значением ϵ_{ij}, а сообщение от вершины x_j к вершине z_i — значением δ_{ij}. При этом, каждая вершина дерева $\mathfrak{T}$ передает сообщение соседней вершине, лишь после получения всех сообщений от остальных соседних вершин. Алгоритм завершается, когда посланы все сообщения. Для алгоритма 2 мы будем применять обозначение $\mathsf{PRECALC}$, а для результата выполнения этого алгоритма с параметрами λ и $\mathbf{a}$ — обозначение $\mathsf{PRECALC}(\lambda, \mathbf{a})$.

Для произвольной пары $(i,j) \in I \times J$, $h_{ij} = 1$, обозначим через $\mathfrak{S}^x_{ij}$ наибольшее поддерево в $\mathfrak{T}$, которое содержит x_j, но не содержит z_i, а через $\mathfrak{S}^z_{ij}$ обозначим наибольшее поддерево в $\mathfrak{T}$, которое содержит z_i, но не содержит x_j.

Предложение 4. *Пусть* $(\gamma, \Delta, E) = \mathsf{PRECALC}(\lambda, \mathbf{a})$. *Тогда для каждой пары* $(i,j) \in I \times J$, $h_{ij} = 1$, *имеют место равенства*

$$\delta_{ij} = \min_{\substack{\mathbf{c}\in\mathcal{C} \\ \mathbf{c}\ -\ \text{связный} \\ j\in\operatorname{supp}\mathbf{c}}} \left\{ \sum_{j'\in J(\mathfrak{S}^x_{ij})\cap\operatorname{supp}\mathbf{c}} (-1)^{a_{j'}} \lambda_{j'} \right\}, \quad \text{где } \delta_{ij} \in \Delta, \tag{11}$$

$$\epsilon_{ij} = \min_{\substack{\mathbf{c}\in\mathcal{C} \\ \mathbf{c}\ -\ \text{связный} \\ j\in\operatorname{supp}\mathbf{c}}} \left\{ \sum_{j'\in J(\mathfrak{S}^z_{ij})\cap\operatorname{supp}\mathbf{c}} (-1)^{a_{j'}} \lambda_{j'} \right\}, \quad \text{где } \epsilon_{ij} \in E. \tag{12}$$

Доказательство. Фиксируем произвольную пару $(i_0, j_0) \in I \times J$, такую что $h_{i_0 j_0} = 1$. Сначала докажем второе равенство для $\epsilon_{i_0 j_0}$. Примем вершину x_{j_0} за корень дерева $\mathfrak{T}$. Относительно вершины x_{j_0} дерево $\mathfrak{T}$ распадается на четное число ярусов $2L$ (поскольку в нем отсутствуют концевые проверочные вершины). Докажем равенства (11) для ребер на четных ярусах и равенства (12) для ребер на нечетных ярусах. Будем делать это по индукции, начиная с нижнего яруса.

Для $2L$-го яруса имеем, очевидно, $\delta_{ij} = (-1)^{a_j}\lambda_j$. С другой стороны, $\mathfrak{S}^x_{ij} = \{x_j\}$ и для любого вектора $\mathbf{c}$, такого что $j \in \operatorname{supp}\mathbf{c}$, имеем $J(\mathfrak{S}^x_{ij}) \cap \operatorname{supp}\mathbf{c} = \{x_j\}$. Следовательно, для $2L$-го яруса равенство (11) выполняется. Для $(2L-1)$-го яруса имеем, по определению,

$$\epsilon_{ij} = \min_{J' \in 2^{J_{ij}}_{\text{odd}}} \left\{ \sum_{j' \in J'} \delta_{ij'} \right\}.$$

