О параллельной параметро-эффективной расшифровке псевдо-булевских функций

В. В. Осокин

В работе рассмотрена задача параллельной параметро-эффективной расшифровки псевдо-булевских функций в рамках модели точной расшифровки. Для интервально-постоянных функций, примерами которых являются псевдо-булевские монотонные функции и разбивающие функции, предложен оптимальный по порядку алгоритм параметро-эффективной расшифровки. Предложен параметро-эффективный алгоритм расшифровки некоторых подклассов интервально-постоянных функций с оптимальной по порядку параллельной сложностью и оптимальной по порядку обычной сложностью.

Ключевые слова: сложность расшифровки функций, параметро-эффективная расшифровка, параллельная расшифровка, псевдо-булевские функции, интервально-постоянные функции, монотонные функции, модель точной расшифровки.

1. Введение

Рассматривается задача расшифровки функций в рамках точной модели расшифровки при помощи запросов на значение функции [1, 2]. Эта задача состоит в следующем. Имеется учитель, который загадывает функцию из некоторого известного класса. Имеется ученик (алгоритм расшифровки), который задает учителю блок вопросов. Каждый вопрос — это значение аргумента функции. Учитель в ответ сообщает значения загаданной функции на данных вопросах. По полученным в ответ результатам ученик формирует следующий блок вопросов и передает их учителю, и процесс итеративно продол-

жается до тех пор пока ученик полностью не разгадает загаданную функцию. Сложность расшифровки — это количество заданных вопросов, а параллельная сложность расшифровки — это количество заданных блоков. Если функции из загадываемого класса зависят от большого числа переменных, но известно, что загаданнная функция зависит существенно от малого числа переменных, и сложность расшифровки зависит от числа существенных переменных (даже если это число неизвестно ученику) и слабо зависит от общего числа переменных (это число всегда известно ученику), то говорят об параметро-эффективной расшифровке.

Мы рассматриваем задачу параллельной параметро-эффективной расшифровки псевдо-булевских функций в рамках точной модели расшифровки при помощи запросов на значение функции.

Рассмотрим такую псевдо-булевскую функцию, что если она принимает одно и то же значение на двух сравнимых наборах $\alpha < \beta$, то она принимает то же значение и на любом наборе γ, таком что $\alpha < \gamma < \beta$. Такие функции будем называть *интервально-постоянными* и обозначать класс таких функций через ICF. Список подклассов ICF включает в себя псевдо-булевские монотонные функции [3] и разбивающие функции [4, 5]. Отсюда следует, что он также содержит класс булевских монотонных функций и некоторые другие классы, описание которых дано в разделе 2.2.

Настоящее исследование базируется на результатах П. Дамашке [6] и В. Осокина по расшифровке булевских монотонных функций и псевдо-булевских разбивающих функций соответственно [7]. Класс ICF является естественным обобщением обоих классов функций. Мы считаем, что свойство из определения ICF является ключевым свойством, позволившим П. Дамашке построить параметро-эффективный параллельный алгоритм расшифровки монотонных функций [6]. В настоящей работе предложено обобщение алгоритмов расшифровки монотонных булевских функций и расшифровки разбивающих функций, которое расшифровывает ICF параметро-эффективно и имеет небольшую параллельную сложность. Доказаны нижние оценки, показывающие, что для некоторых подклассов ICF наш алгоритм имеет и оптимальную по порядку сложность, и оптимальную по порядку параллельную сложность.

Отрицательной стороной полученного алгоритма является то, что в общем случае он не является оптимальным для подклассов ICF, которые могут быть расшифрованы (не параметро-эффективно) за менее чем 2^n запросов на значение функции в худшем случае. В частности, Ж. Ансель показал, что монотонные функции могут быть расшифрованы при помощи всего лишь по порядку $\frac{2^n}{\sqrt{n}}$ запросов на значение функции в худшем случае [8]. В общем случае ни алгоритм П. Дамашке, ни наш алгоритм не является оптимальным для монотонных функций. В связи с этим получен параметро-эффективный (но не параллельный) алгоритм расшифровки ICF и доказана его оптимальность на нескольких подклассах ICF, включая класс булевских монотонных функций.

Неполный список исследований, проведенных в рамках модели точной расшифровки при помощи запросов на значение функции, но не рассматривающих ни задачу параметро-эффективной расшифровки, ни задачу параллельной расшифровки, включает в себя [2, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17].

Некоторые общие результаты по сложности параметро-эффективной (но не параллельной) расшифровки в модели точной расшифровки при помощи запросов на значение функции получены в [18]. И. Вегенер и др. [19] получили точные оценки сложности параметро-эффективной расшифровки некоторых классов булевских функций в этой модели.

Параллельная (но не параметро-эффективная) расшифровка в модели точной расшифровки при помощи запросов на значение функции исследовалась в [20]. Наконец, параллельная параметро-эффективная расшифровка в этой модели рассматривалась в работах П. Дамашке [6, 21, 22].

В большинстве упомянутых работ рассматривается расшифровка булевских функций. В настоящей работе исследована задача параллельной параметро-эффективной расшифровки псевдо-булевских функций. Основные свойства псевдо-булевских функций описаны в [23]. Некоторые классы псевдо-булевских функций описаны в [3].

Параметро-эффективная расшифровка (иначе, расшифровка хунт) исследовалась в рамках нескольких моделей стимулирующего обучения и обучения с учителем. Параметро-эффективный алгоритм расшифровки пороговых функций в рамках модели ограничен-

ной ошибки стимулирующего обучения предложен в [24]. Впоследствии различные аспекты параметро-эффективной расшифровки были рассмотрены в [18, 25]. Пассивное обучение с учителем, то есть расшифровка по случайной выборке обычно изучается в рамках модели, получившей название вероятно примерно точной (PAC) модели обучения [26]. Параметро-эффективная расшифровка по случайной равномерно распределенной выборке в PAC модели является открытой проблемой [27]. Для ознакомления с недавними результатами по пассивной расшифровке хунт см. [28, 29, 30]. Параллельная расшифровка рассматривалась в [20, 31, 32].

Среди направлений, близких к рассматриваемому направлению расшифровки функций, можно выделить теорию тестового распознавания [33, 34].

Автор выражает благодарность д.ф.-м.н. проф. Гасанову Э. Э. за постановку задачи и помощь в работе.

2. Основные понятия

2.1. Базовые определения

Пусть $V_n = \{x_1, \ldots, x_n\}$ — множество булевских переменных. В соответствии с [33] будем называть множество $\{0,1\}^{V_n} = \{f : V_n \to \{0,1\}\}$ *n-мерным булевым кубом над переменными* $x_1, \ldots, x_n$.

Элементы $\{0,1\}^{V_n}$ будем называть *фиксациями переменных из* V_n. Рассматривая функции с переменными из V_n, будем считать, что множество V_n некоторым образом упорядочено. В таком случае мы будем также понимать фиксацию $\alpha \in \{0,1\}^{V_n}$ как набор $\alpha = (\alpha_1, \ldots, \alpha_n)$, где $\alpha_i = \alpha(x_i)$. Мы будем i-ю компоненту фиксации α обозначать через α_i. Для фиксаций $\alpha = (\alpha_1, \ldots, \alpha_n)$ и $\beta = (\beta_1, \ldots, \beta_n)$ пишем $\alpha \circ \beta$, $\circ \in \{\geqslant, \leqslant, =\}$, если для любого $i \in \{1, 2, \ldots, n\}$ выполнено $\alpha_i \circ \beta_i$. Пишем $\alpha < \beta$ ($\alpha > \beta$), если $\alpha \leqslant \beta$ ($\alpha \geqslant \beta$) и $\alpha \neq \beta$. Если для двух фиксаций α и β либо $\alpha \geqslant \beta$, либо $\beta \geqslant \alpha$, то мы называем фиксации α и β *сравнимыми*.

Подмножество $\{0,1\}^{V_n}$, состоящее из фиксаций, фиксирующих в точности i переменных в V_n единицами, $i \in \{1, \ldots, n\}$, будем называть *i-м слоем булевого куба* $\{0,1\}^{V_n}$. Слой с номером $[\frac{n}{2}]$ назовем *средним слоем* куба $\{0,1\}^{V_n}$.

Если $\alpha \in \{0,1\}^{V_n}$, $x_i \in V_n$ и $a \in \{0,1\}$, то через $\alpha_{x_i \leftarrow a}$ обозначим такую фиксацию переменных из V_n, что $\alpha_{x_i \leftarrow a}(x_j) = \alpha(x_j)$ для всех $j \neq i$ и $\alpha_{x_i \leftarrow a}(x_i) = a$. Другими словами, фиксация $\alpha_{x_i \leftarrow a}$ присваивает те же значения всем переменным из V_n, как и фиксация α с тем исключением, что она присваивает значение a переменной x_i. *Перестановкой* фиксации α будем называть фиксацию, соответствующую произвольной перестановке компонент набора α.

