ИНФОРМАТИКА

2008 **октябрь-декабрь** **№ 4**

ЛОГИЧЕСКОЕ ПРОЕКТИРОВАНИЕ

УДК 519.7

А.Д. Закревский, Н.Р. Торопов

РЕАЛИЗАЦИЯ ЭВРИСТИЧЕСКОГО МЕТОДА ДЕКОМПОЗИЦИИ БУЛЕВЫХ ФУНКЦИЙ

Исследуется оригинальный эвристический алгоритм последовательной двухблочной декомпозиции частичных булевых функций по нестрогому разбиению на множестве аргументов. Рассматривается ключевая комбинаторная задача: нахождение пригодного разбиения на множестве аргументов, т. е. такого, по которому функция разделима. Поиск пригодного разбиения существенно ускоряется путем предварительного обнаружения его следов. В рамках экспериментальной системы оценивается эффективность алгоритма, определяются границы его практического применения.

Введение

Под декомпозицией булевой функции принято понимать ее представление в виде композиции функций от меньшего числа переменных. Эта задача многообразна – достаточно заметить, что любая нетривиальная логическая схема, реализующая некоторую булеву функцию от многих переменных, может рассматриваться как композиция функций, реализуемых отдельными элементами. Рассмотрим частный, но важный случай задачи, ставший классическим, поскольку ему посвящена не одна сотня публикаций, а именно *последовательную двухблочную* декомпозицию.

В этом случае задача формулируется следующим образом. На множестве аргументов $\boldsymbol{x} = (x_1, x_2, \ldots, x_n)$ задана булева функция $f(\boldsymbol{x})$. Требуется заменить ее эквивалентной композицией $g(h(\boldsymbol{u}, \boldsymbol{w}), \boldsymbol{w}, \boldsymbol{v}) = f(\boldsymbol{x})$ булевых функций g и h от меньшего числа переменных. При этом должны выполняться отношения $\boldsymbol{x} = \boldsymbol{u} \cup \boldsymbol{w} \cup \boldsymbol{v}$, $\boldsymbol{u} \cap \boldsymbol{w} = \boldsymbol{u} \cap \boldsymbol{v} = \boldsymbol{w} \cap \boldsymbol{v} = \varnothing$, порождающие *нестрогое разбиение* $\boldsymbol{u}/\boldsymbol{v}$ на множестве аргументов $\boldsymbol{x}$. Эта замена называется декомпозицией функции $f(\boldsymbol{x})$ по разбиению $\boldsymbol{u}/\boldsymbol{v}$ ($f(\boldsymbol{x})$ *декомпозируема* по $\boldsymbol{u}/\boldsymbol{v}$) и может упростить логическую схему, реализующую функцию $f(\boldsymbol{x})$ (например, при логическом синтезе в базисе элементов LUT (look up tables)). Она имеет смысл при условии $|\boldsymbol{u}| > 1$ и $|\boldsymbol{v}| > 0$, в противном случае декомпозиция тривиальна – схема не упрощается. Если $|\boldsymbol{w}| = 0$, композиция называется *разделительной*, в противном случае – *неразделительной*. Задача существенно усложняется, если булева функция $f(\boldsymbol{x})$ оказывается *частичной*, будучи определена не на всех элементах булева пространства $M = \{0, 1\}^n$.

Декомпозиция булевой функции является трудной комбинаторной задачей, трудоемкость которой быстро растет с увеличением числа переменных n. Ниже рассматривается оригинальный *эвристический метод* декомпозиции, эффективность которого обеспечивается практически приемлемым компромиссом между скоростью нахождения решения и его надежностью. Программная реализация метода основана на использовании набора специальных макроопераций над длинными (2^n-компонентными) булевыми векторами $\boldsymbol{f}$, которыми можно представлять произвольные булевы функции $f(\boldsymbol{x})$ от n переменных.

В работе [1] было доказано

Утверждение 1. *При равновероятной выборке функции $f(\boldsymbol{x})$ из множества всех булевых функций от n переменных вероятность декомпозируемости функции $f(\boldsymbol{x})$ ограничена сверху величиной $C_n^2 (n-2) \gamma^{2^{n-3}}$, где $\gamma = 88/256$.*

Эта величина быстро стремится к нулю с ростом n (например, при $n = 6, 12, 18$ она принимает соответственно значения 0,012, 10^{-234}, 10^{-15193}), откуда следует, что декомпозируемость случайных булевых функций от многих переменных весьма маловероятна. Следовательно, практический смысл задача декомпозиции имеет лишь для такой функции, которая, как априо-

ри известно, представима некоторой композицией, которую и требуется найти. Ниже рассматривается задача именно в этой постановке.

1. Подготовка исходных данных

Широко распространен подход, согласно которому конкретные примеры исходных данных для программ решения многообразных задач логического проектирования выбираются из специальных библиотек, формируемых с учетом некоторых практических соображений. В данной статье предлагается другой способ, основанный на генерировании случайных примеров с заданными значениями некоторых параметров.

