ТЕОРЕТИЧЕСКИЕ ОСНОВЫ ПРИКЛАДНОЙ ДИСКРЕТНОЙ МАТЕМАТИКИ

DOI 10.17223/20710410/3/1

УДК 519.7

МИНИМИЗАЦИЯ БУЛЕВЫХ ФУНКЦИЙ МНОГИХ ПЕРЕМЕННЫХ В КЛАССЕ ДНФ — ИТЕРАТИВНЫЙ МЕТОД И ПРОГРАММНАЯ РЕАЛИЗАЦИЯ

А. Д. Закревский, Н. Р. Торопов

Объединенный институт проблем информатики НАН Беларуси, г. Минск, Беларусь

E-mail: zakrevskij@tut.by

Предлагается итеративный алгоритм минимизации булевых функций многих переменных, основанный на использовании параллельных операций над соседними элементами в булевом пространстве аргументов. Он включает операцию быстрого нахождения элементов характеристического множества с малым числом соседей и формирования определяемых ими импликант, итеративную процедуру применения этой операции к последовательно сокращаемому характеристическому множеству и операцию приведения множества полученных конъюнктов к корректной ДНФ.

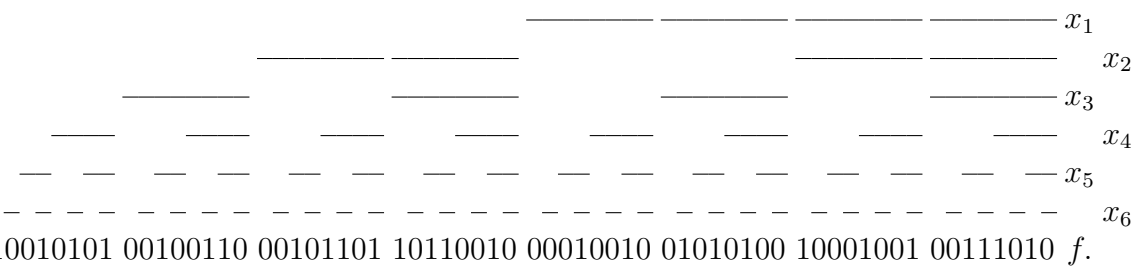
Ключевые слова: *булевы функции, ДНФ, минимизация.*

Введение

Классическая задача минимизации булевых функций в классе дизъюнктивных нормальных форм (ДНФ) может быть сформулирована следующим образом.

Определим произвольную булеву функцию $f(\boldsymbol{x}) \equiv f(x_1, x_2, \ldots, x_n)$ стандартным образом булевым вектором $\boldsymbol{f}$ с 2^n компонентами, пронумерованными слева направо, начиная с нуля, причем компонента с номером k задает значение функции f на наборе значений аргументов, представляющем общепринятый двоичный код $\boldsymbol{b}^k$ числа k.

Например, следующий вектор $\boldsymbol{f}$ представляет булеву функцию f шести переменных x_1, x_2, x_3, x_4, x_5, x_6, принимающую значение 1 на 27 наборах значений аргументов, образующих характеристическое множество $M^1 = \{000000, 000011, 000101, \ldots, 111110\}$ функции f. Порядковые номера соответствующих компонент вектора $\boldsymbol{f}$ равны 0, 3, 5, …, 62.



Заметим, что такой вектор можно интерпретировать как совершенную нормальную форму булевой функции, члены которой представлены единичными компонентами вектора и их кодами. Например, код $\boldsymbol{b}^{10} = 001010$ элемента f_{10} задает полную элементарную конъюнкцию $\overline{x}_1\overline{x}_2x_3\overline{x}_4x_5\overline{x}_6$.

Проблема заключается в том, чтобы представить заданную вектором $\boldsymbol{f}$ функцию f в ДНФ, желательно минимизированной. Заметим, что традиционные методы решения этой задачи связаны с затратами времени, быстро растущими с ростом n, и становятся практически неприемлемы при $n > 20$, когда число членов в совершенной ДНФ исчисляется миллионами [1]. В связи с этим в данной работе предлагается оригинальный метод получения ДНФ булевых функций, число аргументов которых может достигать 25. Метод основан на применении эффективных параллельных операций над булевыми 2^n-векторами, предложенных в работах [2, 3].

К таким операциям относится, в частности, операция конъюнктивного симметрирования вектора $\boldsymbol{f}$ по переменной x_i, обозначаемая ниже через $\mathrm{S}\,\boldsymbol{f} \wedge i$. При ее выполнении вектор $\boldsymbol{f}$ разбивается на 2^{n-1} пар компонент, соседних по переменной x_i, и оба элемента каждой пары получают значение, равное конъюнкции исходных значений этих элементов. Напомним, что *соседними* называются такие компоненты вектора f, которым соответствуют наборы значений аргументов, различающиеся ровно в одном аргументе.

