Сложность расшифровки линейных булевых функций

А. В. Быстрыгова

В работе рассматривается задача точной расшифровки линейной булевой функции арности n, существенно зависящей от k переменных. Получены точные значения сложности расшифровки для малых k, и верхние оценки для общего случая.

Ключевые слова: точная расшифровка функций, линейные булевые функции.

Введение

На практике часто может возникнуть ситуация, когда нам дано некоторое устройство — “черный ящик”, про которое мы знаем некоторую частичную информацию, например, что оно принадлежит некоторому классу устройств, и нам хочется понять, что за устройство перед нами. При этом мы можем подавать на вход устройства некоторые сигналы и снимать с выходов устройства результат. Если такое устройство является конечным автоматом [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11], то раздел науки, занимающийся анализом таких устройств называется эксперименты с автоматами [12, 13, 14, 15, 16]. При этом если у нас есть только одно такое устройство, то эксперимент называется простым, и мы, подавая на вход устройства входные воздействия, понимаем, что его состояние меняется каждый такт. Если у нас имеются несколько копий устройства, то эксперимент называется кратным. Если исследуемое устройство является устройством без памяти, то необходимость в нескольких копиях отпадает. Раздел науки, занимающийся анализом устройств без памяти называется расшифровкой

функций, а в зарубежной литературе Machine Learning (машинное обучение).

Расшифровка функции — это игра между "учеником" (алгоритмом расшифровки) и "учителем", в которой "учитель" загадывает функцию из класса, известного "ученику", и тот должен за наименьшее количество запросов разгадать функцию. Запрос на значение функции — это набор значений переменных функции, ответ на запрос — значение искомой функции на этом наборе.

При расшифровке функций популярны две модели: модель точной расшифровки (exact learning) [17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30] и модель вероятно примерно точной расшифровки (probably approximately correct leaning, PAC) [30, 31, 32, 33, 34, 35]. В модели точной расшифровки "ученик" должен точно расшифровать загаданную "учителем" функцию, то есть полностью восстановить таблицу значений функции, или если эта функция не дискретная, то восстанавливать ее с некоторой точностью (в этом случае этот процесс называется интерполяцией функции [36]). При вероятно примерно точной расшифровке "ученик" не может выбирать набор, значение функции на котором он хочет узнать. Он делает запрос к функции, а "учитель", руководствуясь заранее определенным распределением вероятности на области определения функции, выбирает набор значений аргументов и выдает "ученику" пару: набор и значение функции на этом наборе. Цель "ученика" — восстановить вектор значений загаданной функции так, чтобы вероятность ошибки была небольшой. Еще одна модель расшифровки рассматривается в работах [37, 38], в которой запросом является пара наборов значений аргументов функции, а ответом на запрос является знак разности значений функции на этих наборах. Такого рода запросы применяются при расшифровке функций ранжирования интернет поисковиков.

Одной из первых рассматривалась задача расшифровки монотонных булевых функций. Эта же задача является наиболее полно решенной [18, 19, 20, 21]. В работах [22, 23, 24, 25] получено решение задачи расшифровки функции разбиения булевого куба на подкубы. В работе [26] предлагается общий подход к параллельной параметроэффективной расшифровке интервально-постоянных функций. Этот класс функций включает в себя, в частности, и монотонные функции и функции разбиения булевого куба на подкубы. О параметроэффективной расшифровке говорят, если функции из загадываемого

класса зависят от большого числа переменных, но известно, что загаданная 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"). В работе [27] рассматривалась задача расшифровки пороговых функций, а в работах [28, 29] исследовалась задача расшифровки арифметических сумм монотонных конъюнкций. И, наконец, в работах [30, 31, 32] изучалась задача расшифровки линейных булевых функций.

В данной работе рассматривается задача параметро-эффективной расшифровки линейных булевых функций в рамках модели точной расшифровки.

Известно, что загадана линейная функция арности n с k существенными переменными вида $f(x_0, x_1, ..., x_{n-1}) = a_0x_0 \oplus a_1x_1 \oplus ... \oplus a_{n-1}x_{n-1}$, $a_i \in \{0, 1\}$, $\sum_{i=0}^{n-1} a_i = k$. Необходимо за минимальное количество запросов к функции точно восстановить ее вектор значений.

В работе [30] было получено, что сложность расшифровки функции f, существенно зависящей от k переменных, равна сложности расшифровки функции f, существенно зависящей от $n - k$ переменных, а также в рамках модели точной расшифровки получены верхние оценки: $]\log n[$ для $k = 1$, $3]\log n[-2$ для $k = 2$, $4]\log n[-3$ для $k = 3$.

В работах [30], [31], [32] была рассмотрена эта задача для произвольного k в рамках модели вероятно примерно точной расшифровки PAC. Для этой модели в работе [30] получена оценка $O(k \log n/k)$. В работе [31] получена оценка $O(n^{1-\frac{1}{k}} \log n)$, если известно, что загаданная функция зависит от не более чем k переменных.

В настоящей работе в рамках модели точной расшифровки получены оценки сложности расшифровки линейных булевых функций для случаев $k = 2$ и $k = 3$, отличающиеся от точного значения не более чем на 2, и верхние оценки сложности расшифровки для произвольных k.

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

Основные понятия и формулировка результатов

Пусть Φ —– некоторый класс булевых функций. Тогда $\Phi^n \subseteq \Phi$ есть подкласс, состоящий из функций, зависящих от n переменных $x_0, x_1, \ldots, x_{n-1}$; $\Phi^{k,n} \subseteq \Phi^n$ есть подкласс, состоящий из функций от n переменных $x_0, \ldots, x_{n-1}$, существенно зависящих ровно от k переменных.

Под *запросом* на значение функции будем понимать набор значений переменных функции, под *ответом* на запрос — значение функции на этом наборе. Под *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 из* Φ^n, если значения функции на наборах, сгенерированных условным экспериментом, однозначно определяют таблицу значений функции f при условии, что $f \in \Phi^n$. Скажем, что *алгоритм расшифровывает класс функций* 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.

Пусть $A \in \mathcal{A}(\Phi)$, $f \in \Phi$, тогда обозначим через $\varphi(A, f)$ число запросов на значение функции, требуемое алгоритму A для расшифровки функции f. Будем называть $\varphi(A, f)$ *сложностью алгоритма A на функции f*.

Положим

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

Алгоритм расшифровки, на котором достигается минимум — это такой алгоритм, который работает лучше других алгоритмов на самой плохой функции из класса.