Пусть $\mathbf{c}$ — произвольный связный кодовый вектор, такой что $j \in \operatorname{supp}\mathbf{c}$. Тогда множество $J(\mathfrak{S}^z_{ij}) \cap \operatorname{supp}\mathbf{c}$ состоит из нечетного числа элементов и содержится в J_{ij}. При этом, если $\mathbf{c}$ пробегает множество всех связных кодовых векторов, таких что $j \in \operatorname{supp}\mathbf{c}$, то $J(\mathfrak{S}^z_{ij}) \cap \operatorname{supp}\mathbf{c}$ пробегает множество всех подмножеств в $2^{J_{ij}}_{\text{odd}}$. Поэтому предыдущее равенство принимает вид

$$\epsilon_{ij} = \min_{\substack{\mathbf{c}\in\mathcal{C} \\ \mathbf{c} \text{ — связный} \\ j\in\operatorname{supp}\mathbf{c}}} \left\{ \sum_{j' \in J(\mathfrak{S}^z_{ij}) \cap \operatorname{supp}\mathbf{c}} \delta_{ij'} \right\},$$

где, по уже доказанному, $\delta_{ij'} = (-1)^{a_{j'}}\lambda_{j'}$.

Пусть теперь равенства (11) и (12) доказаны соответственно для всех ярусов $2l'$ и $2l'-1$, где $l < l' \leqslant L$. Докажем, что они справедливы и для ярусов $2l$ и $2l-1$ соответственно. Для $2l$-го яруса имеем, по определению,

$$\delta_{ij} = (-1)^{a_j}\lambda_j + \sum_{i' \in I_{ji}} \epsilon_{i'j},$$

где для каждого значения $\epsilon_{i'j}$ равенство (12) уже доказано. Легко видеть, что найдется такой связный кодовый вектор $\mathbf{c}$, что $j \in \operatorname{supp}\mathbf{c}$ и

$$\epsilon_{i'j} = \sum_{j' \in J(\mathfrak{S}^z_{i'j}) \cap \operatorname{supp}\mathbf{c}} (-1)^{a_{j'}}\lambda_{j'} \quad \text{для всех } i' \in I_{ji}.$$

Ясно, что

$$\mathbf{c} = \arg \min_{\substack{\mathbf{c}\in\mathcal{C} \\ \mathbf{c} \text{ — связный} \\ j\in\operatorname{supp}\mathbf{c}}} \left\{ \sum_{j' \in J(\mathfrak{S}^x_{ij}) \cap \operatorname{supp}\mathbf{c}} (-1)^{a_{j'}}\lambda_{j'} \right\}.$$

Объединяя три последних равенства, получаем равенство (11).

Для $(2l-1)$-го яруса, как и прежде, по определению, имеем

$$\epsilon_{ij} = \min_{J' \in 2^{J_{ij}}_{\mathrm{odd}}} \left\{ \sum_{j' \in J'} \delta_{ij'} \right\},$$

где для всех $\delta_{ij'}$ равенство (11) уже доказано. Положим

$$J^* = \arg \min_{J' \in 2^{J_{ij}}_{\mathrm{odd}}} \left\{ \sum_{j' \in J'} \delta_{ij'} \right\}.$$

Пусть связный кодовый вектор $\mathbf{c}$ таков, что $j \in \operatorname{supp} \mathbf{c}$, $J_{ij} \cap \operatorname{supp} \mathbf{c} = J^*$ и справедливы равенства

$$\delta_{ij'} = \sum_{j'' \in J(\mathfrak{S}^x_{ij'}) \cap \operatorname{supp} \mathbf{c}} (-1)^{a_{j''}} \lambda_{j''} \quad \text{для всех } j' \in J^*.$$

Нетрудно убедиться, что такой вектор $\mathbf{c}$ существует, и что

$$\mathbf{c} = \arg \min_{\substack{\mathbf{c} \in \mathcal{C} \\ \mathbf{c} \text{ — связный} \\ j \in \operatorname{supp} \mathbf{c}}} \left\{ \sum_{j' \in J(\mathfrak{S}^z_{ij}) \cap \operatorname{supp} \mathbf{c}} (-1)^{a_{j'}} \lambda_{j'} \right\}.$$

Объединяя четыре последних равенства, получаем равенство (12).

