# Параллельный доступ к базам данных и булевы функции

М.Н. Назаров

## 1. Введение

Задача вычисления булевых функций на нескольких независимых наборах переменных достаточно известна. Например, интересные результаты были получены в [1] и [2]. Однако, совершенно новый подход к решению данной задачи был реализован в [3]. В этой работе представлен параллельный алгоритм вычисления булевой функции на нескольких независимых наборах, который может быть интерпретирован как алгоритм параллельного извлечения информации из базы данных по нескольким независимым адресам. Исследован случай максимального распараллеливания алгоритмов при полиномиально ограниченном количестве процессоров. Главным ограничением является необходимость считывания всех запрашиваемых бит за одно обращение к внешнему носителю. Это ограничение связано с тем, что скорость такого обращения существенно ниже скорости работы процессора. Поэтому общее время доступа к базе данных может быть минимально при одном обращении к внешнему носителю, хотя это и потребует дополнительных вычислений.

Вызывает интерес задача криптографической защиты базы данных при соблюдении всех описанных выше ограничений. Кроме того, большое практическое значение имеет задача минимизации числа процессоров при соблюдении главного ограничения алгоритмов. Данная работа посвящена решению указанных задач.

Автор благодарит профессора А.Е. Андреева за научное руководство, а также В.Б. Кудрявцева и Э.Э. Гасанова за внимание и поддержку.

## 2. Основные определения и исходный алгоритм

Рассмотрим специальную параллельную модель, описывающую некоторую абстрактную базу данных, в которой информация представлена в виде таблиц для булевых функций $n$ переменных $f(x_1, x_2, \ldots, x_n)$. Более точно, база данных представляет собой набор булевых функций реализованных таблицами, входные последовательности каждой из которых являются некоторой фиксированной областью в памяти. Процессоры могут различать индексы этих булевых функций и получать значение функции (один бит) на выходе. Другими словами, каждый набор значений переменных является адресом, по которому хранится один бит информации. Под доступом к базе данных понимается считывание информации с внешних носителей по определенным адресам, что фактически означает вычисление булевой функции на заданных наборах переменных. Отметим, что процессоры могут выполнять запросы к внешней памяти (то есть базе данных), затрачивая специальное время, обозначенное **extime**.

Опишем некоторые предварительные результаты из [3], учитывая, что сложность всех алгоритмов оценивается по трем параметрам:

**ptime** — время извлечения информации из базы данных;

**bp** — количество процессоров;

**mem** — размер базы данных.

Пусть $GF(q)$ — поле Галуа размерности $q = 2^k$ $(k \geqslant 0)$. Рассмотрим множество $GF(q)^4$ как четырехмерное линейное пространство. Через $l(A, u)$ обозначим прямую, проходящую через точку $A$ параллельно вектору $\vec{h} = (1, u, u^2, u^3)$. Если $U$ является подмножеством $GF(q)$ размерности $r^3$, то через $SL_4(U)$ обозначим множество всех прямых $l(A, u)$ таких, что $A \in GF(q)^4$ и $u \in U$. Кроме того, определим набор точек $l^{\#}(A, u) = l(A, u) \setminus A$. В работе [3] описан алгоритм **A**, который для любой входной последовательности наборов $A_1, A_2, \ldots, A_r$, где $A_i \in GF(q)^4$ возвращает последовательность $u_1, u_2, \ldots, u_r$, $(u_i \in U \subseteq GF(q))$ такую, что для любой пары $j_1, j_2$, $(j_1 \neq j_2)$ выполнено

$$l^{\#}(A_{j1}, u_{j1}) \cap l^{\#}(A_{j2}, u_{j2}) = \varnothing.$$

Отметим, что этот алгоритм имеет следующую сложность:

$$\mathbf{bp}(\mathbf{A}) = \mathrm{O}(r^7 k^2),$$
$$\mathbf{ptime}(\mathbf{A}) = \mathrm{O}(\log r + \log k).$$

Опишем теперь процедуры, необходимые для построения алгоритма $DP_{r,f}$ [3], который производит параллельное вычисление булевой функции $f : \{0,1\}^n \to \{0,1\}$ на $r$ независимых наборах переменных $X_1, X_2, \ldots, X_r$.

Исходное пространство $\{0,1\}^n$ разобьем на два подпространства

$$\{0,1\}^n = \{0,1\}^{4k} \times \{0,1\}^{n-4k},$$

где $k = \lceil 3\log r + \log n \rceil$. Тогда все входы $X_1, X_2, \ldots, X_r$ можно представить в виде $X_i = A_i B_i$, $i = 1, 2, \ldots, r$, где $A_i \in \{0,1\}^{4k}$, а $B_i \in \{0,1\}^{n-4k}$. Первое подпространство рассмотрим как четырехмерное линейное пространство $GF(q)^4$, где $q = 2^k$.

Для всех $A \in GF(q)^4$ рассмотрим булевы функции
$f_A : \{0,1\}^{n-4k} \to \{0,1\}$ такие, что $f_A(B) = f(A,B)$.

Для всех $l \in SL_4(U)$ рассмотрим булевы функции
$g_l : \{0,1\}^{n-4k} \to \{0,1\}$ такие, что $g_l(B) = \bigoplus\limits_{A \in l} f_A$.