Заметим,что при числе переменных, превышающем 5, удобно представлять вектор $\boldsymbol{f}$ в виде булевой матрицы размером 2^5 на 2^{n-5}, отображая её 32-компонентные строки словами в компьютерной памяти (что адекватно для большинства современных компьютеров). Тогда любые два элемента вектора $\boldsymbol{f}$, соседние по переменной x_i, будут принадлежать к одному слову, если $i < 6$, и к разным словам в противном случае, что можно использовать для ускорения вычислений.

1. Построение булевой матрицы соседства $\boldsymbol{N}$

На первом этапе предлагаемого метода посредством n-кратного применения операции $\mathrm{S}\,\boldsymbol{f} \wedge i$ строится булева *матрица соседства* N размером $n \times 2^n$. Каждая строка $\boldsymbol{n}_i$ этой матрицы представляет результат выполнения операции $\mathrm{S}\,\boldsymbol{f} \wedge i$ при конкретном значении параметра i (от 1 до n). Матрица N отображает структуру характеристического множества M^1 функции $f(\boldsymbol{x})$, на котором эта функция принимает значение 1. Элемент n_i^k матрицы N принимает значение 1, если и только если k-й элемент вектора $\boldsymbol{f}$ равен 1 и имеет соседа по переменной x_i, также со значением 1. Таким образом, i-я строка этой матрицы отображает (единицами) подмножество элементов из M^1, имеющих в этом же множестве соседей по переменной x_i, а k-й столбец показывает, по каким переменным имеет соседей соответствующий элемент из M^1, представленный k-й компонентой вектора $\boldsymbol{f}$.

Продемонстрируем получение матрицы $\boldsymbol{N}$ на примере того же вектора $\boldsymbol{f}$, задающего случайную булеву функцию шести переменных:

$$f = 10010101\ 00100110\ 00101101\ 10110010\ 00010010\ 01010100\ 10001001\ 00111010,$$

$$N = \begin{array}{l} 00010000\ 00000100\ 00001001\ 00110010\ 00010000\ 00000100\ 00001001\ 00110010 \\ 00000101\ 00100010\ 00000101\ 00100010\ 00000000\ 00010000\ 00000000\ 00010000 \\ 00000100\ 00000100\ 00100000\ 00100000\ 00010000\ 00010000\ 00001000\ 00001000 \\ 00010001\ 00100010\ 00000000\ 00100010\ 00000000\ 01000100\ 10001000\ 00100010 \\ 00000101\ 00000000\ 00000101\ 10100000\ 00000000\ 01010000\ 00000000\ 00001010 \\ 00000000\ 00000000\ 00001100\ 00110000\ 00000000\ 00000000\ 00000000\ 00110000. \end{array}$$

2. Матричное представление ДНФ

В аналогичной форме булевыми вектором $\boldsymbol{g}$ и матрицей $\boldsymbol{D}$ той же размерности, называемыми *вектором решения* и *матрицей решения*, предлагается представлять и искомую ДНФ, рассматриваемую как некоторую совокупность интервалов булева пространства $M = \{0,1\}^n$. В векторе $\boldsymbol{g}$ отмечаются некоторые содержащиеся в интервалах элементы множества M^1, по одному для каждого интервала, а в столбцах матрицы $\boldsymbol{D}$ отмечаются внутренние переменные этих интервалов.

Рассматривая векторы $\boldsymbol{f}$ и $\boldsymbol{g}$ как множества отмеченных в них элементов пространства M, а матрицы $\boldsymbol{N}$ и $\boldsymbol{D}$ — как подмножества декартова произведения множеств $\boldsymbol{x}$ и M, сформулируем очевидное

Утверждение 1. $\boldsymbol{g} \subseteq \boldsymbol{f}$ и $\boldsymbol{D} \subseteq \boldsymbol{N}$.

Другими словами, вектор решения $\boldsymbol{g}$ и матрица решения $\boldsymbol{D}$ могут быть получены соответственно из вектора $\boldsymbol{f}$ и матрицы $\boldsymbol{N}$ заменой в них некоторых единиц на нули, как это и делается в описываемом в данной статье методе.