Функция $f(x_0, x_1, ..., x_{n-1})$ является *линейной булевой функцией арности n с k существенными переменными и нулевым свободным членом*, если $f(x_0, x_1, ..., x_{n-1}) = a_0 x_0 \oplus a_1 x_1 \oplus ... \oplus a_{n-1} x_{n-1}$, $a_i \in \{0, 1\}$, $\sum_{i=0}^{n-1} a_i = k$, где $\oplus$ — означает сложение по модулю 2.

Будем рассматривать задачу для $1 \leqslant k < \frac{n}{2}$, так как согласно работе [30], сложность расшифровки функции f, существенно зависящей от k переменных, равна сложности расшифровки функции f, существенно зависящей от $n-k$ переменных.

Пусть L — множество линейных булевых функций с нулевым свободным членом. Поскольку в данной работе мы будем исследовать только такие функции, то обозначим

$$\varphi(n,k) = \varphi(L,n,k).$$

Если a — вещественное число, то через $]a[$ обозначим наименьшее целое не меньшее a, через $[a]$ обозначим наибольшее целое не большее a. Под $\log n$ будем понимать двоичный логарифм от n.

Теорема 1. *Для любого натурального n, $n > 2$, справедливы неравенства $2]\log n[-2 \leqslant \varphi(n,2) \leqslant 2]\log n[-1$, причем если $\log n - [\log n] \in \{0\} \cup (1/2,1)$, то $\varphi(n,2) = 2]\log n[-1$.*

Теорема 2. *Для любого натурального n, $n > 5$, справедливы соотношения*

1) $3]\log n[-4 \leqslant \varphi(n,3) \leqslant 3]\log n[-2$, *если* $\log n - [\log n] \in (0,1/3]$

2) $3]\log n[-3 \leqslant \varphi(n,3) \leqslant 3]\log n[-2$, *если* $\log n - [\log n] \in (1/3,2/3]$

3) $\varphi(n,3) = 3]\log n[-2$, *если* $\log n - [\log n] \in \{0\} \cup (2/3,1)$.

Теорема 3. *Для любого натурального n, $n > 5$, имеют место неравенства:*

1) $\varphi(n,4) \leqslant (]\log n[+2)(]\log n[-1)/2 + 4]\log n[-3$,

2) $\varphi(n,5) \leqslant (]\log n[+2)(]\log n[-1)/2 + 5]\log n[-3$,

3) $\varphi(n,6) \leqslant (]\log n[+2)(3]\log n[-5)/2 + 6]\log n[-4$,

4) $\varphi(n,7) \leqslant (]\log n[+2)(3]\log n[-5)/2 + 7]\log n[-4$,

5) *если* $8 \leqslant k < n/2$, *то*

$$\varphi(n,k) \leqslant (k - 2^{[\log k]} + 2)k^{[\log k]/2}\cdot]\log n[^{[\log k]} + \\ + k^{([\log k]+1)/2}\cdot]\log n[^{[\log k]-1} + k]\log n[/2.$$

Очевидно, что любую линейную функцию можно расшифровать за не более чем n запросов (нужно запрашивать значение функции на двоичных наборах веса 1). Поэтому алгоритмы, упомянутые в доказательстве теоремы 2, целесообразно использовать для небольших k, не близких к $\frac{n}{2}$, то есть когда предложенные в теоремах оценки на количество запросов меньше n.

Оценки сложности расшифровки функций от двух переменных

Приведем сначала классическую мощностную нижнюю оценку.

Лемма 1 (мощностная нижняя оценка)**.** *Для любого класса булевых функций Φ, любого алгоритма A, расшифровывающего класс Φ, для любого натурального n существует такая функция f из Φ^n, что $\varphi(A, f) \geqslant]\log|\Phi^n|[$.*

Доказательство. Любой запрос $a = (a_0, a_1, \ldots, a_{n-1})$ на значение функции разбивает множество функций Φ^n на два подмножества: те функции, которые на наборе a принимают значение 0, и те — которые на этом наборе принимают значение 1.

Чтобы найти функцию, которую произвольным фиксированным алгоритмом трудно расшифровать, нам достаточно "прятать" искомую функцию в множестве большей мощности. В этом случае каждый запрос будет сокращать множество потенциальных функций в лучшем случае в 2 раза. И, значит, если число запросов будет меньше чем $]\log|\Phi^n|[$, то расшифровываемая функция будет находиться в множестве мощности не менее 2, а, значит, еще не определена однозначно.

Лемма доказана. □

Лемма 2. *Для любого натурального n, $n > 2$, справедливо неравенство $\varphi(n, 2) \geqslant 2]\log n[-2$, причем если $\log n - [\log n] \in \{0\} \cup (1/2, 1)$, то $\varphi(n, 2) \geqslant 2]\log n[-1$.*

Доказательство. Множество $L^{2,n}$ — линейных функций, существенно зависящих от двух переменных, имеет мощность $C_n^2 = n(n-1)/2 = n^2/2 - n/2$. Так как при $n > 2$ справедливо неравенство $n^2/2 - n/2 > n^2/4$, то с учетом леммы 1 имеем

$$\varphi(n, 2) \geqslant]\log C_n^2[>]\log n^2/4[=]2\log n[-2,$$

т.е.

$$\varphi(n,2) \geqslant]2\log n[-1.$$

Обозначим $[\log n] = m$, $c = \log n - m$, $c \in [0,1)$, т.е. $\log n = m + c$.

Если $c = 0$, то $2]\log n[= 2m =]2\log n[$.

Если $c \in (0, 1/2]$, то $2]\log n[= 2m+2$, а $]2\log n[=]2m+2c[= 2m+1$, т.е. $]2\log n[= 2]\log n[-1$.

Если $c \in (1/2, 1)$, то $2]\log n[= 2m+2$, а $]2\log n[=]2m+2c[= 2m+2$, т.е. $]2\log n[= 2]\log n[$.

Лемма доказана. □

Теперь получим верхнюю оценку для $\varphi(n,2)$.

Пусть B — некоторое подмножество множества переменных $\{x_0, x_1, ..., x_{n-1}\}$. Под *запросом на множестве* B будем понимать запрос значения функции f на наборе, в котором все переменные из B установлены в 1, а остальные в 0. Будем обозначать его $f(B)$.

