Московский государственный университет имени
М. В. Ломоносова
Факультет вычислительной математики и кибернетики
Кафедра математической кибернетики

Курсовая работа по теме

«Сложность расшифровки счетчика делимости на три»

Студент 318 группы
М. М. Сакович

Научный руководитель
д.ф-м.н., доцент. С. Н. Селезнева

Москва, 2017

Содержание

Введение **3**

1. Основные определения **5**

2. Постановка задачи **6**

3. Основная часть **7**

3.1. Решение 7

3.2. Результаты 10

Литература **11**

Введение

Впервые понятие «черного ящика» для задач расшифровки ввел В.К. Коробков в 1963 году в [1], где он рассматривал задачу расшифровки класса монотонных функций. Пусть задан оператор A_f, вычисляющий для произвольной точки $\tilde{\alpha} = (\alpha_1, \alpha_2, \ldots, \alpha_n) \in B^n$ значение функция алгебры логики из некоторого класса $f(x_1, x_2, \ldots, x_n)$ в этой точке. Функция может существенно зависеть от любого числа своих переменных. Образно говоря, мы имеем дело с «черным ящиком», который имеет n входов и один выход и про который известно, что он реализует некоторую функцию алгебры логики от n переменных из определенного класса K. Нам нужно определить какую именно функцию он реализует.

Рассмотрим множество $\{F\}$ алгоритмов, решающих указанную задачу. То есть для произвольной функции алгебры логики $f(x_1, \ldots, x_n)$ из заданного класса K любой алгоритм F с помощью оператора A_f определяет от каких переменных существенно зависит функция f. Очевидно, что любой паре — алгоритму F и функции $f(x_1, \ldots, x_n)$ можно сопоставить число $\phi(F, f)$ — число обращений к оператору A_f в процессе нахождения существенных переменных функции f с помощью алгоритма F. Оценим качество алгоритма F функцией $\phi(F, n) = \max_{f \in K_n} \phi(F, f)$, где K_n — множество всех функций от n переменных из класса K. Класс K можно характеризовать функцией $\phi(n) = \min_F \phi(F, n)$, где минимум берется по всем алгоритмам F, решающим поставленную задачу.

Деревья решений являются адекватной моделью вычисления, если нужно идентифицировать неизвестный объект из заданного класс, задавая вопросы и получая конечное число ответов. Запросы могут задаваться условно, т.е. вопрос может зависеть от предыдущих ответов. Цель состоит в том, чтобы минимизировать наихудшее число запросов, которое является глубиной дерева решений.

Булевые деревья решений являются наиболее важным подклассом деревьев решений. Известная булева функция f должна быть оценена на неизвестном входном наборе a. Мы можем запросить биты a_i набора a. По сути, делая запрос, в зависимости от полученного ответа мы переходим в левую или правую часть поддерева. Мерой сложности является количество запросов. Все запросы имееют одинаковую стоимость. Заметим, что минимальная глубина дерева принятия решений равна минимальной глубине программ ветвления или диаграмм двоичных решений.[2]

Здесь рассматривется еще один класс задач. Дается класс F булевых функций от n переменных и мы хотим выяснить, какие от каких перменных существенно зависит функция $f \in F$.

В статье [3] были получены результаты для классов OR, PAR и THR. Класс OR содержит все $OR_S, S \subseteq 1, \ldots, n$, т.е. дизъюнкция всех x_i, где $i \in S$. Фиктивные переменные заменяются константой 0. Так же рассматривался класс $OR(k)$, содержащий все OR_s, где $|S| = k$.

Для классов OR и $OR(k)$ были получены нижние оценки n и $\lceil \log_2(\binom{k}{n}) \rceil$ соответсвенно. Класс $OR(k)$ можно распознать за $k\lceil \log_2(\frac{n}{k}) \rceil + 2k - 2$ запроса.

Классы PAR и $PAR(k)$ для функции четности, которая возвращает значение 1, если число единиц на входе четное, и 0 иначе.

Для класса PAR нижняя оценка n. Для класса $PAR(k)$ был получен алгоритм из $O(k \log_2(\frac{n}{k}))$ запросов.