Пусть каждой функции $f_A$ и $g_l$ соответствует своя таблица, хранимая на отдельном внешнем носителе и обрабатываемая своим процессором. Совокупность этих функций представляет собой всю базу данных, то есть позволяет реализовать булеву функцию $f$. Процессор может обратиться к любой из этих функций, затратив некоторое время **extime**. Для наглядности можно объединить все эти таблицы в одну и получить представление функции $f$ (базы данных) в виде таблицы булевых функций от $n - 4k$ переменных.

Здесь $m$ равняется количеству точек в первом подпространстве разбиения, то есть $m = |GF(q)^4| = 2^{4k}$, а $p$ равняется количеству прямых в множестве $SL_4(U)$, то есть $p = |SL_4(U)|$. Каждый столбец таблицы хранится на своем носителе и обслуживается отдельным процессором. Главное ограничение алгоритма означает, что для любых $r$ входов из каждого столбца таблицы можно извлечь только один бит.

| $y_1$ | $y_2$ | $\dots$ | $y_{n-4k}$ | $f_{A1}$ | $f_{A2}$ | $\dots$ | $f_{Am}$ | $g_1$ | $g_2$ | $\dots$ | $g_p$ |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | $\dots$ | 0 | $*$ | $*$ | $\dots$ | $*$ | $*$ | $*$ | $\dots$ | $*$ |
| 0 | 0 | $\dots$ | 1 | $*$ | $*$ | $\dots$ | $*$ | $*$ | $*$ | $\dots$ | $*$ |
| $\dots$ | $\dots$ | $\dots$ | $\dots$ | $\dots$ | $\dots$ | $\dots$ | $\dots$ | $\dots$ | $\dots$ | $\dots$ | $\dots$ |
| 1 | 1 | $\dots$ | 1 | $*$ | $*$ | $\dots$ | $*$ | $*$ | $*$ | $\dots$ | $*$ |

Таблица 1.

Используем описанные функции для реализации $f$. Заметим, прежде всего, что

$$g_l(B)\oplus\left(\bigoplus_{A^*\in l\setminus\{A\}} f_{A^*}(B)\right)=\left(\bigoplus_{A\in l} f_A\right)\oplus\left(\bigoplus_{A^*\in l\setminus\{A\}} f_{A^*}(B)\right)=f_A(B).$$

При помощи алгоритма $\mathbf{A}(A_1,A_2,\dots,A_r)$, упомянутого выше, можно найти $r$ элементов $(u_1,u_2,\dots,u_r)$ из $U$ таких, что для любой пары $j_1\neq j_2$

$$l^{\#}(A_{j1},u_{j1})\cap l^{\#}(A_{j2},u_{j2})=\varnothing.$$

Следовательно, можно построить следующую систему

$$f(A_i,B_i)=g_{l(A_i,u_i)}(B_i)\oplus\left(\bigoplus_{A^*\in l^{\#}(A_i,u_i)} f_{A^*}(B_i)\right),\qquad i=1,2,\dots,r.$$

Из свойств алгоритма $\mathbf{A}$ следует, что любая из функций $f_A$ или $g_l$ может участвовать только в одном уравнении последней системы. Иначе, при пересечении прямых могла возникнуть ситуация, когда необходимо было бы вычислить два значения $f_A(B_1)$ и $f_A(B_2)$ для одной точки $A$, но на различных наборах $B_1$ и $B_2$, а это привело бы к нарушению главного ограничения алгоритма.

Дадим непосредственное описание процедур алгоритма $DP_{r,f}$, следуя работе [3].

Введем следующие обозначения:

$$\vec{A}=(A_1,A_2,\dots,A_r),\ \vec{B}=(B_1,B_2,\dots,B_r),\ \vec{u}=(u_1,u_2,\dots,u_r)=\mathbf{A}(\vec{A}).$$

Определим две вспомогательные процедуры:

$$\mathbf{B}_1(A,u,A^*)=\begin{cases}1, & A^*\in l^{\#}(A,u);\\ 0, & A^*\notin l^{\#}(A,u).\end{cases}$$

$$\mathbf{B}_2(A,u,l)=\begin{cases}1, & l=l(A,u);\\ 0, & l\neq l(A,u).\end{cases}$$

Далее построим процедуры:

$$\mathbf{B}_3(A,\vec{A},\vec{B},\vec{u})=\bigvee_{i=1}^{r}\mathrm{B}_1(A_i,u_i,A)\cdot B_i,$$

$$\mathbf{B}_4(l,\vec{A},\vec{B},\vec{u})=\bigvee_{i=1}^{r}\mathrm{B}_2(A_i,u_i,l)\cdot B_i.$$

Заметим, что две последние процедуры получают на выходе корректные входы для системы, реализующей $f(A_i,B_i)$. Действительно, для любого $A\in GF(q)^4$ выполнено следующее, если $A\in l^{\#}(A_i,u_i)$, то $\mathbf{B}_3(A,\vec{A},\vec{B},\vec{u})=B_i$ и $\mathbf{B}_4(l(A_i,u_i),\vec{A},\vec{B},\vec{u})=B_i$. Введем обозначения

$$B_A=\mathbf{B}_3(A,\vec{A},\vec{B},\vec{u})\text{ и }B_l=\mathbf{B}_4(l(A_i,u_i),\vec{A},\vec{B},\vec{u}).$$

Определим переменные

$$Y_A=f_A(B_A),\ A\in GF(q)^4\text{ и }Z_l=g_l(B_l),\ l\in SL_4(U).$$

Пусть $\vec{Z}$ представляет собой набор переменных $Z_l$, где $l\in SL_4(U)$, а $\vec{Y}$ — это набор переменных $Y_A$, где $A\in GF(q)^4$.