Для рассматриваемого примера искомая ДНФ будет выглядеть следующим образом:

$$g = 10010100\ 00000110\ 00101000\ 10010000\ 00000010\ 01000000\ 10000001\ 00001000,$$

$$D = \begin{matrix} 00010000\ 00000100\ 00000000\ 00010000\ 00000000\ 00000000\ 00000001\ 00000000 \\ 00000100\ 00000010\ 00000000\ 00000000\ 00000000\ 00000000\ 00000000\ 00000000 \\ 00000000\ 00000000\ 00100000\ 00000000\ 00000000\ 00000000\ 00000000\ 00000000 \\ 00000000\ 00000010\ 00000000\ 00000000\ 00000000\ 00000000\ 10000000\ 00000000 \\ 00000100\ 00000000\ 00000000\ 10000000\ 00000000\ 01000000\ 00000000\ 00001000 \\ 00000000\ 00000000\ 00001000\ 00010000\ 00000000\ 00000000\ 00000000\ 00000000. \end{matrix}$$

В более известной форме эта ДНФ задается троичной матрицей, столбцы которой представляют элементарные конъюнкции ($\overline{x}_1\overline{x}_2\overline{x}_3\overline{x}_4\overline{x}_5\overline{x}_6, \overline{x}_2\overline{x}_3\overline{x}_4x_5x_6, \ldots$ и т. д.)

$$\begin{matrix} 1\ 0-0-0\,0\,0\,0-1\,1\ 1-1 \\ 2\ 0\,0-0-1\,1\,1\,1\,0\,0\ 1\,1\,1 \\ 3\ 0\,0\,0\,1\,1-0\ 1\,1\,0\,1\,0\,0\,1 \\ 4\ 0\,0\,1\,1-0\ 1\,0\,0\,1\,0-1\,1 \\ 5\ 0\,1-0\ 1\,1\,0-1\,1-0\ 1- \\ 6\ 0\,1\,1\,1\,0\,0-0-0\,1\,0\ 1\,0. \end{matrix}$$

Число p конъюнкций в полученной ДНФ равно весу вектора решения $\boldsymbol{g}$, а длина ДНФ (сумма рангов конъюнкций) равна $pn - q$, где q — число единиц в матрице $\boldsymbol{D}$.

Заметим, что введенные выше булевы матрицы и векторы при программной реализации предлагаемого метода обрабатываются во внутренних циклах и, следовательно, должны быть представлены в оперативной памяти, что ограничивает их допустимый размер. Учитывая параметры персонального компьютера, можно утверждать, что данный метод достаточно быстро реализуется на нем, если число переменных минимизируемой булевой функции не превышает 25.

3. Выделение элементов с малым числом соседей

Построение реализующей функцию f ДНФ целесообразно начать с поиска элементов ядра решения — обязательных простых импликант высокого ранга. Назовем обязательной такую простую импликанту функции f, которая входит в любую кратчайшую

ДНФ этой функции. Поиск облегчается предварительным выделением в векторе $\boldsymbol{f}$ элементов характеристического множества M^1 с небольшим числом соседей, поскольку именно такие элементы могут определять импликанты высокого ранга, отображаемые элементарными конъюнкциями со многими литералами.

Число соседей у элементов вектора $\boldsymbol{f}$ со значением 1 можно представить конечнозначным вектором $\boldsymbol{w}$

$$\begin{array}{l} f = 10010101\ 00100110\ 00101101\ 10110010\ 00010010\ 01010100\ 10001001\ 00111010 \\ w = 0..2.3.3\ \ ..2..22.\ \ ..1.23.3\ 1.62..3.\ \ ...2..0.\ \ .2.3.2..\ 1...3..1\ \ ..332.3., \end{array}$$

однако более удобно, в перспективе программной реализации последующих вычислений, рассортировать элементы по числу соседей и представить результат серией булевых векторов $\boldsymbol{m}_i$, в которых единицами отмечены элементы с i соседями.

Для рассматриваемого примера, при $i < 4$, получим

$$\begin{array}{rl} f = & 10010101\ 00100110\ 00101101\ 10110010\ 00010010\ 01010100\ 10001001\ 00111010, \\ \boldsymbol{m}_0 = & 10000000\ 00000000\ 00000000\ 00000000\ 00000010\ 00000000\ 00000000\ 00000000, \\ \boldsymbol{m}_1 = & 00000000\ 00000000\ 00100000\ 10000000\ 00000000\ 00000000\ 10000001\ 00000000, \\ \boldsymbol{m}_2 = & 00010000\ 00100110\ 00001000\ 00010000\ 00010000\ 01000100\ 00000000\ 00001000, \\ \boldsymbol{m}_3 = & 00000101\ 00000000\ 00000101\ 00000010\ 00000000\ 00010000\ 00001000\ 00110010. \end{array}$$

Эти векторы, образующие соответствующую булеву матрицу $\boldsymbol{M}$, легко получить эффективными покомпонентными операциями над строками матрицы $\boldsymbol{N}$, что существенно ускоряет поиск подходящих импликант.

