О параллельной расшифровке разбиений булевого куба

В. В. Осокин

В работе рассматривается класс $\Phi^{k,n}$ дискретных функций, определенных на булевом кубе $\{0,1\}^n$ и существенно зависящих от не более чем $k \leqslant n$ переменных. Каждая функция f из $\Phi^{k,n}$ задает разбиение куба $\{0,1\}^n$ на непересекающиеся грани и сопоставляет уникальное число каждой грани результирующего разбиения. Мы доказываем нижнюю оценку сложности расшифровки функций из $\Phi^{k,n}$ и предлагаем алгоритм расшифровки функций из $\Phi^n = \Phi^{n,n}$, оптимальный в том смысле, что число запросов на значение функции, требуемое алгоритму для расшифровки произвольной функции из $\Phi^{k,n}$, по порядку совпадает с нижней оценкой. Построенный алгоритм может быть эффективно распараллелен и при максимальном распараллеливании его сложность по порядку равна k.

Введение

Неформально задача расшифровки функций из некоторого класса Φ состоит в построении ученика (алгоритма расшифровки), который смог бы успешно играть с учителем в следующую игру. Учитель загадывает некоторую функцию из класса Φ. Ученик должен расшифровать загаданную функцию, то есть полностью определить ее таблицу значений, задав учителю минимальное число вопросов (запросов на значение функции). Под запросом понимается точка из области определения функции. Ответом учителя является значение функции в данной точке.

Определим класс Φ^n дискретных функций, сопоставляющих каждой вершине куба $\{0,1\}^n$ натуральное число. Для каждой $f \in \Phi^n$ если

f сопоставляет двум разным вершинам a и b одно и то же число m, то $f(G) = m$, где G — наименьшая грань куба $\{0,1\}^n$, такая что $a \in G$ и $b \in G$. Другими словами, каждая $f \in \Phi^n$ разбивает куб $\{0,1\}^n$ на грани и приписывает уникальное число каждой грани результирующего разбиения. Функции из Φ^n будем называть *разбивающими*.

В работе исследуется сложность расшифровки функций из Φ^n при помощи запросов на значение функции. Легко показывается, что сложность расшифровки функций из Φ^n в худшем случае равна 2^n.

Для практики особый интерес представляют функции из класса Φ^n, число существенных переменных которых мало по сравнению с n. Поэтому, естественен вопрос о существовании алгоритма расшифровки функций из Φ^n, сложность которого слабо (например, логарифмически) зависит от числа несущественных переменных неизвестной функции. Алгоритмы, обладающие подобным свойством, обычно называют *параметро-эффективными* [1]. Пусть $\Phi^{k,n}$ — подмножество Φ^n, состоящее из функций, зависящих существенно от не более чем k переменных. В разделе 4 настоящей работы показано, что сложность расшифровки функций из $\Phi^{k,n}$ асимптотически не меньше $k \log n + 2^k$ и построен оптимальный по порядку параметро-эффективный алгоритм расшифровки функций из Φ^n.

В разделе 5 настоящей работы рассмотрена ситуация, в которой учитель может отвечать сразу на целый блок сгенерированных учеником запросов на значение функции. В такой ситуации встает вопрос минимизации числа блоков запросов на значение функции, подаваемых учеником учителю в процессе расшифровки (минимизации *параллельной сложности* алгоритма расшифровки).

Упомянутый выше оптимальный по порядку параметро-эффективный алгоритм расшифровки функций из Φ^n является по существу условным: при выборе следующего запроса на значение функции он пользуется знаниями о значениях функции на предыдущих поданных им наборах. Отсюда следует, что его параллельная сложность не оптимальна. В идеале хотелось бы иметь безусловный параметро-эффективный алгоритм расшифровки, поскольку параллельная сложность такого алгоритма равна единице. В разделе 5 показано, что безусловного близкого к оптимальному параметро-эффективного алгоритма расшифровки разбивающих функций не существует, до-

казана зависимость сложности расшифровки от параллельной сложности и построен оптимальный по порядку параметро-эффективный алгоритм расшифровки разбивающих функций с оптимальной параллельной сложностью расшифровки.

Автор выражает благодарность профессору Э.Э.Гасанову за постановку задачи и помощь в работе.

1. Актуальность темы и близкие работы

Начнем с интерпретации класса Φ^n разбивающих функций. Следуя векторной модели информационного поиска документов [2], будем понимать булевы переменные $\{x_1, \ldots, x_n\}$ как множество из n слов, набор $a \in \{0,1\}^n$ — как документ, а булев куб $\{0,1\}^n$ — как множество всевозможных документов. Согласно определению Φ^n каждая $f \in \Phi^n$ разбивает множество документов $\{0,1\}^n$ на непересекающиеся классы и приписывает уникальную тему (натуральное число) каждой грани результирующего разбиения. Как видно из определения Φ^n, каждый такой класс задается конъюнкцией литер над некоторыми словами из $\{x_1, \ldots, x_n\}$.

Поясним сказанное на следующем примере. Рассмотрим разбивающую функцию с двумя существенными переменными: «условная» и «расшифровка». Первая тема на множестве документов, составленных из n слов, задается конъюнкцией «условная»&«расшифровка», вторая — конъюнкцией ¬«условная»&«расшифровка», третья — конъюнкцией ¬«расшифровка». Очевидно,

$$\text{«условная»\&«расшифровка»} \vee \neg\text{«условная»\&«расшифровка»} \vee \vee\neg\text{«расшифровка»} \equiv 1.$$

Конъюнкции в этой ДНФ задают 3 непересекающиеся грани n-мерного куба (то есть 3 класса в множестве из 2^n документов), а функция, сопоставляющая уникальное натуральное число каждой конъюнкции этой ДНФ, является разбивающей функцией от n переменных, существенно зависящей от двух своих переменных: «условная» и «расшифровка».

В работе изучается сложность расшифровки функций из Φ^n. В терминах приведенной интерпретации класса Φ^n решается задача построения оптимального алгоритма, который для произвольного разбиения множества документов на темы (при помощи некоторой разбивающей функции) строит классификатор, который для каждого документа возвращает тему, которой этот документ принадлежит.

Как будет далее показано, сложность расшифровки функций из Φ^n равна 2^n (максимально возможная). Поэтому, в настоящей работе основное внимание уделяется параметро-эффективной расшифровке функций из Φ^n. Параметро-эффективная расшифровка дискретных (в основном, булевых) функций активно изучается в нескольких моделях стимулирующего обучения и обучения с пассивным/активным учителем.

Одной из первых попыток эффективной работы с несущественными переменными стал параметро-эффективный алгоритм расшифровки пороговых функций в модели ограниченной ошибки стимулирующего обучения [3]. В последствии различные аспекты параметро-эффективной расшифровки в модели ограниченной ошибки изучались в [1, 4].

Расшифровка при пассивном учителе или расшифровка на основе случайной выборки обычно изучается в рамках так называемой вероятно примерно точной (PAC) модели обучения [5]. Параметро-эффективная расшифровка на основе случайной выборки при равномерном распределении в PAC-модели является одной из открытых проблем в теории обучения. Заинтересованный читатель найдет информацию о последних исследованиях в данном направлении в [6, 7, 8].

В настоящей работе мы фокусируемся на параметро-эффективной расшифровке при активном учителе. В отличие от пассивной расшифровки в активной модели расшифровки ученику разрешается задавать вопросы о неизвестной функции. Задача параметро-эффективной активной расшифровки рассматривалась в различных моделях расшифровки [1, 9]. Мы рассматриваем задачу в рамках точной модели расшифровки при помощи запросов на значение функции [10, 11, 12]. Некоторые общие результаты, касающиеся сложности параметро-эффективной расшифровки в этой модели даны в [4, 13].

