Серия «Математика»
2010. Т. 3, № 4. С. 33—43

Онлайн-доступ к журналу:
http://isu.ru/izvestia

ИЗВЕСТИЯ
Иркутского
государственного
университета

УДК 519.7

Вычислительная оценка сложности полиномиальных представлений булевых функций *

А. С. Казимиров, С. Ю. Реймеров
Восточно-Сибирская государственная академия образования

Аннотация. В статье рассматривается сложность полиномиальных представлений булевых функций. Предложены условия, которым должна удовлетворять функция, имеющая сложность больше заданной. С помощью генетического алгоритма минимизации получены верхние оценки сложности таких булевых функций, и как следствие получена новая верхняя оценка сложности для всех булевых функций.

Ключевые слова: булевы функции; полиномиальные представления; минимизация; генетические алгоритмы.

Любую булеву функцию можно представить множеством различных термов и наибольший интерес представляют термы с определенными свойствами. В данной работе рассматривается минимизация полиномиальных представлений булевых функций.

Под полиномиальной формой или полиномом P_f, представляющей функцию $f(x_1, \ldots, x_n)$, будем понимать формулу вида $P_f = P_1 \oplus \ldots \oplus P_k$, где $P_i = x_{i_1}^{\alpha_{i_1}} \cdot \ldots \cdot x_{i_t}^{\alpha_{i_t}}$, $\alpha_{i_j} \in \{0, 1\}$, $x^0 = \overline{x}, x^1 = x$, также P_i может быть равно 1.

Количество слагаемых k будем далее называть сложностью полинома и обозначать $|P_f|$.

Через $L(f)$ обозначим сложность полинома, представляющего функцию f и имеющего минимальное число слагаемых: $L(f) = \min\limits_{P_f} |P_f|$. Далее $L(f)$ будем называть сложностью функции.

Под сложностью $L(n)$ класса всех n-местных функций будем понимать сложность самой сложной функции: $L(n) = \max\limits_{f \in F^n} L(f)$.

В статье [1] получена оценка $L(7) \leq 28$. В данной работе получены результаты, позволяющие понизить эту оценку.

* Работа выполнена при финансовой поддержке РФФИ, грант 09–01–00476а.

LP-эквивалентность. *Остаточной* функцией f по i-му аргументу будем называть функцию $f^{\alpha}_{x_i}(x_1,\ldots,x_n) = f(x_1,\ldots,x_{i-1},\alpha,x_i,\ldots,x_n)$. В зависимости от значения α будем говорить о нулевой и единичной остаточных.

Функцию $f'_{x_i}(x_1,\ldots,x_n) = f^0_{x_i}(x_1,\ldots,x_n) \oplus f^1_{x_i}(x_1,\ldots,x_n)$ будем называть *производной* функции f по i-му аргументу.

Рассмотрим функцию $f(x_1,\ldots,x_n)$. Функция $g(x_1,\ldots,x_n)$ называется *LP-эквивалентной* к $f(x_1,\ldots,x_t)$, если можно построить последовательность функций $f_0,\ldots,f_n$, в которой $f_0 = f$, $f_t = g$ и f_{k+1} получается из f_k одним из следующих преобразований:

$$\left[\begin{array}{l} f_{k+1}(x_1,\ldots,x_i,\ldots,x_j,\ldots,x_n) = f_k(x_1,\ldots,x_j,\ldots,x_i,\ldots,x_n);\\ f_{k+1}(x_1,\ldots,x_n) = f_k(x_1,\ldots,\overline{x}_i,\ldots,x_n);\\ f_{k+1}(x_1,\ldots,x_n) = x_i(f_k)^0_{x_i}(x_1,\ldots,x_n) \oplus \overline{x}_i \cdot (f_k)'_{x_i}(x_1,\ldots,x_n);\\ f_{k+1}(x_1,\ldots,x_n) = x_i(f_k)^1_{x_i}(x_1,\ldots,x_n) \oplus \overline{x}_i \cdot (f_k)'_{x_i}(x_1,\ldots,x_n). \end{array}\right.$$

Все булевы функции разбиваются на LP-классы. Под представителем LP-класса будем понимать наименьшую функцию (в двоичной записи ее вектора) из класса.