Построим процедуру

$$\mathbf{B}_6(A,u,\vec{Z},\vec{Y})=\left(\bigvee_{l\in SL_4(U)}Z_l\cdot\mathbf{B}_2(A,u,l)\right)\oplus\left(\bigoplus_{A^*\in GF(q)^4}Y_{A^*}\cdot\mathbf{B}_1(A,u,A^*)\right).$$

Заметим, что в первых скобках реализуется $Z_{l(A,u)}$, а во вторых скобках — $\bigoplus\limits_{A^*\in l^{\#}(A,u)}Y_{A^*}$. Следовательно,

$$\mathbf{B}_6(A_i, u_i, \vec{Z}, \vec{Y}) = Z_{l(A_i,u_i)} \oplus \left( \bigoplus_{A \in l^{\#}(A_i,u_i)} Y_A \right) =$$

$$= g_{l(A_i,u_i)}(B_i) \oplus \left( \bigoplus_{A \in l^{\#}(A_i,u_i)} f_A(B_i) \right) = f(A_i, B_i).$$

Таким образом, получены все процедуры, необходимые для описания алгоритма и решения поставленных задач.

**Алгоритм** $DP_{r,f}$

1) Применим алгоритм $\mathbf{A}$ ко входу $(A_1, A_2, \ldots, A_r)$, то есть
$$\vec{u} = \mathbf{A}(\vec{A}).$$
2) Применим процедуру $\mathbf{B}_5$, чтобы построить все корректные входы для булевых функций базы данных, то есть $\mathbf{B}_5$:
   i) для любых $A \in GF(q)^4$ вычислим $B_A = \mathbf{B}_3(A, \vec{A}, \vec{B}, \vec{u})$;
   ii) для любых $l \in SL_4(U)$ вычислим $B_l = \mathbf{B}_4(l, \vec{A}, \vec{B}, \vec{u})$.
3) Вычислим выходы булевых функций базы данных, то есть
   i) для любых $A \in GF(q)^4$ вычислим $Y_A = f_A(B_A)$;
   ii) для любых $l \in SL_4(U)$ вычислим $Z_l = g_l(B_l)$.

   Заметим, что это единственный шаг, на котором происходит попарно независимое, в силу применения алгоритма $\mathbf{A}$, обращение к внешним носителям.
4) Применим процедуру $\mathbf{B}_7$ для вычисления выходной последовательности алгоритма, то есть $\mathbf{B}_7$:
   для любых $i = 1, 2, \ldots, r$ вычислим $f(A_i, B_i) = \mathbf{B}_6(A, u, \vec{Z}, \vec{Y})$.

## 3. Доступ к криптографически защищенной базе данных

Рассмотрим задачу криптографической защиты базы данных при соблюдении всех ограничений алгоритма $DP_{r,f}$. Как следует из

предыдущего раздела, для этого необходимо криптографически защитить таблицу 1. Выбор конкретной криптосистемы выходит за рамки данной работы, поэтому для нас достаточно только предположить, что надежность используемой криптосистемы достаточно высока. Очевидно, что в случае шифрования столбцов нашей таблицы произойдет нарушение главного ограничения алгоритма, так как перед извлечением нужной информации потребуется считать некоторые столбцы целиком для их расшифровки.

Будем зашифровывать двоичные последовательности, соответствующие строкам таблицы 1. Таким образом, исходные сообщения для криптосистемы это двоичные последовательности длины $|GF(q)^4| + |SL_4(U)|$. Каждая строка заменяется на ее криптограмму. Шифрование базы данных будем производить при ее предварительной обработке для алгоритма $DP_{r,f}$, поэтому здесь этот процесс рассматриваться не будет.

Нетрудно показать, что алгоритм $DP_{r,f}$ использует только соответствующие $r$ строк таблицы 1 для любых $r$ входов. Следовательно, для корректной работы алгоритма необходимо расшифровать эти строки. Значит, за один шаг алгоритма потребуется считать $r$ бит из каждого столбца таблицы 1, что делает невозможным одно обращение к внешним носителям. Для устранения противоречия с главным ограничением вновь воспользуемся методами алгоритма $DP_{r,f}$. Будем считать, что нам необходимо вычислить значения каждой булевой функции $f_A$ и $g_l$ от $r$ независимых наборов переменных $B_1$, $B_2$, $\ldots$,$B_r$. Тогда мы получим задачу, решаемую алгоритмом $DP_{r,f}$, и нам необходимо специальным образом представить таблицу 1.

Каждый столбец, соответствующий $f_A$ и $g_l$, разобьем аналогично разбиению функции $f$, описанному во втором разделе, учитывая, что входные функции зависят теперь от $n - 4k$ переменных и, следовательно, поменяются значения величин $n$ и $k$. Таким образом, каждый столбец таблицы 1 будет представлен несколькими булевыми функциями, каждой из которых соответствует свой внешний носитель информации и свой процессор. Точное количество таких частей будет оценено ниже.