В [14] даются точные оценки сложности параметро-эффективной расшифровки дизъюнкций, линейных и пороговых функций, в то время как в [15] рассматривается параметро-эффективная параллельная расшифровка монотонных булевых функций.

Среди направлений, близких к рассматриваемому направлению расшифровки функций (машинного обучения с учителем [16, 17]), можно выделить теорию тестового распознавания [18, 19] и теорию информационного поиска [2, 20].

Настоящая работа является продолжением исследований, приведенных в [21, 22].

2. Постановка задачи и формулировка результатов

Пусть $V_n = \{x_1, \ldots, x_n\}$ — множество булевых переменных. Следуя [18] множество $\{0,1\}^{V_n} = \{f : V_n \to \{0,1\}\}$ будем называть n-мерным булевым кубом над переменными $x_1, \ldots, x_n$. Отображение $a \in \{0,1\}^{V_n}$ будем также называть *набором*, интерпретируя его как вектор $a = (a_1, \ldots, a_n)$, где $a_i = a(x_i)$. И наоборот, набор $a = (a_1, \ldots, a_n) \in \{0,1\}^n$ будем называть *фиксацией переменных* из V_n, интерпретируя его как отображение $a \in \{0,1\}^{V_n}$, такое что $a(x_i) = a_i$. Если $a \in \{0,1\}^{V_n}$, $x_i \in V_n$ и $z \in \{0,1\}$, то через $a_{x_i \leftarrow z}$ обозначим отображение из $\{0,1\}^{V_n}$, такое что $a_{x_i \leftarrow z}(x_j) = a(x_j)$ для всех $j \neq i$ и $a_{x_i \leftarrow z}(x_i) = z$.

Переменная x_i называется *существенной переменной* функции $f : \{0,1\}^{V_n} \to \mathbb{N}$, если существует набор $a \in \{0,1\}^{V_n}$, такой что $f(a) \neq f(a_{x_i \leftarrow \overline{a}(x_i)})$. Если x_i является существенной переменной функции f, будем говорить, что функция f *существенно зависит от* x_i.

Частичной фиксацией p переменных из некоторого множества V назовем отображение из V в $\{0,1,*\}$. Переменная $x \in V$ *фиксирована* фиксацией p, если $p(x) \neq *$. Пусть $W \subseteq V$ — переменные, фиксированные фиксацией p. Пусть a — частичная фиксация переменных множества V', содержащего $V \backslash W$, такая что a фиксирует все переменные из $V \backslash W$. Тогда p/a — это фиксация переменных из V, совпадающая с p на всех переменных, фиксированных p, и совпадающая с a на остальных переменных.

Пусть $V' \subseteq V$ и p — частичная фиксация переменных из V, оставляющая в точности переменные из V' не фиксированными. Тогда если $f : \{0,1\}^V \to \mathbb{N}$, то f_p обозначает функцию над $\{0,1\}^{V'}$, задаваемую равенством $f_p(a) = f(p/a)$ для всех $a \in \{0,1\}^{V'}$. Функция f_p называется *проекцией* f.

Всюдуопределенную дискретную функцию $f : \{0,1\}^{V_n} \to \mathbb{N}$, такую что для любого $i \in f(\{0,1\}^{V_n})$ множество $f^{-1}(i)$ является гранью булевого куба, будем называть *разбивающей*. Разбивающая функция f разбивает булев куб $\{0,1\}^{V_n}$ на грани и сопоставляет каждой грани разбиения уникальное натуральное число. Через Φ^n обозначим множество всех разбивающих функций от n переменных. Пусть $\Phi^{k,n}$ — множество функций в Φ^n, зависящих от не более чем k переменных, $k \leqslant n$. Такие функции мы будем называть *k-существенными разбивающими функциями*.

Запросом на значение функции $f : \{0,1\}^{V_n} \to \mathbb{N}$ будем называть набор из $\{0,1\}^{V_n}$. *Ответом на запрос на значение функции* f — значение функции f на этом наборе.

алгоритмом расшифровки будем понимать условный эксперимент, который последовательно генерирует запросы на значение функции в зависимости от ответов на предыдущие запросы. Будем говорить, что алгоритм расшифровывает функцию f, если значения функции f на наборах из результата условного эксперимента (то есть, на множестве сгенерированных им запросов на значение функции) однозначно определяют таблицу значений функции f. Алгоритм расшифровки будем называть *безусловным*, если он является безусловным экспериментом. В противном случае будем называть его *условным*.

Пусть Q — некоторый класс псевдобулевских функций, $Q^n \subseteq Q$ — класс функций из Q, зависящих от n переменных, $Q^{k,n} \subseteq Q^n$ — класс функций из Q^n, существенно зависящих от не более чем $k \leqslant n$ своих переменных.

Будем говорить, что алгоритм *расшифровывает класс* Q, если для любого $n \in \mathbb{N}$ алгоритм расшифровывает любую функцию из Q^n при условии, что число n было заранее сообщено алгоритму. Последнее условие необходимо хотя бы для того, чтобы алгоритм знал, какой длины запросы ему подавать учителю. Множество алгоритмов расшифровки Q обозначим через $\mathcal{A}(Q)$.

Пусть $\varphi(F, f)$ — число запросов на значение функции, которое требуется алгоритму F для расшифровки функции f. Положим

$$\varphi(Q, n) = \min_{F \in \mathcal{A}(Q)} \max_{f \in Q^n} \varphi(F, f).$$

Заметим, что здесь использован min, а не inf, поскольку при фиксированном n число алгоритмов расшифровки функций из $Q^n \subseteq Q$ конечно. Величину $\max_{f \in Q} \varphi(F, f)$. будем называть *сложностью алгоритма F на классе Q*. Функцию $\varphi(Q, n)$ будем называть *сложностью расшифровки Q*.

Положим

$$\varphi(Q, n, k) = \min_{F \in \mathcal{A}(Q)} \max_{f \in Q^{k,n}} \varphi(F, f).$$

Заметим, что $\varphi(Q, n) = \varphi(Q, n, n)$.

Множество последовательно сделанных алгоритмом F запросов на значение функции при расшифровке функции f будем называть (F, f)-*независимым*, если при формировании каждого запроса из этого множества алгоритм расшифровки не использует информацию о значениях функции f на ранее поданных наборах из этого множества. Через $\varphi_p(F, f)$ обозначим минимальное число (F, f)-независимых множеств, таких что в объединении они дают все множество запросов, сделанных F при расшифровке f. Обозначим

$$\mathcal{A}_q(Q, n, k) = \{F \in \mathcal{A}(Q) : \max_{f \in Q^{k,n}} \varphi(F, f) \leqslant q\}.$$

Положим

$$\varphi_p(Q, n, k, q) = \min_{F \in \mathcal{A}_q(Q,n,k)} \max_{f \in Q^{k,n}} \varphi_p(F, f).$$

Величину $\max_{f \in Q} \varphi_p(F, f)$ будем называть *параллельной сложностью алгоритма F на классе Q*.

В настоящей работе исследуется сложность расшифровки класса Ф разбивающих функций. Следующее утверждение показывает, что не существует эффективного (полиномиального) алгоритма расшифровки функций из Ф.

Утверждение 1. *Для любого $n \in \mathbb{N}$ выполнено $\varphi(\Phi, n) = 2^n$.*

В разделе 4 доказана нижняя оценка сложности расшифровки функций из $\Phi^{k,n}$ и построен оптимальный параметро-эффективный алгоритм расшифровки функций из Φ. Имеет место следующая теорема.

Теорема 1. *Для $k, n \in \mathbb{N}$ выполнено*

$$\max(k \log n, 2^k) \lesssim \varphi(\Phi, n, k) \lesssim k \log n + 2^k$$

при $n \to \infty$.