В [2] показано, что если функция g эквивалентна f, то $L(f) = L(g)$. Там же можно найти подробное описание классов LP-эквивалентности.

Построение сложных функций. Из разложения Шеннона

$$f(x_0,\ldots,x_n) = \overline{x}_0 \cdot f(0,x_1,\ldots,x_n) \oplus x_0 \cdot f(1,x_1,\ldots,x_n)$$

получается верхняя оценка:

$$L(f) \leqslant L(f^0_{x_i}) + L(f^1_{x_i}).$$

Ее можно обобщить следующим образом:

$$L(n+1) \leqslant L(n) + L(n).$$

Таким образом, функция не может иметь сложность α, если

$$L(f^0_{x_i}) + L(f^1_{x_i}) < \alpha.$$

Построим множество $(n+1)$-местных функций F_α следующим образом: возьмем всевозможные пары n-местных функций f_1 и f_2 таких, что $L(f_1) + L(f_2) \geq \alpha$, и для каждой пары сконструируем функцию $f : f^0_{x_1} = f_1$ и $f^1_{x_1} = f_2$.

В дальнейшем будем использовать выражение *функция f пары* (α,β), которое означает, что функция $f = x_0 f_1 \oplus \overline{x}_0 f_2$, где $L(f_1) = \alpha, L(f_2) = \beta$.

Из верхней оценки разложения функции на две остаточных следует, что если существуют $(n+1)$-местные функции сложности α, то все они будут содержаться в F_α. Функции из F_α будем в дальнейшем называть функциями-претендентами на сложность α.

Для получения функций-претендентов на сложность 28 надо перебрать все пары таких функций f_1 и f_2, сложности которых представлены одним из столбцов таблицы I.

Каждая функция из пар $(15, 14)$ и $(15, 13)$ будет иметь LP-эквивалентную функцию в парах $(14, 15)$ и $(13, 15)$, соответственно. Таким образом, достаточно перебрать функции-претенденты из пар $(13, 15)$, $(14, 14)$, $(14, 15)$ и $(15, 15)$.

Таблица I. Сложности функций для построения F_{28}

$L(f_1)$	*13*	**14**	**14**	15	15	**15**
$L(f_2)$	*15*	**14**	**15**	13	14	**15**

Рассмотрим произвольную функцию $f = x_0 f_1 \oplus \overline{x}_0 f_2$ пары $(13, 15)$, где $L(f_1) = 13$ и $L(f_2) = 15$. Функция f является функцией-претендентом. Пусть $f_3 = f_1 \oplus f_2$. Если $L(f_3) = 15$, то можно построить эквивалентную к f функцию-претендент $x_0 f_3 \oplus \overline{x}_0 f_2$, которая будет в паре $(15, 15)$. Если $L(f_3) = 14$, то можно построить эквивалентную к f функцию-претендент $x_0 f_3 \oplus \overline{x}_0 f_2$, которая будет в паре $(14, 15)$. Если $L(f_3) = 13$, то можно построить эквивалентную к f функцию-претендент $x_0 f_1 \oplus \overline{x}_0 f_3$, которая будет иметь сложность, меньшую 27, а значит, ее не надо минимизировать. Таким образом, все функции-претенденты, получаемые в переборе пары $(13, 15)$, имеют либо эквивалентные в других парах, либо вообще не могут дать искомую сложность 28.

Таким образом, для поиска функций сложности 28 достаточно перебрать пары $(14, 14)$, $(14, 15)$ и $(15, 15)$.

Для получения функций-претендентов на сложность 27 надо перебрать пары таких функций f_1 и f_2, сложности которых представлены одним из столбцов таблицы II.

Таблица II. Сложности функций для построения F_{27}

$L(f_1)$	*12*	*13*	*13*	14	**14**	**14**	15	15	15	**15**
$L(f_2)$	*15*	*14*	*15*	13	**14**	**15**	12	13	14	**15**

Пары $(14, 13)$, $(15, 12)$, $(15, 13)$ и $(15, 14)$ имеют симметричные пары, поэтому их перебор не даст новых функций с точностью до LP-эквивалентности.