Из всего сказанного следует, что криптографическая защита базы данных заключается в шифровании всех строк таблицы 1 и последу-

ющем разбиении каждого столбца на части.

Построим теперь алгоритм, организующий доступ к зашифрованной базе данных. Назовем этот алгоритм $CDP_{r,f}$.

**Алгоритм** $CDP_{r,f}$

1) Применим процедуру, которую обозначим через **D**, для считывания $r$ строк таблицы 1, соответствующих наборам $B_1, B_2, \ldots, B_r$ из базы данных.

   i) $\forall A \in GF(q)^4$ применим алгоритм $DP_{r,f_A}$ для вычисления каждой функции $f_A$ на входе $\vec{B} = (B_1, B_2, \ldots, B_r)$ и из результатов сформируем булеву функцию $F_A = (f_A(B_1), f_A(B_2), \ldots, f_A(B_r))$.

   ii) $\forall l \in SL_4(U)$ применим алгоритм $DP_{r,g_l}$ для вычисления каждой функции $g_l$ на входе $\vec{B} = (B_1, B_2, \ldots, B_r)$ и из результатов сформируем булеву функцию $G_l = (g_l(B_1), g_l(B_2), \ldots, g_l(B_r))$.

   Полученные результаты объединим в виде таблицы из $r$ строк, соответствующих наборам $B_1, B_2, \ldots, B_r$.

| $y_1$ | $y_2$ | $\ldots$ | $y_{n-4k}$ | $F_{A1}$ | $F_{A2}$ | $\ldots$ | $F_{Am}$ | $G_{l1}$ | $G_{l2}$ | $\ldots$ | $G_{lp}$ |
|---|---|---|---|---|---|---|---|---|---|---|---|
| набор $B_1$ | | | | $*$ | $*$ | $\ldots$ | $*$ | $*$ | $*$ | $\ldots$ | $*$ |
| набор $B_2$ | | | | $*$ | $*$ | $\ldots$ | $*$ | $*$ | $*$ | $\ldots$ | $*$ |
| $\ldots$ | | | | $\ldots$ | $\ldots$ | $\ldots$ | $\ldots$ | $\ldots$ | $\ldots$ | $\ldots$ | $\ldots$ |
| набор $B_r$ | | | | $*$ | $*$ | $\ldots$ | $*$ | $*$ | $*$ | $\ldots$ | $*$ |

Таблица 2.

   Данная таблица представляет собой $r$ зашифрованных строк таблицы 1, соответствующих наборам $B_1, B_2, \ldots, B_r$.

2) Применим процедуру, которую обозначим через **C**, для расшифровки всех строк таблицы 2. Расшифрованные строки опять разместим в этой же таблице.

3) Применим процедуру **A** к входному вектору $\vec{A}$, то есть

$$\vec{u} = \mathbf{A}(\vec{A}).$$

4) Применим процедуру $\mathbf{B}_5$ для построения правильных входов булевых функций базы данных, то есть $\mathbf{B}_5$:
   i) для любых $A \in GF(q)^4$ вычислим $B_A = \mathbf{B}_3(A, \vec{A}, \vec{B}, \vec{u})$;
   ii) для любых $l \in SL_4(U)$ вычислим $B_l = \mathbf{B}_4(l, \vec{A}, \vec{B}, \vec{u})$.

5) Вычислим все функции, необходимые для дальнейшей работы алгоритма
   i) для любых $A \in GF(q)^4$ вычислим $Y_A = F_A(B_A)$;
   ii) для любых $l \in SL_4(U)$ вычислим $Z_l = G_l(B_l)$.

   Отметим, что на этом шаге не осуществляется обращение к базе данных. Все необходимые значения выбираются в таблице 2.

6) Применим процедуру $\mathbf{B}_7$ для вычисления выходной последовательности алгоритма, то есть $\mathbf{B}_7$:
   для любых $i = 1, 2, \ldots, r$ вычислим $f(A_i, B_i) = \mathbf{B}_6(A, u, \vec{Z}, \vec{Y})$.

Оценим сложность алгоритма $CDP_{r,f}$ по трем параметрам, указанным в первом разделе. Для этого докажем следующую теорему, учитывая, что $U$ — подмножество $GF(q)^4$ размерности $r^3$ и $k = \lceil 3\log r + \log n \rceil$.

**Теорема 1.** *Для любых положительных целых $n$ и $r$, $r \leqslant 2^{\frac{n}{16}}$ и для любых булевых функций $f$ от $n$ переменных выполнено:*

$$\mathbf{ptime}(CDP_{r,f}) = \mathrm{O}(\log r + \log n) + \mathbf{extime} + \mathbf{crypttime},$$

$$\mathbf{bp}(CDP_{r,f}) = \mathrm{O}((r^{25}n^4)(n - 12\log r - 4\log n)^5 \cdot \cdot(\log r + \log(n - 12\log r - 4\log n))^2),$$

$$\mathbf{mem}(CDP_{r,f}) \leqslant 2^n \left(1 + \tfrac{1}{n}\right) \left(1 + \tfrac{1}{n - 12\log r - 4\log n}\right).$$