Следствие 1. *Для $k, n \in \mathbb{N}$ если $k \log n \not\asymp 2^k$, то*

$$\varphi(\Phi, n, k) \sim k \log n + 2^k$$

при $n \to \infty$.

В разделе 5 доказано, что не существует безусловного близкого к оптимальному параметро-эффективного оптимального алгоритма расшифровки разбивающих функций, доказано, что параллельная сложность оптимального параметро-эффективного алгоритма расшифровки не может быть по порядку меньше k и построен оптимальный параметро-эффективный на Φ алгоритм расшифровки с оптимальной по порядку параллельной сложностью.

Теорема 2. *Для $k, n \in \mathbb{N}$ и произвольной константы $c > 2$ если $2^k < (\frac{c}{2} - 1)k \log n$, то*

$$\varphi_p(\Phi, n, k, c \cdot k \log n) \asymp k$$

при $k, n \to \infty$.

3. Расшифровка разбивающих функций

Рассмотрим разбивающую функцию f от n переменных, все переменные которой являются существенными. Пусть функция f принимает 2^n различных значений. Другими словами, f разбивает n-мерный булев куб на 2^n граней размерности 0. Если ничего не известно

о значениях функции f, то алгоритму расшифровки потребуется как минимум 2^n запросов на значение функции, чтобы расшифровать f. Отсюда следует, что

$$\varphi(\Phi, n) = 2^n.$$

Таким образом, оптимальный алгоритм расшифровки тривиален: в качестве запросов на значения функции он использует все 2^n возможных набора и утверждение 1 доказано.

4. Параметро-эффективная расшифровка разбивающих функций

Перейдем к построению параметро-эффективного оптимального по порядку алгоритма расшифровки разбивающих функций. Для начала покажем, что число запросов на значение функции, необходимых в худшем случае только для того, чтобы узнать все существенные переменные k-существенной разбивающей функции не может быть асимптотически меньше $k \log(n-k)$.

Лемма 1. *Для $k, n \in \mathbb{N}$ выполнено*

$$\varphi(\Phi, n, k) \gtrsim k \log(n-k)$$

при $n \to \infty$.

Доказательство. Пусть имеется алгоритм F расшифровки k-существенных разбивающих функций при помощи запросов на значение функции. Пусть даже этот алгоритм получает k на вход (это только упрощает его жизнь). Наша задача состоит в том, чтобы построить k-существенную разбивающую функцию, такую что алгоритму F потребуется асимптотически как минимум $k \log(n-k)$ запросов на значение функции, чтобы восстановить f.

Пусть $V_n = \{x_1, \ldots, x_n\}$ — множество переменных функции f, $V_k = \{x_{i_1}, \ldots, x_{i_k}\}$ — подмножество V_n, состоящее из существенных переменных функции f.

Идея заключается в том, чтобы выбрать небольшое подмножество $V_{\log k}$ существенных переменных и использовать их, чтобы прятать

оставшиеся существенные переменные друг от друга. Другими словами, мы хотим иметь такую функцию f, что когда F узнает какую-либо информацию о переменной $x_j \in V_k \backslash V_{\log k}$, он не узнает ровным счетом ничего про другие переменные из $V_k \backslash V_{\log k}$. Если мы покажем, что асимптотически алгоритму F требуется как минимум $\log(n-k)$ запросов на значение функции, чтобы расшифровать переменную из $V_k \backslash V_{\log k}$ и что мощность множества $V_{\log k}$ равна $o(k)$, то этим мы закончим доказательство леммы.

Без потери общности будем полагать, что $k = 2^m$ для некоторого натурального m. Пусть $V_{\log k} = \{x_{i_1} \ldots, x_{i_{\log k}}\}$. Заметим, что множество $\{0,1\}^{V_{\log k}}$ содержит в точности k наборов. Перенумеруем все пары (a, z), $a \in \{0,1\}^{V_{\log k}}$, $z \in \{0,1\}$, и обозначим эту нумерацию через N. Выберем произвольным образом $k - \log k$ наборов в $\{0,1\}^{V_{\log k}}$ отобразим их взаимно однозначно в множество $V_k \backslash V_{\log k}$. Это отображение обозначим через L и рассмотрим набор $b \in \{0,1\}^{V_n}$. Пусть

$$a = (b(x_{i_1}), \ldots, b(x_{i_{\log k}})) \in \{0,1\}^{V_{\log k}}.$$

Положив $f(b) = N(a, b(L(a)))$, мы полностью определим функцию f на $\{0,1\}^{V_n}$.

Говоря неформально, мы рассмотрели маленький булев куб $\{0,1\}^{V_{\log k}}$ с в точности k вершинами и разбивающую функцию на этом кубе, сопоставляющую уникальное натуральное число каждой из k вершин этого куба. Заметим, что $\dim(\{0,1\}^{V_{\log k}}) = \log k$. Далее мы увеличили куб до размерности k путем «разбиения» каждой из его k вершин при помощи уникальной неиспользованной существенной переменной функции f.

В качестве примера положим $k = 3$ и рассмотрим 1-мерный булев куб $\{0,1\}^{\{x_{i_1}\}}$. Используя представление в виде ДНФ, описанное выше разбиение куба $\{0,1\}^{\{x_{i_1}\}}$ может быть записано в виде $x_{i_1} \vee \overline{x}_{i_1} \equiv 1$, а увеличение размерности в виде преобразования $x_{i_1} \vee \overline{x}_{i_1} \equiv 1 \to x_{i_1}(x_{i_2} \vee \overline{x}_{i_2}) \vee \overline{x}_{i_1}(x_{i_3} \vee \overline{x}_{i_3}) \equiv 1$. Теперь мы имеем разбиение 3-мерного куба $\{0,1\}^{\{x_{i_1}, x_{i_2}, x_{i_3}\}}$ на 4 грани. После сопоставления уникальных натуральных чисел граням мы получаем 3-существенную разбивающую функцию от 3 переменных. Эту функцию легко преобразовать в 3-существенную функцию от n переменных путем добавления $n-3$ несущественных переменных. Полученная функция —

это в точности та, что необходима нам для управления имеющимся алгоритмом расшифровки.

Заметим, что мы до сих пор не фиксировали номера $i_1, \ldots, i_k$ существенных переменных функции f. Переменные $x_{i_1}, \ldots, x_{i_{\log k}}$ являются вспомогательными, потому полагаем $i_1 = 1, \ldots, i_{\log k} = \log k$. На самом деле, мы полагаем, что натуральные числа $i_{\log k+1}, \ldots, i_k$ — это единственное свойство функции f, неизвестное алгоритму расшифровки (это предположение еще больше упрощает жизнь алгоритму F). Заметим, что чтобы расшифровать i_j, $j \in \{\log k + 1, \ldots, k\}$, алгоритм расшифровки должен делать запросы на значение функции, используя такие наборы $b \in \{0,1\}^{V_n}$, что $L(b(x_1), \ldots, b(x_{\log k})) = x_{i_j}$ и только такие. Таким образом, чтобы найти нижнюю оценку числа запросов на значение функции, мы всего лишь должны подсчитать минимальное число запросов, сделанных алгоритмом F при расшифровке некоторой переменной x_{i_j}, $j \in \{\log k + 1, \ldots, k\}$, и затем умножить это число на $k - \log k$.