Лемма 3. *Если* $B \subseteq \{x_0, x_1, ..., x_{n-1}\}$ *и в* B *содержится нечетное число существенных переменных функции* f *из* L^n*, то одну из существенных переменных множества* B *можно найти за не более* $]\log |B|[$ *запросов.*

Доказательство. Разделим B произвольным образом на два непересекающихся множества B_1 и B_2, отличающиеся по мощности не более чем на 1. Запросим значение на B_1. Если оно равно 0 (то есть в B_1 четное число существенных переменных, а значит во множестве B_2 нечетное), положим $B = B_2$, иначе, положим $B = B_1$. Затем опять разделим B на два множества и т.д, до тех пор пока B состоит из более чем одной переменной. Получившееся в итоге множество B состоит из одной переменной, которая и является существенной. Поскольку каждый раз множество B делится надвое, то для нахождения существенной переменной потребуется не более $]\log |B|[$ запросов.

Лемма доказана. □

Как следствие получаем следующий результат, ранее опубликованный в [30].

Следствие 1. *Для любого натурального* n *имеет место равенство* $\varphi(n,1) =]\log n[$.

Доказательство. Согласно мощностной нижней оценке $\varphi(n,1) \geqslant]\log|L^{1,n}|[=]\log n[$. С другой стороны, поскольку любая функция из $L^{1,n}$ имеет одну существенную переменную, то согласно лемме 3 ее можно найти не более чем за $]\log n[$ запросов. □

Введем B^i — подмножество переменных множества B, у которых в двоичном представлении номера переменной i-й бит равен 1 (нумерация идет от младших значимых битов, начиная с 0).

Например, пусть $n = 8$, $B = \{x_1, x_{3,}, x_4\}$. Тогда $B^0 = \{x_1, x_3\}$, $B^1 = \{x_3\}$, $B^2 = \{x_4\}$.

Лемма 4. *Для любого натурального n, $n > 2$, справедливо неравенство $\varphi(n,2) \leqslant 2]\log n[-1$.*

Доказательство. Пусть $B = \{x_0, x_1, ..., x_{n-1}\}$. Запросим значение искомой функции f на множествах B^i, $i = 0, 1, ...,]\log n[-1$, т.е. сделаем $]\log n[$ запросов.

Если на B^i функция f равна 0, то в B^i содержится четное число существенных переменных, т.е. в нашем случае либо 0, либо 2, и, значит, i-й бит в номерах обоих существенных переменных одинаков.

Если $f(B^i) = 1$, то в B^i содержится нечетное число существенных переменных, что в нашем случае означает, что в B^i содержится ровно одна существенная переменная. Следовательно, номера существенных переменных по i-му биту отличаются.

Поскольку номера существенных переменных отличны, то обязательно существует хотя бы один бит, на котором они отличаются. Пусть эти номера отличаются в p-м бите. Тогда $f(B^p) = 1$, и в B^p содержится ровно одна существенная переменная. Согласно лемме 3 мы можем найти эту переменную за $]\log|B^p|[$ запросов. Учитывая, что $|B^p| \leqslant 2^{]\log n[-1}$, имеем, что на нахождение этой переменной мы потратим не более $]\log n[-1$ запросов.

Остается заметить, что поскольку мы опросили значение функции f на каждом из запросов B^i, то мы для каждого $i \in \{0, 1, ...,]\log n[-1\}$ знаем совпадают ли i-е биты номеров существенных переменных. Тем самым, зная одну переменную, мы автоматически узнаем и вторую существенную переменную.

Таким образом, задав не более чем $2]\log n[-1$ запросов, мы узнаем обе существенные переменные функции. Лемма доказана. □

Теорема 1 является простым следствием лемм 2 и 4.

Оценки сложности расшифровки функций от трех переменных

Лемма 5. *Для любого натурального n, $n > 5$, справедливо неравенство $\varphi(n,3) \geqslant 3]\log n[-4$, причем*

1) если $\log n - [\log n] \in (1/3, 2/3]$, то $\varphi(n,3) \geqslant 3]\log n[-3$;

2) если $\log n - [\log n] \in \{0\} \cup (2/3, 1)$, то $\varphi(n,3) \geqslant 3]\log n[-2$.

Доказательство. Множество $L^{3,n}$ — линейных функций, существенно зависящих от трех переменных, имеет мощность $C_n^3 = n(n-1)(n-2)/6 = n^3/6 - n^2/2 + n/3$. Так как при $n > 5$ справедливо неравенство $n^3/6 - n^2/2 + n/3 > n^3/8$, то с учетом леммы 1 имеем

$$\varphi(n,3) \geqslant]\log C_n^3[>]\log(n^3/8)[=]3\log n[-3,$$

т.е.

$$\varphi(n,3) \geqslant]3\log n[-2.$$

Обозначим $[\log n] = m$, $c = \log n - m$, $c \in [0,1)$, т.е. $\log n = m + c$.

Если $c = 0$, то $3]\log n[= 3m =]3\log n[$.

Если $c \in (0, 1/3]$, то $3]\log n[= 3m+3$, а $]3\log n[=]3m+3c[= 3m+1$, т.е. $]3\log n[= 3]\log n[-2$.

Если $c \in (1/3, 2/3]$, то $3]\log n[= 3m+3$, а $]3\log n[=]3m+3c[= 3m+2$, т.е. $]3\log n[= 3]\log n[-1$.

Если $c \in (2/3, 1)$, то $3]\log n[= 3m+3$, а $]3\log n[=]3m+3c[= 3m+3$, т.е. $]3\log n[= 3]\log n[$.

Лемма доказана. □

Лемма 6. *Для любого натурального n, $n > 3$, справедливо неравенство $\varphi(n,3) \leqslant 3]\log n[-2$.*

Доказательство. Пусть $B = \{x_0, x_1, ..., x_{n-1}\}$. Запросим значение f на B^0. Если $f(B^0) = 1$, положим $C = B^0$, иначе $C = B \setminus B^0$. В C содержится нечетное количество существенных переменных. Согласно лемме 3 одну из существенных переменных можно найти за $]\log|C|[$ запросов. Поскольку $|C| \leqslant 2^{]\log n[-1}$, имеем, что для нахождения одной существенной переменной мы потратили не более $]\log n[-1$ запросов.

Теперь применяя лемму 4 к B, найдем за не более $2]\log n[-2$ запросов остальные существенные переменные (мы уже знаем значение

f на B^0, поэтому не нужно заново запрашивать это значение, следовательно потребуется за один запрос меньше).

Таким образом, задав не более $1 + (]\log n[-1) + (2]\log n[-2) = 3]\log n[-2$ запросов, мы узнаем все существенные переменные функции f.

Лемма доказана. □

Теорема 2 является следствием лемм 5 и 6.

Верхняя оценка для общего случая

Рассмотрим вспомогательную задачу нахождения существенной переменной функции. Такая задача для функций разбиения булевого куба на подкубы рассматривалась в [24].