**Доказательство.** Докажем приведенные оценки, используя оценки для алгоритма $DP_{r,f}$ из [3]. Рассмотрим первый шаг алгоритма. Для каждого конкретного $A \in GF(q)^4$ мы имеем

$$\mathbf{bp}(DP_{r,fA}) = \mathrm{O}((r^{13}(n - 4k)^5)(\log r + \log(n - 4k))^2),$$
$$\mathbf{ptime}(DP_{r,fA}) = \mathrm{O}(\log r + \log(n - 4k)) + \mathbf{extime}.$$

Для каждого конкретного $l \in SL_4(U)$:

$$\mathbf{bp}(DP_{r,gl}) = \mathbf{bp}(DP_{r,fA}) = \mathrm{O}((r^{13}(n-4k)^5)(\log r + \log(n-4k))^2),$$
$$\mathbf{ptime}(DP_{r,gl}) = \mathbf{ptime}(DP_{r,fA}) = \mathrm{O}(\log r + \log(n-4k)) + \mathbf{extime}.$$

Учитывая, что

$$|GF(q)^4| = 2^{4k} \text{ и } |SL_4(U)| = \frac{|GF(q)^4| \cdot |U|}{|GF(q)|} = 2^{4k} \cdot \frac{|U|}{2^k} \leqslant 2^{4k} \cdot \frac{1}{n},$$

получаем

$$\mathbf{bp}(\mathbf{D}) = (|GF(q)^4| + |SL_4(U)|) \cdot \mathbf{bp}(DP_{r,fA}) =$$
$$= \mathrm{O}(2^{4k}\left(1 + \frac{1}{n}\right) r^{13}(n-4k)^5(\log r + \log(n-4k))^2),$$

где $k = \lceil 3\log r + \log n \rceil$. Таким образом,

$$\mathbf{bp}(\mathbf{D}) = \mathrm{O}(2^{4k}\left(1 + \frac{1}{n}\right) r^{13}(n-4k)^5(\log r + \log(n-4k))^2) =$$
$$= \mathrm{O}(r^{12}n^4\left(1 + \frac{1}{n}\right) r^{13}(n - 12\log r - 4\log n)^5(\log r + \log(n-4k))^2) =$$
$$= \mathrm{O}(r^{25}n^4(n - 12\log r - 4\log n)^5(\log r + \log(n - 12\log r - 4\log n))^2),$$

$$\mathbf{ptime}(\mathbf{D}) = \mathrm{O}(\log r + \log(n-4k)) + \mathbf{extime} =$$
$$= \mathrm{O}(\log r + \log(n - 12\log r - 4\log n)) + \mathbf{extime}.$$

На втором шаге мы расшифровываем строки таблицы 2. Будем считать, что для этого нам не требуется новых процессоров и эта процедура выполняется за время **crypttime**. Точное значение последней величины зависит от используемой криптосистемы и возможности ее распараллеливания. Получаем, что

$$\mathbf{bp}(\mathbf{C}) = \mathbf{bp}(\mathbf{D}), \mathbf{ptime}(\mathbf{C}) = \mathbf{crypttime}.$$

Для третьего шага алгоритма, согласно оценкам для $DP_{r,f}$, имеем

$$\mathbf{bp}(\mathbf{A}) = \mathrm{O}(r^7k^2), \quad \mathbf{ptime}(\mathbf{A}) = \mathrm{O}(\log r + \log k).$$

Для четвертого шага из [3] следует

$$\mathbf{bp}(\mathbf{B}_5) = \mathrm{O}(|GF(q)^4| \cdot (rk^2 + rn)) + \mathrm{O}(|SL_4(U)| \cdot (rk + rn)),$$
$$\mathbf{ptime}(\mathbf{B}_5) = \mathrm{O}(\log k + \log r).$$

Учитывая, что

$$|GF(q)^4| = 2^{4k} \text{ и } |SL_4(U)| \leqslant 2^{4k} \cdot \frac{1}{n},$$

получаем

$$\mathbf{bp}(\mathbf{B}_5) = \mathrm{O}(r2^{4k} \cdot (k^2 + n)) + \mathrm{O}(r2^{4k} \cdot \left(1 + \frac{k}{n}\right)).$$

На пятом шаге потребуется обратиться к каждому столбцу таблицы 2, хранящейся в оперативной памяти. Так как каждый столбец обрабатывается своим процессором, то можно оценить время как $\mathrm{O}(1)$. Количество процессоров не увеличится.

На шестом шаге алгоритма $CDP_{r,f}$ имеем

$$\mathbf{bp}(\mathbf{B}_7) = r \cdot \mathbf{bp}(\mathbf{B}_6), \quad \mathbf{ptime}(\mathbf{B}_7) = \mathbf{ptime}(\mathbf{B}_6),$$

где

$$\mathbf{bp}(\mathbf{B}_6) = \mathrm{O}(|SL_4(U)| \cdot k + |GF(q)^4| \cdot k^2) = \mathrm{O}(k2^{4k} \cdot \left(k + \frac{1}{n}\right)),$$
$$\mathbf{ptime}(\mathbf{B}_6) = \mathrm{O}(k).$$

Из всего вышеизложенного следуют оценки для алгоритма $CDP_{r,f}$.