Рассмотрим пару $(12, 15)$. Если $f_3 = f_1 \oplus f_2$ имеет сложность меньше 14, то для всех таких функций можно построить LP-эквивалентные из f_3 и f_1, для которых оценка сложности меньше 27. Если $L(f_3) = 15$, то для таких функций можно построить LP-эквивалентные функции из f_3 и f_2, которые принадлежат паре $(15, 15)$.

Рассмотрим пару $(13, 14)$. Если $f_3 = f_1 \oplus f_2$ имеет сложность меньше 14, то для всех таких функций можно построить LP-эквивалентные из

f_3 и f_1, для которых оценка сложности меньше 27. Если $L(f_3) = 15$, то для таких функций можно построить LP-эквивалентные функции из f_3 и f_2, которые принадлежат паре $(15, 14)$.

Рассмотрим пару $(13, 15)$. Если f_3 имеет сложность меньше 14, то для всех таких функций можно построить LP-эквивалентные из f_3 и f_1, для которых оценка сложности меньше 27. Если $L(f_3) = 15$, то для таких функций можно построить LP-эквивалентные функции из f_3 и f_2, которые принадлежат паре $(15, 15)$.

Таким образом, для поиска функций сложности 27 достаточно перебрать пары $(14, 14)$, $(14, 15)$ и $(15, 15)$, то есть те же пары, что и для поиска функций сложности 28.

6-местные функции сложности 14. Для поиска 7-местных функций сложности 27 и 28 необходимо получить все 6-местные функции сложности 14 и 15. В работе [1] были найдены все функции сложности 15. Таким образом, для дальнейшего решения задачи необходимо найти функции сложности 14.

Для поиска функций сложности 14 необходимо перебирать пары таких 5-местных функций f_1 и f_2, сложности которых представлены одним из столбцов таблицы III.

Таблица III. Сложности функций для построения F_{14}

$L(f_1)$	*5*	*6*	*6*	**7**	**7**	**7**	8	8	**8**	**8**	9	9	9	9	**9**
$L(f_2)$	*9*	*8*	*9*	**7**	**8**	**9**	6	7	**8**	**9**	5	6	7	8	**9**

Повторяем рассуждения, аналогичные построению сложных функций сложности 27 и 28.

Исключаем симметричные пары. После чего остается рассмотреть пары $(5, 9)$, $(6, 8)$, $(6, 9)$, $(7, 7)$, $(7, 8)$, $(7, 9)$, $(8, 8)$, $(8, 9)$ и $(9, 9)$.

Рассмотрим пару $(5, 9)$: функции f_1, f_2 и $f_3 = f_1 \oplus f_2$, где $L(f_1) = 5$, $L(f_2) = 9$. Поскольку $L(5) = 9$, то $L(f_3) \leq 9$. Допустим, что $L(f_3) \leq 8$, тогда выполнится $L(x_0 f_1 \oplus \overline{x}_0 f_3) \leq 13$. Если же $L(f_3) = 9$, то функция $x_0 f_1 \oplus \overline{x}_0 f_3$ будет просмотрена в паре $(9, 9)$, так как $L(f_3) = 9$ и $L(f_2) = 9$. Таким образом, можно исключить пару $(5, 9)$.

Аналогичным образом классы пар $(6, 8)$ и $(6, 9)$ сводятся к парам, в построении которых не участвуют функции сложности 6. В итоге, чтобы получить все классы функций сложности 14, достаточно перебрать пары $(7, 7)$, $(7, 8)$, $(7, 9)$, $(8, 8)$, $(8, 9)$ и $(9, 9)$.

Для дальнейшей оценки перебора этих функций воспользуемся таблицей IV, в которой отражено количество функций той или иной сложности.

В таблице V отражен количественный результат перебора. Полный перебор всех функций f_1 и f_2 из пар функций-претендентов на сложность 14 будет больше, чем в приведенной таблице. Например, количество функций-претендентов в паре $(7, 7)$ равно 1 688 168 546 855 523 216.

Перебор можно уменьшить, если воспользоваться свойствами LP-эквивалентности.