Классы $THR, THR(k)$ и $THR_t(k)$ порождаются пороговой функцией. Пороговая функция является основой дискретных нейронных сетей. Пороговое значение t определяет, сожержит ли вход по крайней мере t единиц. Для классов THR и $THR(k)$ пороговое значение неизвестно. Нижние оценки для классов $THR, THR(k)$ и $THR_t(k)$ равны $n-1+\lceil \log_2(n+1) \rceil$, $\lceil \log_2(kC_n^k+2) \rceil$ и $\lceil \log_2(C_n^k) \rceil$ соответсвенно. Так же были получены алгоритмы: за $n-1+\lceil \log_2(n+1) \rceil$ запросов для THR, за $2(k-1)\log_2(\frac{n-1}{k-1}) + 6k - 6 + \lceil \log_2(n+2) \rceil$ запросов для $THR(k)$ и за $(k-1)\log_2(\frac{n-1}{k-1}) + 3k - 3 + \lceil \log_2(n+2) \rceil$ запросов для $PAR_t(k)$.

В данной работе рассмотрим еще несколько классов: A_0, A_1, A_2.

1. Основные определения

Пусть $E_2 = \{0,1\}$. Набор $(\alpha_1, \alpha_2, \ldots, \alpha_n)$, где $\alpha_i \in E_2, 1 \leq i \leq n$, называется булевым или двоичным набором (вектором). Элементы набора называют координатами. Число n называется длинной набора. Далее двоичный набор длины n будем обозначать $\tilde{\alpha}$.

Множество всех двоичных наборов длины n образует n-мерный булев(или двоичный) куб, который называют также единичным n-мерным кубом и обычно обозначают B^n.

Весом набора $\tilde{\alpha}$ (обозначение $|\tilde{\alpha}|$) называют число его координат, равных 1, т.е.

$$|\tilde{\alpha}| = \sum_{i=1}^{n} \alpha_i.$$

Наборы $\tilde{\alpha} \in B^n$ называют вершинами куба B^n. Множество всех вершин куба B^n, имеющих вес k, называется k-м слоем куба B^n (обозначение B_k^n).

Набор, все координаты которого равны 0, будем называть нулевым. Набор, все координаты которого равны 1 - единичным. Вес нулевого набора равен 0, а вес единичного - n.

Наборы будем называть соседними, если они различаются только в одной координате.

Функция $f(x_1, \ldots, x_n)$, определенная на множестве $B^n = \{0,1\}^n$ и принимающая знаечения из множества $\{0,1\}$, называется функцией алгебры логики (булевой функцией). Множество всех булевых функций обозначим P_2.

Переменная x_i $(1 \leq i \leq n)$ функции $f(x_1, \ldots, x_{i-1}, x_i, x_{i+1}, \ldots, x_n)$ называется существенной, если можно указать такие наборы $\tilde{\alpha}$ и $\tilde{\beta}$, соседние по i-ой компоненте, (т.е $\tilde{\alpha} = (\alpha_1, \ldots, \alpha_{i-1}, 0, \alpha_{i+1}, \ldots, \alpha_n)$ и $\tilde{\beta} = (\alpha_1, \ldots, \alpha_{i-1}, 1, \alpha_{i+1}, \ldots, \alpha_n)$), что $f(\tilde{\alpha}) \neq f(\tilde{\beta})$. В противном случае переменная назывется фиктивной. Если у функции все переменные фиктивные, то она является константой.

Теорема. Нижняя оценка для класса F, мощности $|F|$, равна $\lceil \log_2(|F|) \rceil$.

Теорема. Для всех $n \geq 1$ верно

$$\sum_{k=0}^{n} C_n^k = 2^n. [5]$$

Теорема. Средняя сложность алгоритма бинарного поиска $L^{cp}(n) \leq \log_2(n)$. [6]

2. Постановка задачи

Рассмотрим функции: $\tau_0^n(x_1,\ldots,x_n), \tau_1^n(x_1,\ldots,x_n), \tau_2^n(x_1,\ldots,x_n) \in P_2^n$, такие что

$$\tau_i(\tilde{\alpha}) = \begin{cases} 1, & |\tilde{\alpha}| \bmod 3 = i, \\ 0, & |\tilde{\alpha}| \bmod 3 \neq i. \end{cases}$$

Пусть $A_i(n) = \{\tau_i^k(x_{i_1},\ldots,x_{i_k}) \mid 1 \leq i_1 < i_2 < \ldots < i_k \leq n,\, k = 0,\ldots,n\}\, i \in \{0,1,2\}$. Попытаемся узнать, от каких перменных существенно зависит функция $f \in A_i(n), i = 0,1,2$.
Задача состоит в том, чтобы узнать, достигается ли нижняя оценка для классов $A_0(n), A_1(n), A_2(n)$, и построить алгоритм, для которого она достигается.