Псевдо-булевской функцией над V_n называется отображение $f : \{0,1\}^{V_n} \to \mathbb{N}$, где $\mathbb{N}$ есть множество натуральных чисел. Будем говорить, что функция f *присваивает значение* a фиксации α, если $f(\alpha) = a$. Переменная $x_i \in V_n$ называется *существенной переменной функции* $f : \{0,1\}^{V_n} \to \mathbb{N}$, если существует такая фиксация $\alpha \in \{0,1\}^{V_n}$, что $f(\alpha) \neq f(\alpha_{x_i \leftarrow \overline{\alpha(x_i)}})$. Если x_i — существенная переменная, будем говорить, что f *существенно зависит от* x_i.

Частичной фиксацией β *переменных* из V_n будем называть отображение из V_n в $\{0,1,*\}$. Будем говорить, что переменная $x \in V_n$ *фиксирована* частичной фиксацией β, если $\beta(x) \neq *$. Пусть $W \subseteq V_n$ — переменные, фиксированные частичной фиксацией β. Пусть α — частичная фиксация переменных из некоторого множества $V \subseteq V_n$, содержащего $V_n - W$, такая что α фиксирует все переменные из $V_n - W$. Тогда через β/α обозначим фиксацию переменных из V_n, которая согласуется с β на всех переменных, фиксированных β, и согласуется с α на остальных переменных.

Рассмотрим частичную фиксацию β переменных из V_n, которая не фиксирует в точности переменные из $W \subseteq V_n$. Пусть $G = \{\beta/\alpha | \alpha \in \{0,1\}^{V_n}\}$. Множество $G \subseteq \{0,1\}^{V_n}$ будем называть *гранью куба, задаваемой частичной фиксацией* β. Будем говорить, что фиксация β *порождает* G и что переменные в W являются G*-не фиксированными*. Обозначим фиксацию $\beta/1$ через 1^G, а фиксацию $\beta/0$ через 0^G, где 1 и 0 — фиксации, соответствующие единичному и нулевому наборам соответственно. Неформально, 1^G и 0^G — наибольшая и наименьшая фиксации в G. Если $G \subseteq \{0,1\}^{V_n}$ — это грань минимальной мощности, содержащая фиксации α и β, $\alpha, \beta \in \{0,1\}^{V_n}$, то будем говорить, что G *задана фиксациями* α и β. Очевидно, если α и β сравнимы, например $\alpha \geqslant \beta$, то $1^G = \alpha$ и $0^G = \beta$, где G — грань, задаваемая фиксациями α и β.

Пусть $V \subseteq V_n$ и β — частичная фиксация переменных из V_n, которая не фиксирует в точности переменные из V. Тогда для $f : \{0,1\}^{V_n} \to \mathbb{N}$ обозначим через f_β функцию над $\{0,1\}^V$, такую что $f_\beta(\alpha) = f(\beta/\alpha)$ для всех $\alpha \in \{0,1\}^V$. Функция f_β называется *проекцией* f, задаваемой частичной фиксацией β.

Будем писать $g(n) = \bar{o}(1)$, если $\lim\limits_{n\to\infty} g(n) = 0$; $A(n) = \bar{o}(B(n))$, если $A(n) = B(n) \cdot \bar{o}(1)$. Скажем, что $A(n)$ *асимптотически не превосходит* $B(n)$ при $n \to \infty$ и обозначим $A \lesssim B$ (или $B \gtrsim A$), если существует $g(n) = \bar{o}(1)$ такое, что начиная с некоторого номера n_0, $A(n) \leqslant (1 + g(n)) \cdot B(n)$. Если $A \lesssim B$ и $B \lesssim A$, то будем говорить, что A и B *асимптотически равны* при $n \to \infty$ и обозначать $A \sim B$. Будем писать $A(n) \preccurlyeq B(n)$ (или $B(n) \succcurlyeq A(n)$), если существует такая положительная константа c, что, начиная с некоторого номера n_0, $A(n) \leqslant c \cdot B(n)$. Если $A \preccurlyeq B$ и $B \preccurlyeq A$, то будем говорить, что A и B *равны по порядку* при $n \to \infty$ и обозначать $A \asymp B$.

2.2. Классы функций, рассматриваемые в работе

В рамках данного исследования мы используем следующие обозначения для классов функций. Пусть Φ — некоторый класс функций. Тогда $\Phi^n \subseteq \Phi$ есть подкласс, состоящий из функций от n переменных $x_1, \ldots, x_n$; $\Phi^{k,n} \subseteq \Phi^n$ есть подкласс, состоящий из функций от n переменных $x_1, \ldots, x_n$, существенно зависящих не более чем от k переменных; $\Phi^{=n} \subseteq \Phi^n$ есть подкласс, состоящий из функций от n переменных $x_1, \ldots, x_n$, существенно зависящих от всех n своих переменных.

Пусть $f : \{0,1\}^{V_n} \to \mathbb{N}$ — псевдо-булевская функция. Если для любых фиксаций $\alpha, \beta, \gamma \in \{0,1\}^{V_n}$, таких что $\alpha > \gamma > \beta$ из $f(\alpha) = f(\beta)$ следует, что $f(\alpha) = f(\gamma)$, то будем называть f *интервально-постоянной*. Другими словами, если такая функция присваивает одно и то же значение двум сравнимым наборам, то она присваивает то же значение любому набору из грани, задаваемой этими двумя наборами. Класс всех таких функций будем обозначать через ICF (*interval constant functions*).

Рассмотрим такую псевдо-булевскую функцию, что если она присваивает одно и то же значение двум произвольным фиксациями, то она присваивает то же значение любой фиксации из грани, за-

даваемой этими двумя фиксациями. Такие функции будем называть *разбивающими функциями* [4, 5] и обозначать соответствующий класс как SPL (*splitting functions*). Очевидно, что класс интервально-постоянных функций содержит класс разбивающих функций, то есть SPL $\subset$ ICF.

Рассмотрим такую псевдо-булевскую функцию, что для любых двух фиксаций α и β из того, что $\alpha \leqslant \beta$ следует, что $f(\alpha) \leqslant f(\beta)$. Такие функции будем называть *псевдо-булевскими монотонными функциями* [3] и обозначать соответствующий класс через PM. Очевидно, что класс ICF содержит класс псевдо-булевских монотонных функций, то есть PM $\subset$ ICF. Класс булевских монотонных функций обозначим через M, M $\subset$ PM. Фиксацию $\alpha \in \{0,1\}^{V_n}$ будем называть *верхним нулем* функции $f \in \mathrm{M}^n$, если $f(\alpha) = 0$ и $f(\beta) = 1$ для любой фиксации $\beta > \alpha$. Аналогично α является *нижней единицей*, если $f(\alpha) = 1$ и $f(\beta) = 0$ для любой фиксации $\beta < \alpha$.

Симметрическая функция, зависящая от всех своих переменных — это функция, присваивающая одно и то же значение любым перестановкам произвольной фиксации. Функция с несущественными переменными является симметрической, если ее проекция, задаваемая фиксацией, фиксирующей в точности ее существенные переменные, является симметрической функцией. Класс всех симметрических интервально-постоянных функций будем обозначать через SICF. Класс симметрических интервально-постоянных функций с не более чем $c+1$ различными значениями обозначим через SICF_c. Подмножество SICF, состоящее из булевских монотонных функций, назовем множеством пороговых функций и обозначим через THR [19]. Рассмотрим булевскую функцию, являющуюся дизъюнкцией некоторых своих переменных. Класс таких функций будем обозначать через OR [19]. Очевидно, OR $\subset$ THR.

Очевидно, пересечение классов псевдо-булевских монотонных функций и псевдо-булевских разбивающих функций не пусто и не совпадает ни с одним из этих классов. В некотором смысле ICF (наряду с псевдо-булевскими монотонными функциями) может пониматься как естественное псевдо-булевское обобщение класса булевских монотонных функций.