Таблица IV. Распределение 5-местных функций по сложностям

Сложность	Число классов	Число функций
0	1	1
1	1	243
2	4	24 948
3	19	1 351 836
4	137	39 365 190
5	971	545 193 342
6	3572	2 398 267 764
7	2143	1 299 295 404
8	86	11 460 744
9	2	7 824
Всего	6936	4 294 967 296

Допустим, необходимо найти функцию f. Рассмотрим все функции $h = x_1h_1 \oplus \overline{x}_1h_2$, где h_1 — представитель LP-класса. Построим все функции $g = x_1g_1 \oplus \overline{x}_1g_2$, LP-эквивалентные h. Среди построенных функций будет получена и функция f.

Тогда для каждой функции f найдется некоторая LP-эквивалентная функция h с h_1 — представителем класса.

Таблица V. Количество функций-претендентов на сложность 14 с использованием LP-классов

Пара	Функции-претенденты
$(7,7)$	2 784 390 050 772
$(7,8)$	24 560 374 392
$(7,9)$	16 766 832
$(8,8)$	985 623 984
$(8,9)$	672 864
$(9,9)$	15 648
Всего	2 809 953 504 492

Основное тождество минимизации. Очевидно, что любая функция $f(x_0,\ldots,x_n)$ может быть представлена в виде:

$$f(x_0,\ldots,x_n) = x_0 \cdot f_1(x_1,\ldots,x_n) \oplus \overline{x}_0 \cdot f_2(x_1,\ldots,x_n) \oplus f_3(x_1,\ldots,x_n).$$

Здесь f_1, f_2 и f_3 — функции меньшего числа переменных. Подставив вместо x_0 поочередно 0 и 1, получаем следующие равенства:

$$\begin{cases} f(0,x_1,\ldots,x_n) = f_2(x_1,\ldots,x_n) \oplus f_3(x_1,\ldots,x_n); \\ f(1,x_1,\ldots,x_n) = f_1(x_1,\ldots,x_n) \oplus f_3(x_1,\ldots,x_n). \end{cases}$$

Преобразуем эти равенства таким образом, чтобы функции f_1 и f_2 выражались через известные остаточные функции исходной f и единственную неизвестную функцию f_3:

$$\begin{cases} f_2(x_1,\ldots,x_n) = f(0,x_1,\ldots,x_n) \oplus f_3(x_1,\ldots,x_n); \\ f_1(x_1,\ldots,x_n) = f(1,x_1,\ldots,x_n) \oplus f_3(x_1,\ldots,x_n). \end{cases}$$

Из такого представления следует, что

$$L(f) = \min_{f_3 \in F^n} (L(f^0_{x_0} \oplus f_3) + L(f^1_{x_0} \oplus f_3) + L(f_3)),$$

то есть для минимизации $(n+1)$-местной функции f достаточно перебрать все n-местные функции f_3. Однако, реализация этого метода минимизации $(n+1)$-местной функции связана с необходимостью предварительного вычисления и хранения библиотеки сложностей всех n-местных функций. Уже при $n = 5$ размер библиотеки составляет 2^{32} байт. Для работы с 7-местными функциями требуется библиотека размером 2^{64} байт. На сегодняшний день проблематично как получение такой библиотеки, так и ее хранение. Возможности сегодняшней вычислительной техники позволяют получать точные значения $L(f)$ для произвольных функций с местностью не более 6 аргументов, но возможно понизить оценку сложности генетическими алгоритмами.

Алгоритмы минимизации. Следующие алгоритмы использовались при минимизации функций-претендентов.

Алгоритм 1. Минимизация функции полным перебором.

Пусть f — $(n+1)$-местная функция, для которой необходимо найти сложность; `lib` — массив сложностей n-местных функций (индексом каждой сложности является двоичное представление вектора функции); `f1` — $f^0_{x_1}$; `f2` — $f^1_{x_1}$; `hasNext(f3)` — метод, который сообщает остались ли еще функции после `f3`.

```
int L = -1;
for(u_int64 f3 = 0; hasNext(f3); f3++) {
  int nL = lib[f3] + lib[f3 ^ f1] + lib[f3 ^ f2];
  if (nL < L || L == -1)
    L = nL;
}
```

Таким образом происходит перебор всех функций `f3` и в `L` хранится $L(f)$.