Рассмотрим произвольную частичную фиксацию p переменных из V_n, оставляющую в точности переменные из $V_n \backslash V_{\log k}$ не фиксированными. Проекция f_p от $n - \log k$ переменных зависит существенно лишь от переменной $x_{i_j} = L(p(x_1), \ldots, p(x_{\log k}))$, $j \in \{\log k + 1, \ldots, k\}$. Как было замечено ранее, необходимо подсчитать минимальное число запросов на значение функции, необходимое, чтобы расшифровать существенную переменную функции f_p, а затем умножить это число на число $k - \log k$ переменных в множестве $V_k \backslash V_{\log k}$.

Можно легко показать, что число запросов на значение функции, требуемое для расшифровки произвольной 1-существенной функции от n переменных, ограничено снизу числом $\log n$ [14, 21]. Идея состоит в том, чтобы отвечать на каждый новый запрос алгоритма таким образом, чтобы ответ уменьшил мощность множества возможных существенных переменных не более чем в двое. Отсюда немедленно следует, что алгоритму F требуется как минимум $\log(n-k)$ запросов на значение функции, чтобы расшифровать x_{i_j}, единственную существенную переменную функции f_p. Это заканчивает доказательство леммы.

Используя лемму и результат предыдущего раздела, моментально получаем следующее утверждение.

Утверждение 2. *Для $k, n \in \mathbb{N}$ выполнено*

$$\varphi(\Phi, n, k) \gtrsim \max(k \log(n-k), 2^k)$$

при $k, n \to \infty$.

Мы показали, что минимальное число запросов на значение функции, требуемое алгоритму расшифровки для того, чтобы расшифровать k-существенную разбивающую функцию от n переменных, асимптотически ограничена снизу как величиной 2^k, так и величиной $k \log(n-k)$. Первое — это число запросов, необходимое, чтобы расшифровать k-существенную разбивающую функцию от k переменных, а второе — число запросов, необходимое для того, чтобы найти существенные переменные неизвестной функции от n переменных.

Один из подходов к построению параметро-эффективных алгоритмов расшифровки заключается в том, чтобы сначала определить существенные переменные неизвестной функции, а затем расшифровать проекцию этой функции, задаваемую частичной фиксацией, которая оставляет в точности существенные переменные не фиксированными [1]. В настоящей работе используется несколько иной подход: в цикле выполняются следующие действия. Сначала мы расшифровываем некоторые существенные переменные проекции искомой функции, задаваемой частичной фиксацией, которая оставляет в точности уже найденные существенные переменные не фиксированными. После этого используется полный перебор для того, чтобы найти новую проекцию, которая существенно зависит от хотя бы одной переменной. Когда таких проекций больше не существует, цикл обрывается.

Перейдем к описанию параметро-эффективного по существу условного оптимального алгоритма расшифровки разбивающих функций. Алгоритм последовательно расшифровывает одну существенную переменную за другой. После определения очередной существенной переменной алгоритм фиксирует уже найденные существенные переменные таким образом, чтобы получившаяся проекция искомой функции зависела существенно хотя бы от одной переменной.

Прежде всего, запросим значения функции f на нулевом и единичном наборах. Если они совпадают, то разбивающая функция — константа, и алгоритм завершает свою работу.

Рассмотрим произвольную цепь булевого куба, состоящую из $n+1$ вершин, пусть один конец этой цепи — нулевая вершина куба, другой — единичная. Запросим значение функции f в средней точке этой цепи. Очевидно, это значение не совпадает хотя бы с одним из значений на концах цепи. Выберем произвольную концевую вершину цепи, значение в которой отличается от значения в средней точке и рассмотрим цепь, образованную этой вершиной и средней вершиной цепи. Будем продолжать описанные действия, пока не получим цепь (ребро), состоящую из двух вершин. Переменная, задающая направление этого ребра, является существенной и, значит, мы нашли одну переменную (очевидно, не более чем за $]\log n[$ запросов на значение функции).

Фиксируем найденные к настоящему моменту существенные переменные таким образом, чтобы полученная при этом проекция функции f существенно зависела хотя бы от одной переменной. Если такой фиксации не существует, то найдены все существенные переменные функции f, алгоритм «дорасшифровывает» функцию f, сделав максимум 2^k запросов, и завершает работу. Заметим, что для поиска подходящей фиксации нам потребуется не более $2 \cdot 2^m$ запросов на значение функции, где m — число расшифрованных к настоящему моменту существенных переменных: достаточно запросить по две крайние точки в гранях, полученных при каждой из возможных 2^m фиксаций найденных существенных переменных.

Повторяем описанные действия на грани размерности $n - m$, задаваемой найденной подходящей фиксацией.

Очевидно, сложность описанного алгоритма по порядку не превосходит $k \log n + 2^k$. Воспользовавшись результатами работ [21] и [22], несложно построить алгоритм, сложность которого на классе разбивающих функций не превосходит $k \log n + 2^k$ уже не по порядку, а асимптотически. Тем самым, имеет место теорема 1.

5. Параллельная параметро-эффективная расшифровка разбивающих функций

Начнем с нижней оценки сложности параллельной расшифровки разбивающих функций. Стандартный подход при доказательстве нижних оценок сложности расшифровки функций — построение алгоритма ответов на запросы, который для каждого алгоритма расшифровки строит функцию, на которой алгоритм расшифровки делает достаточное для оценки число запросов.

Рассмотрим конъюнкции $K_1, \ldots, K_k$, k-четное, задаваемые следующими отношениями:

$$\begin{cases} K_1 = x_1, K_2 = \overline{x}_1\overline{x}_2, K_2' = \overline{x}_1 x_2; \\ K_3 = K_2' x_3, K_4 = K_2'\overline{x}_3\overline{x}_4, K_4' = K_2'\overline{x}_3 x_4; \\ \ldots \\ K_{j-1} = K_{j-2}' x_{j-1}, K_j = K_{j-2}'\overline{x}_{j-1}\overline{x}_j, K_j' = K_{j-2}'\overline{x}_{j-1}x_j, \\ \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad j \leqslant k, j \text{ — четное.} \end{cases}$$

Рассмотрим разбивающую функцию $f \in \Phi^{k,n}$ от переменных множества $V_n = \{y_1, \ldots, y_n\}$, существенно зависящую от переменных $y_{i_1} = x_1, \ldots, y_{i_k} = x_k$, такую что $f(K_j) = j$ для любого $j \in \{1, \ldots, k\}$. Множество всевозможных таких функций будем обозначать $\Phi^{k,n}_{plb}$. Здесь plb — сокращение от «parallel lower bound»: класс $\Phi^{k,n}_{plb}$ мы будем использовать для доказательства нижней оценки сложности параллельной расшифровки функций из Φ. Само собой, $\Phi^{k,n}_{plb} \subset \Phi^{k,n}$.

Заметим одно интересное свойство функции $f \in \Phi^{k,n}_{plb}$: если для некоторого набора $a \in \{0,1\}^{V_n}$ выполнено $a(x_{i-1}) = a(x_i)$ для некоторого четного i, то значение $f(a)$ функции f на этом наборе не зависит от переменной x_j для любого $j > i$. Это позволяет учителю не давать никакой информации о существенных переменных с индексами больше заданного пока ему удается обеспечивать для любого запроса, поданного алгоритмом расшифровки к настоящему моменту, равенство хотя бы одной пары «соседних» переменных, индексы которых не превосходят заданный. Указанное свойство функций из

$\Phi_{plb}^{k,n}$ играет ключевую роль при доказательстве нижней оценки сложности параллельной расшифровки функций из Φ.

Далее мы предложим алгоритм ответов на запросы, который будет выбирать сложную для алгоритма расшифровки функцию из множества $\Phi_{plb}^{k,n}$. Мы последовательно рассмотрим алгоритмы для трех случаев: в первом случае запросы от алгоритма расшифровки поступают блоками длины 1 (алгоритм C_1), во втором случае от алгоритма расшифровки поступает 1 большой блок запросов (алгоритм C_2), в третьем случае никаких ограничений на вид поступающих запросов не накладывается, то есть строится алгоритм C_3 ответов на запросы в общем случае.