Будем говорить, что *алгоритм находит существенную переменную функции f из L^n*, если по значениям функции на наборах, сгенерированных алгоритмом, мы хотя бы для одной переменной функции f однозначно можем сказать, что она является существенной. Скажем, что *алгоритм находит существенную переменную для класса линейных функций L*, если для любого $n \in \mathbb{N}$ он находит существенную переменную для любой функции из L^n при условии, что он получает n в виде входного параметра. Обозначим множество алгоритмов, находящих существенную переменную для класса L через $\mathcal{F}(L)$.

Пусть $A \in \mathcal{F}(L)$, $f \in L$, тогда обозначим через $\phi(A, f)$ число запросов на значение функции, требуемое алгоритму A для нахождения какой-либо существенной переменной функции f.

Положим

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

Как следствие леммы 3 получаем следующее утверждение.

Следствие 2. *Если k нечетное число, то $\phi(n, k) \leqslant]\log n[$.*

Поскольку дальше мы будем получать только верхние оценки сложности, то можем считать, что число n является степенью двойки. Если n не степень двойки, добавим фиктивные переменные, чтобы количество переменных стало степенью двойки.

Лемма 7. *Имеет место неравенство* $\phi(n,4) \leqslant (2+]\log n[)(]\log n[-1)/2+]\log n[-1$.

Доказательство. Вводим множество всех переменных $B = \{x_0, x_1, .., x_{n-1}\}$. Вводим переменную — счетчик $current$. Присваиваем ей значение $current = \log n - 1$. Переходим к шагу 1.

Шаг 1. Делим множество B на два равных по мощности непересекающихся множества B_0 и B_1. Во множество B_0 помещаем все переменные из B, в двоичном представлении номера которых бит с номером $current$ установлен в 0, а $B_1 = B \setminus B_0$.

Тогда возможен один из следующих случаев: в B_0 попало 0 существенных, в B_1 4; или 1 и 3; или 2 и 2; или 3 и 1; или 4 и 0.

Сделаем запрос на множестве B_0. Если это значение равно 1, то значит в B_0 нечетное число существенных переменных. Согласно лемме 3 можем найти одну из них, после чего останавливаем вычисления.

Если значение равно 0, переходим к **шагу 2**.

Шаг 2. Если в B_0 2 существенные переменные, то $current$-й бит в двоичном представлении их номеров равен 0, соответственно $current$-й бит двоичного представления номеров двух других существенных переменных равен 1. Если распределение 4 и 0 или 0 и 4, то $current$-й бит всех существенных переменных одинаковый.

Запросим значение функции на множествах $B_0^0, B_0^1, ..., B_0^{current-1}$, сформированных по множеству B_0. Если в B_0 2 существенных переменных, то опрос найдет нам B_0^i, на котором f равна 1, тогда согласно лемме 3 найдем одну из этих переменных и останавливаем вычисления. Если распределение 4 и 0 или 0 и 4, и на каком-то B_0^i функция равна 1, то аналогично найдем одну из этих переменных и остановим вычисления. Иначе имеет место распределение 0 и 4 или 4 и 0, и на всех B_0^i функция равна 0, то переходим к **шагу 3**.

Шаг 3. Уменьшаем на один значение счетчика $current$, то есть устанавливаем $current = current - 1$. Переходим к **шагу 1**.

Посмотрим, что происходит на шаге 3. Имеем, что или все существенные переменные находятся в B_0, или они все лежат в B_1. Уменьшается значение $current$ на один и выполняется шаг 1. То есть половина множества B_0 с предыдущего шага 1 входит в B_1 на новом шаге 1. Другая половина множества B_0 с предыдущего шага 1 попадает во множество B_0 на новом шаге 1. Половина множества B_1 с предыдущего шага попадают во множество B_0 на новом шаге

1. Другая половина множества B_1 с предыдущего шага попадают во множество B_1 на новом шаге 1.

Если после шага 3 $current = 1$, то на шаге 2 обязательно найдется хотя бы одна существенная переменная, поскольку мы уже не сможем спрятать 4 переменные в одном множестве.

Оценим количество запросов, которое нам потребуется, чтобы найти одну существенную переменную.

Шаг 3 будет выполнен не более $]\log n[-2$ раз. В самом начале алгоритма и после каждого такого шага будет выполнен шаг 1.

То есть шаг 1 и шаг 2 будут выполнены не более $]\log n[-2+1 =]\log n[-1$ раз.

На шаге 1 мы тратим 1 запрос, а на i-м шаге 2 мы тратим не более $]\log n[-i$ запросов.

Если на шаге 1 $f(B_0) = 1$, то применяем лемму 3 к B_0, учитывая, что $|B_0| \leqslant 2^{]\log n[-1}$, для нахождения одной существенной переменной тратим не более $]\log n[-1$ запросов.

Если на шаге 2 найдется множество B_0^i, содержащее одну существенную переменную, то аналогично применяем лемму 3 к этому B_0^i, то есть тратим не более $]\log n[-1$ запросов.

Эти $]\log n[-1$ запросов учитываем один раз, так как один раз применяем лемму 3.

Итого получилось, что для нахождения одной существенной переменной из четырех мы потратим не более $(1+]\log n[-1)+(1+]\log n[-2)+(1+]\log n[-3)+...+(1+1)+]\log n[-1 = (2+]\log n[)(]\log n[-1)/2+]\log n[-1$ запросов.

Лемма доказана. □

Алгоритм, описанный в лемме 7, дальше будем называть **Алгоритм 1**.

Рассмотрим пример применения Алгоритма 1.

Пусть $n = 16$. Переменные x_4, x_5, x_6, x_7 существенные.

$current = 3$.

Шаг 1. Разбиение представлено на рисунке 1.a), $f(B_0) = 0$.

Шаг 2. $f(B_0^0) = 0, f(B_0^1) = 0, f(B_0^2) = 0$. Значит, третий бит двоичного представления всех существенных переменных одинаков.

Шаг 3. $current = 2$.

Шаг 1. Разбиение представлено на рисунке 1.b), $f(B_0) = 0$.

B_0	B_1
0,1,2,3,4,5,6,7	8,9,10,11,12,13,14,15

a)

B_0	B_1
0,1,2,3, 8,9,10,11	4,5,6,7 12,13,14,15

b)

B_0	B_1
0,1, 4,5, 8,9, 12,13	2,3, 6,7, 10,11, 14,15

c)

Рис. 1

Шаг 2. $f(B_0^0) = 0, f(B_0^1) = 0$. Значит, второй бит двоичного представления всех существенных переменных одинаков.