Алгоритм 2. Минимизация функций с дополнительной библиотекой.

Для всех 6-местных функций, сложности которых меньше 14, выполняется $\min(L(f_1), L(f_2)) \leq 4$, а функции сложности 15 принадлежат

одному LP-классу. На основе этого в [8] был предложен оптимизированный алгоритм.

Пусть `fs` — двумерный массив, в котором есть все функции сложностей до 5 (первый индекс — сложность, второй — номер функции); `nllib` — библиотека, в которой хранятся сложности всех 5-местных функций; `f1` — $f^0_{x_1}$; `f2` — $f^1_{x_1}$.

```
UInt8 ts = 255, ts1, ts2, ts3;
int fsi;
for(int i = 0; i < 5; i++) {
    for(int j = 0; j < fs[i].size(); j++) {
        fsi = fs[i][j];
        ts1 = nllib[fsi ^ f1];
        ts2 = nllib[fsi ^ f2];
        if (ts1 + ts2 + i < ts)
            ts = ts1 + ts2 + i;
        ts3 = nllib[fsi ^ f1 ^ f2];
        if (ts1 + ts3 + i < ts)
            ts = ts1 + ts3 + i;
        if (ts2 + ts3 + i < ts)
            ts = ts2 + ts3 + i;
    }
}
```

Хранение библиотеки 5-местных функций. Для минимизации 6-местной функции необходимо иметь библиотеку сложностей 5-местных функций с произвольным доступом к ней, то есть хранить ее в оперативной памяти.

Очевидно, что для хранения сложности одной функции достаточно 1 байта. Поскольку количество 5-местных функций составляет $2^{2^5} = 4\ 294\ 967\ 296$, то для хранения сложностей всех 5-местных функций достаточно 4 GiB памяти.

Поскольку $L(5) \leq 9$, то можно закодировать данные так, чтобы в 1 байте хранились значения сложностей сразу для двух функций. Таким образом, объем всей библиотеки сокращается до 2 GiB.

Можно хранить в памяти только представителей LP-классов функций и вычислять сложности функции “на лету”, определяя класс и сложность класса, к которому она принадлежит. Такое решение требует меньше памяти, но объем вычислений значительно увеличивается. Поэтому необходимо найти некоторый компромисс между используемой памятью и объемом промежуточных вычислений.

Обозначим через Q множество всех n-местных функций. А через $Q^{'}$ — множество n-местных функций, у которых первое значение вектора равно 0, т.е. $Q^{'} = \{f \mid f \in L,\ f(0,\ldots,0) = 0\}$.

Предложение. *Любая функция из множества Q имеет LP-эквивалентную функцию в множестве $Q^{'}$.*

Доказательство. Достаточно показать, что для любой функции $f \in Q$ существует LP-эквивалентная функция из класса $Q^{'}$.

Рассмотрим функцию $f \in L \setminus Q^{'}$. Если $f(1,0,\ldots,0) = 0$, то построим LP-эквивалентную функцию $g(x_1,\ldots,x_n) = f(\overline{x}_1,x_2,\ldots,x_n)$. Очевидно, что $g \in L^{'}$.

Если $f(1,0,\ldots,0) = 1$, то построим LP-эквивалентную функцию $g(x_1,\ldots,x_n) = f(x_1,\ldots,x_n) \oplus \overline{x}_1 \cdot f(1,x_2,\ldots,x_n)$. Очевидно, что $g \in L^{'}$.

Таким образом, получается, что для любой функции $f \in Q \setminus Q^{'}$ можно построить LP-эквивалентную функцию $g \in Q^{'}$.

Следовательно достаточно хранить только функции, у которых первое значение вектора равно 0. А для всех остальных существуют LP-эквивалентные. Таким образом удалось сократить размер библиотеки до 1 GiB. □

Минимизация 7-местных функций. Время выполнения алгоритма полного перебора при минимизации 7-местных функций недостижимо. Тривиальных методов значительного сокращения полного перебора не было выявлено, поэтому было решено использовать некие оценочные алгоритмы. Такими алгоритмами могут быть случайная выборка или генетический алгоритм минимизации.