В частности, мы доказали равенство (12) и для $\epsilon_{i_0 j_0}$, а значит, и для произвольного ϵ_{ij}. Теперь для того, чтобы доказать равенство (11) для произвольного δ_{ij}, достаточно провести такие же рассуждения, что и для значений δ_{ij} расположенных на $2l$-м ярусе дерева с заданным корнем.

Предложение доказано.

Положим $(\gamma, \Delta, E) = \mathsf{PRECALC}(\lambda, \mathbf{a})$. Для каждого $i \in I$ определим отображение $\phi_i \colon J_i \to 2^{J_i}$ по формуле

$$\phi_i(j) = \arg \min_{J' \in 2^{J_{ij}}_{\mathrm{odd}}} \left\{ \sum_{j' \in J'} \delta_{ij'} \right\}, \quad j \in J_i,$$

где $\delta_{ij'} \in \Delta$ и $2^{J_{ij}}_{\text{odd}}$ — множество всех подмножеств в J_{ij}, состоящих из нечетного числа элементов. Легко видеть, что определенное таким образом семейство $\Phi(\Delta) = \{\phi_i \mid i \in I\}$ является семейством направляющих.

Предложение 5. *Пусть* $(\gamma, \Delta, E) = \mathsf{PRECALC}(\lambda, \mathbf{a})$. *Тогда кодовый вектор* $\mathbf{c} = \mathsf{DOMINO}_{\Phi(\Delta)}(j_0)$ *является оптимальным решением задачи (7), и имеет место равенство*

$$w_\lambda(\mathbf{a} - \mathbf{c}) = w_\lambda(\mathbf{a}) + \gamma_{j_0}.$$

Доказательство. Тот факт, что вектор $\mathbf{c}$ является связным кодовым вектором, таким что $j_0 \in \operatorname{supp} \mathbf{c}$, следует из предложения 2. Из вида направляющих отображений следует, что

$$\sum_{j \in J(\mathfrak{S}^z_{ij_0}) \cap \operatorname{supp} \mathbf{c}} (-1)^{a_j} \lambda_j = \epsilon_{ij_0} \quad \text{для всех } i \in I_{j_0}.$$

Тогда

$$\gamma_{j_0} = (-1)^{a_{j_0}} \lambda_{j_0} + \sum_{i \in I_{j_0}} \epsilon_{ij_0} = \sum_{j \in \operatorname{supp} \mathbf{c}} (-1)^{a_j} \lambda_j.$$

Согласно предложению 4 получаем

$$\mathbf{c} = \arg \min_{\substack{\mathbf{c}' \in \mathcal{C} \\ \mathbf{c}' \text{ — связный} \\ j_0 \in \operatorname{supp} \mathbf{c}'}} \left\{ \sum_{j \in J(\mathfrak{S}^z_{ij_0}) \cap \operatorname{supp} \mathbf{c}'} (-1)^{a_j} \lambda_j \right\} \quad \text{для всех } i \in I_{j_0}.$$

Следовательно,

$$\mathbf{c} = \arg \min_{\substack{\mathbf{c}' \in \mathcal{C} \\ \mathbf{c}' \text{ — связный} \\ j_0 \in \operatorname{supp} \mathbf{c}'}} \left\{ \sum_{j \in \operatorname{supp} \mathbf{c}'} (-1)^{a_j} \lambda_j \right\}.$$

С другой стороны, для произвольного вектора $\mathbf{c}'$ имеем

$$w_\lambda(\mathbf{a} - \mathbf{c}') - w_\lambda(\mathbf{a}) = \sum_{j \in \operatorname{supp} \mathbf{c}'} (-1)^{a_j} \lambda_j.$$

Объединяя два последних соотношения, получаем

$$\mathbf{c} = \arg \min_{\substack{\mathbf{c}' \in \mathcal{C} \\ \mathbf{c}' \text{ — связный} \\ j_0 \in \operatorname{supp} \mathbf{c}'}} \left\{ w_\lambda(\mathbf{a} - \mathbf{c}') \right\}.$$

Кроме того,

$$w_\lambda(\mathbf{a} - \mathbf{c}) - w_\lambda(\mathbf{a}) = \sum_{j \in \operatorname{supp} \mathbf{c}} (-1)^{a_j} \lambda_j = \gamma_{j_0}.$$

Предложение доказано.