$$\mathbf{bp}(CDP_{r,f}) = \max(\mathbf{bp}(\mathbf{D}), \mathbf{bp}(\mathbf{C}), \mathbf{bp}(\mathbf{A}), \mathbf{bp}(\mathbf{B}_5), \mathbf{bp}(\mathbf{B}_7)) =$$
$$= \mathrm{O}((r^{25}n^4)(n - 12\log r - 4\log n)^5(\log r + \log(n - 12\log r - 4\log n))^2).$$

$$\mathbf{ptime}(CDP_{r,f}) =$$
$$= \mathbf{ptime}(\mathbf{D}) + \mathbf{ptime}(\mathbf{C}) + \mathbf{ptime}(\mathbf{A}) + \mathbf{ptime}(\mathbf{B}_5) + \mathbf{ptime}(\mathbf{B}_7) =$$
$$= \mathrm{O}(\log r + \log(n - 12\log r - 4\log n)) + \mathbf{extime} + \mathbf{crypttime} +$$
$$+ \mathrm{O}(\log r + \log k) + \mathrm{O}(k) = \mathrm{O}(\log r + \log n) + \mathbf{extime} + \mathbf{crypttime}.$$

Для оценки величины $\mathbf{mem}(CDP_{r,f})$ вспомним, что каждый столбец таблицы 1 разбивается по алгоритму $DP_{r,f}$. Учитывая, что

$$m = |GF(q)^4|,\ p = |SL_4(U)| \text{ и } \mathbf{mem}(DP_{r,fA}) \leqslant 2^{n-4k} \cdot \left(1 + \frac{1}{n-4k}\right),$$

получаем

$$\mathbf{mem}(CDP_{r,f}) \leqslant (|GF(q)^4| + |SL_4(U)|) \cdot 2^{n-4k} \cdot \left(1 + \frac{1}{n-4k}\right) \leqslant$$
$$\leqslant 2^{4k}\left(1 + \frac{1}{n}\right) \cdot 2^{n-4k}\left(1 + \frac{1}{n-4k}\right) = 2^n\left(1 + \frac{1}{n}\right) \cdot \left(1 + \frac{1}{n-4k}\right) =$$
$$= 2^n\left(1 + \frac{1}{n}\right) \cdot \left(1 + \frac{1}{n - 12\log r - 4\log n}\right).$$

Что и требовалось доказать.

## 4. Сложность алгоритма $DP_{r,f}$ при ограниченном числе процессоров

Отметим, что задача доступа к базе данных [3] решалась при условии максимального распараллеливания алгоритмов, что существенно увеличило количество процессоров в оценке сложности. Найдем минимальное количество процессоров, необходимое алгоритму $DP_{r,f}$ для соблюдения главного ограничения задачи, а именно однократного обращения к базе данных, и получим новые оценки сложности.

Рассмотрим задачу параллельного извлечения информации из базы данных по $r$ независимым адресам за одно обращение к внешним носителям при минимально возможном количестве процессоров.

Алгоритм $DP_{r,f}$ разбивает базу данных на определенное количество частей, каждая из которых обслуживается своим процессором. При обращении к базе данным по любым $r$ адресам допускается только одно обращение каждого процессора к соответствующей ему части и вся информация должна быть считана за один шаг. Поэтому, для выполнения главного ограничения алгоритма необходимо минимум столько процессоров, на сколько частей разбита база данных. Заметим, что для работы алгоритма $DP_{r,f}$ требуется представить булеву

функцию $f$ через булевы функции $f_A^{(j)}$ и $g_l^{(j)}$ : $\{0,1\}^{n-4k} \to \{0,1\}$, где $A \in GF(q)^4$, $l \in SL_4(U)$. Каждая такая функция это часть разбиения базы данных, обслуживаемая отдельным процессором. Следовательно, количество частей в $DP_{r,f}$ не может превышать величину

$$|GF(q)^4|+|SL_4(U)| \leqslant (2^{4k}+2^{4k}\cdot\frac{1}{n}) = 2^{4k}\cdot\left(1+\frac{1}{n}\right) \leqslant 16r^{12}n^4\left(1+\frac{1}{n}\right).$$

Таким образом, количество процессоров алгоритма можно ограничить величиной $\mathrm{O}(r^{12}n^4)$.

Оценим время работы алгоритма при таком ограниченном количестве процессоров. Для этого нам потребуется рассмотреть все шаги алгоритма $DP_{r,f}$, описанного во втором разделе данной работы.

**Теорема 2.** *Для любых положительных целых $n$ и $r$, $r \leqslant 2^{\frac{n}{16}}$ и для любых булевых функций $f$ от $n$ переменных время работы алгоритма $DP_{r,f}$ при количестве процессоров, ограниченном величиной $\mathrm{O}(r^{12}n^4)$, можно оценить как*

$$\mathbf{ptime}(DP_{r,f}) = \\ = \mathrm{O}(r(n+\log^2 r+\log^2 n+\log r\log n)(\log r+\log n))+\mathbf{extime}.$$

**Доказательство.** Рассмотрим первый шаг алгоритма $DP_{r,f}$. Его сложность можно оценить следующим образом [3].

$$\mathbf{bp}(\mathbf{A}) = \mathrm{O}(r^7(\log r+\log n)^2), \quad \mathbf{ptime}(\mathbf{A}) = \mathrm{O}(\log r+\log(\log r+\log n)).$$

Количество процессоров, необходимое на данном шаге, удовлетворяет нашему ограничению и, следовательно, оценка времени не изменится.