Случайная выборка — это алгоритм, в котором в качестве функции f_3 берется некоторое случайное множество 6-местных функций и среди них выбирается одна такая, которая дает наименьшую оценку сложности исходной 7-местной функции. Каждая из 6-местных функций также минимизируется алгоритмом случайной выборки, а функции сложности 5 уже существуют в загруженной библиотеке сложностей.

Алгоритм случайной выборки оказался малоэффективным, так как среднее число функций f_3, которые позволяют минимизировать функцию стремится к одной, поэтому количество используемых операций стремится к числу операций в полном переборе.

По той причине, что случайная выборка показала себя как неэффективный метод минимизации, использовался генетический алгоритм. Данный алгоритм на 6-местных функциях статистически минимизировал функции как алгоритм полного перебора, но его время выполнения (для 6-местных функций) не превышало 5 секунд.

В итоге для минимизации использовался алгоритм генетической минимизации в классическом своем представлении: со скрещиванием, минимизацией и отбором по функциям с наименьшей сложностью. Подробнее этот алгоритм описан в работе [3].

Минимизация функций-претендентов. Генерация функций-претендентов и минимизация производились на кластере, созданном из 14 узлов учебной аудитории.

Генерация и последующая минимизация были осуществлены двумя MPI-приложениями.

Первое приложение генерировало функции-претенденты: представителю каждого LP-класса необходимой сложности (функция f_1) подставлялась в пару каждая функция f_2 необходимой сложности. Перебор функций-претендентов одного LP-класса независим от перебора функций-претендентов другого класса, поэтому возможно параллельное рассмотрение различных классов. Часть функций в процессе перебора удалось исключить за счет оценки сложности функции разложением ее на две остаточных. Полученные функции-претенденты нуждались в последующей минимизации. Минимизация каждой отдельной функции — независимая задача, поэтому распараллеливание происходило на уровне отдельных функций.

Этими двумя программами были найдены 6–местные функции сложности 14. Результат данной работы представлен в таблице VI.

Таблица VI. Результаты минимизации функций-претендентов на сложность 14

Группа	Число функций
(7,7) (7,8) (7,9)	0
(8,8)	1586
(8,9)	332
(9,9)	992
Всего	2414

Таким образом, было найдено 2414 функций, которые дают сложность 14. Эти функции были найдены с точностью до LP-классов, то есть все LP-классы 6-местных функций, дающих сложность 14, присутствуют в этих 2414. Также присутствовали функции, которые принадлежали уже представленным LP-классам.

На следующем этапе было написано MPI-приложение, которое для каждой функции находило все функции, принадлежащие ее LP-классу. После работы этого приложения были найдены все представители каждого из этих LP-классов.

Таблица VII. 6-местные функции сложности 14 и 15

Сложность	Число классов	Число функций
14	62	65 393 568
15	1	96
Всего	63	65 393 664

Результаты поиска LP-классов 6-местных функций и дальнейшего вычисления всех представителей этих классов можно представить таблицей VII.

Для дальнейшего поиска 7-местных сложных функций необходимо также перебрать пары функций для генерации функций-претендентов.

Объем перебора этих пар можно представить таблицей VIII.

Таблица VIII. Количество функций-претендентов на сложность 27 и выше

Пара	Функции-претенденты
$(14, 14)$	4 054 401 216
$(14, 15)$	5 952
$(15, 15)$	96
Всего	4 054 407 264

Аналогично поиску функций на сложность 14 были написаны приложения для поиска функций на сложность 27 и выше. Для минимизации использовался генетический алгоритм.

Минимизация функций-претендентов на сложность 27 производилась в несколько однотипных этапов. На первом этапе параметры генетического алгоритма были подобраны таким образом, чтобы время минимизации всех функций-претендентов не превышало двух-трех дней. Далее на кластере все функции-претенденты были минимизированы и получен файл функций с оценкой сложности больше или равной 27.