4. Нахождение простых импликант

Обозначим через $\boldsymbol{t}^k$ троичный вектор, получаемый из вектора $\boldsymbol{b}^k$ (кода элемента f_i) присвоением значения $-$ компонентам, отмеченным единицами в столбце $\boldsymbol{n}^k$ матрицы $\boldsymbol{N}$. Его можно интерпретировать как некоторый интервал Int_k пространства $M = \{0,1\}^n$, а также как соответствующую элементарную конъюнкцию, которая может оказаться простой импликантой функции f.

Утверждение 2. Вектор $\boldsymbol{t}^k$ представляет обязательную простую импликанту функции f, если и только если $\mathrm{Int}_k \subseteq M^1$.

Утверждение 3. Для каждой обязательной простой импликанты функции f в матрице $\boldsymbol{N}$ найдется столбец $\boldsymbol{n}^k$, соответствующий которому троичный вектор $\boldsymbol{t}^k$ представляет эту импликанту.

Легко находятся обязательные простые импликанты ранга n — они непосредственно представляются элементами множества M^1, не имеющими соседей. Они отмечаются в векторе $\boldsymbol{m}_0$ и представляются соответствующими столбцами матрицы $\boldsymbol{N}$, не содержащими единиц. Аналогично, все обязательные простые импликанты ранга $n-1$ отмечены в векторе $\boldsymbol{m}_1$ и также представлены соответствующими столбцами матрицы $\boldsymbol{N}$ — они содержат по одной единице.

Выявление обязательных простых импликант меньшего ранга несколько сложнее. Однако оно достаточно просто для ранга $n-2$ и $n-3$, осуществляясь посредством покомпонентных операций над некоторыми столбцами матрицы соседства $\boldsymbol{N}$.

Утверждение 4. Предположим, что k-й элемент вектора $\boldsymbol{f}$ имеет двух соседей. Тогда вектор $\boldsymbol{t}^k$ представляет обязательную простую импликанту функции f, если и только если j-я компонента вектора $\boldsymbol{f}$ равна единице, где $\boldsymbol{b}^j = \boldsymbol{b}^k \oplus \boldsymbol{n}^k$.

Например, $\boldsymbol{b}^{27} \oplus \boldsymbol{n}^{27} = 011011 \oplus 100001 = 111010$, $j = 58$ и $f_{58} = 1$, следовательно, троичный вектор $\boldsymbol{t}^{27} = -1101-$ представляет обязательную простую импликанту.

Утверждение 5. Предположим, что k-й элемент вектора $\boldsymbol{f}$ имеет трех соседей. Тогда вектор $\boldsymbol{t}^k$ представляет обязательную простую импликанту функции f, если и только если $\boldsymbol{n}^k \leqslant \boldsymbol{n}^j$ (вектор $\boldsymbol{n}^k$ поглощается вектором $\boldsymbol{n}^j$), где $\boldsymbol{b}^j = \boldsymbol{b}^k \oplus \boldsymbol{n}^k$.

Доказательство этого утверждения основано на том факте, что векторы $\boldsymbol{b}^k$ и $\boldsymbol{b}^j$ являются противолежащими элементами в интервале Int_k ранга 3 и вместе со своими соседями покрывают все восемь элементов этого интервала (рис. 1).

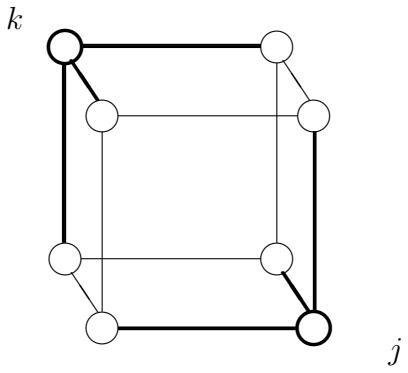


Рис. 1

Утверждение 6. Предположим, что k-й элемент вектора $\boldsymbol{f}$ имеет четырех соседей. Тогда вектор $\boldsymbol{t}^k$ представляет обязательную простую импликанту функции f, если и только если $\boldsymbol{n}^k \leqslant \boldsymbol{n}^j$ для трех (это минимальное число) значений индекса j, которые можно определить в соответствии с рис. 2.