Псевдо-булевская функция над $\{0,1\}^{V_n}$, $V_n = \{x_1, \ldots, x_n\}$ называется *2-разделяющей*, если существует такая перестановка $(i_1, \ldots, i_n)$,

что для переменных $y_1 = x_{i_1}, \ldots, y_n = x_{i_n}$ выполнено: a) если y_i не является существенной, то y_j не является существенной при $j > i$; b) если переменная y_{2i-1} является существенной и β — частичная фиксация, фиксирующая $\beta(y_{2i-1}) = 1$, то проекция f_β не зависит от переменной y_j при $j > 2i - 1$; c) если переменная y_{2i} является существенной и β — частичная фиксация, фиксирующая $\beta(y_{2i-1}) = \beta(y_{2i}) = 0$, то проекция f_β не зависит от переменной y_j при $j > 2i$. Класс Φ функций будем называть *2-разделяющим*, если для любых $n, k \in \mathbb{N}$ класс $\Phi^{k,n}$ содержит 2-разделяющую функцию с k существенными переменными. Далее показано, что классы булевских монотонных функций и разбивающих функций являются 2-разделяющими.

2.3. Используемая модель расшифровки

Описываемое исследование проведено в рамках *модели точной расшифровки при помощи запросов на значение функции* [1, 2]. В этой модели функция f от n переменных задана при помощи черного ящика и задача ученика (алгоритма расшифровки) состоит в том, чтобы расшифровать f, то есть полностью восстановить ее таблицу значений. Чтобы расшифровать загаданную функцию, алгоритм расшифровки может делать *запросы на значение функции*, то есть подавать произвольные наборы из $\{0,1\}^{V_n}$ черному ящику. Алгоритм расшифровки подает запросы блоками, учитель одновременно отвечает на все вопросы из одного блока. В *стандартной постановке* алгоритм расшифровки должен расшифровать загаданную функцию, использовав но возможности наименьшее число запросов на значение функции. В *параллельной постановке* алгоритм расшифровки должен минимизировать число блоков, требуемых для расшифровки [6].

В работе исследуется *сложность расшифровки* в худшем случае. В разделе 2.4 даны точные определения сложности алгоритмов расшифровки и классов функций. Будем говорить, что алгоритм расшифровки класса Φ является *эффективным на* Φ, если его сложность на Φ^n не более чем полиномиальна по n. Алгоритм является *параметро-эффективным на* Φ, если сложность алгоритма на $\Phi^{k,n}$ как функция от k и n зависит от n не более чем логарифмически. [25].

Отечественные исследователи впервые рассматривали задачу расшифровки в 60-е годы XX века [2]. В англоязычной литерату-

ре модель точной расшифровки была предложена Д.Англуин в конце 1980-х. Из многих типов вопросов учителю, предложенных Д.Англуин, наибольшее развитие получили запросы на значение функции и запросы на эквивалентность функций. Учитель, который может отвечать на оба типа запросов, был даже назван *минимально адекватным учителем*. Учитель, рассматриваемый в настоящий работе наряду с многими другими работами не отвечает на запросы на эквивалентность функций. Более того, такой учитель не позволяет эффективно расшифровывать классы ICF, PM, SPL, M. Несмотря на это, такая модель позволяет строить параллельные параметро-эффективные алгоритмы для всех подклассов ICF, рассмотренных в разделе 2.2.

2.4. Сложность расшифровки

Пусть Φ — некоторый класс псевдо-булевских функций. Будем говорить, что алгоритм *расшифровывает* класс Φ, если для любого $n \in \mathbb{N}$ он расшифровывает любую функцию из Φ^n при условии, что он получает n в виде входного параметра. Обозначим множество алгоритмов расшифровки класса Φ через $\mathcal{A}(\Phi)$. Любой элемент множества $\mathcal{A}(\Phi)$ можно понимать и как единичный алгоритм, получающий n на вход, и как последовательность алгоритмов, такую что n-й алгоритм последовательности расшифровывает функции из Φ^n. Такое определение удобно при рассмотрении асимптотического поведения сложности алгоритмов расшифровки.

Пусть $\varphi(A, f)$ — это число запросов на значение функции, требуемое алгоритму A для расшифровки функции f. Будем называть $\varphi(A, f)$ *сложностью алгоритма A на функции f*. Положим

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

Заметим, что здесь используется min, а не inf, поскольку число алгоритмов расшифровки функций из Φ^n конечно. Будем называть величину $\varphi(A, \Phi) = \max\limits_{f \in \Phi} \varphi(A, f)$ сложностью алгоритма A на классе Φ. Функцию $\varphi(\Phi, n)$ назовем *сложностью расшифровки* Φ. Если сложность алгоритма A на Φ^n по порядку равна сложности расшиф-

ровки Φ при $n \to \infty$, будем говорить, что алгоритм A *оптимальный на* Φ.

Положим, что

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

Заметим, что $\varphi(\Phi, n) = \varphi(\Phi, n, n)$. Функцию $\varphi(\Phi, n, k)$ будем называть *сложностью параметро-эффективной расшифровки класса* Φ. Если сложность алгоритма A на $\Phi^{k,n}$ по порядку равна сложности параметро-эффективной расшифровки Φ при $n, k \to \infty$, будем говорить, что алгоритм A является *оптимальным параметро-эффективным на* Φ.

Пусть $\varphi_p(A, f)$ — это число блоков запросов на значение функции, требуемых алгоритму A для расшифровки функции f. Величину $\varphi_p(A, f)$ будем называть *параллельной сложностью алгоритма* A *на функции* f. Величину $\varphi_p(A, \Phi) = \max\limits_{f \in \Phi} \varphi_p(A, f)$ назовем *параллельной сложностью алгоритма* A *на классе* Φ.

Обозначим $\mathcal{A}_q(\Phi, n, k) = \{A \in \mathcal{A}(\Phi) : \varphi(A, \Phi^{k,n}) \leqslant q\}$.

Положим

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

3. Основные результаты

Теорема 1. *Пусть* $\Phi \subseteq \mathrm{ICF}$, $A \in \mathcal{A}(\Phi)$ *и для некоторой функции* $\psi(n)$ *выполнено* $\varphi(A, \Phi^n) \preccurlyeq \psi(n)$ *при* $n \to \infty$. *Тогда существует такой алгоритм расшифровки* $A' \in \mathcal{A}(\Phi)$, *что* $\varphi(A', \Phi^{k,n}) \preccurlyeq k \log n + \sum_{i=1}^{k} \psi(i)$ *при* $n, k \to \infty$.

Теорема 2. *Имеют место следующие асимптотические неравенства*

$$\varphi(\mathrm{OR}, n, k) \gtrsim k \log \frac{n}{k}, \ \varphi(\mathrm{M}, n, k) \gtrsim k \log \frac{n}{k} + \sqrt{\frac{2}{\pi}} \frac{2^k}{\sqrt{k}},$$
$$\varphi(\mathrm{SPL}, n, k) \gtrsim \max(k \log(n - k + 1), 2^k)$$

при $n, k \to \infty$.

Теорема 3. *Имеют место следующие асимптотические равенства*

- $\varphi(\mathrm{ICF}, n, k) \asymp \varphi(\mathrm{PM}, n, k) \asymp \varphi(\mathrm{SPL}, n, k) \asymp k \log n + 2^k$ *при* $n, k \to \infty$,
- $\varphi(\mathrm{M}, n, k) \asymp k \log n + \frac{2^k}{\sqrt{k}}$ *при* $n, k \to \infty$,
- $\forall c = \mathrm{const}\ \ \varphi(\mathrm{SICF}_c, n, k) \asymp \varphi(\mathrm{THR}, n, k) \asymp \varphi(\mathrm{OR}, n, k) \asymp k \log n$ *при* $n \to \infty$ *и* $\log k = o(\log n)$.

Теорема 4. *Пусть* $\Phi \subseteq \mathrm{ICF}$. *Существует такой алгоритм* $A \in \mathcal{A}(\Phi)$, *что* $\varphi(A, \Phi^{k,n}) \preccurlyeq k \log n + 2^k$ *при* $n, k \to \infty$ *и* $\varphi_p(A, \Phi^{k,n}) \preccurlyeq k$ *при* $n, k \to \infty$.

Теорема 5. *Пусть* $\Phi \subseteq \mathrm{ICF}$ — *2-разделяющий класс. Тогда для любого* $A \in \mathcal{A}(\Phi)$ *выполнено* $\varphi(A, \Phi^{k,n}) \succcurlyeq k \log n$ *при* $n, k \to \infty$. *Если для некоторого* $A \in \mathcal{A}(\Phi)$ *выполнено* $\varphi(A, \Phi^{k,n}) \preccurlyeq k \log n$, *то* $\varphi_p(A, \Phi^{k,n}) \succcurlyeq k$ *при* $k \to \infty$, *где* $\Phi \in \mathrm{ICF}, \mathrm{PM}, \mathrm{SPL}, \mathrm{M}$.