Для произвольного вектора $\mathbf{a} \in \mathbb{F}_2^n$, $\mathbf{a} = (a_j)$, рассмотрим задачу

$$w_\lambda(\mathbf{a} - \mathbf{c}) \to \min, \quad \mathbf{c} \in \mathcal{C} \text{ — связный кодовый вектор.} \tag{13}$$

Для решения этой задачи, объединим алгоритмы DOMINO_Φ и $\mathsf{PRECALC}$ в рамках единого алгоритма.

Алгоритм 3 (двухэтапный алгоритм домино). Алгоритм принимает на вход и возвращает следующие параметры и результаты.

- **Параметры:** векторы $\lambda \in \mathbb{R}^n$ и $\mathbf{a} \in \mathbb{F}_2^n$.
- **Результат:** вектор $\mathbf{c} \in \mathcal{C}$.

1) **Вычислить** $(\gamma, \Delta, E) = \mathsf{PRECALC}(\lambda, \mathbf{a})$.
2) **Вычислить** $j_0 = \arg\min_{j \in J} \{\gamma_j\}$.
3) **Вычислить** $\mathbf{c} = \mathsf{DOMINO}_{\Phi(\Delta)}(j_0)$.
4) **Вернуть** результат $\mathbf{c}$.

Алгоритм 3 будем обозначать как OPDIFF, а результат его выполнения с параметрами λ и $\mathbf{a}$ — как $\mathsf{OPDIFF}(\lambda, \mathbf{a})$. Из предложения 5 и описания алгоритма 3 немедленно получается

Теорема 6. *Вектор* $\mathbf{c} = \mathsf{OPDIFF}(\lambda, \mathbf{a})$ *является оптимальным решением задачи (13).*

6. Итеративный алгоритм декодирования

Предложение 7. *Если вектор* $\mathbf{e} \in \hat{\mathbf{r}} + \mathcal{C}$ *не является оптимальным решением задачи (5), то найдется связный кодовый вектор* $\mathbf{c} \in \mathcal{C}$, *такой что* $w_\lambda(\mathbf{e}) > w_\lambda(\mathbf{e} - \mathbf{c})$.

Доказательство. Пусть вектор $\mathbf{f} \in \hat{\mathbf{r}} + \mathcal{C}$ таков, что $w_\lambda(\mathbf{e}) > w_\lambda(\mathbf{f})$. Положим $\mathbf{c}' = \mathbf{e} - \mathbf{f} \in \mathcal{C}$. Для любого индекса $j_0 \in \operatorname{supp} \mathbf{c}'$ определим множество $[j_0] = \{j \in \operatorname{supp} \mathbf{c}' \mid J[x_{j_0}, x_j] \subset \operatorname{supp} \mathbf{c}'\}$. Очевидно, если $j_1 \in [j_0]$, то $[j_1] = [j_0]$, так что множество $\operatorname{supp} \mathbf{c}'$ можно дизъюнктивно и однозначно разбить на подмножества вида $[j]$. Пусть

$$\operatorname{supp} \mathbf{c}' = \bigcup_{\beta=1}^{B} J_\beta, \quad \text{где } J_\beta = [j] \text{ для всех } j \in J_\beta.$$

Тогда вектор $\mathbf{c}'$ можно представить в виде

$$\mathbf{c}' = \sum_{\beta=1}^{B} \mathbf{c}_\beta,$$

где $\operatorname{supp} \mathbf{c}_\beta = J_\beta$ для всех $\beta = 1, \ldots, B$. Очевидно, векторы $\mathbf{c}_\beta$ являются связными кодовыми векторами.