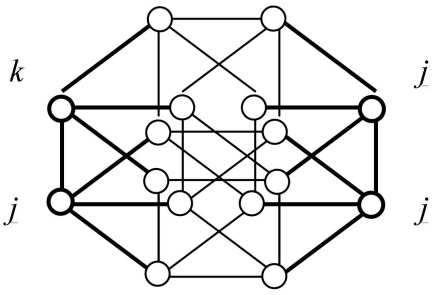


Рис. 2

Возникает вопрос о практической целесообразности проверки компонент вектора $\boldsymbol{f}$ на выполнение условий, сформулированных в утверждениях 4 – 6. Предположим, что задана булева функция от n переменных с вероятностью $1/2$ значения 1 в каждой компоненте, независимо друг от друга. Рассмотрим некоторую компоненту со значением 1 и k соседями. Насколько вероятно, что эта компонента определяет соответствующую простую импликанту ранга $n - k$?

Утверждение 7. Вероятность такого события равна $1/2^t$, где $t = 2^k - (k + 1)$.

Эта вероятность быстро стремится к нулю. Например, при $k = 2, 3, 4$ и 5 она равна $1/2$, $1/16$, $1/2048$ и $1/67\,208\,864$ соответственно, будучи независима от n.

Поэтому в предлагаемом эвристическом алгоритме анализу подвергаются лишь такие единичные компоненты вектора $\boldsymbol{f}$, число соседей у которых не превышает трех.

5. Алгоритм получения частичного решения

В предлагаемом алгоритме последовательно находятся импликанты высокого ранга. Результат фиксируется введением единиц в вектор решения $\boldsymbol{g}$ (сначала $\boldsymbol{g} = \boldsymbol{0}$), определением соответствующих им столбцов матрицы решения $\boldsymbol{D}$ и корректировкой вектора $\boldsymbol{f}^*$, представляющего множество еще не покрытых элементов характеристического множества M^1.

1. $\boldsymbol{g} := \boldsymbol{m}_0$, $\boldsymbol{f}^* := \boldsymbol{f} \setminus \boldsymbol{m}_0$. Так находятся импликанты ранга n (в данном примере это две импликанты, задаваемые векторами 000000 и 100110 и определяемые элементами вектора $\boldsymbol{f}$ с номерами 0 и 38).

2. Последовательно рассматриваются элементы вектора $\boldsymbol{f}$, отмеченные одновременно в векторах $\boldsymbol{m}_1$ и $\boldsymbol{f}^*$. Для очередного элемента f^k берется соответствующий ему набор $\boldsymbol{b}^k$, компонента которого, отмеченная единицей в столбце $\boldsymbol{n}^k$ матрицы соседства $\boldsymbol{N}$, заменяется на символ «—». Результат представляет простую импликанту ранга $n-1$. Соответственно корректируются векторы $\boldsymbol{g}$ и $\boldsymbol{f}^*$: в первый добавляется одна единица, а из второго удаляются две.

Так, в данном примере последовательно рассматриваются элементы с номерами $k = 18, 24, 48$ и 55, которым соответствуют наборы 010010, 011000, 110000 и 110111 и которые определяют простые импликанты ранга $n-1$: $01{-}010, 0110{-}0, 110{-}00, {-}10111$. Вектор решения принимает значение

$$g = 10000000\ 00000000\ 00100000\ 10000000\ 00000010\ 00000000\ 10000001\ 00000000$$

и соответственно (в результате удаления поглощаемых элементов они отмечены подчеркиванием) меняется значение вектора $\boldsymbol{f}^*$:

$$f^* = \underline{0}0010101\ 00100110\ 00\underline{0}01\underline{10}\ \underline{00}\underline{0}10010\ 0001\underline{00}\underline{0}\ 01010100\ \underline{0}\underline{000}\underline{0}\underline{00}\underline{0}\ 00111010.$$

3. На этом этапе рассматриваются последовательно элементы f^k с двумя соседями, отмеченные одновременно в векторах $\boldsymbol{m}_2$ и $\boldsymbol{f}^*$, и строятся импликанты ранга $n-2$.

Сначала находятся элементы f^k, удовлетворяющие условию утверждения 4. Соответствующие компоненты вектора $\boldsymbol{g}$ принимают значение 1, столбцы $\boldsymbol{d}^k$ матрицы $\boldsymbol{D}$ остаются равными столбцам $\boldsymbol{n}^k$ матрицы $\boldsymbol{N}$, и из вектора $\boldsymbol{f}^*$ удаляются единицы, покрываемые интервалами Int_k.