Во всех трех алгоритмах мы будем использовать понятие *текущей пары расшифровываемых переменных*: мы полагаем, что в любой момент работы алгоритма среди переменных $x_1, \ldots, x_k$ существует пара переменных x_{i-1}, x_i, i — четное, такая что все переменные с номерами, меньшими $i-1$, уже расшифрованы, а о переменных с номерами, превосходящими i, алгоритм расшифровки не имеет никакой информации кроме той, что они не совпадают с уже расшифрованными.

Алгоритм C_1. В начале работы алгоритма F расшифровки разбивающих функций алгоритм F не имеет никакой информации о существенных переменных разгадываемой функции.

Когда F подает первый набор a, мы отвечаем 1, если в этом наборе единиц больше чем нулей, и 2 в противном случае. Если мы ответили 1, то алгоритм F узнал о существенных переменных только то, что $a(x_1) = 1$, то есть что переменная x_1 «лежит среди единиц» набора a. Если же мы ответили 2, то F узнал, что $a(x_1) = 0$ и $a(x_2) = 0$, то есть что и x_1, и x_2 «лежат среди нулей» набора a. Заметим, что в худшем для нас (то есть для алгоритма C_1 ответов на запросы) случае после подачи набора a алгоритм F сократил множество возможных вариантов для переменных x_1 и x_2 вдвое. При этом он ровным счетом ничего не узнал об остальных существенных переменных $x_3, \ldots, x_k$.

Для последующих наборов поступаем аналогичным образом: если в очередном наборе среди компонент, которые все еще могут соответствовать переменным x_1 и x_2, единиц больше, чем нулей, то отвечаем 1, иначе отвечаем 2. Если в очередном запросе среди таких компонент нет ни двух нулей, ни двух единиц, то

- выбираем из этих компонент любые две, пусть их номера — i_1 и i_2, $i_1, i_2 \in \{1, \ldots, n\}$, полагаем $x_1 = y_{i_1}, x_2 = y_{i_2}$ и считаем переменные x_1 и x_2 расшифрованными;
- алгоритм F к настоящему моменту уже подал как минимум $[\log n]$ запросов;
- алгоритм расшифровки F до сих пор не имеет ровным счетом никакой информации о переменных $x_3, \ldots, x_n$. Мы начинаем прятать переменные x_3 и x_4 так же, как прятали x_1 и x_2: если в текущем наборе среди компонент, не равных i_1 и i_2, единиц больше чем нулей, отвечаем 3, иначе отвечаем 4.

На все последующие запросы мы отвечаем аналогичным образом с одним но: если переменные x_{i-1} и x_i — текущие расшифровываемые переменные и для поданного алгоритмом набора a выполнено $a(x_{j-1}) = a(x_j) = b$, $b \in \{0, 1\}$, $j < i - 1$, j — четное, то значение на этом наборе уже однозначно определено: это $j - b$. Именно такой ответ мы и даем алгоритму F на запрос a.

Этим мы заканчиваем описание первого алгоритма ответов на запросы алгоритма расшифровки. Заметим, что он заставляет любой алгоритм расшифровки F сделать как минимум $[\frac{k}{2}][\log(n-k)]$ запросов на значение функции: чтобы определить переменные x_1 и x_2 ему нужно как минимум $[\log n]$ запросов, чтобы определить x_3 и x_4 — как минимум $[\log(n-2)]$ запросов и т. д. Отсюда следует, что если всего алгоритм расшифровки сделал N запросов на значение функции, то он расшифровал не более $[2\frac{N}{\log(n-k)}]$ переменных.

Мы доказали следующую лемму.

Лемма 2. *Существует такой алгоритм ответов на запросы, что для любого алгоритма F расшифровки функций из Φ на любом множестве запросов мощности N, поданном алгоритмом F (необязательно в виде одного блока), он расшифрует не более чем $[2\frac{N}{\log(n-k)}]$ переменных.*

Будем неформально называть сформулированное в лемме 2 свойство описанного алгоритма «$2s$ *переменных за* $s \log n$ *запросов*».

Как мы видим, в случае алгоритма C_1 нам приходится отдавать по две переменные примерно каждые $\log n$ запросов: если требуется

последовательно отвечать на каждый очередной запрос алгоритма расшифровки, то мы (учитель) не можем «прятать» две переменные дольше чем $[\log n]$ запросов. На самом деле, чтобы долго «прятать» две переменные, нам достаточно построить два равных столбца в матрице запросов. И если нам приходится строить эти столбцы по мере поступления запросов от алгоритма расшифровки, то есть по мере формирования матрицы запросов, то в худшем для нас случае мы сможем построить столбцы длины лишь $[\log n]$. Если же мы сразу имеем матрицу запросов в виде одного блока, то при $N \gg \log n$ мы можем выбрать существенно более длинные столбцы: если в блоке N запросов, то мы можем использовать не менее $\frac{3}{4}N$ запросов, для того, чтобы «прятать» две существенные переменные, не выдавая никакой информации о других существенных переменных. Именно это замечание и лежит в основе алгоритма C_2.

Алгоритм C_2. Этот алгоритм имеет доступ сразу ко всем N запросам на значение функции. Согласно границе Плоткина [23] в любой булевской матрице размера $N \times n$, $N < \frac{n}{2}$, существует два столбца, расстояние Хэмминга (число различных компонент) между которыми не превосходит $[\frac{N}{2}]$. Выберем такие два столбца в блоке (матрице) запросов, полученном от алгоритма расшифровки. Рассмотрим тот столбец из этих двух, в котором среди упомянутых различных компонент не менее половины единиц, и обозначим соответствующую ему переменную через x_1. Переменную, соответствующую второму столбцу, обозначим через x_2.

Вспомним нашу цель: мы должны сопоставить каждой строке матрицы запросов некоторое значение расшифровываемой алгоритмом расшифровки и одновременно создаваемой нами функции $f \in \Phi_{plb}^{k,n}$. Поэтому, положим $f(a) = 1$, если $a(x_1) = 1$ и $f(a) = 2$, если $a(x_1) = 0$ и $a(x_2) = 0$. Если $a(x_1) = 0$, а $a(x_2) = 1$, то пока не будем сопоставлять набору a никакое значение функции f. Очевидно, какую-либо информацию об остальных существенных переменных можно получить только на тех наборах, где $x_1 = 0$, а $x_2 = 1$, а по построению таких наборов в матрице запросов не более чем $[\frac{N}{4}]$.

Выберем в матрице запросов столбцы, соответствующие переменным x_3 и x_4, таким образом, чтобы информацию о существенных переменных не из множества $\{x_1, x_2, x_3, x_4\}$ могло содержать только

$[\frac{N}{4^2}]$ запросов (строк) этой матрицы. Для этого для каждого непомеченного ($f(a) \neq 1$ и $f(a) \neq 2$)) запроса a, такого что $a(x_3) = 1$, положим $f(a) = 3$, для каждого непомеченного запроса a, такого что $a(x_3) = 0$ и $a(x_4) = 0$, положим $f(a) = 4$.