На втором этапе генетический алгоритм был настроен так, что его точность была повышена, а время минимизации всех оставшихся функций не превышало двух-трех дней, был получен файл с функциями, имеющими верхнюю оценку сложности больше или равную 27.

Оставшиеся функции были минимизированы генетическим алгоритмом минимизации с большим перебором чем на втором этапе, после чего всех функций были найдены представления сложности меньше 27.

Таким образом, удалось показать, что не существует 7-местных функций сложности 27 и 28:

Теорема. $L(7) \leq 26$.

Действие теоремы распространяется на большее число переменных:

Следствие. $L(n) \leqslant \frac{26}{128} \cdot 2^n$.

Данная оценка лучше асимптотической оценки

$$L(n) \leq \frac{2^n(2\log_2 n + 2)}{n},$$

полученной в [4], при $n \leq 70$.

Список литературы

1. Винокуров С. Ф. Верхняя оценка сложности булевых функций в классе ПНФ / С. Ф. Винокуров, А. С. Казимиров // Algebra and Model Theory 4. – Novosibirsk: Novosibirsk State Tech. University, 2003. – P. 160–165.
2. Казимиров А. С. Оценка числа классов LP-эквивалентности булевых функций / А. С. Казимиров // Вестн. Бурят. ун-та. Сер. 13, Математика и информатика. – 2005. – Вып. 2. – С. 17–22.

3. Казимиров А. С. Параллельные генетические алгоритмы в задачах минимизации булевых функций / А. С. Казимиров // Вестн. ТГУ. Приложение. – 2006. – № 17. – С. 226–230.
4. Кириченко К. Д. Верхняя оценка сложности полиномиальных нормальных форм булевых функций / К. Д. Кириченко // Дискретная математика. – 2005. – Т. 17, № 3. – С. 80–88.
5. Реймеров С. Ю. О сложности полиномиальных представлений булевых функций 7 переменных / С. Ю. Реймеров // Студент и научно-технический прогресс: Математика : материалы XLVIII междунар. науч. студ. конф. – Новосибирск : Новосиб. гос. ун-т, 2010. – С. 177.
6. Рябец Л. В. Нахождение минимальных полиномов булевых функций с использованием специальной операторной формы / Л. В. Рябец // Технологии Microsoft в теории и практике программирования : тез. докл. – Новосибирск, 2006. – С. 215–217.
7. Even S. On minimal modulo 2 sums of products for switching functions / Even S // IEEE Trans. Electron. Comput. – 1967. – Vol. EC-16, N 10. – P. 671–674.
8. Gaidukov A. Algorithm to derive minimum ESOPs for 6-variable functions / A. Gaidukov // Proceedings of the 5th International Workshop on Boolean Problems 2002. Freiberg, Germany, Sept. 19-20. – 2002. – P. 141–148.

A. S. Kazimirov, S. U. Reymerov

Computational bound on complexity of polynomial representations of boolean functions

Abstract. This paper concerns complexity of exclusive-or-sum-of-products expressions representing Boolean functions. Conditions for functions having complexity over a given number are introduced. Genetic minimization algorithm gave obtained upper bounds on complexity of such functions. As a result a new upper bound on complexity for all Boolean functions is obtained.

Keywords: boolean functions; ESOPs; exclusive-or-sum-of-products; minimization; genetic algorithms.

Казимиров Алексей Сергеевич, кандидат физико-математических наук, Восточно-Сибирская государственная академия образования, 664011, Иркутск, ул. Н. Набережная, 6, тел.: (3952) 240477 (a.kazimirov@gmail.com)

Реймеров Сергей Юрьевич, аспирант, Восточно-Сибирская государственная академия образования, 664011, Иркутск, ул. Н. Набережная, 6, тел.: (3952) 240477 (sergeyreym@gmail.com)

Alexey Kazimirov, East Siberian State Academy of Education, 6, N. Naberezhnaya St., Irkutsk, 664011, phone: (3952) 240477 (a.kazimirov@gmail.com)

Sergey Reymerov, East Siberian State Academy of Education, 6, N. Naberezhnaya St., Irkutsk, 664011, postgraduate student, phone: (3952) 240477 (sergeyreym@gmail.com)