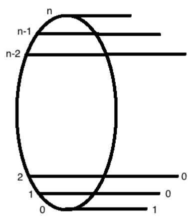

Рис. 2.1: Класс $A_0(n)$.

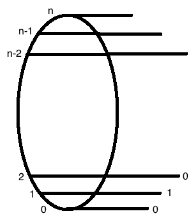

Рис. 2.2: Класс $A_1(n)$.

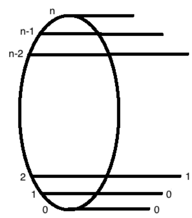

Рис. 2.3: Класс $A_2(n)$.

3. Основная часть

3.1. Решение

Чтобы дать нижнюю оценку, посчитаем мощность классов $A_i(n)$ $(i = 0, 1, 2)$.

Лемма 1. *Мощность классов $A_0(n)$ и $A_1(n)$ равна 2^n.*

Доказательство. Рассмотрим класс $A_0(n)$. Он порождается функцией $\tau_0(x)$, зависящей от n переменных. Из этих n переменных можем выбрать 0 существенных переменных, т.е. C_n^0, 1 существенную перменную, т.е. C_n^1 и.т.д.. Таким образом,

$$|A_0(n)| = C_n^0 + C_n^1 + \cdots + C_n^n = \sum_{k=0}^{n} C_n^k = 2^n.$$

Аналогично, $|A_1(n)| = 2^n$. $\square$

Лемма 2. *Мощность класса $A_2(n)$ равна $2^n - n$.*

Доказательство. Класс $A_2(n)$ порождается функцией $\tau_2(x)$, которая завивит от n переменных. В отличие от классов $A_0(n)$ и $A_1(n)$, если функция из класса $A_2(n)$ существенно зависит от одной переменной, то она никогда не примет значение 1, значит мы не сможем распознать её. Таким образом

$$|A_2(n)| = C_n^0 + C_n^2 + \cdots + C_n^n = \sum_{k=0}^{n} C_n^k - C_n^1 = 2^n - n.$$

$\square$

Зная мощности классов, можно дать нижнюю оценку

$$\begin{aligned}
&1. \lceil \log_2 |A_0(n)| \rceil = \log_2 2^n = n \\
&2. \lceil \log_2 |A_1(n)| \rceil = \log_2 [2^n] = n \\
&3. \lceil \log_2 |A_2(n)| \rceil = \lceil \log_2 (2^n - n) \rceil.
\end{aligned}$$

Лемма 3. *При любых $n > 2$ верно:*

$$\lceil \log_2 (2^n - n) \rceil = n$$

Доказательство. Из свойств логарифмов:

$$\lceil \log_2(2^n - n) \rceil = \lceil \log_2(2^n) + \log_2(1 - \frac{n}{2^n}) \rceil =$$
$$= \lceil n + \log_2(1 - \frac{n}{2^n}) \rceil$$

При $n > 2$ значение $\frac{n}{2^n} < 1$, убывает и ограничено нулем снизу.
Значит $-1 < \log_2(1 - \frac{n}{2^n}) < 0$. Отсюда следует, что $\lceil \log_2(1 - \frac{n}{2^n}) \rceil = 0$, откуда $\lceil n + \log_2(1 - \frac{n}{2^n}) \rceil = n$. □

Посмотрим, достигаюся ли нижние оценки для классов $A_i(n)$ $(i = 0, 1, 2)$.

Построим алгоритм из n запросов для функции из класса $A_0(n)$.
Алгоритм A_0.
Шаг 1. Рассмотрим функцию f из класса $A_0(n)$ на нулевом наборе

$$\tau_0(0, 0, \ldots, 0) = \{0 \bmod 3 = 0\} = 1.$$

Не задавая вопроса, мы знаем, что любая функция из класса $A_0(n)$ на нулевом наборе имеет значение 1.
Шаг i. Рассмотрим набор $\alpha_i = (0, 0, \ldots, 0, 1, 0, \ldots, 0)$, в котором на i-м месте стоит 1, а на всех остальных местах 0. Если $f(\alpha_i) = 1$, то переменная x_i не влияет на значение функции, т.е. x_i - фиктивная переменная, в противном случае x_i - существенная
Применим шаг i для всех $i = 1, 2, \ldots, n$.
Т.о. делаем n запросов $\alpha_1, \alpha_2, \ldots, \alpha_n$ и для каждой из перменных $x_1, x_2, \ldots, x_n$ узнаем, является она существенной или фиктивной. Отметим, что данный алгоритм является безусловным.