Поскольку $w_\lambda(\mathbf{e}) > w_\lambda(\mathbf{f})$, то

$$\sum_{j \in \operatorname{supp} \mathbf{c}'} (-1)^{e_j} \lambda_j = w_\lambda(\mathbf{e} - \mathbf{c}') - w_\lambda(\mathbf{e}) < 0,$$

и, следовательно,

$$\sum_{\beta=1}^{B} \left[\sum_{j \in \operatorname{supp} \mathbf{c}_\beta} (-1)^{e_j} \lambda_j \right] < 0.$$

Поэтому найдется такой индекс β_0, что

$$w_\lambda(\mathbf{e} - \mathbf{c}_{\beta_0}) - w_\lambda(\mathbf{e}) = \sum_{j \in \operatorname{supp} \mathbf{c}_{\beta_0}} (-1)^{e_j} \lambda_j < 0.$$

Итак, $\mathbf{c}_{\beta_0}$ — связный кодовый вектор, такой что $w_\lambda(\mathbf{e}) > w_\lambda(\mathbf{e} - \mathbf{c}_{\beta_0})$.

Предложение доказано.

Из теоремы 6 и доказанного предложения сразу получается

Следствие 8. *Если вектор* $\mathbf{e} \in \hat{\mathbf{r}} + \mathcal{C}$ *не является оптимальным решением задачи (5), то для вектора* $\mathbf{c} = \mathsf{OPDIFF}(\lambda, \mathbf{e})$ *верно неравенство*

$$w_\lambda(\mathbf{e}) > w_\lambda(\mathbf{e} - \mathbf{c}).$$

Теперь мы можем представить итеративный алгоритм, который реализует упомянутый в разделе 2 общий подход к задаче (5).

Алгоритм 4 (алгоритм декодирования методом спуска). Алгоритм принимает на вход и возвращает следующие параметры и результаты.

- **Параметры:** число $M \in \mathbb{N} \cup \{+\infty\}$ и векторы $\lambda \in \mathbb{R}^n$ и $\mathbf{e}^{(0)} \in \hat{\mathbf{r}} + \mathcal{C}$.
- **Результат:** вектор $\mathbf{e} \in \hat{\mathbf{r}} + \mathcal{C}$.

1) **Вычислить** $\mathbf{c} = \mathsf{OPDIFF}(\lambda, \mathbf{e}^{(0)})$ и установить $t = 0$.
2) **Пока** $w_\lambda(\mathbf{e}^{(t)}) > w_\lambda(\mathbf{e}^{(t)} - \mathbf{c})$ и $t < M$, **выполнять:**
   а) увеличить t на единицу.
   б) $\mathbf{e}^{(t)} = \mathbf{e}^{(t-1)} - \mathbf{c}$.
   в) $\mathbf{c} = \mathsf{OPDIFF}(\lambda, \mathbf{e}^{(t)})$.
3) **Если** $w_\lambda(\mathbf{e}^{(t)}) \leqslant w_\lambda(\mathbf{e}^{(t)} - \mathbf{c})$, **то** вернуть результат $\mathbf{e} = \mathbf{e}^{(t)}$, **иначе** объявить об отказе.

Алгоритм 4 будем обозначать как $\mathsf{DESCEND}$, а результат его выполнения с параметрами M, λ и $\mathbf{e}^{(0)}$ — как $\mathsf{DESCEND}(M, \lambda, \mathbf{e}^{(0)})$. Отметим, что и при $M = +\infty$ данный алгоритм выполняется за конечное число итераций (хотя бы ввиду конечности класса $\hat{\mathbf{r}} + \mathcal{C}$). Из следствия 8 и описания алгоритма 4 очевидным образом вытекает

Теорема 9. *Вектор* $\mathbf{e} = \mathsf{DESCEND}(M, \lambda, \mathbf{e}^{(0)})$, *в случае успешного выполнения алгоритма, является оптимальным решением задачи (5) при любом* $\mathbf{e}^{(0)} \in \hat{\mathbf{r}} + \mathcal{C}$. *При этом, если* $M = +\infty$, *то алгоритм всегда выполняется успешно.*

7. Оценка сложности

В этом разделе мы оценим сложность алгоритма DESCEND при $n \to \infty$.