Продолжая аналогичным образом, мы можем так выбирать столбцы, соответствующие очередной паре переменных (x_{i-1}, x_i), i-четное, что множество запросов, на которых алгоритм расшифровки может получить какую-либо информацию о переменных x_j, $j > i$, $j \leqslant k$, сокращается как минимум в 4 раза после такого выбора. Другими словами, «отдавая» всего лишь две переменные алгоритму расшифровки, мы сокращаем множество запросов, на которых алгоритм расшифровки может узнать что-то новое, в 4 раза. Отсюда после выбора $2m$ столбцов (переменных) останется лишь $\frac{N}{4^m}$ запросов, для которых мы не фиксировали значение f. Заметим, что если в матрице с n столбцами меньше, чем $\log n$ строк, то среди ее столбцов всегда найдется два равных столбца. Поэтому, нас интересует, при каких m выполнено $\frac{N}{4^m} < \log n$. Имеем: $4^m > \frac{N}{\log n}$, $m > \log_4 \frac{N}{\log n}$, $2m > \log \frac{N}{\log n}$. Значит, «отдав» $[\log \frac{N}{\log n}]$ переменных, «необработанной» (с еще не фиксированными значениями f) останется подматрица исходной матрицы с как минимум двумя равными столбцами. Сопоставив паре равных столбцов очередную пару переменных, мы закончим построение функции f. Очевидно, суммарно алгоритм расшифровки может расшифровать не более чем $[\log \frac{N}{\log n}] + 2$ переменные функции f.

Мы доказали следующую лемму.

Лемма 3. *Существует такой алгоритм ответов на запросы, что для любого алгоритма F расшифровки функций из Φ на любом блоке запросов мощности $N > 2 \log n$, поданном алгоритмом F, он расшифрует не более чем $[\log \frac{N}{\log n}] + 2$ переменных.*

Будем неформально называть сформулированное в лемме 3 свойство описанного алгоритма «$\log s$ *переменных за* $s \log n$ *запросов*».

Перейдем к основной задаче — построению алгоритма C_3 ответов на запросы, который заставляет любой «близкий к оптимальному» алгоритм расшифровки функций из Φ, подавать достаточно много блоков запросов на значение функции, чтобы расшифровать функ-

цию, загадываемую алгоритмом ответов на запросы. Алгоритм C_3 объединяет свойства алгоритмов C_1 и C_2.

Из лемм 2 и 3, то есть из свойства алгоритма C_1 ответов на запросы отдавать «$2s$ переменных за $s \log n$ запросов» и свойства алгоритма C_2 отдавать «$\log s$ *переменных за* $s \log n$ *запросов*», следует, что алгоритм C_3 мог бы использовать следующий подход: пользуемся алгоритмом C_1, пока не встретили достаточно большой блок. На этом блоке используем C_2. Затем снова используем C_1, пока не встретим очередной достаточно большой блок, на котором снова используем C_2, и так далее.

Теперь дадим формальное описание алгоритма C_3.

Нам понадобится определение *внутренней части блока*. Пусть алгоритм расшифровки последовательно подает 3 блока: B_1, B_2 и B_3. Рассмотрим ситуацию, в которой мы хотим на блоке B_1 использовать алгоритм C_1, на блоке B_2 — C_2, а на блоке B_3 — снова C_1 (предполагаем, что мощности этих блоков позволяют так сделать). Будем последовательно (в порядке их следования в блоке) отвечать на запросы из B_1. Когда C_1 закончит свою работу на B_1, у нас останется текущая пара переменных, которая до конца не раскрыта алгоритму расшифровки. Поэтому, используем необходимое число первых наборов из блока B_2, чтобы «доотдать» текущую пару переменных. К оставшимся наборам блока B_2 применим алгоритм C_2. При этом когда C_2 будет заканчивать свою работу, то есть когда число непомеченных наборов в блоке B_2 станет меньшим, чем $\log n$, мы не будем использовать C_2 на этих непомеченных наборах, а снова запустим на них C_1, после чего продолжим использовать C_1 и на блоке B_3. Часть блока B_2, на которой мы использовали алгоритм C_2 ответов на запросы, будем называть *внутренней*.

Алгоритм C_3.

Повторяем в цикле:

- используем алгоритм C_1, пока не получим от алгоритма расшифровки блок, внутренняя часть которого больше $2 \log n$;
- окончательно раскрываем текущую пару переменных, используя первые наборы из этого блока. Используем алгоритм C_2 для наборов из внутренней части.

Теперь мы готовы сформулировать основную лемму.

Лемма 4. *Пусть k, n фиксированы и сложность некоторого алгоритма расшифровки F на классе $\Phi^{k,n}$ разбивающих функций удовлетворяет неравенству*

$$\varphi(F, \Phi^{k,n}) \leqslant (1+\varepsilon)k \log n$$

для некоторой константы $\varepsilon > 0$. Тогда существует константа $\delta > 0$, такая что для параллельной сложности алгоритма F на этом классе выполнено

$$\varphi_p(F, \Phi^{k,n}) \geqslant \delta k.$$

Доказательство. Из условия утверждения следует, что для любой функции $f \in \Phi^{k,n}$ выполнено $\varphi(F, f) \leqslant (1+\varepsilon)k \log n$. Чтобы доказать утверждение, нам требуется найти такую функцию $f \in \Phi^{k,n}$, что $\varphi_p(F, f) \geqslant \delta k$. Покажем, что алгоритм C_3 ответов на запросы дает именно такую функцию f.

Рассмотрим блоки, поданные алгоритмом расшифровки F в случае, когда в роли учителя выступает алгоритм C_3. Пусть k_b — число блоков, на внутренних частях которых алгоритм C_3 использует алгоритм C_2 (будем называть такие блоки *большими*), k_s — число остальных блоков (будем называть такие блоки *маленькими*). Пусть N_i — мощность внутренней части i-го большого блока. Тогда по лемме 3 на запросах из внутренних частей больших блоков алгоритм F расшифрует не более чем $\sum_{i=1}^{k_b} [\log \frac{N_i}{\log n}]$ переменных. Данная величина достигает максимального значения, если все внутренние части имеют одинаковую мощность. Отсюда, вспоминая, что суммарная мощность внутренних частей по условию не может превосходить $(1+\varepsilon)k \log n$, получаем что на запросах на внутренних частях больших блоков F не расшифрует более чем $k_b [\log \frac{(1+\varepsilon)k}{k_b}]$ переменных.

Теперь найдем число переменных, расшифрованных на запросах, обработанных алгоритмом C_1 (то есть на маленьких блоках и на частях больших блоков, не являющихся внутренними). Ни в каком маленьком блоке не может быть больше $4 \log n$ запросов, поскольку иначе в нем бы обязательно была внутренняя часть и блок был бы

большим. Значит, всего в маленьких блоках не более чем $4k_s \log n$ запросов. Если учесть еще части больших блоков, не являющиеся внутренними, получаем, что всего на дополнении к внутренним частям больших блоков делается не более чем $4(k_s + k_b)\log n$ запросов. Значит, по лемме 2 здесь алгоритм расшифровки может определить не более чем $8(k_s + k_b)$ переменных.

В итоге получаем, что алгоритм расшифровки F в сумме расшифрует не более чем $k_b[\log \frac{(1+\varepsilon)k}{k_b}]+8(k_s+k_b)$ переменных. Поскольку алгоритм F должен расшифровать все переменные, требуется, чтобы эта величина была не меньше k. Предположим теперь, что доказываемое утверждение не выполнено, то есть для любой константы δ выполнено $k_b + k_s < \delta k$. Следовательно, и $k_b < \delta k$. Возьмем $\alpha = \frac{k}{k_b}$. Тогда $k_b = \frac{k}{\alpha} < \delta k$ и $\frac{1}{\alpha} < \delta$. Возьмем $\delta = \frac{1}{16(1+\varepsilon)}$. Тогда при $\varepsilon > 0$

$$k_b \log \frac{(1+\varepsilon)k}{k_b} + 8(k_s + k_b) = k\frac{\log((1+\varepsilon)\alpha)}{\alpha} + 8(k_s + k_b) <$$
$$< k\frac{\log((1+\varepsilon)\alpha)}{\alpha} + 8\delta k < (\delta \log((1+\varepsilon)\frac{1}{\delta}) + 8\delta) \cdot k =$$
$$= \frac{\log(16(1+\varepsilon)^2) + 8}{16(1+\varepsilon)}k = \frac{12 + 2\log(1+\varepsilon)}{16(1+\varepsilon)}k < \frac{14}{16}k < k.$$

Противоречие. Лемма доказана.