Теорема 1. *Сложность расшифровки класса $A_0(n)$ равна n.*

Доказательство. Как показано выше, нижняя оценка равна n. Алгоритм A_0 дает верхнюю оценку, которая равна n. Следовательно сложность расшифровки класса $A_0(n)$ равна n. □

Построим алгоритм из n запросов для функции из класса $A_1(n)$.
Алгоритм A_1.
Шаг 1. Рассмотрим функцию f из класса $A_1(n)$ на нулевом наборе

$$\tau_1(0, 0, \ldots, 0) = \{0 \bmod 3 = 0\} = 0.$$

Из этого следует, что не задавая вопроса, мы знаем, что любая функция из класса $A_1(n)$ на нулевом наборе имеет значение 0. Заметим, что если функция f существенно зависит от всех своих переменных, то на любом наборе из первого слоя, функция f принимает значение 1.

Шаг i. Рассмотрим набор $\alpha_i = (0, 0, \dots, 0, 1, 0, \dots, 0)$, в котором на i-м месте стоит 1, а на всех остальных местах 0. Если $f(\alpha_i) = 0$, то переменная x_i не влияет на значение функции, т.е. x_i - фиктивная переменная, в противном случае x_i - существенная

Применим шаг i для всех $i = 1, 2, \dots, n$.

Т.о. делаем n запросов $\alpha_1, \alpha_2, \dots, \alpha_n$ и для каждой из перменных $x_1, x_2, \dots, x_n$ узнаем, является она существенной или фиктивной. Отметим, что данный алгоритм также является безусловным.

Теорема 2. *Сложность расшифровки класса $A_1(n)$ равна n.*

Доказательство. Как показано выше, нижняя оценка равна n. Алгоритм A_1 дает верхнюю оценку, которая равна n. Следовательно сложность расшифровки класса $A_1(n)$ равна n. □

Построим алгоритм распознавания функции из класса $A_2(n)$

Алгоритм A_2.

Шаг 1. Рассмотрим функцию f из класса $A_2(n)$ на наборе $\tilde{\alpha} = (1, 1, 0, \dots, 0)$. Если $f(\tilde{\alpha}) = 1$, то переходим к шагу 3. В противном случае переходим к шагу 2.

Шаг 2. Заменим в наборе $\tilde{\alpha}$ первый встречающийся 0 на 1. Если $f(\tilde{\alpha}) = 1$, то переходим к шагу 3. Если набор $\tilde{\alpha}$ - единичный и $f(\tilde{\alpha}) = 0$, то все переменные функции f - фиктивные, иначе повторяем шаг 2.

Шаг 3. Не нарушая общности рассуждений, пусть последняя единица в наборе $\tilde{\alpha}$ стоит на k-м месте ($k \leq n$). Обозначим такой набор как $\tilde{\alpha_k}$. Тогда переменная x_k - существенная. Среди переменных $x_1, x_2, \dots, x_{k-1}$ ровно одна существенная, т.к. $\tau_2(\tilde{\alpha})$ может принять первый раз значение 1 только в том случае, если на месте двух первых встретившихся существенных переменных в наборе $\tilde{\alpha}$ стоят единицы, а так как добавление единицы на k-е место обращает функцию $\tau_2(\tilde{\alpha})$ в 1, то x_k - вторая существенная переменная, а первая нахоится среди первых $k - 1$ переменных.

Шаг 4. Найдем первую существенную переменную. Для этого построим набор

$$\alpha_{\lceil \frac{k-1}{2} \rceil_{k-1}} = (1, 1, \dots, 1, 0, \dots, 0, 1, 0, \dots, 0),$$

который получается из набора $\tilde{\alpha}$ заменой 1, стоящих на местах $\lceil \frac{k-1}{2} \rceil, \lceil \frac{k-1}{2} \rceil + 1, \dots, k - 1$ на 0.