Предложение 10. *Если максимальная степень вершин графа $\mathfrak{T}$ лежит в классе $O(1)$, $n \to \infty$, то сложность алгоритма* DOMINO_Φ *лежит в классе $O(n)$, $n \to \infty$.*

Доказательство. Во-первых, ясно, что каждая проверочная вершина графа $\mathfrak{T}$ обрабатывается алгоритмом домино не более одного раза. Выполнение шага (2.a), то есть поиск в множестве J_i такого индекса j, что $\operatorname{val} x_j = 1$, занимает время порядка $O(1)$. Такое же время, очевидно, затрачивается на выполнение шага (2.b). Итак, общее время выполнения алгоритма имеет порядок роста $O(n)$.

Предложение 11. *Если максимальная степень вершин графа $\mathfrak{T}$ лежит в классе $O(1)$, $n \to \infty$, то сложность алгоритма* PRECALC *лежит в классе $O(n)$, $n \to \infty$.*

Доказательство. Процедура инициализации алгоритма имеет, очевидно, сложность порядка $O(n)$. Далее, при наших предположениях вычисление значений ϵ_{ij}, δ_{ij} и γ_j по формулам (8)–(10) имеет сложность порядка $O(1)$, поскольку зависит лишь от степеней проверочных и символьных вершин. Пусть d — максимальная степень вершин графа $\mathfrak{T}$. Тогда множество $\{(i,j) \in I \times J \mid h_{ij} = 1\}$ содержит не более dn элементов, и, следовательно, на вычисление всех ϵ_{ij} и δ_{ij} затрачивается время порядка $O(n)$. То же верно и для процедуры вычисления значений γ_j. Таким образом, сложность алгоритма PRECALC лежит в классе $O(n)$.

Предложение 12. *Если максимальная степень вершин графа $\mathfrak{T}$ лежит в классе $O(1)$, $n \to \infty$, то сложность алгоритма* OPDIFF *лежит в классе $O(n)$, $n \to \infty$.*

Доказательство. Сложность вычисления (γ, Δ, E), согласно предыдущему утверждению, имеет порядок роста $O(n)$. Вычисление индекса с минимальным значением γ_j занимает, очевидно, линейное по n

время. Вычисление вектора $\mathbf{c}$, по предложению 10, также занимает время порядка $O(n)$. Таким образом, алгоритм OPDIFF имеет сложность порядка $O(n)$.

Получаем очевидное следствие.

Следствие 13. *Если максимальная степень вершин графа $\mathfrak{T}$ лежит в классе $O(1)$, $n \to \infty$, то сложность алгоритма* DESCEND *с параметром $M < +\infty$ лежит в классе $O(n)$, $n \to \infty$.*

Теорема 1 является простым следствием теоремы 9 и следствия 13.

8. Приложение

Предложение 14. *Плотность условной вероятности того, что при передаче вектора $\mathbf{c} \in \mathcal{C}$ будет принят вектор $\mathbf{r} \in \mathbb{R}^n$, имеет вид*

$$p(\mathbf{r} \,|\, \mathbf{c}) = \alpha(\mathbf{r}) \exp\left(-L_c \sum_{j\colon r_j\tilde{c}_j<0} |r_j| \right),$$

где $\alpha(\mathbf{r}) > 0$.

Доказательство. Квадрат расстояния между $\mathbf{r}$ и $\tilde{\mathbf{c}}$ равен

$$||\mathbf{r} - \tilde{\mathbf{c}}||^2 = \mathbf{r}^2 + \tilde{\mathbf{c}}^2 - 2\langle \mathbf{r}, \tilde{\mathbf{c}} \rangle = \mathbf{r}^2 + na^2 - 2\langle \mathbf{r}, \tilde{\mathbf{c}} \rangle .$$

Подставляя это выражение в (1), получаем

$$p(\mathbf{r} \,|\, \mathbf{c}) = \frac{\exp\left(-\frac{1}{2\sigma^2}(\mathbf{r}^2 + na^2)\right)}{(\sigma\sqrt{2\pi})^n} \exp\left(\frac{1}{\sigma^2} \langle \mathbf{r}, \tilde{\mathbf{c}} \rangle \right). \tag{14}$$