Шаг 3. $current = 1$.

Шаг 1. Разбиение представлено на рисунке 1.c), $f(B_0) = 0$.

Шаг 2. $f(B_0^0) = 1$. Значит, в двоичном представлении двух существенных переменных первый бит равен 0, а у двух других переменных равен 0.

В B_0^0 нечетное число существенных переменных. Применяя лемму 3 к $B_0^0 = \{x_1, x_5, x_9, x_{13}\}$, найдем одну из них. Для этого запрашиваем $f(\{x_1, x_5\})$. Оно равно 1. Поэтому существенная переменная либо x_1,

либо x_5. Запрашиваем значение на $f(\{x_1\})$. Оно равно 0. Поэтому существенная переменная x_5.

Одна существенная переменная найдена. Мы потратили 11 запросов, а по лемме 7 должны были потратить не более 12 запросов.

Лемма 8. *Справедливо неравенство* $\phi(n,6) \leqslant (2+]\log n[)(]\log n[-2)+]\log n[-1$.

Доказательство. Для расшифровки f применим Алгоритм 1 с небольшими изменениями.

Вводим множество всех переменных $B = \{x_0, x_1, .., x_{n-1}\}$.

Вводим переменную — счетчик $current$. Присваиваем ей значение $current = \log n - 1$. Переходим к шагу 1.

Шаг 1. Делим множество B на два равных по мощности множества B_0 и B_1. Во множество B_0 помещаем все переменные из B, в двоичном представлении номера которых бит с номером $current$ установлен в 0, а $B_1 = B \setminus B_0$.

Тогда может случиться одна из следующих ситуаций: в B_0 попало 0 существенных, в B_1 6; или 1 и 5; или 2 и 4; или 3 и 3; или 4 и 2; или 5 и 1; или 6 и 0. Сделаем запрос на множестве B_0. Если это значение равно 1, то значит в B_0 нечетное число существенных переменных, согласно лемме 3, найдем одну из них и останавливаем вычисления.

Если значение равно 0, переходим к шагу 2.

Шаг 2. Предположим, что в B_0 2 существенные переменные. Запросим значение функции на множествах $B_0^0, B_0^1, ..., B_0^{current-1}$, сформированных по множеству B_0. Если в B_0 2 существенные переменные, то опрос найдет нам B_0^i, на котором f равна 1, тогда применяя лемму 3 к этому B_0^i, найдем одну из этих переменных и останавливаем вычисления.

Если же существенная переменная не найдена, то в B_0 не 2 существенные переменные. Предположим, что в B_1 2 существенные переменные. Аналогично предыдущим действиям попробуем найти существенную переменную. Если она найдется, то останавливаем вычисления.

Если вновь не удалось найти существенную переменную, то распределение переменных по двум множествам либо 0 и 6, либо 6 и 0. Переходим к шагу 3.

Шаг 3. Уменьшаем на один значение счетчика $current$, то есть $current = current - 1$. Переходим к шагу 1.

Если после шага 3 $current = 2$, то на шаге 2 обязательно найдется хотя бы одна существенная переменная, так как на шаге 1 в какое-то из множеств B_0 и B_1 попадет 4 переменные.

Оценим количество запросов, которое нам потребуется.

Шаг 3 будет выполнен не более $]\log n[-3$ раз. В самом начале алгоритма и после каждого такого шага будет выполнен шаг 1.

Шаг 1 и шаг 2 будут выполнены не более $]\log n[-3+1 =]\log n[-2$ раз.

На шаге 1 мы тратим 1 запрос, а на i-м шаге 2 мы тратим не более $2(]\log n[-i)$ запросов.

Если на шаге 1 $f(B_0) = 1$, то применяем лемму 3 к B_0, учитывая, что $|B_0| \leqslant 2^{]\log n[-1}$, для нахождения одной существенной переменной тратим не более $]\log n[-1$ запросов.

Если на шаге 2 найдется множество B_j^i, $j \in \{0,1\}$, содержащее одну существенную переменную, то аналогично применяем лемму 3 к этому B_j^i, то есть тратим не более $]\log n[-1$ запросов.

Эти $]\log n[-1$ запросов учитываем один раз, так как один раз применяем лемму 3.

Итого получилось, что для нахождения одной существенной переменной из 5 мы потратим не более $(1 + 2(]\log n[-1)) + (1 + 2(]\log n[-2)) + (1 + 2(]\log n[-3)) + ... + (1 + 2 \cdot 2)+]\log n[-1 = (]\log n[+2)(]\log n[-2)+]\log n[-1$ запросов.

Лемма доказана. □

Лемма 9. *Если* $k = 4p$, $p \in \mathbb{N}$, *то* $\phi(n,k) \leqslant (]\log n[-]\log k[+1)(1 + 2\sum_{i=1}^{p-1}(\phi(n,2i)-]\log n[) + \phi(n,2p)-]\log n[)+]\log n[$.

Доказательство. Доказывать будем по индукции.

База индукции. $k = 4$. Справедливость утверждения леммы следует из леммы 7.

Шаг индукции. Пусть утверждение верно для любого $k < 4p$. Применим Алгоритм 1 нахождения одной существенной переменной с некоторыми изменениями для $k = 4p$.

Вводим множество всех переменных $B = \{x_0, x_1, .., x_{n-1}\}$.

Вводим переменную — счетчик $current$. Присваиваем ей значение $current = \log n - 1$. Переходим к шагу 1.

Шаг 1. Делим множество B на два равных по мощности множества B_0 и B_1. Во множество B_0 помещаем все переменные из B, в

двоичном представлении номера которых бит с номером $current$ установлен в 0, а $B_1 = B \setminus B_0$.

Тогда может оказаться, что в B_0 попало t существенных, а в B_1 $k - t$, где t принимает одно из значений $0, 1, 2, 3, 4, \ldots, k$. Сделаем запрос на множестве B_0. Если это значение равно 1, то в B_0 нечетное число существенных переменных, согласно лемме 3, найдем одну из них, останавливаем вычисления. Если значение равно 0, переходим к шагу 2.

Шаг 2. Переберем все возможные распределения (t и $(k-t)$), где t - четное. Распределения (0 и k) и (k и 0) рассмотрим в последнюю очередь.

Предположим, что распределение 2 и $k-2$. Запустим алгоритм нахождения одной существенной переменной во множестве с меньшим числом существенных переменных при условии, что в нем ровно 2 существенные переменные. Если мы найдем существенную переменную, то останавливаем вычисления. Иначе делаем следующее предположение.