Неформально, лемма 4 показывает, что среди близких к оптимальным (в смысле обычной сложности) алгоритмов расшифровки разбивающих функций не стоит искать алгоритмы с параллельной сложностью, по порядку меньшей чем k: таких алгоритмов не существует.

Из леммы 4 получаем следующее следствие.

Следствие 2. *Для $k, n \in \mathbb{N}$ и произвольной константы $c > 1$ найдется константа $\delta > 0$, такая что $\varphi_p(\Phi, n, k, c \cdot k \log n) > \delta k$.*

Для доказательства теоремы 2 осталось построить оптимальный параметро-эффективный алгоритм расшифровки разбивающих функций, параллельная сложность которого на классе разбивающих функций по порядку не превосходит k.

Пусть $V_n = \{x_1, \ldots, x_n\}$ — множество булевых переменных и $W \subseteq V_n$ — его подмножество. Пусть f — k-существенная разбивающая функция от переменных из V_n. Рассмотрим булев куб $\{0,1\}^{V_n}$ и частичную фиксацию p переменных из V_n, оставляющую в точности переменные из W не фиксированными. Пусть

$$G = \{p/a | a \in \{0,1\}^{V_n}\}$$

— грань булевого куба $\{0,1\}^{V_n}$. Будем говорить, что фиксация p индуцирует G и что переменные в W являются G-не фиксированными. Обозначим фиксацию $p/1$ через 1^G, а фиксацию $p/0$ через 0^G (см рис. 1). Говоря неформально, это наибольший и наименьший наборы грани G соответственно.

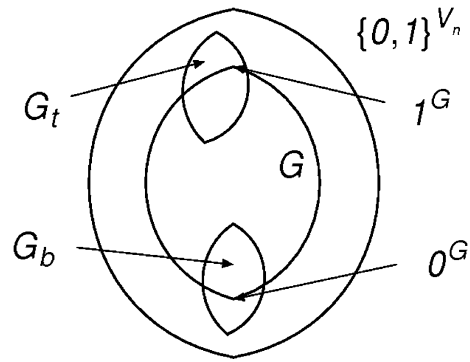

Рис. 1. Грань куба $\{0,1\}^{V_n}$.

Обозначим грань $f^{-1}(f(1^G))$ через G_t и $f^{-1}(f(0^G))$ через G_b. Пусть p_{G_t} — это частичная фиксация, индуцирующая G_t, а p_{G_b} — частичная фиксация, индуцирующая G_b.

Утверждение 3. *Пусть* $x \in \{0,1\}^{V_n}$. *Если* $p_G(x) = *$ *и* $p_{G_t}(x) \neq *$, *то* $p_{G_t}(x) = 1$. *Если* $p_G(x) = *$ *и* $p_{G_b}(x) \neq *$, *то* $p_{G_b}(x) = 0$.

Доказательство. Докажем первую часть, вторая часть утверждения доказывается аналогично. Пусть $p_{G_t}(x) = 0$. Тогда $1^G(x) = 0$, поскольку набор 1^G принадлежит грани G_t. Но 1^G — это наибольший набор в G, откуда $a(x) = 0$ для любого набора $a \in G$. Значит,

$p_G(x) = 0$, противоречие нашему предположению, что $p_G(x) = *$. Утверждение доказано.

Рассмотрим произвольную грань булевого куба $G \subseteq \{0,1\}^{V_n}$, такую что $f(0^G) \neq f(1^G)$. Предположим, что по крайней мере одна существенная переменная функции f является G-не фиксированной. Пусть A — матрица, такая что

- каждая строка матрицы A принадлежит грани G;
- все столбцы A, соответствующие G-не фиксированным переменным из V_n попарно различны. Более того, каждый такой столбец не является ни единичным, ни нулевым.

В [14] похожее множество строк было названо разделяющим множеством. Мы также будем использовать это название и называть матрицу A G-разделяющей, поскольку она сопоставляет уникальный столбец каждой G-не фиксированной переменной в V_n и, значит, «разделяет» G-не фиксированные переменные.

Через J обозначим множество всех известных переменных функции f. Положим $b = f(0^G)$, $t = f(1^G)$.

Рассмотрим рекурсивную процедуру $S(A, b, t)$, которая расшифровывает как минимум одну G-не фиксированную существенную переменную функции f.

Пусть v — набор. Для каждой строки A_i матрицы A делаем запрос на значение функции при помощи набора A_i и полагаем $v_i = 0$, если $f(A_i) = b$, $v_i = 1$, если $f(A_i) = t$ и $v_i = 2$ иначе. Возможны два варианта.

a) матрица A имеет строку c, такую что $f(c) \neq b$ и $f(c) \neq t$. Пусть G^1 — такая грань куба, что $0^{G^1} = c$ и $1^{G^1} = 1^G$. Аналогично пусть G^0 — такая грань куба, что $0^{G^0} = 0^G$ и $1^{G^0} = c$ (см. рис. 2). Легко заметить, что все строки матрицы $A \vee c$ принадлежат грани G^1, а все строки матрицы $A \wedge c$ принадлежат грани G^0. Мы последовательно запускаем $S(A \vee c, f(c), t)$ и $S(A \wedge c, b, f(c))$.

b) для каждой строки c матрицы A либо $f(c) = t$, либо $f(c) = b$. Находим в матрице A столбец, равный v. Переменная, соответствующая этому столбцу, является существенной, мы добавляем ее в J.

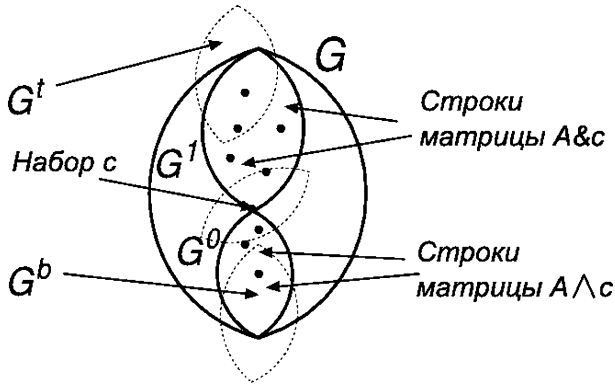

Рис. 2. Матрица A имеет «разделяющую» строку c.

Утверждение 4. *Для любой матрицы A, возникающей в процессе выполнения процедуры S и такой, что на каждой ее строке функция f принимает одно из крайних значений (t или b), выполнено:*

1) *всегда существует один и только один столбец матрицы A, равный v;*
2) *переменная, соответствующая этому столбцу, является существенной;*
3) *множество J не содержит эту переменную.*

Доказательство. Понятно, что для любых двух различных граней G_1 и G_2, задаваемых одной разбивающей функцией, существует переменная x, которая является и G_1-, и G_2-фиксированными, причем $p_{G_1}(x) \neq p_{G_2}(x)$. Пусть x — такая переменная для граней G_t и G_b. Она является G-не фиксированной, поскольку $1^G(x) = p_{G_t}(x) \neq p_{G_b}(x) = 0^G(x)$. Из утверждения 1 следует, что $p_{G_t}(x) = 1, p_{G_b}(x) = 0$. Для i-й строки матрицы A имеем $f(A_i) = t \Leftrightarrow A_i \in G_t \Rightarrow A_i(x) = p_{G_t}(x) = 1 = v_i$. Аналогично, если $f(A_i) = b$, то $A_i(x) = v_i$. Таким образом, равенство $A_i(x) = v_i$ выполнено для каждой строки матрицы A, и столбец матрицы A, соответствующий переменной x — это тот столбец, что мы искали.