Если же условие утверждения 4 не выполняется, из двух соседей элемента f^k выбирается один сосед, желательно еще не покрытый, в столбце $\boldsymbol{d}^k$ остается лишь одна соответствующая единица и выполняется операция, предусмотренная для элемента с одним соседом.

В данном примере (он оказывается довольно простым) это приводит к окончательному решению — покрытию всех элементов характеристического множества M^1, получению пары векторов

$$g = 10010100\ 00000110\ 00101000\ 10010000\ 00000010\ 01000000\ 10000001\ 00001000,$$
$$f^* = 00000000\ 00000000\ 00000000\ 00000000\ 00000000\ 00000000\ 00000000\ 00000000$$

и построению матрицы решения $\boldsymbol{D}$, получаемой из $\boldsymbol{N}$ удалением одной единицы в столбцах, отмеченных в векторе

$$\boldsymbol{m}_2 = 00010000\ 00100110\ 00001000\ 00010000\ 00010000\ 01000100\ 00000000\ 00001000,$$

и удалением всех единиц в столбцах, не отмеченных в векторе g :

$$D = \begin{matrix} 00010000\ 00000100\ 00000000\ 00010000\ 00000000\ 00000000\ 00000001\ 00000000 \\ 00000100\ 00000010\ 00000000\ 00000000\ 00000000\ 00000000\ 00000000\ 00000000 \\ 00000000\ 00000000\ 00100000\ 00000000\ 00000000\ 00000000\ 00000000\ 00000000 \\ 00000000\ 00000010\ 00000000\ 00000000\ 00000000\ 00000000\ 10000000\ 00000000 \\ 00000100\ 00000000\ 00000000\ 10000000\ 00000000\ 01000000\ 00000000\ 00001000 \\ 00000000\ 00000000\ 00001000\ 00010000\ 00000000\ 00000000\ 00000000\ 00000000. \end{matrix}$$

4. Если вектор $\boldsymbol{f}^*$ остается отличным от нуля, рассматриваются элементы с тремя соседями, отмеченные одновременно в векторах $\boldsymbol{m}_3$ и $\boldsymbol{f}^*$. Находятся элементы, удовлетворяющие условию утверждения 5, в решение вводятся соответствующие импликанты ранга $n-3$. При невыполнении данного условия находятся определяемые этими элементами импликанты более высокого ранга: $n-1$ и $n-2$.

В результате описанной процедуры обработки элементов вектора $\boldsymbol{f}$, имеющих не более трех соседей, мы получаем множество импликант, образующих *частичное решение* S и вектор $\boldsymbol{f}^*$, представляющий *остаток* — совокупность не покрытых этим решением элементов множества M^1.

6. Итеративный алгоритм

Идея этого эвристического алгоритма состоит в следующем. Сначала он находит частичное решение для вектора $\boldsymbol{f}$, затем выполняет такую же операцию для остатка $\boldsymbol{f}^*$, дополняя множество получаемых импликант и соответственно упрощая вектор $\boldsymbol{f}^*$ (удаляя из него некоторые единицы). Если после упрощения $\boldsymbol{f}^*$ в нем остаются единицы, выполняется следующая итерация и так далее, пока $\boldsymbol{f}^*$ не станет равным нулю.

Получаемые при этом импликанты могут быть далеко не обязательными и даже не простыми. Поэтому в заключение они приводятся к простым (понижением ранга), а также устраняются получаемые дубли.

Продемонстрируем алгоритм на конкретном примере. Пусть $n = 19$ и каждый элемент вектора $\boldsymbol{f}$ принимает значение 1 с вероятностью $p = 1/4$. При этом предположении был сгенерирован случайный вектор $\boldsymbol{f}$ с 147 232 единицами и построена матрица соседства $\boldsymbol{N}$ с $2^{19} = 534\,288$ столбцами и следующим распределением столбцов по числу единиц в них ($N(i)$ столбцов содержат по i единиц):

$$\begin{array}{rcccccccccccccccccc} i & = & 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 & 11 & 12 & 13 & 14 & 15 & 16, \\ N(i) & = & 296 & 2087 & 7180 & 16352 & 25357 & 29868 & 26913 & 19406 & 11379 & 5401 & 2087 & 690 & 172 & 35 & 8 & 1 & 0. \end{array}$$

При первой итерации алгоритма обрабатываются элементы множества M^1 с числом соседей $i = 0, 1, 2$ и 3, всего $296 + 2087 + 7180 + 16352 = 25915$ элементов. Находятся определяемые ими обязательные простые импликанты, общим числом $296 + 2082 + {} + 1974 + 80 = 4432$. Общее число найденных при первой итерации простых импликант (вместе с необязательными) равно 24 379. Они отмечаются в векторе решения $\boldsymbol{g}$, и покрываемые ими элементы удаляются из переменного вектора остатка $\boldsymbol{f}^*$ (вначале равного $\boldsymbol{f}$). Число единиц в векторе $\boldsymbol{f}^*$ становится равным 81 910.