Другими словами, любой оптимальный на $\Phi^{k,n}$ алгоритм расшифровки из $\mathcal{A}(\Phi)$ имеет параллельную сложность на $\Phi^{k,n}$, большую или равную k по порядку при $n \to \infty$ и $2^k \preccurlyeq k \log n$, где $\Phi \in \mathrm{ICF}, \mathrm{PM}, \mathrm{SPL}, \mathrm{M}$.

Теорема 6. *Для* $k, n \in \mathbb{N}$ *и произвольной константы* $c > 2$, *такой что* $2^k < (\frac{c}{2} - 1) k \log n$, *выполнено*

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

при $k, n \to \infty$, *где* $\Phi \in \mathrm{ICF}, \mathrm{PM}, \mathrm{SPL}, \mathrm{M}$.

Последняя теорема есть следствие теорем 4 и 5 и утверждает, что алгоритм из теоремы 4 имеет минимальную по порядку параллельную сложность на $\Phi^{k,n}$ среди всех оптимальных на $\Phi^{k,n}$ алгоритмов из $\mathcal{A}(\Phi)$ при $n \to \infty$ и $2^k \preccurlyeq k \log n$, где $\Phi \in \mathrm{ICF}, \mathrm{PM}, \mathrm{SPL}, \mathrm{M}$.

4. Нижние оценки сложностей расшифровки подклассов ICF

Рассмотрим два широко используемых метода получения нижних оценок сложностей расшифровки классов функций при помощи запросов на значение функции.

Предположим, что Ψ — некоторый класс псевдо-булевских функций от n переменных и что общее число различных значений функций из Ψ не превосходит m, $m \in \mathbb{N}$. Тогда сложность на Ψ^n любого алгоритма расшифровки Ψ не меньше $\log_m(|\Psi|)$. Это следует из того факта, что если некоторый алгоритм расшифровывает некоторую функцию из Ψ^n, сделав менее чем s запросов на значение функции, то результатами работы этого алгоритма могут быть не более чем m^s различных функций. Отсюда $m^s \geqslant |\Psi|$ и $s \geqslant \log_m(|\Psi|)$. Нижние оценки, полученные при помощи этого метода, будем называть *мощностными нижними оценками.*

Второй метод является конструктивным, он заключается в построении алгоритма ответов на запросы на значение функции, играющего роль учителя и отвечающего на запросы любого алгоритма расшифровки из $\mathcal{A}(\Psi)$. Отвечая на запросы алгоритма A расшифровки, он пытается выбирать ответы таким образом, чтобы максимизировать общее число запросов, которое должен сделать алгоритм A, чтобы расшифровать загаданную функцию. Нижние оценки, полученные при помощи этого метода, будем называть *конструктивными нижними оценками.*

4.1. Дизъюнкции переменных

Описанный метод получения мощностных нижних оценок может быть использован, например, чтобы мгновенно получить нижнюю оценку сложности параметро-эффективной расшифровки класса OR. В самом деле, $\binom{n}{k}$ функций из $\mathrm{OR}^{k,n}$ существенно зависят от k переменных. Отсюда сложность расшифровки OR не меньше $[\log \binom{n}{k}] > k \log \frac{n}{k}$. В [19] показано, что сложность параметро-эффективной расшифровки OR асимптотически равна $k \log \frac{n}{k}$ при $k = o(n), n \to \infty$.

Заметим, что если известны существенные переменные функции из OR, то известна и сама функция. Отсюда по крайней мере $k \log \frac{n}{k}$ запросов на значение функции требуется в худшем случае только для того, чтобы расшифровать существенные переменные функции из $\mathrm{OR}^{k,n}$.

4.2. Булевские монотонные функции

Перейдем к получению мощностной нижней оценки сложности параметро-эффективной расшифровки класса булевских монотонных функций. Хорошо известно, что сложность расшифровки M равна $\binom{n}{[\frac{n}{2}]} + \binom{n}{[\frac{n}{2}]+1}$ [8], в некотором смысле более полное описание алгоритма Анселя можно найти в [13]. Заметим, что это значение асимптотически равно $2\sqrt{\frac{2}{\pi}}\frac{2^n}{\sqrt{n}}$ при $n \to \infty$.

Лемма 1. *Имеет место асимптотическое неравенство*

$$\varphi(\mathrm{M}, n, k) \gtrsim k \log \frac{n}{k} + \sqrt{\frac{2}{\pi}}\frac{2^k}{\sqrt{k}} \quad при \quad n, k \to \infty.$$

Доказательство. Хорошо известно, что $|M^k| \geqslant 2^{\binom{k}{[\frac{k}{2}]}}$. Ограничим число $|M^{=k}|$ монотонных функций, каждая из которых существенно зависит от k переменных.

Пусть α, β, γ — это три произвольных набора из среднего слоя k-мерного булевого куба $\{0,1\}^{V_k}$. Пусть поэлементная дизъюнкция этих трех наборов есть единичный набор: $\alpha \vee \beta \vee \gamma = (1, 1, \ldots, 1)$. Рассмотрим множество M' монотонных функций от k переменных, такое что для любой функции из M' наборы α, β, γ являются нижними единицами этой функции, и все нижние единицы этой функции принадлежат среднему слою. Заметим, что функции из M' существенно зависят от всех k своих переменных. В самом деле, переменной достаточно содержаться в по крайней мере одной конъюнкции минимальной ДНФ монотонной функции, чтобы быть существенной переменной этой функции. Но три конъюнкции, соответствующие нижним единицам α, β, γ по определению содержат все k переменных. Понятно, что мощность M' равна $2^{\binom{k}{[\frac{k}{2}]}-3}$. Поскольку каждая функция из M' существенно зависит от всех k своих переменных, мы получаем, что $|M^{=k}| \geqslant |M'| = 2^{\binom{k}{[\frac{k}{2}]}-3}$. Значит, имеет место неравенство $|M^{k,n}| \geqslant 2^{\binom{k}{[\frac{k}{2}]}-3}\binom{n}{k}$. Логарифмируя, получаем неравенство $\varphi(\mathrm{M}, n, k) > \log\binom{n}{k} + \binom{k}{[\frac{k}{2}]} - 3$. Используя хорошо известное неравенство для биномиальных коэффициентов $(\frac{n}{k})^k \leqslant \binom{n}{k} \leqslant (\frac{n \cdot e}{k})^k$ и асимп-

тотическое равенство для центральных биномиальных коэффициентов $\binom{k}{\frac{k}{2}} \sim \sqrt{\frac{2}{\pi}}\frac{2^k}{\sqrt{k}}$, $k \to \infty$, получаем утверждение леммы.

Заметим, что асимптотически второе слагаемое из оценки в лемме 1 всего лишь в два раза меньше сложности $\varphi(\mathrm{M}, k)$ расшифровки M при $k \to \infty$. Как будет далее показано, число запросов на значение функции из первого слагаемого необходимо только для расшифровки существенных переменных функций из $\mathrm{M}^{k,n}$ (даже если алгоритм расшифровки заранее знает k).

4.3. Разбивающие функции

В разделе будет получена конструктивная нижняя оценка сложности параметро-эффективной расшифровки разбивающих функций. Мы покажем, что для любого алгоритма $A \in \mathcal{A}(\mathrm{SPL})$ число запросов на значение функции, необходимое в худшем случае только для расшифровки существенных переменных функции из $\mathrm{SPL}^{k,n}$ асимптотически не меньше $k\log(n-k+1)$ при $n, k \to \infty$.

Лемма 2. *Имеет место асимптотическое неравенство*

$$\varphi(\mathrm{SPL}, n, k) \gtrsim k\log(n-k+1) \quad \textit{при} \quad n, k \to \infty.$$

Доказательство этой леммы приведено в [7, лемма 1].

Теорема 2 сразу следует из доказанных нижних оценок.

5. Нижняя оценка параллельной сложности параметро-эффективной расшифровки 2-разделяющих подклассов ICF

Напомним, что псевдо-булевская функция над $\{0,1\}^{V_n}$, $V_n = \{x_1, \ldots, x_n\}$ называется 2-разделяющей, если существует такая перестановка $(i_1, \ldots, i_n)$, что для переменных $y_1 = x_{i_1}, \ldots, y_n = x_{i_n}$ выполнено

– если y_i не является существенной, то y_j не является существенной при $j > i$,

– если переменная y_{2i-1} является существенной и β — частичная фиксация, фиксирующая $\beta(y_{2i-1}) = 1$, то проекция f_β не зависит существенно от переменной y_j при $j > 2i-1$;

– если переменная y_{2i} является существенной и β — частичная фиксация, фиксирующая $\beta(y_{2i-1}) = \beta(y_{2i}) = 0$, то проекция f_β не зависит существенно от переменной y_j при $j > 2i$.