Предположим, что распределение 4 и $k-4$. Запустим алгоритм нахождения одной существенной переменной во множестве с меньшим числом существенных переменных при условии, что в нем ровно 4 существенные переменные. Если мы найдем существенную переменную, то останавливаем вычисления. Иначе делаем следующее предположение.

Повторяем аналогичные рассуждение для распределений t и $k - t$, где $t = 2, 4, ..., k - 2$.

Если переменная не найдена ни при одном их этих распределений, то делаем вывод, что имеет место распределение (0 и k) или (k и 0). Переходим к шагу 3.

Шаг 3. Уменьшаем на один значение счетчика $current$, то есть $current = current - 1$. Переходим к шагу 1.

Оценим количество запросов.

Если после шага 3 $current =]\log k[- 1$, то на шаге 2 обязательно найдется хотя бы одна существенная переменная, так как на шаге 1 во множества B_0 и B_1 попадут по $2^{]\log k[-1}$ переменных, поэтому в этих множествах не будет распределения существенных переменных 0 и k или k и 0. Поэтому шаг 3 будет выполнен не более $]\log n[-]\log k[$ раз.

В самом начале алгоритма и после каждого такого шага будет выполнен шаг 1. Шаг 1 и шаг 2 будут выполнены не более $]\log n[-]\log k[+1$ раз.

На шаге 1 мы тратим 1 запрос.

На шаге 2 для каждого $t \in \{2, 4, ..., 2p-2\}$ мы дважды запускаем алгоритм нахождения одной существенной переменной в предположении, что распределение существенных переменных по множествам t и $k-t$, то есть тратим не более $2(\phi(n,2)-]\log n[) + 2(\phi(n,4)-]\log n[) + ... + 2(\phi(n,2p-2)-]\log n[)$ запросов. Еще на шаге 2 мы один раз запускаем алгоритм нахождения одной существенной переменной для $t = 2p$ и тратим не более $\phi(n,2p)-]\log n[$ запросов.

Из каждой скобки вычитаем $]\log n[$, так как эти запросы делаем, применяя лемму 3, которую используем всего один раз, после чего все вычисления останавливаются. Эти запросы мы добавим отдельным слагаемым в итоговой оценке.

Если на шаге 1 $f(B_0) = 1$, то применяем лемму 3 к B_0, учитывая, что $|B_0| \leqslant 2^{]\log n[-1}$, для нахождения одной существенной переменной тратим не более $]\log n[$ запросов.

Если на шаге 2 найдется множество B^i_j, $j \in \{0,1\}$, содержащее одну существенную переменную, то аналогично применяем лемму 3 к этому B^i, то есть тратим не более $]\log n[$ запросов.

Получается, что для нахождения одной существенной переменной из $k = 4p$ мы потратим не более $(]\log n[-]\log k[+1)(1 + 2\sum_{i=1}^{p-1}(\phi(n,2i)-]\log n[) + \phi(n,2p)-]\log n[)+]\log n[$ запросов.

Лемма доказана. □

Лемма 10. *Если $k = 4p+2$, $p \in \mathbb{N}$, то $\phi(n,k) \leqslant (]\log n[-]\log k[+1)(1+ 2\sum_{i=1}^{p}(\phi(n,2i)-]\log n[))+]\log n[$*

Доказательство. Доказывать будем по индукции.

База индукции. $k = 6$. Справедливость утверждения леммы следует из леммы 8.

Шаг индукции. Пусть утверждение верно для любого $k < 4p+2$.

Дальше доказательство практически повторяет доказательство леммы 9. Отличие только в шаге 2 и в оценке количества запросов, выполняемых на шаге 2: на шаге 2 не будет проверяться распределение $2p$ и $2p$.

На шаге 2 будет выполнено не более $2(\phi(n,2)-]\log n[) + 2(\phi(n,4)-]\log n[) + ... + 2(\phi(n,2p-2)-]\log n[) + 2(\phi(n,2p)-]\log n[)$ запросов.

Получается, что для нахождения одной существенной переменной из $k = 4p+2$ мы потратим не более $(]\log n[-]\log k[+1)(1 + 2\sum_{i=1}^{p}(\phi(n,2i)-]\log n[))+]\log n[$ запросов.

Лемма доказана. □

Введем рекурсивно функцию $d(n,p)$:

$$d(n,1) = 2]\log n[,\ \ d(n,p) = p]\log n[\cdot d(n,[p/2]).$$

Лемма 11. *Если $n > 1$, то $d(n,p)$ — возрастающая по p функция.*

Доказательство. Доказывать будем индукцией по p.

База индукции. Покажем что $d(n,2) \geqslant d(n,1)$: $d(n,2) - d(n,1) = 2]\log n[\cdot d(n,1) - d(n,1) = 2]\log n[\cdot(2]\log n[-1) > 0$.

Предположение индукции. Пусть для любого k, $1 < k < p$, выполняется $d(n,k-1) < d(n,k)$. Докажем, что $d(n,p-1) < d(n,p)$.

Если $p = 2k+1$, то $d(n,p) - d(n,p-1) = p]\log n[\cdot d(n,k) - (p-1)]\log n[\cdot d(n,k) =]\log n[\cdot d(n,k)$. По предположению индукции $d(n,k) > d(n,1) > 0$. Получаем $d(n,p) > d(n,p-1)$.

Если $p = 2k$, то $d(n,p) - d(n,p-1) = p]\log n[\cdot d(n,k) - (p-1)]\log n[\cdot d(n,k-1) = p]\log n[\cdot(d(n,k) - d(n,k-1)) + d(n,k-1)$. По предположению индукции $d(n,k) > d(n,k-1) > 0$. Получаем $d(n,p) > d(n,p-1)$.

Лемма доказана. □

Лемма 12. *Для любых натуральных p справедливо неравенство*

$$d(n,p) \leqslant 2(p]\log n[)^{[\log p]+1} \cdot 2^{-[\log p]([\log p]+1)/2}.$$

Доказательство. Доказывать будем по индукции.

База индукции. Очевидно выполнение леммы для $d(n,1)$.

Предположение индукции. Пусть лемма верна для любого t, $t < p$. Докажем, что лемма верна для p.

Пусть $k = [p/2]$, тогда $k \leqslant p/2$ и $[\log k] = [\log p] - 1$.