При второй итерации рассматривается новая матрица $\boldsymbol{N}$, представляющая отношение соседства на элементах множества M^1, отмеченных в векторе остатка $\boldsymbol{f}^*$. В этом

векторе выявляются элементы, имеющие не более трех соседей в этом же векторе, находятся соответствующие ими импликанты. Векторы $\boldsymbol{g}$ и $\boldsymbol{f}^*$ получают новые значения. Если после этого $\boldsymbol{f}^* \neq \boldsymbol{0}$, выполняется следующая итерация.

Так для рассматриваемого примера выполняются четыре итерации, прежде чем вектор $\boldsymbol{f}^*$ становится равным $\boldsymbol{0}$ — все элементы множества M^1 оказываются покрытыми 61 477 импликантами со следующим распределением их по рангам (ранг – число импликант):

$$19 - 2\,458, 18 - 33\,668, 17 - 25\,237, 16 - 114.$$

Как уже говорилось, не все эти импликанты оказываются простыми. Поэтому в заключение выполняются две процедуры приведения полученного решения к корректному виду. Первая из них упрощает импликанты, обращаясь к исходной матрице соседства $\boldsymbol{N}$ и удаляя из импликант некоторые литералы, если это возможно. Она приводит к новому распределению импликант по рангам:

$$19 - 296, 18 - 20\,933, 17 - 38\,617, 16 - 1\,631.$$

Вторая процедура устраняет дубли среди полученных импликант, в результате чего число последних сокращается до 60 972, а распределение их по рангам принимает следующий вид:

$$19 - 296, 18 - 20\,933, 17 - 38\,353, 16 - 1\,390.$$

7. Компьютерные эксперименты

Изложенный выше итеративный алгоритм был программно реализован на C++ и испытан на компьютере (Pentium IV, 2.8 GHz). Чтобы было удобней работать с булевыми векторами, в которых перечислены соседи для рассматриваемых элементов вектора $\boldsymbol{f}$, матрица соседства $\boldsymbol{N}$ предварительно транспонировалась.

Эксперименты проводились на множестве псевдослучайных булевых функций, с двумя параметрами: *числом переменных* n и *плотностью единиц* r, определяющей ожидаемое число единиц q в представляющем булеву функцию f векторе $\boldsymbol{f}$: $r = 32q/2^n$. Например, при $r = 16$ рассматривается абсолютно случайная булева функция, принимающая на каждом элементе булева пространства значение 1 с вероятностью $1/2$.

Прежде всего, были определены границы практической применимости предложенного алгоритма в пространстве параметров n и r. Дело в том, что при достаточно большом значении параметра r алгоритм может прекратить свою работу после некоторого числа итераций, поскольку числа $N(0), N(1), N(2)$ и $N(3)$ могут оказаться равными нулю, в то время как вектор $\boldsymbol{f}^*$ будет оставаться отличным от $\boldsymbol{0}$.

Например, если $n = 17$ и $r = 15$, алгоритм останавливается после восьми итераций, найдя всего 636 импликант. Но при $n = 17$ и $r = 14$ он находит после 90 итераций 20 077 импликант, полностью покрывающих множество M^1 (после последующего упрощения этих импликант и устранения дублей их число сокращается до 19 811). Следовательно, пара $(n, r) = (17, 14)$ является элементом верхней границы области применения алгоритма.

В табл. 1 представлены основные характеристики этой границы, полученные экспериментально. Строки таблицы соответствуют числу переменных n (от 4 до 23) и соответствующим им максимальным значениям параметра r, при которых программа работает корректно. Кроме того, в строках показаны следующие величины:

N — число единиц в векторе $\boldsymbol{f}$ (число импликант в совершенной ДНФ функции f);

C — число импликант в полученной ДНФ;
S — длина полученной ДНФ (сумма рангов импликант);
Q_k — число импликант ранга k в полученной ДНФ (при $k = n, n-1, n-2, n-3, n-4$);
It — число итераций;
T – затраченное время, в с.