На самом деле, существует лишь одна G-не фиксированная переменная x, такая что $* \neq G_t(x) \neq G_b(x) \neq *$. Это следует из того,

что все столбцы матрицы A, не являющиеся ни нулевыми, ни единичными, попарно различны, а значит, не может существовать двух столбцов с различными номерами, равных v. Отсюда сразу получаем, что x — это существенная переменная функции f, и это единственная переменная, «разделяющая» G_t и G_b в G. Здесь следует заметить, что каждую существенную переменную функции f алгоритм расшифровывает лишь один раз в процессе своей работы. Это следует из того, что никакая переменная не может быть одновременно и G_1, и G_0-не фиксированной, где G_1 и G_0 — грани из описания процедуры $S(A, b, t)$. Утверждение доказано.

Теперь опишем рекурсивную процедуру P, которая расшифровывает все существенные переменные функции f.

Мы начинаем с полного перебора всех возможных частичных фиксаций переменных из V_n, таких что в точности переменные из $V_n - J$ остаются не фиксированными. Возможны два варианта.

a) мы находим частичную фиксацию p, такую что $f(p/1) = f(p/0)$. В этом случае по крайней мере одна существенная переменная функции f является G-не фиксированной, где $0^G = p/0$ и $1^G = p/1$. Значит, мы фиксируем произвольную G-разделяющую матрицу A и запускаем $S(A, f(0^G), f(1^G))$;

b) такой частичной фиксации не существует. В этом случае J уже содержит все существенные переменные функции f и мы выходим из процедуры.

Теперь мы готовы перейти к описанию собственно алгоритма F расшифровки k-существенных разбивающих функций от n переменных для любого k. Алгоритм F инициализирует множество J, полагая $J = \varnothing$, и запускает процедуру P. После этого F выбирает произвольную частичную фиксацию, которая оставляет в точности переменные в J не фиксированными и делает $2^{|J|}$ запросов на значение функции, чтобы завершить процесс расшифровки.

Лемма 5. *Для приведенного выше алгоритма F и любой функции $f \in \Phi^{k,n}$ выполнено $\varphi(F, f) \leqslant 2(k \log n + 2^k)$ и $\varphi_p(F, f) \leqslant 3k$.*

Доказательство. Для доказательства леммы необходимо подсчитать общее число запросов на значение функции, сделанное алгоритмом F. Можно легко показать, что число строк любой разделяющей матрицы A не превосходит $[\log n]$. Предположим, что полный проход процедуры S добавляет k' существенных переменных в множество J. Мы можем сопоставить бинарное дерево этому проходу, имеющее в точности k' листьев. В каждой вершине этого дерева мы делаем не более чем $[\log n]$ запросов на значение функции поскольку эти запросы являются строками разделяющей матрицы. Поскольку число вершин бинарного дерева не превосходит удвоенного числа его листьев, мы получаем отсюда, что общее число запросов на значение функции, сделанных во время этого прохода, не превышает $2k'[\log n]$. Таким образом, общее число запросов на значение функции, сделанных во всех проходах процедуры S, не превосходит $2k[\log n]$.

Теперь заметим, что полный перебор в процедуре P делает не более чем $2^{|J|}$ запросов на значение функции. Поскольку $|J|$ в любой момент не превосходит k и $\sum_{i=1}^{k} 2^k < 2^{k+1}$, мы заключаем, что все запуски полного перебора требуют не более чем 2^{k+1} запросов на значение функции. На самом деле, легко показать, что они требуют не более 2^k запросов (см. [22]).

Осталось заметить, что параллельная сложность описанного алгоритма не может превышать суммы числа вершин описанных в лемме 5 бинарных деревьев и числа самих деревьев, а значит она не превосходит $3k$. Лемма доказана.

Теорема 2 непосредственно следует из следствия 2 и леммы 5.

Список литературы

[1] Blum A., Hellerstein L., Littlestone N. Learning in the presence of finitely or infinitely many irrelevant attributes // Journal of Computer and System Sciences. 50. P. 32–40. 1995.

[2] Manning C. D., Raghavan P., Schutze H. Introduction to Information Retrieval // Cambridge University Press, 2008.

[3] Littlestone N. Learning Quickly When Irrelevant Attributes Abound: A New Linear-threshold Algorithm // Machine Learning. 2 (4). P. 285–318. 1987.

[4] Bshouty N., Hellerstein L. Attribute efficient learning with queries // Journal of Computer and System Sciences. 56. P. 310–319. 1998.

[5] Valiant L. A Theory of the Learnable // Comm. ACM. 27. P. 1134–1142. 1984.

[6] Blum A. Learning a Function of r Relevant Variables // COLT 2003. Open problems.

[7] Mossel E., O'Donnell R., Servedio R. Learning juntas // Proceedings of the 35th Annual ACM Symposium on Theory of Computing. 2003.

[8] Arpe J., Reischuk R. Learning Juntas in the Presence of Noise // Theoret. Comput. Sci. 384 (1). P. 2–21. 2007.

[9] Feldman V. Attribute-Efficient and Non-adaptive Learning of Parities and DNF Expressions // The Journal of Machine Learning Research archive. Vol. 8. P. 1431–1460. 2007.

[10] Angluin D. Queries and Concept Learning // Machine Learning. Vol. 2. P. 319–342. 1988.

[11] Коробков В. К. О монотонных функциях алгебры логики // Проблемы кибернетики. Вып. 13. М.: Наука, 1965.

[12] Ансель Ж. О числе монотонных булевых функций от n переменных // Кибернетич. сб. Новая серия. Вып. 5. М.: Мир, 1968. С. 53–57.

[13] Damaschke P. Adaptive Versus Nonadaptive Attribute-Efficient Learning // Machine Learning. 41. P. 197–215. 2000.

[14] Uehara R., Tsuchida K., Wegener I. Optimal attribute-efficient learning of disjunction, parity, and threshold functions // Lecture Notes In Computer Science. Vol. 1208. 1997.

[15] Damaschke P. Parallel Attribute-Efficient Learning of Monotone Boolean Functions // Lecture Notes in Computer Science. Algorithm Theory — SWAT 2000. P. 473–479.

[16] Mitchell T. Machine Learning. McGraw Hill Higher Education, 1997.

[17] Kearns M. J., Vazirani U. V. An Introduction to Computational Learning Theory. The MIT Press, 1994.

[18] Кудрявцев В. Б., Андреев А. Е., Гасанов Э. Э. Теория тестового распознавания. М.: ФИЗМАТЛИТ, 2007.

[19] Кудрявцев В. Б., Гасанов Э. Э., Долотова О. А., Погосян Г. Р. Теория тестирования логических устройств. М.: ФИЗМАТЛИТ, 2006.

[20] Гасанов Э. Э., Кудрявцев В. Б. Теория хранения и поиска информации. М.: ФИЗМАТЛИТ, 2002.

[21] Осокин В. В. Асимптотически оптимальный алгоритм расшифровки разбиения булевого куба на подкубы // Интеллектуальные системы. Т. 11. 2007. С. 587–606.

[22] Осокин В. В. О сложности расшифровки разбиения булевого куба на подкубы // Дискретная математика. Т. 20, вып. 2. 2008. С. 46–62.

[23] Плоткин М. Двоичные коды с заданным минимальным расстоянием // Кибернетический сборник. 7. С. 60–73. 1963.