По определению и предположению индукции имеем

$$d(n,p) = p]\log n[\cdot d(n,k) \leqslant$$
$$\leqslant p]\log n[\cdot 2(k]\log n[)^{[\log k]+1}\cdot 2^{-[\log k]([\log k]+1)/2} \leqslant$$
$$\leqslant p]\log n[\cdot 2(\frac{p}{2}]\log n[)^{[\log p]}\cdot 2^{-[\log p]([\log p]-1)/2} =$$
$$= 2(p]\log n[)^{[\log p]+1}\cdot 2^{-[\log p]([\log p]-1)/2-[\log p]} =$$
$$= 2(p]\log n[)^{[\log p]+1}\cdot 2^{-[\log p]([\log p]+1)/2}.$$

Лемма доказана. □

Лемма 13. *Для любых натуральных n, $n > 1$, и p справедливо неравенство $\phi(n,2p) \leqslant d(n,p)$.*

Доказательство. Пусть $k = [p/2]$. Согласно леммам 9 и 10 имеем

$$\phi(n,2p) \leqslant (]\log n[-]\log(2p)[+1)(1+2\sum_{i=1}^{k}(\phi(n,2i)-]\log n[))+]\log n[\leqslant$$
$$\leqslant 2]\log n[\cdot\sum_{i=1}^{k}(\phi(n,2i)-]\log n[)+]\log n[\leqslant 2]\log n[\cdot\sum_{i=1}^{k}\phi(n,2i).$$

Будем доказывать лемму по индукцией по p.

База индукции. Из леммы 4 и определения функции $d(n,p)$ следует $\phi(n,2) \leqslant d(n,1)$.

Предположение индукции. Пусть для любого t, $t < p$ верно $\phi(n,2t) \leqslant d(n,t)$.

Согласно лемме 11 функция $d(n,i)$ возрастающая по i, поэтому с учетом предположения индукции получаем

$$\phi(n,2p) \leqslant 2]\log n[\cdot\sum_{i=1}^{k}\phi(n,2i) \leqslant 2]\log n[\cdot\sum_{i=1}^{k}d(n,i) \leqslant$$
$$\leqslant 2k]\log n[\cdot d(n,k) \leqslant p]\log n[\cdot d(n,k) = d(n,p).$$

Лемма доказана. □

Лемма 14. *Если k четное число, то*

$$\phi(n,k) \leqslant d(n,k/2) \leqslant 2k^{[\log k]/2}\cdot]\log n[^{[\log k]}.$$

Доказательство. Согласно леммам 12 и 13 верно

$$\phi(n,k) \leqslant d(n,k/2) \leqslant 2(k]\log n[/2)^{[\log(k/2)]+1} \cdot 2^{-[\log(k/2)]([\log(k/2)]+1)/2}.$$

Учитывая, что $[\log(k/2)] = [\log k - 1] = [\log k] - 1$ и $[\log k] > \log k - 1$, получаем

$$\begin{aligned}\phi(n,k) &\leqslant 2(k]\log n[)^{[\log k]} \cdot 2^{-([\log k]+([\log k]-1)[\log k]/2)} = \\ &= 2(k]\log n[)^{[\log k]} \cdot 2^{-([\log k]+1)[\log k]/2} < 2(k]\log n[)^{[\log k]} \cdot 2^{-\log k[\log k]/2} = \\ &= 2]\log n[^{[\log k]} \cdot k^{[\log k]} \cdot k^{-[\log k]/2} = 2k^{[\log k]/2} \cdot]\log n[^{[\log k]}.\end{aligned}$$

Лемма доказана. □

Докажем теорему 3

1) Понятно, что $\varphi(n,4) \leqslant \varphi(n,3) + \phi(n,4)$.

Используя леммы 6 и 7, получаем

$$\varphi(n,4) \leqslant (]\log n[+2)(]\log n[-1)/2 + 4]\log n[-3.$$

2) Используя следствие 2 и пункт 1 данной теоремы, получаем

$$\varphi(n,5) \leqslant \varphi(n,4) + \phi(n,5) \leqslant (]\log n[+2)(]\log n[-1)/2 + 5]\log n[-3.$$

3) Используя лемму 8 и пункт 2 данной теоремы, получаем

$$\varphi(n,6) \leqslant \varphi(n,5) + \phi(n,6) \leqslant (]\log n[+2)(3]\log n[-5)/2 + 6]\log n[-4.$$

4) Используя следствие 2 и пункт 3 данной теоремы, получаем

$$\varphi(n,7) \leqslant \varphi(n,6) + \phi(n,7) \leqslant (]\log n[+2)(3]\log n[-5)/2 + 7]\log n[-4.$$

5) Пусть $k > 7$, $p = [k/2]$, $k = 2^m + c$, $0 \leqslant c < 2^m$. Тогда, применяя следствие 2 и леммы 4 и 14, получаем

$$\varphi(n,k) \leqslant \varphi(n,2) + \sum_{i=3}^{k} \phi(n,i) <$$
$$< \sum_{i=2}^{p} \phi(n,2i) + d(n,1) + p]\log n[= \sum_{i=1}^{p} d(n,i) + p]\log n[=$$
$$= \sum_{i=1}^{2^{m-1}-1} d(n,i) + \sum_{i=2^{m-1}}^{p} d(n,i) + p]\log n[\leqslant$$
$$\leqslant (2^{m-1}-1)d(n,2^{m-1}-1) + (p-(2^{m-1}-1))d(n,p) + p]\log n[\leqslant$$
$$\leqslant 2k^{m/2}\cdot]\log n[^{m}\cdot\frac{1}{2}(k-2^m+2) +$$
$$+ 2(2^m-2)^{(m-1)/2}\cdot]\log n[^{m-1}\cdot(2^{m-1}-1) + p]\log n[\leqslant$$
$$\leqslant k^{m/2}\cdot]\log n[^{m}\cdot(k-2^m+2) + 2^{m(m+1)/2}\cdot]\log n[^{m-1}+p]\log n[\leqslant$$
$$\leqslant (k-2^{[\log k]}+2)k^{[\log k]/2}\cdot]\log n[^{[\log k]} +$$
$$+ k^{([\log k]+1)/2}\cdot]\log n[^{[\log k]-1}+k]\log n[/2.$$

Теорема доказана.

Список литературы

[1] В.Б.Кудрявцев, С.В. Алешин, А.С. Подколзин. Введение в теорию автоматов. Издательство «Наука», Москва, 1985.

[2] Алешин С.В. Полугруппы и группы автоматов // Интеллектуальные системы. — 2013. — Т. 17, вып. 1–4. — С. 129–141.

[3] Иванов И.Е. О некоторых свойствах автоматов с магазинной памятью // Интеллектуальные системы. — 2014. — Т. 18, вып. 1. — С. 243–252.