Для второго шага алгоритма $DP_{r,f}$ ранее мы имели оценки

$$\mathbf{bp}(\mathbf{B}_5) \leqslant \mathrm{O}(r^{13}n^5)+\mathrm{O}(r^{13}n^4(\log r+\log n)^2), \\ \mathbf{ptime}(\mathbf{B}_5) = \mathrm{O}(\log r+\log(\log r+\log n)).$$

Указанное количество процессоров превышает ограничение теоремы, а значит необходимо заново оценить время этой процедуры при ограниченном числе процессоров.

Рассмотрим этот шаг подробнее. Процедура $\mathbf{B}_5$ состоит из двух подпунктов:

i) для любых $A \in GF(q)^4$ вычислим $B_A = \mathbf{B}_3(A, \vec{A}, \vec{B}, \vec{u})$;
ii) для любых $l \in SL_4(U)$ вычислим $B_l = \mathbf{B}_4(l, \vec{A}, \vec{B}, \vec{u})$.

Для первого подпункта необходимо $|GF(q)^4| \cdot \mathbf{bp}(\mathbf{B}_3)$ процессоров, которые выполнят его за время $\mathbf{ptime}(\mathbf{B}_3)$, а для второго подпункта требуется $|SL_4(U)| \cdot \mathbf{bp}(\mathbf{B}_4)$ процессоров и время выполнения равно $\mathbf{ptime}(\mathbf{B}_4)$. Как было отмечено выше, количество процессоров у нас ограничено величиной $|GF(q)^4| + |SL_4(U)| \leqslant \mathrm{O}(r^{12}n^4)$. Выделим из этой величины $|GF(q)^4|$ процессоров на первый подпункт и $|SL_4(U)|$ на второй. Так как $|GF(q)^4| \cdot \mathbf{bp}(\mathbf{B}_3)$ процессоров выполняет первый подпункт за время $\mathbf{ptime}(\mathbf{B}_3)$, то $|GF(q)^4|$ процессоров выполнит эту же процедуру за время $\mathbf{ptime}(\mathbf{B}_3) \cdot \mathbf{bp}(\mathbf{B}_3)$. Аналогично, для второго подпункта получаем время $\mathbf{ptime}(\mathbf{B}_4) \cdot \mathbf{bp}(\mathbf{B}_4)$. Воспользуемся для процедур $\mathbf{B}_3$ и $\mathbf{B}_4$ оценками из [3].

$$\mathbf{bp}(\mathbf{B}_3) = \mathrm{O}(r \cdot (k^2 + n)), \quad \mathbf{ptime}(\mathbf{B}_3) = \mathrm{O}(\log k + \log r),$$
$$\mathbf{bp}(\mathbf{B}_4) = \mathrm{O}(r \cdot (k + n)), \quad \mathbf{ptime}(\mathbf{B}_4) = \mathrm{O}(\log k + \log r).$$

Таким образом, при ограниченном количестве процессоров

$$\mathbf{ptime}(\mathbf{B}_5) = \mathrm{O}(r\cdot(k^2+n)\cdot(\log k+\log r))+\mathrm{O}(r\cdot(k+n)\cdot(\log k+\log r)) =$$
$$= \mathrm{O}(r \cdot (n + (\log r + \log n)^2) \cdot (\log r + \log(\log r + \log n))).$$

На третьем шаге алгоритма происходит обращение к базе данных. Наше ограничение на число процессоров устанавливалось исходя из этого шага, поэтому новых процессоров не требуется. Время выполнения оценивается как $\mathbf{extime}$.

Рассмотрим четвертый шаг (процедура $\mathbf{B}_7$).

Для всех $i = 1, 2, \ldots, r$ вычисляется

$$f(A_i, B_i) = \mathbf{B}_6(A_i, u_i, \vec{Z}, \vec{Y}) =$$
$$= \left( \bigvee_{l \in SL_4(U)} Z_l \cdot \mathbf{B}_2(A, u, l) \right) \oplus \left( \bigoplus_{A^* \in GF(q)^4} Y_{A^*} \cdot \mathbf{B}_1(A, u, A^*) \right).$$

Оценки сложности этой процедуры до ограничения количества процессоров были таковы

$$\mathbf{bp}(\mathbf{B}_7) = \mathrm{O}(s \cdot r^{13} n^4 (\log r + \log n)^2), \quad \mathbf{ptime}(\mathbf{B}_7) = \mathrm{O}(\log r + \log n).$$

Указанное количество процессоров превышает ограничение теоремы, поэтому необходимо заново определить величину $\mathbf{ptime}(\mathbf{B}_7)$.