В последующих рассуждениях полагаем, что k четное. Рассуждения в случае нечетного k аналогичны. Пусть f — 2-разделяющая функция от n переменных с k существенными переменными, без ограничения общности будем считать, что $y_1 = x_1, \ldots, y_n = x_n$. Для $s \leqslant k/2$ обозначим через a_s^f значение функции f на любом наборе α, таком что $\alpha(x_1) = 0, \alpha(x_2) = 1, \ldots, \alpha(x_{2s-3}) = 0, \alpha(x_{2s-2}) = 1$ и $\alpha(x_{2s-1}) = 1$; через b_s^f обозначим значение функции f на любом наборе α, таком что $\alpha(x_1) = 0, \alpha(x_2) = 1, \ldots, \alpha(x_{2s-3}) = 0, \alpha(x_{2s-2}) = 1$ и $\alpha(x_{2s-1}) = 0, \alpha(x_{2s}) = 0$. Пусть c^f обозначает значение функции f на любом наборе α, таком что $\alpha(x_1) = 0, \alpha(x_2) = 1, \ldots, \alpha(x_{k-1}) = 0$, $\alpha(x_k) = 1$.

Класс функций Φ является 2-разделяющим, если для любых $n, k \in \mathbb{N}$ класс $\Phi^{k,n}$ содержит 2-разделяющую функцию с k существенными переменными. Покажем, что классы SPL и M являются 2-разделяющими (следовательно, классы PM и ICF также являются 2-разделяющими).

Рассмотрим такие конъюнкции $C_1, \ldots, C_k$, что

$$\begin{cases} C_1 = x_1, C_2 = \overline{x}_1\overline{x}_2, C_2' = \overline{x}_1 x_2; \\ C_3 = C_2' x_3, C_4 = C_2'\overline{x}_3\overline{x}_4, C_4' = C_2'\overline{x}_3 x_4; \\ \ldots \\ C_{j-1} = C_{j-2}' x_{j-1}, C_j = C_{j-2}'\overline{x}_{j-1}\overline{x}_j, C_j' = C_{j-2}'\overline{x}_{j-1} x_j, \\ \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad j \leqslant k, j \text{ четное}. \end{cases}$$

Рассмотрим разбивающую функцию $f \in \mathrm{SPL}^{k,n}$ от переменных $V_n = \{x_1, \ldots, x_n\}$ с существенными переменными $x_1, \ldots, x_k$, такую что $f(C_j) = j$ для любых $j \in \{1, \ldots, k\}, f(C_k') = k+1$. Очевидно, f является 2-разделяющей и потому класс SPL также является 2-разделяющим.

Теперь рассмотрим монотонную функцию $f(x_1, \ldots, x_n) = x_1 \vee x_2(x_3 \vee x_4(x_5 \vee x_6(\ldots(x_{k-1} \vee x_k))))$ [6]. Функция f является 2-разделяющей и потому M является 2-разделяющим классом.

В оставшейся части раздела полагаем, что задан 2-разделяющий класс Φ. Рассмотрим 2-разделяющую функцию $f(y_1, \ldots, y_n)$ из $\Phi^{k,n}$ с k существенными переменными $x_1, \ldots, x_k$. Рассмотрев биективные отображения множества $\{x_1, \ldots, x_k\}$ на любые другие подмножества $\{y_1, \ldots, y_n\}$ мощности k, получаем $\binom{n}{k} - 1$ новых 2-разделяющих функций из Φ, которые отличаются лишь выбором существенных переменных. Далее мы описываем алгоритм ответов на запросы на значение функции, который для любого алгоритма расшифровки генерирует функцию из этого класса, такую что алгоритм расшифровки имеет на ней большую параллельную сложность.

Мы последовательно предъявим алгоритмы для трех случаев. В первом случае учитель получает запросы на значение функции один за другим и должен сразу же отвечать на каждый запрос (алгоритм C_1). Во втором случае учитель получает один большой блок запросов (алгоритм C_2). В третьем случае мы не накладываем никаких ограничений на то, как подаются запросы на значение функции, то есть третий случай является общим случаем (алгоритм C_3). Предлагаемые алгоритмы являются обобщениями алгоритмов, приведенных в [7].

Нам понадобится понятие *текущей пары расшифровываемых переменных*. Предполагается, что в любой момент времени существует пара переменных x_{i-1}, x_i, i четное, такая что все переменные с индексами меньше $i-1$ уже расшифрованы, и алгоритм расшифровки не имеет никакой информации о других существенных переменных кроме той, что они не совпадают с уже расшифрованными существенными переменными.

5.1. Алгоритм C_1, последовательно отвечающий на каждый запрос на значение функции

Изначально алгоритм расшифровки A не имеет никакой информации о существенных переменных загадываемой функции.

Когда A подает первый набор α на значение функции, отвечаем a_1^f, если запрос содержит больше единиц, чем нулей, и b_1^f иначе. В пер-

вом случае алгоритм A узнает только то, что $\alpha(x_1) = 1$, то есть что переменная x_1 находится среди единиц набора α. Во втором случае A узнает, что $\alpha(x_1) = 0$ и $\alpha(x_2) = 0$, то есть что и x_1, и x_2 находятся среди нулей набора α. Заметим, что в худшем для нас (то есть для алгоритма C_1 ответов на запросы на значение функции) случае алгоритм расшифровки A сузил множество возможных вариантов для переменных x_1 и x_2 вдвое, подав запрос α. При этом A не получил никакой информации о переменных $x_3, \ldots, x_k$.

С новыми запросами на значение функции поступаем аналогичным образом. Рассмотрим множество компонент очередного запроса на значение функции, которые все еще могут соответствовать существенным переменным x_1 и x_2. Если число единиц в этом множестве превосходит число нулей, отвечаем a_1^f, иначе отвечаем b_1^f. Если это множество не содержит ни двух нулей, ни двух единиц, выбираем любые два элемента этого множества; пусть число единичных компонент равно i_1, а число нулевых компонент равно i_2, $i_1, i_2 \in \{1, \ldots, n\}$. Тогда полагаем, что i_1-я компонента соответствует переменной x_1, i_2-я компонента соответствует переменной x_2 и что переменные x_1 и x_2 расшифрованы. К настоящему моменту алгоритм A сделал как минимум $[\log n]$ запросов на значение функции и он до сих пор не имеет никакой информации по переменным $x_3, \ldots, x_n$. Начинаем прятать переменные x_3 и x_4 таким же образом, как прятали x_1 и x_2: если множество компонент вновь поданного запроса содержит больше единиц, чем нулей среди компонент, не соответствующих ни i_1, ни i_2, отвечаем a_2^f, иначе отвечаем b_2^f.

Мы отвечаем на все остальные запросы аналогичным образом с одним исключением: если переменные x_{i-1} и x_i являются текущими расшифровываемыми переменными и для вновь поданного запроса α либо $\alpha(x_{j-1}) = 1$, либо $\alpha(x_{j-1}) = \alpha(x_j) = 0$, $j < i - 1$, j четное, то значение загадываемой функции уже определено на этом запросе, и мы используем это значение для ответа на запрос.

На этом описание первого алгоритма ответов на запросы завершено. Заметим, что этот алгоритм заставляет любой алгоритм расшифровки сделать как минимум $[\frac{k}{2}][\log(n-k)]$ запросов на значение функции: алгоритм расшифровки должен сделать как минимум $[\log n]$ запросов, чтобы расшифровать переменные x_1 и x_2, как минимум $[\log(n-2)]$ запросов, чтобы расшифровать x_3 и x_4 и так да-

лее. Отсюда если алгоритм расшифровки всего делает N запросов на значение функции, он расшифровывает не более чем $[2\frac{N}{\log(n-k)}]$ существенных переменных.

Мы доказали следующую лемму.

Лемма 3. *Для любого 2-разделяющего класса* Φ *существует алгоритм ответов на запросы на значение функции, такой что любой алгоритм* $A \in \mathcal{A}(\Phi)$ *расшифровывает не более чем* $[2\frac{N}{\log(n-k)}]$ *переменных, сделав* N *запросов на значение функции.*

Свойство алгоритма C_1 из леммы будем называть «$2s$ переменных за $s \log n$ запросов».