Далее, скалярное произведение $\langle \mathbf{r}, \tilde{\mathbf{c}} \rangle$ можно записать в виде

$$\sum_{j=1}^{n} r_j\tilde{c}_j = a\left(\sum_{j\colon r_j\tilde{c}_j\geqslant 0} |r_j| - \sum_{j\colon r_j\tilde{c}_j<0} |r_j| \right) = a\sum_{j=1}^{n} |r_j| - 2a \sum_{j\colon r_j\tilde{c}_j<0} |r_j|.$$

Подставляя последнее выражение в (14), получаем

$$p(\mathbf{r} \,|\, \mathbf{c}) = \alpha(\mathbf{r}) \exp\left(-\frac{2a}{\sigma^2} \sum_{j\colon r_j\tilde{c}_j<0} |r_j| \right),$$

где

$$\alpha(\mathbf{r}) = \frac{\exp\left(-\frac{1}{2\sigma^2}(\mathbf{r}^2 + na^2) + \frac{a}{\sigma^2}\sum_{j=1}^{n}|r_j|\right)}{(\sigma\sqrt{2\pi})^n} > 0.$$

Предложение 15. *Пусть вероятности всех кодовых слов одинаковы. Тогда вероятность того, что был передан вектор* $\mathbf{c} \in \mathcal{C}$*, при условии, что принят вектор* $\mathbf{r} \in \mathbb{R}^n$*, равна*

$$P(\mathbf{c}\,|\,\mathbf{r}) = \beta(\mathbf{r})p(\mathbf{r}\,|\,\mathbf{c}), \quad \textit{где } \beta(\mathbf{r}) > 0.$$

Доказательство. Имеем

$$P(\mathbf{c}\,|\,\mathbf{r}) = \frac{p(\mathbf{r},\mathbf{c})}{p(\mathbf{r})} = \frac{p(\mathbf{r}\,|\,\mathbf{c})P(\mathbf{c})}{\sum_{\mathbf{c}'\in\mathcal{C}} p(\mathbf{r}\,|\,\mathbf{c}')P(\mathbf{c}')} = \frac{p(\mathbf{r}\,|\,\mathbf{c})}{\sum_{\mathbf{c}'\in\mathcal{C}} p(\mathbf{r}\,|\,\mathbf{c}')}.$$

Благодарности

Автор выражает благодарность сотрудникам кафедры математической теории интеллектуальных систем механико-математического факультета МГУ Гасанову Эльяру Эльдаровичу и Пантелееву Павлу Анатольевичу за руководство и неоценимую помощь в проведении исследований.

Список литературы

[1] Berlekamp E. R., McEliece R. J., Van Tilborg H. C. A. On the inherent intractability of certain coding problems // IEEE Trans. Inf. Theory. Vol. 24. Issue 5. P. 384–386. May 1978.

[2] Bruck J., Naor M. The hardness of decoding linear codes with preprocessing // IEEE Trans. Inf. Theory. Vol. 36. P. 381–385. March 1990.

[3] Gallager R. Low-density parity-check codes // IRE Trans. Inf. Theory. Vol. IT–8. No. 1. P. 21–28. Jan. 1962.

[4] MacKay D.J. Good Error-Correcting Codes Based on Very Sparse Matrices // IEEE Trans. Inf. Theory. Vol. 45. No. 2. P. 399–431. March 1999.

[5] Etzion T., Trachtenberg A., Vardy A. Which Codes Have Cycle-Free Tanner Graphs? // IEEE Trans. Inf. Theory. Vol. 45. No. 6. P. 2173–2181. September 1999.

[6] Wiberg N. Codes and decoding on general graphs / Ph. D. dissertation. Linkoping, Sweden: Linkoping Univ., 1996.

[7] Moon T. K. Error Correction Coding: Mathematical Methods and Algorithms. NJ: Wiley, 2005.