Для вычисления каждого произведения $Z_l \cdot \mathbf{B}_2(A, u, l)$ в алгоритме $DP_{r,f}$ использовалось $\mathbf{bp}(\mathbf{B}_2)$ процессоров и происходило это за время $\mathbf{ptime}(\mathbf{B}_2)$, а для любого произведения $Y_{A^*} \cdot \mathbf{B}_1(A, u, A^*)$ использовалось $\mathbf{bp}(\mathbf{B}_1)$ процессоров и затрачивалось время $\mathbf{ptime}(\mathbf{B}_1)$. Произведений первого типа должно быть вычислено $|SL_4(U)|$, а произведений второго типа $|GF(q)^4|$. Однако, по ограничению теоремы у нас всего $|GF(q)^4| + |SL_4(U)|$ процессоров, то есть по одному процессору на произведение каждого типа. На вычисление каждого произведения $Z_l \cdot \mathbf{B}_2(A, u, l)$ одним процессором будет затрачено время $\mathbf{bp}(\mathbf{B}_2) \cdot \mathbf{ptime}(\mathbf{B}_2)$, а на каждое $Y_{A^*} \cdot \mathbf{B}_1(A, u, A^*)$ время $\mathbf{bp}(\mathbf{B}_1) \cdot \mathbf{ptime}(\mathbf{B}_1)$. Таким образом, все указанные произведения будут вычислены за время $\max(\mathbf{bp}(\mathbf{B}_2) \cdot \mathbf{ptime}(\mathbf{B}_2), \mathbf{bp}(\mathbf{B}_1) \cdot \mathbf{ptime}(\mathbf{B}_1))$. Далее легко доказать, что время вычисления всех дизъюнкций оценивается как $\mathrm{O}(\log |SL_4(U)|)$, а на вычисление всех сумм по модулю 2 необходимо времени $\mathrm{O}(\log |GF(q)^4|)$. Следовательно, время выполнения всей процедуры $\mathbf{B}_6$ можно оценить как

$$\mathbf{ptime}(\mathbf{B}_6) = \max(\mathbf{bp}(\mathbf{B}_2) \cdot \mathbf{ptime}(\mathbf{B}_2), \mathbf{bp}(\mathbf{B}_1) \cdot \mathbf{ptime}(\mathbf{B}_1)) + \\ + \max(\mathrm{O}(\log |SL_4(U)|), \mathrm{O}(\log |GF(q)^4|)).$$

Из [3] мы имеем следующие оценки

$$\mathbf{bp}(\mathbf{B}_1) = \mathrm{O}((\log r + \log n)^2), \quad \mathbf{ptime}(\mathbf{B}_1) = \mathrm{O}(\log(\log r + \log n)),$$
$$\mathbf{bp}(\mathbf{B}_2) = \mathrm{O}(\log r + \log n), \quad \mathbf{ptime}(\mathbf{B}_2) = \mathrm{O}(\log(\log r + \log n)),$$
$$\log |SL_4(U)| = \mathrm{O}(\log r + \log n), \log |GF(q)^4| = \mathrm{O}(\log r + \log n).$$

Отсюда получаем, что

$$\mathbf{ptime}(\mathbf{B}_6) = \mathrm{O}((\log r + \log n)^2 \cdot \log(\log r + \log n)).$$

Окончательно, для четвертого шага алгоритма определяем

$$\mathbf{ptime}(\mathbf{B}_7) = \mathrm{O}(r \cdot (\log r + \log n)^2 \cdot \log(\log r + \log n)).$$

Время работы всего алгоритма при ограниченном количестве процессоров определяется как

$$\mathbf{ptime}(DP_{r,f}) = \mathbf{ptime}(\mathbf{A}) + \mathbf{ptime}(\mathbf{B}_5) + \mathbf{extime} + \mathbf{ptime}(\mathbf{B}_7) =$$
$$= \mathrm{O}(\log r + \log(\log r + \log n)) +$$
$$+ \mathrm{O}(r \cdot (n + (\log r + \log n)^2) \cdot (\log r + \log(\log r + \log n))) +$$
$$+ \mathbf{extime} + \mathrm{O}(r \cdot (\log r + \log n)^2 \cdot \log(\log r + \log n)) =$$
$$= \mathrm{O}(r \cdot (n + (\log r + \log n)^2) \cdot (\log r + \log(\log r + \log n))) + \mathbf{extime}.$$

Что и требовалось доказать.

Очевидно, что размер базы данных не увеличится при ограничении количества процессоров. То есть

$$\mathbf{mem}(DP_{r,f}) \leqslant 2^n \left(1 + \tfrac{1}{n}\right).$$

Следовательно, окончательно получаем следующие новые оценки алгоритма $DP_{r,f}$ при ограниченном количестве процессоров:

$$\mathbf{ptime}(DP_{r,f}) = \mathrm{O}(r(n + \log^2 r + \log^2 n + \log r \log n)(\log r + \log n)),$$
$$\mathbf{bp}(DP_{r,f}) = \mathrm{O}(r^{12} n^4),$$
$$\mathbf{mem}(DP_{r,f}) \leqslant 2^n \left(1 + \tfrac{1}{n}\right).$$

Таким образом, общее количество процессоров в оценках уменьшается в $\mathrm{O}(rn(\log r + \log n)^2)$ раз по сравнению с алгоритмом $DP_{r,f}$, а время работы процессоров увеличивается в $\mathrm{O}(r(n + \log^2 r + \log^2 n + \log r \log n))$ раз. Однако, такое увеличение времени работы может оказаться не существенным по отношению ко времени считывания информации с внешних носителей.

## Список литературы

[1] Paul W.J. Realizing Boolean functions on disjoint set of variables // Theoret. Comput. Sci. 2. 1976. P. 383–396.

[2] Улиг Д. О синтезе самокорректирующихся схем из функциональных элементов с небольшим числом зависимых элементов // ДАН СССР. 15. 1974. С. 558–562.

[3] Andreev A.E., Clementi A.E.F., Rolim J.D.P. On the parallel computation of Boolean functions on unrelated inputs // Proc. of 4-th Israeli Symposium on Theory of Computing and Systems (ISTCS'96). 1996.