[4] Часовских А.А. Условия полноты линейно-p-автоматных функций // Интеллектуальные системы. — 2014. — Т. 18, вып. 3. — С. 203–252.

[5] Александров Д.Е. Об оценках автоматной сложности распознавания классов регулярных языков // Интеллектуальные системы. — 2014. — Т. 18, вып. 4. — С. 161–190.

[6] Гасанов Э.Э. Прогнозирование периодических сверхсобытий автоматами // Интеллектуальные системы. — 2015. — Т. 19, вып. 1. — С. 23–34.

[7] Иванов И.Е. О сохранении периодических последовательностей автоматами с магазинной памятью с однобуквенным магазином // Интеллектуальные системы. — 2015. — Т. 19, вып. 1. — С. 145–160.

[8] Летуновский А.А. Выразимость линейных автоматов относительно расширенной суперпозиции // Интеллектуальные системы. — 2015. — Т. 19, вып. 1. — С. 161–170.

[9] Гербус В.Г. О связи функций автомата и автоматной функции // Интеллектуальные системы. — 2015. — Т. 19, вып. 2. — С. 109–116.

[10] Миронов А.М. Критерий реализуемости функций на строках вероятностными автоматами Мура с числовым выходом // Интеллектуальные системы. — 2015. — Т. 19, вып. 2. — С. 149–160.

[11] Терехина И.Ю. Модель невлияния для квантовых автоматов // Интеллектуальные системы. — 2015. — Т. 19, вып. 2. — С. 183–190.

[12] *Пантелеев П.А.* Об отличимости состояний решетчатых автоматов // Интеллектуальные системы. — 2004. — Т. 8, вып. 1–4. — С. 529–542.

[13] *Кирнасов А.Е.* Об отношении сложностей условного и безусловного установочного экспериментов // Интеллектуальные системы. — 2005. — Т. 9, вып. 1–4. — С. 433–444.

[14] *Уваров Д.В.* О сложности кратных диагностических экспериментов для подмножеств состояний автоматов // Интеллектуальные системы. — 2005. — Т. 9, вып. 1–4. — С. 485–504.

[15] *Кудрявцев В.Б., Грунский И.С., Козловский В.А.* Анализ и синтез автоматов по их поведению // Интеллектуальные системы. — 2006. — Т. 10, вып. 1–4. — С. 345–448.

[16] *Пантелеев П.А.* Об отличимости состояний автомата при искажениях на входе // Интеллектуальные системы. — 2007. — Т. 11, вып. 1–4. — С. 653–678.

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

[18] Ансель Ж. О числе монотонных булевых функций n переменных. — Кибернетический сборник, новая серия, вып. 5, 1968, С. 53–57.

[19] Sokolov N.A. (1982). On the optimal evaluation of monotonic Boolean functions // USSR Computational Mathematics and Mathematical Physics, Volume 22, Issue 2, 1982, Pages 207-220.

[20] Damaschke, P. (2003). On Parallel Attribute-Efficient, Learning // Journal of Computer and System Sciences, Volume 67, Issue 1, August 2003, 46-62.

[21] В. В. Осокин. О расшифровке монотонных булевых функций с несущественными переменными // Дискретная математика, 22:3 (2010), 134–145.

[22] Осокин В.В. Асимптотически оптимальный алгоритм расшифровки разбиения булевого куба на подкубы // Интеллектуальные системы. — 2007. — Т. 11, вып. 1–4. — С. 635–652.

[23] В. В. Осокин. О сложности расшифровки разбиения булевого куба на подкубы // Дискретная математика, 20:2 (2008), 46–62.

[24] Воронин Б.В., Осокин В.В. О сложности расшифровки существенных переменных функции, задающей разбиение булевого куба // Интеллектуальные системы. — 2008. — Т. 12, вып. 1–4. — С. 159–178.

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

[26] Осокин В.В. О параллельной параметро-эффективной расшифровке псевдо-булевских функций // Интеллектуальные системы. — 2010. — Т. 14, вып. 1–4. — С. 429–458.

[27] Zolotykh N.Yu., Shevchenko V.N. (1997). Lower Bounds for the Complexity of Learning Half-Spaces with Membership Queries // ALT'98, Otzenhausen, Germany, October 8-10, 1998.

[28] Nakamura A., Abe N. (1995) Exact learning of linear combinations of monotone terms from function value queries // Theoretical Computer Science, Volume 137, Issue 1, 159-176, 1995.

[29] Гасанов Э.Э., Ниязова З.А. Расшифровка арифметических сумм малого числа монотонных конъюнкций // Материалы XI Международного семинара Дискретная математика и ее приложения (Москва, 18-23 июня 2012 г.). — Изд-во механико-математического ф-та МГУ Москва, 2012. — С. 335–337.

[30] Ryuhei Uehara, Kensei Tsuchida, and Ingo Wegener. Optimal Attribute-Efficient Learning Of Disjunction, Parity, And Threshold Functions // EuroCOLT '97 Proceedings of the Third European Conference on Computational Learning Theory, 1997.

[31] Adam R. Klivans, Rocco A. Servedio. Toward Attribute Efficient Learning of Decision Lists and Parities // The Journal of Machine Learning Research Volume 7, 2006.

[32] Vitaly Feldman. On Attribute Efficient and Non-adaptive Learning of Parities and DNF Expressions // The Journal of Machine Learning Research Volume 8, 2007

[33] Valiant L. G. A theory of the learnable // ACM Press New York, NY, USA, 1984, Volume 27, Issue II, 1134-1142.

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

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

[36] Костюченко О.В. Сплайновая интерполяция с плавающими узлами // Интеллектуальные системы. — 2007. — Т. 11, вып. 1–4. — С. 715-720.

[37] Гасанов Э.Э. Расшифровка линейных функций ранжирования // Материалы XI Международного семинара "Дискретная математика и ее приложения посвященного 80-летию со дня рождения академика О.Б.Лупанова (Москва, 18-23 июня 2012 г.). Изд-во мех-мат фак-та МГУ. 2012. С. 332-334.

[38] Хегай С.И. Расшифровка полиномиальных функций ранжирования // Интеллектуальные системы. 2015. 19:1. 213 – 230.

Complexity of learning linear Boolean functions

A. V. Bistrigova

Abstract: The exact learning of a linear Boolean function of k relevant variables is considered in this work. Here we obtain the complexity of learning such functions for $k < 4$ and upper bounds for other values of k.

Keywords: linear Boolean functions, exact learning.