Если $f(\alpha_{\lceil \frac{k-1}{2} \rceil_{k-1}}) = 0$, то существенная переменная находится среди $x_{\lceil \frac{k-1}{2} \rceil}, \dots, x_{k-1}$, в противном случае существенная переменная среди $x_1, \dots, x_{\lfloor \frac{k-1}{2} \rfloor}$.

Т.о мы сузили область поиска вдвое. Аналогичным образом будем заменять половину оставшихся 1 на 0, в зависимости того, в какую половину попадает существенная переменная, до тех пор, пока не останется одна 1, соответсвующая переменной x_i.

Переменная x_i - является существенной. Пусть $\alpha_{ik} = (0, \dots, 0, 1, 0, \dots, 0, 1, 0, \dots, 0)$, где на i-м и k-м местах стоят 1, а на всех

остальных 0. Т.к. x_i и x_k - существенные, то $f(\alpha_{ik}) = 1$.
Отметим, что на шагах 1-4 алгоритм является условным.
Шаг 5. Осталось найти все существенные переменные среди $n - k$ оставшихся. Для этого составим $n - k$ безусловных запросов:

$$\begin{aligned}
\alpha_{ik_1} &= (0, \dots, 0, 1, 0, \dots, 0, 1, 1, 0, \dots, 0) \\
\alpha_{ik_2} &= (0, \dots, 0, 1, 0, \dots, 0, 1, 0, 1, \dots, 0) \\
&\dots \\
\alpha_{ik_n} &= (0, \dots, 0, 1, 0, \dots, 0, 1, 0, 0, \dots, 1)
\end{aligned}$$

Если $f(\alpha_{ik_j}) = 1$ (где $j = k + 1, \dots, n$), то переменная x_j - фиктивная. В противном случае x_j - существенная.
Посчитаем, сколько запросов нам понадобилось.
На первом шаге 1 запрос. На втором шаге нам понадобилось $k - 2$ запроса, где $k \leq n$. На 5-м шаге $n-k$. Четвертый шаг по сути является алгоритмом бинарного поиска, сложность которого $\leq \log_2(n)$. Таким образом, в худшем случае сложность приведенного алгоритма равна $1 + k - 2 + n - k + \log_2(n) = n - 1 + log_2(n)$.

Теорема 3. *Ассимптотическая сложность расшифровки класса $A_2(n)$ равна n.*

Доказательство. Как показано выше, нижняя оценка при $n > 2$ равна n. Рассмотрим случаи, когда $n = 1$ и $n = 2$.
При $n = 1$ функция $f(|\alpha|) \in A_2(n)$ является константой, так как никогда не примет значение 1. При $n = 2$ нижняя оценка равна 1. Получим верхнюю оценку. Сделаем запрос $\alpha(1, 1)$. Если $f(\alpha) = 1$, то обе переменные существенные, в противном случае функция является константой. Таким образом для $n = 1, 2$ нижняя и верхняя оценки совпадают и равны $n - 1$.
При $n > 2$ алгоритм A_2 дает верхнюю ассимптотическую оценку, которая равна n. Следовательно сложность расшифровки класса $A_2(n)$ ассимптотически равна n. □

3.2. Результаты

Для классов $A_0(n)$ и $A_1(n)$ получена нижняя оптимальная оценка в n шагов, которая достигается на представленом безусловном алгоритме. Для класса $A_2(n)$ получена сложность ассимптотически равная n.

Литература

[1] Коробков В.К. *О монотонных функциях алгербы логики.* Сб. "Проблемы кибернетики вып 13, М., "Наука 1965, 5-27.

[2] Ingo Wegener *The complexity of boolean functions.* Wiley-Teubner, 1987.

[3] Ryuhei Uehara, Kensei Tsuchida, Ingo Wegener *Optimal attribute-efficient learning of disjunction, parity, and threshold functions.* 1996.

[4] Гаврилов Г.П., Сапоженко А.А. *Сборник задач по дискретной математике.* М.:ФИЗМАТЛИТ, 2004.

[5] Слезнева С.Н. *Лекции по "Избранным вопросам дискретной математики" 3-й курс, 318 группа.* 2016.

[6] Алексеев В.Б. *Введение в теорию сложности алгоритмов.* М.: Изд. отдел ф-та ВМиК МГУ, 2002.