Используя алгоритм C_1 для ответа на запросы на значение функции, нам приходится раскрывать две существенные переменные после подачи примерно $\log n$ запросов. На самом деле, все что требуется для того, чтобы успешно прятать две переменные — это выбрать два равных столбца в матрице запросов. Если эти столбцы необходимо выбирать на лету (то есть по мере появления новых запросов), то в худшем для нас случае мы можем получить столбцы длины не более $\log n$. С другой стороны, если мы получаем матрицу запросов в виде одного большого блока и если $N \gg \log n$, то мы можем выбрать намного более длинные столбцы. Более точно, если блок состоит из N запросов на значение функции, то можно использовать как минимум $\frac{3}{4}N$ запросов этого блока, чтобы прятать две существенные переменные, не раскрывая никакой информации об остальных существенных переменных. Этот факт является ключевым для алгоритма C_2, к описанию которого мы переходим.

5.2. Алгоритм C_2, отвечающий на блок запросов

Этот алгоритм имеет доступ сразу ко всем N запросам на значение функции. Согласно границе Плоткина [35] в любой булевской матрице размера $N \times n$, $N < \frac{n}{2}$, существует два столбца, таких что расстояние Хэмминга между ними не превосходит $\frac{N}{2}$. Заметим, что асимптотически указанное свойство выполняется даже если убрать ограничение $N < \frac{n}{2}$. Выберем такие два столбца в блоке (матрице) поданных алгоритмом расшифровки запросов. Рассмотрим тот из этих двух столбцов, что имеет больше единиц, чем нулей среди

отличающихся компонент этих двух столбцов. Переменную, соответствующую этому столбцу, обозначим через x_1, переменную, соответствующую второму столбцу, обозначим через x_2.

Напомним, что наша задача — сопоставить значение создаваемой 2-разделяющей функции f с k существенными переменными каждой строке матрицы запросов. Положим $f(\alpha) = a_1^f$, если $\alpha(x_1) = 1$ и $f(\alpha) = b_1^f$, если $\alpha(x_1) = 0$ и $\alpha(x_2) = 0$. Если $\alpha(x_1) = 0$ и $\alpha(x_2) = 1$, то временно не будем присваивать значение запросу α. Очевидно, алгоритм расшифровки может получить какую-либо информацию о других существенных переменных только при помощи запросов α, таких что $\alpha(x_1) = 0$ и $\alpha(x_2) = 1$. По построению в матрице запросов имеется не более чем $[\frac{N}{4}]$ таких строк.

Выберем в матрице запросов столбцы, соответствующие переменным x_3 и x_4, таким образом, что информация о существенных переменных вне множества $\{x_1, x_2, x_3, x_4\}$ может содержаться только в $[\frac{N}{4^2}]$ запросах (строках) этой матрицы. Для этого полагаем $f(\alpha) = a_2^f$ для любого непомеченного ($f(\alpha) \neq a_1^f$ и $f(\alpha) \neq b_1^f$) запроса α, такого что $\alpha(x_3) = 1$, и $f(\alpha) = b_2^f$ для любого непомеченного запроса α, такого что $\alpha(x_4) = 0$.

Продолжая но аналогии, можно выбрать столбцы, соответствующие очередной паре переменных (x_{i-1}, x_i), i четное, таким образом, что множество запросов, которое дает какую-либо информацию алгоритму расшифровки о переменных x_j, $j > i$, $j \leqslant k$, уменьшается как минимум в 4 раза после такого выбора. Другими словами, мы уменьшаем множество запросов на значение функции, которые могут дать новую информацию алгоритму расшифровки, как минимум в 4 раза. Отсюда следует, что когда будет выбрано $2m$ столбцов (переменных), останется лишь $\frac{N}{4^m}$ непомеченных запросов. Заметим, что если булевская матрица с n столбцами имеет менее $\log n$ строк, то в ней имеется как минимум два равных столбца. Значит, необходимо найти такое m, что $\frac{N}{4^m} < \log n$. Имеем $4^m > \frac{N}{\log n}$, $2m > \log \frac{N}{\log n}$. Отсюда после раскрытия $[\log \frac{N}{\log n}]$ переменных подматрица матрицы запросов, состоящая из еще не помеченных запросов, имеет как минимум два равных столбца. Сопоставление очередной пары существенных переменных этим двум столбцам заканчивает построение функции f.

Очевидно, всего алгоритм расшифровки может расшифровать не более чем $[\log \frac{N}{\log n}] + 2$ существенных переменных функции f.

Мы доказали следующую лемму.

Лемма 4. *Для любого 2-разделяющего класса* Φ *существует такой алгоритм ответов на запросы на значение функции, что любой алгоритм* $A \in \mathcal{A}(\Phi)$ *расшифровывает не более чем* $[\log \frac{N}{\log n}] + 2$ *существенных переменных, подав блок из* $N > 2 \log n$ *запросов.*

Свойство алгоритма C_2 из леммы будем называть «$\log s$ переменных за $s \log n$ запросов».

5.3. Алгоритм C_3, позволяющий получить нижнюю оценку в общем случае

Перейдем к основной задаче — построению алгоритма C_3 ответов на запросы, который заставляет любой «близкий к оптимальному» алгоритм расшифровки функций из произвольного 2-разделяющего класса подавать достаточно много блоков запросов на значение функции, чтобы расшифровать функцию, загадываемую алгоритмом ответов на запросы. Алгоритм C_3 объединяет свойства алгоритмов C_1 и C_2.

Из лемм 3 и 4, 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 5. *Пусть* Φ *— 2-разделяющий класс. Пусть* k, n *фиксированы и для алгоритма* $A \in \mathcal{A}(\Phi)$ *выполнено* $\varphi(A, \Phi^{k,n}) \leqslant (1+\varepsilon)k\log n$ *для некоторого* $\varepsilon > 0$. *Тогда найдется константа* $\delta > 0$, *такая что* $\varphi_p(A, \Phi^{k,n}) \geqslant \delta k$.

Доказательство этой леммы приведено в [7, лемма 4].

Неформально, лемма 5 показывает, что среди близких к оптимальным (в смысле обычной сложности) алгоритмов расшифровки 2-разделяющих функций не стоит искать алгоритмы с параллельной сложностью, по порядку меньшей чем k: таких алгоритмов не существует.

Теорема 5 сразу следует из леммы 5.

6. Параметро-эффективные алгоритмы расшифровки подклассов ICF

6.1. Условная расшифровка

Начнем с простого алгоритма, расшифровывающего произвольные функции в ICF параметро-эффективно. По существу этот алгоритм последовательно производит обыкновенный бинарный поиск каждой существенной переменной загаданной функции. Сначала запрашиваются наборы $\overline{0}$ и $\overline{1}$. Если $f(\overline{0}) = f(\overline{1})$, то загаданная функция есть константа, и алгоритм заканчивает свою работу. В противном случае запрашивается произвольный набор α из среднего слоя $\{0,1\}^{V_n}$. Если $f(\alpha) \neq f(\overline{1})$, то определяем β как частичную фиксацию переменных из V_n, фиксирующую x единицей, если $\alpha(x) = 1$, и $*$ иначе. Если $f(\alpha) = f(\overline{1})$ определим β как частичную фиксацию переменных из V_n, фиксирующую переменную x нулем, если $\alpha(x) = 0$ и $*$ иначе. Очевидно, f_β зависит от не более чем $]\frac{n}{2}[$ переменных и существенно зависит хотя бы от одной переменной. Повторяем описанные действия для этой проекции. После не более чем $\log n$ шагов мы получим проекцию загаданной функции, единственная переменная которой является существенной.

Теперь предположим, что расшифрованы некоторые существенные переменные функции f. Пусть существует такая частичная фиксация переменных загаданной функции, что она фиксирует только уже расшифрованные переменные и что полученная проекция существенно зависит хотя бы от одной переменной. Назовем такую фиксацию подходящей фиксацией расшифрованных переменных. Мы снова используем бинарный поиск, чтобы найти очередную существенную переменную проекции, задаваемой этой частичной фиксацией. Затем мы находим новую подходящую фиксацию, фиксирующую все расшифрованные существенные переменные, и повторяем описанные действия для соответствующей проекции. Если в какой-то момент подходящая фиксация не существует, то загаданная функция уже расшифрована, и алгоритм заканчивает свою работу.

Легко видеть, что сложность описанного алгоритма на $\Phi^{k,n}$ для любого $\Phi \subseteq \mathrm{ICF}$ не превосходит $k \log n + 2^k$ по порядку при $n, k \to \infty$. Интересно, что для некоторых подклассов ICF можно добиться мень-

шей по порядку сложности. Чтобы убедиться в этом, докажем теорему 1.