Таблица 1

Экспериментальные характеристики границы области применения алгоритма

n	r	N	C	S	Q_n	Q_{n-1}	Q_{n-2}	Q_{n-3}	Q_{n-4}	It	T
4	16	5	3	10	1	2	0	0	0	1	0,00
5	16	16	8	27	0	3	5	0	0	1	0,00
6	16	31	14	62	1	4	9	0	0	1	0,00
7	16	62	31	169	1	12	18	0	0	1	0,00
8	16	138	58	360	2	18	28	10	0	2	0,00
9	16	274	107	744	0	15	72	20	0	3	0,00
10	16	556	194	1513	0	14	127	53	0	3	0,00
11	16	1083	379	3355	1	35	250	93	0	4	0,01
12	16	2203	735	7127	0	31	456	242	6	5	0,03
13	16	4337	1414	15057	0	45	835	526	8	7	0,09
14	16	8734	2780	32266	0	64	1579	1116	21	12	0,35
15	15	16404	5313	67324	1	168	3258	1858	27∗	13	1,01
16	14	31021	10181	139827	1	412	6671	3073	24	13	3,12
17	14	61150	19811	291507	1	684	12842	6224	60	90	24,37
18	13	114347	38007	500357	4	1809	26670	9476	48	21	42,26
19	12	212620	72343	1220742	7	4787	53693	13822	34	12	148
20	11	392995	137215	2462996	34	12639	104999	19505	38	10	610
21	11	786345	270642	5121875	54	21491	207248	41776	73	18	2499
22	10	1442274	512529	10255122	127	58852	399029	54478	43	10	8858
23	9	2620069	966357	20386490	481	158666	740594	66597	19	8	31064

Табл. 2 отображает поведение алгоритма при числе переменных $n = 17$. Видно, как растет число итераций It и время решения T при возрастании плотности единиц r. При этом распределение полученных импликант по рангам быстро изменяется в пользу импликант меньшего ранга.

Таблица 2

Поведение алгоритма при числе переменных $n = 17$

n	r	N	C	S	Q_{17}	Q_{16}	Q_{15}	Q_{14}	Q_{13}	It	T
17	2	12285	7856	127689	2283	5283	290	0	0	2	0,27
17	4	20427	11117	177205	1147	8162	1802	6	0	2	0,53
17	6	28594	13761	215604	417	8406	4887	51	0	3	1,90
17	8	36507	15621	240722	135	6370	8883	233	0	3	4,04
17	10	44756	17348	263178	41	3864	12456	986	1	4	6,49
17	12	52999	18691	279341	6	1880	13899	2896	10	7	8,43
17	14	61150	19811	291507	1	684	12842	6224	60	90	24,52

Поведение алгоритма при максимально возможном для него числе переменных ($n = 24$) показано в табл. 3. При увеличении плотности единиц r на единицу время решения T растет более чем вдвое, достигая 6,8 ч при $r = 4$.

Таблица 3

Поведение алгоритма при числе переменных $n = 24$

n	r	N	C	S	Q_{24}	Q_{23}	Q_{22}	Q_{21}	It	T
24	1	1047350	685881	15982597	223163	446889	15829	0	2	1693,45
24	2	1571532	919682	21231157	148570	701045	70035	32	2	4068,76
24	3	2095590	1124293	25759214	85794	853387	184905	207	3	10695,15
24	4	2619724	1297946	29532155	44585	889335	362864	1162	3	24348,00

Заключение

В предлагаемом алгоритме минимизации булевых функций многих переменных (до 24) реализованы следующие идеи. 1) Роль элементарных операндов играют булевы векторы с 2^n компонентами, представляющие произвольные булевы функции n переменных. 2) Строится булева матрица соседства $\boldsymbol{N}$ размером $n \times 2^n$, отображающая структуру характеристического множества M^1. 3) С ее помощью быстро находятся простые импликанты четырех старших рангов, покрывающие часть множества M^1. 4) Структура остатка представляется новой матрицей $\boldsymbol{N}$, находятся новые импликанты высокого ранга и т. д. Итерации прекращаются, когда множество M^1 становится пустым. 5) В заключение полученные импликанты доводятся до простых и устраняются дубли. Алгоритм реализуется эффективной программой, которая может оказаться полезной при решении систем булевых уравнений, верификации логических схем и решении других трудоемких задач теории булевых функций.

ЛИТЕРАТУРА

1. *Закревский А. Д.* Логический синтез каскадных схем. М., 1981.
2. *Zakrevskij A. D.* Parallel operations over neighbors in Boolean space //Proceedings of the Sixth International Conference CAD DD-07. Minsk, 2007. V. 2. P. 613.
3. *Закревский А. Д.* Программирование вычислений в многомерном булевом пространстве // 7-я Российская конф. с международным участием «Новые информационные технологии в исследовании сложных структур». Томск, 2008.