Доказательство. Вспомним, что предыдущий алгоритм использовал полный перебор при поиске подходящей частичной фиксации. Вместо полного перебора можно использовать алгоритм A. Точнее, будем использовать алгоритм A для расшифровки проекции загаданной функции, задаваемой частичной фиксацией, которая приписывает $*$ всем расшифрованным существенным переменным и 1 всем остальным переменным. Снова используем алгоритма A, чтобы расшифровать проекцию загаданной функции, задаваемую частичной фиксацией, которая приписывает $*$ всем расшифрованным существенным переменным и 0 всем остальным переменным. Если эти две проекции равны как функции, мы делаем вывод, что все существенные переменные загаданной функции уже расшифрованы. Иначе пусть α — одна из фиксаций уже найденных существенных переменных, на которой эти две проекции принимают разные значения. Тогда частичная фиксация всех переменных, которая приписывает $\alpha(x)$ переменной x, если x — уже расшифрованная существенная переменная, и $*$ иначе, является подходящей фиксацией.

Заметим, что если мы параметро-эффективно расшифровываем класс $\Phi \subseteq \mathrm{SICF}$, то в качестве $\psi(n)$ можно взять верхнюю оценку сложности на $\Phi^{=n}$, а не на Φ^n. В самом деле, каждая из двух проекций из приведенного выше доказательства либо зависит от всех своих переменных, либо является константной функцией. Такой выбор $\psi(n)$ предпочтителен для многих подклассов SICF. К примеру, для расшифровки функции из OR необходимо сделать n запросов на значение функции. Если же мы знаем, что функция из OR существенно зависит от всех своих переменных, то функция полностью определена, и запросы делать не требуется. Даже для всего класс SICF в худшем случае требуется не более $c \log n$ запросов для расшифровки функции из SICF^n с $c+1$ различными значениями, если известно, что все n переменных являются существенными. Для функций из THR^n требуется $\log n$ запросов.

Очевидно, описанный алгоритм не дает выигрыша при рассмотрении классов ICF, PM, SPL, поскольку $\psi(n)$ по порядку не меньше 2^n при $n \to \infty$ для любого из этих классов. С другой стороны, ал-

горитм Анселя [8] расшифровки класса M имеет сложность на M^n, равную $\binom{n}{[\frac{n}{2}]} + \binom{n}{[\frac{n}{2}]+1}$. Несложно показать, что $\sum_{i=1}^{k} \psi(i)$ по порядку не превосходит $\frac{2^k}{\sqrt{k}}$ при $k \to \infty$. Вспоминая лемму 1, заключаем, что описанный алгоритм оптимален для класса монотонных функций. Если $\log k = o(\log n)$, то наш алгоритм также является оптимальным параметро-эффективным для классов SICF_c, THR, OR. В [19] приведены алгоритмы расшифровки THR, OR с более точными оценками сложности. Преимущество нашего алгоритма заключается в его универсальности.

Мы доказали теорему 3.

6.2. Практически безусловный (параллельный) алгоритм расшифровки

Использованный прием, вообще говоря, не работает, если предпринимается попытка минимизировать параллельную сложность расшифровки некоторого класса $\Phi \subset \mathrm{ICF}$. Дело в том, что для эффективного применения этого приема в случае параллельной расшифровки требуется алгоритм, имеющий небольшую параллельную сложность на Φ^n. Это не выполнено, например, в случае алгоритма Анселя расшифровки монотонных функций: его параллельная сложность сравнима с числом запросов на значение функции в худшем случае.

В серии публикаций [6, 21, 22] П. Дамашке предъявляет параметро-эффективный алгоритм расшифровки функций, имеющий небольшую параллельную сложность на $\mathrm{M}^{k,n}$. Сложность его алгоритма по порядку равна $k \log n + 2^k$ при $n, k \to \infty$, а параллельная сложность равна k по порядку при $n, k \to \infty$. Лемма 11 из [6] утверждает, что этот алгоритм является оптимальным параметро-эффективным на M, но это утверждение ошибочно, как мы убедились выше ($k \log n + 2^k$ по порядку не равно $k \log n + \frac{2^k}{\sqrt{k}}$). Далее мы приводим обобщение алгоритма П.Дамашке, которое расшифровывает класс ICF и является оптимальным параметро-эффективным на SPL, PM, ICF. Параллельная сложность на ICF этого алгоритма по порядку равна k при $n, k \to \infty$.

В доказательстве следующей леммы используются идеи из доказательства теоремы 4.1 из [21].

Лемма 6. *Существует алгоритм* SPLIT*, который для любой функции* $f \in \mathrm{ICF}^n$ *с по крайней мере одной существенной переменной либо расшифровывает некоторую существенную переменную функции* f*, либо находит две частичных фиксации переменных из* V_n*, таких что множества переменных задаваемых ими проекций не пересекаются и каждая проекция существенно зависит хотя бы от одной переменной. Предполагается, что алгоритм получает на вход* $n, f(\overline{0})$ *и* $f(\overline{1})$*. Алгоритм подает не более двух блоков запросов на значение функций с суммарным количеством запросов, не превосходящим* $3]\log n[$.

Доказательство. Без ограничения общности полагаем, что n — степень двойки. В противном случае добавляем фиктивные переменные таким образом, чтобы полученная функция f' была функцией от n' переменных, где n' — степень двойки. Каждой переменной сопоставим уникальную бинарную кодировку ее номера, то есть слово $b_1, \ldots, b_r$, где $r = \log n$. Если рассматривать эти слова как столбцы матрицы, то полученная матрица A размера $r \times n$ не имеет одинаковых столбцов. Запрашиваем значения функции f на строках матрицы A и на их поэлементных отрицаниях в виде оного блока из $2\log n$ запросов.

1) Предположим, что среди поданных запросов существует запрос α, такой что $f(\alpha) \neq f(\overline{0})$ и $f(\alpha) \neq f(\overline{1})$. Рассмотрим частичную фиксацию β_1 переменных функции f, такую что $\beta_1(x) = 1$ если $\alpha(x) = 1$ и $\beta_1(x) = *$ иначе. Рассмотрим частичную фиксацию β_0 переменных функции f, такую что $\beta_0(x) = 0$, если $\alpha(x) = 0$ и $\beta_0(x) = *$ иначе. Множества переменных функций f_{β_1} и f_{β_0} не пересекаются и каждая из этих функций имеет но крайней мере одну существенную переменную. Алгоритм SPLIT заканчивает свою работу.
2) Предположим, что среди пар противоположных запросов существует пара (α_1, α_2), такая что $f(\alpha_1) = f(\alpha_2)$. Очевидно, либо $f(\alpha_1) \neq f(\overline{1})$, либо $f(\alpha_1) \neq f(\overline{0})$. Без ограничения общности полагаем, что $f(\alpha_1) \neq f(\overline{1})$. Рассмотрим частичную фиксацию β_1

переменных функции f, такую что $\beta_1(x) = 1$ если $\alpha_1(x) = 1$ и $\beta_1(x) = *$ иначе. Рассмотрим частичную фиксацию β_2 переменных функции f, такую что $\beta_2(x) = 1$ если $\alpha_2(x) = 1$ и $\beta_2(x) = *$ иначе. Множества переменных функций f_{β_1} и f_{β_2} не пересекаются и каждая из проекций существенно зависит хотя бы от одной переменной. Алгоритм SPLIT заканчивает свою работу.

3) Без ограничения общности полагаем, что для каждой строки α матрицы A выполнено равенство $f(\alpha) = f(\overline{1})$. В противном случае заменим все строки матрицы A, которым присвоено значение $f(\overline{0})$, на противоположные. Определим матрицу B. положив ее i-ю строку равной поэлементной конъюнкции первых i строк матрицы A: $B_i = A_1 \& \ldots \& A_i$. Подаем строки матрицы B в виде одного блока запросов. Заметим, что набор B_r имеет ровно 1 единичную компоненту.
   а) Предположим, что среди строк матрицы B существует строка α, такая что $f(\alpha) \neq f(\overline{0})$ и $f(\alpha) \neq f(\overline{1})$. Тогда алгоритм SPLIT находит две подходящие фиксации способом, описанным в 1, и завершает свою работу.
   б) Если $f(B_r) = f(\overline{1})$, то переменная, соответствующая этой компоненте, является существенно переменной и алгоритм SPLIT завершает свою работу. Теперь предположим, что существует номер i, такой что $f(B_i) = f(\overline{1})$ и $f(B_{i+1}) = f(\overline{0})$. Обозначим через G грань, задаваемую наборами B_i и $\overline{0}$. Рассмотрим набор $B_i \& \overline{B_{i+1}}$ (то есть набор, противоположный набору B_{i+1} в грани G). По определению $B_{i+1} = B_i \& A_{i+1}$. Отсюда B_{i+1} и A_{i+1} присваивают одно и то же значение любой x, такой что $B_i(x) = 1$. Следовательно, противоположные наборы $\overline{B_{i+1}}$ и $\overline{A_{i+1}}$ также присваивают одно и то же значение любой переменной x, такой что $B_i(x) = 1$. Отсюда $B_i \& \overline{B_{i+1}} = B_i \& \overline{A_{i+1}} \leqslant \overline{A_{i+1}}$. Вспомним, что $f(\overline{A_{i+1}}) = f(\overline{0})$. Поскольку $\overline{0} \leqslant B_i \& \overline{B_{i+1}} \leqslant \overline{A_{i+1}}$, мы заключаем, что $f(B_i \& \overline{B_{i+1}}) = f(\overline{0})$. Рассмотрим частичную фиксацию β_1 переменных функции f, такую что $\beta_1(x) = 1$ если $(B_i \& \overline{B_{i+1}})(x) = 1$, $\beta_1(x) = 0$ если $B_i(x) = 0$ и $\beta_1(x) = *$ иначе. Рассмотрим частичную фиксацию β_2 переменных функции f, такую что $\beta_2(x) = 1$ если $B_{i+1}(x) = 1$, $\beta_2(x) = 0$ если $B_i(x) = 0$ и $\beta_2(x) = *$ иначе. Множества переменных

функций f_{β_1} и f_{β_2} не пересекаются и каждая из проекций существенно зависит хотя бы от одной переменной. Алгоритм SPLIT завершает свою работу.

Лемма доказана.

Лемма 7. *Существует алгоритм* LEARN_VARS*, который для любой функции* $f \in \mathrm{ICF}^n$ *с* k *существенными переменными расшифровывает некоторое число* k' *существенных переменных функции* f*,* $1 \leqslant k' \leqslant k$*, подав для этого не более чем* $6k'] \log n[$ *запросов на значение функции в виде не более чем* $2k'$ *блоков. Мы полагаем, что алгоритм получает на вход* $n, f(\overline{0})$ *и* $f(\overline{1})$*.*

Доказательство. Алгоритм является рекурсивным. Запускаем SPLIT на всем булевом кубе $\{0,1\}^{V_n}$. Если он возвращает две частичные фиксации, запускаем LEARN_VARS на гранях, соответствующих этим частичным фиксациям. Иначе добавляем расшифрованную переменную в ответ.

Работу описанного алгоритма можно представить в виде бинарного дерева с k' концевыми вершинами. В каждой концевой вершине LEARN_VARS расшифровывает одну (уникальную) существенную переменную загаданной функции. В остальных вершинах он находит пару подходящих проекций загаданной функции. В каждой вершине LEARN_VARS делает не более чем $3] \log n[$ запросов в виде не более чем 2 блоков. Учитывая бинарность дерева, получаем, что всего алгоритм делает не более $6k'] \log n[$ запросов на значение функции в виде не более чем $2k'$ блоков. Лемма доказана.

Перейдем к описанию алгоритма расшифровки функций из ICF. Сначала запрашиваются $f(\overline{0})$ и $f(\overline{1})$ в виде одного блока. Запускаем LEARN_VARS, чтобы расшифровать $k_1 \leqslant k$ существенных переменных загаданной функции $f \in \mathrm{ICF}^{k,n}$, сделав не более чем $6k_1] \log n[$ запросов в виде не более чем $2k_1$ блоков. Затем используем полный перебор, чтобы найти подходящую частичную фиксацию, фиксирующую найденные существенные переменные. Полный перебор подает 2^{k_1} запросов в виде одного блока. Если подходящая фиксация не существует, то загаданная функция расшифрована. Иначе повторяем описанные действия для проекции, задаваемой найденной частичной

фиксацией, и расшифровываем следующие k_2 существенные переменные. Поскольку $\sum_i k_i = k$, мы заключаем, что всего требуется не более чем $6k]\log n[+\sum_{j=1}^{k} 2^j$ запросов в виде $3k+1$ блоков.

Мы доказали теорему 4.

Теорема 6 следует из теорем 4 и леммы 5.

Список литературы

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

[2] Коробков В. О монотонных функциях алгебры логики // Проблемы кибернетики. 1965. Т. 13. С. 5–28.

[3] Foldes S., Hammer P. L. Monotone, Horn and Quadratic Pseudo-Boolean Functions // J. UCS. 2000. Vol. 6. N 1. P. 97–104.

[4] Осокин В. В. Асимптотически оптимальный алгоритм расшифровки разбиения булевого куба на подкубы // Интеллектуальные системы. 2007. Т. 11, вып. 1–4. С. 587–606.

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

[6] Damaschke P. On Parallel Attribute-Efficient, Learning // Journal of Computer and System Sciences. Vol. 67. Iss. 1. August 2003. P. 46–62.

[7] Осокин В. В. О параллельной расшифровке разбиений булевого куба // Интеллектуальные системы. 2009. Т. 13, вып. 1–4. С. 427–454.

[8] Hansel G. Nombre minimal de contacts de fermeture nécessaire pour réaliser une fonction booléenne symétrique de n variables // C. R. Acad. Sci. Paris. 1964. 258. P. 6037–6040.

[9] Choi S., Jung K., Kirn J. Almost Tight Upper Bound for Finding Fourier Coefficients of Bounded Pseudo-Boolean Functions // COLT 2008. P. 123–134.

[10] Torvik V. I. Data Mining and Knowledge Discovery: a Guided Approach based on Monotone Boolean Functions / Ph. D. in Engineering Science. May 24, 2002. Louisiana State University.

[11] Boros E., Hammer P. L., Ibaraki T., Kawakami K. Polynomial-Time Recognition of 2-Monotonic Positive Boolean Functions Given by an Oracle // SIAM Journal on Computing. 1997. Vol. 26. Iss. 1. P. 93–109.

[12] Zolotykh N. Yu., Shevchenko V. N. Lower Bounds for the Complexity of Learning Half-Spaces with Membership Queries // ALT'98. Otzenhausen, Germany. October 8–10, 1998.

[13] Kovalerchuck B., Triantapliyllou E., Deshpande A. S., Vityaev E. Interactive learning of monotone Boolean functions // Information Sciences: an International Journal. 1996. Vol. 94. Iss. 1–4. P. 87–118.

[14] Nakamura A., Abe N. Exact learning of linear combinations of monotone terms from function value queries // Theoretical Computer Science. 1995. Vol. 137. Iss. 1. P. 159–176.

[15] Makino K., Ibaraki T. The maximum latency and identification of positive Boolean functions // SIAM Journal on Computing. 1997. Vol. 26. Iss. 5. P. 1363–1383.

[16] Gainanov D. N. On one criterion of the optihality of an algorithm for evaluating monotonic boolean functions // USSR Computational Mathematics and Mathematical Physics. 1985. Vol. 24. Iss. 4. P. 176–181.

[17] Sokolov N. A. On the optimal evaluation of monotonic Boolean functions // USSR Computational Mathematics and Mathematical Physics. 1982. Vol. 22. Iss. 2. P. 207–220.

[18] Bshouty N., Hellerstein L. Attribute efficient learning in query and mistake-bound models // Journal of Computer and System Sciences. 1998. 56. P. 310-319.

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

[20] Bshouty N. H. Exact learning of formulas in parallel // Machine Learning. 1997. Vol. 26. Iss. 1. P. 25–41.

[21] Damaschke P. Adaptive versus nonadaptive attribute-efficient learning // Machine Learning. 2000. 41. P. 197–215.

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

[23] Boros E., Hammer P. L. Pseudo-boolean optimization // Discrete Applied Mathematics. 2002. Vol. 123. Iss. 1–3. P. 155–225.

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

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

[26] Valiant L. G. A theory of the learnable // ACM Press New York, NY, USA. 1984. Vol. 27. Iss. II. P. 1134–1142.

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

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

[29] Kolountzakis M., Markakis E., Mehta A. Learning symmetric Juntas in time $n^{o(k)}$ // In the proceedings of «Interface between Harmonic Analysis and Number Theory». 2005.

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

[31] Balcazar J. L., Diaz J., Gavalda R., Watanabe O. An optimal parallel algorithm for learning DFA // Proceedings of the seventh annual conference on Computational learning theory. 1994. P. 208–217.

[32] Vitter J. S., Lin J. Learning in Parallel // Inf. Comput. 1992. 96 (2). P. 179–202.

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

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

[35] Plotkin M. Binary codes with specified minimum distance // IRE Transactions on Information Theory. 1960. 6. P. 445–450.