В рассматриваемой задаче декомпозиции такими параметрами являются: n – число аргументов функции $f(\boldsymbol{x}) = f(x_1, x_2, \ldots, x_n)$, $p = |\boldsymbol{u}|$ (мощность множества $\boldsymbol{u}$), $q = |\boldsymbol{v}|$ и r – степень неопределенности функции $f(\boldsymbol{x})$ (более точно эта величина будет определена ниже).

В ходе подготовки примера из множества $\boldsymbol{x}$ выбираются случайным образом p переменных, образующих множество $\boldsymbol{u}$, и q переменных, образующих множество $\boldsymbol{v}$. Остальные переменные из множества $\boldsymbol{x}$ образуют множество $\boldsymbol{w}$. Затем генерируются случайные булевы функции $h(\boldsymbol{u}, \boldsymbol{w})$ и $g(x, \boldsymbol{w}, \boldsymbol{v})$. Эти операции достаточно просты и выполняются с использованием датчиков случайных булевых векторов с равномерным распределением.

Более сложной оказывается композиция функций $h(\boldsymbol{u}, \boldsymbol{w})$ и $g(x, \boldsymbol{w}, \boldsymbol{v})$, приводящая к получению функции $f(\boldsymbol{x})$. При изложении предлагаемого ниже метода решения этой задачи удобно пользоваться языком макроопераций над длинными 2^n-компонентными булевыми векторами $\boldsymbol{f}$. Наряду со стандартными покомпонентными операциями над векторами одинаковой размерности ($\boldsymbol{a} \vee \boldsymbol{b}$, $\boldsymbol{a} \oplus \boldsymbol{b}$ и т. д.) в него входят высокоэффективные операции над соседними элементами булева пространства, выполняемые параллельно во всех 2^{n-1} парах соседних элементов [2]. Пока воспользуемся следующей из таких операций:

$\boldsymbol{f} - k$ – присвоение значения 0 аргументу x_k (получение функции $f(x_k = 0)$),

а также обобщающей ее операцией $\boldsymbol{f} - \boldsymbol{u}$, при выполнении которой значение 0 присваивается всем аргументам, отмеченным единицами в n-компонентном векторе $\boldsymbol{u}$.

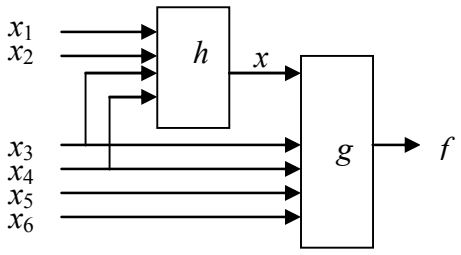
Воспользуемся также операцией $\boldsymbol{h} \times \boldsymbol{u}$ отображения функции $h(\boldsymbol{u})$ в интервал пространства M, соответствующий конъюнкции инверсий переменных, не отмеченных в $\boldsymbol{u}$. Все элементы остальных интервалов с такими же внешними переменными получают при этом значение 0.

В терминах этих операций программа вычисления вектора $\boldsymbol{f}$, представляющего искомую функцию $f(\boldsymbol{x})$, выглядит так:

$$\begin{aligned}
&\boldsymbol{a} := \boldsymbol{h} \times (\boldsymbol{u}, \boldsymbol{w}) - \boldsymbol{v};\\
&\boldsymbol{b} := \boldsymbol{g}_0 \times (\boldsymbol{w}, \boldsymbol{v}) - \boldsymbol{u};\\
&\boldsymbol{c} := \boldsymbol{g}_1 \times (\boldsymbol{w}, \boldsymbol{v}) - \boldsymbol{u};\\
&\boldsymbol{f} := \ \overline{\boldsymbol{a}}\, \boldsymbol{b} \vee \boldsymbol{a}\, \boldsymbol{c},
\end{aligned}$$

где векторы $\boldsymbol{g}_0$ и $\boldsymbol{g}_1$ представляют две половинки вектора $\boldsymbol{g}$, задающие коэффициенты разложения функции $g(x, \boldsymbol{w}, \boldsymbol{v})$ по переменной x.

Покажем эту программу на рисунке, где $n = 6$, а множества $\boldsymbol{u} = (x_1, x_2)$, $\boldsymbol{w} = (x_3, x_4)$ и $\boldsymbol{v} = (x_5, x_6)$ заданы соответствующими булевыми шестимерными векторами $\boldsymbol{u} = 110000$, $\boldsymbol{w} = 001100$ и $\boldsymbol{v} = 000011$.

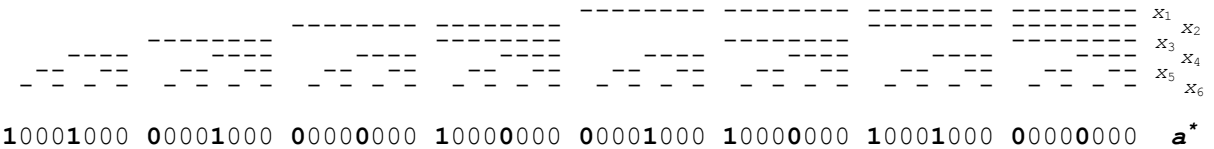


Логическая схема

Допустим, что сгенерированные булевы функции $h(\boldsymbol{u}, \boldsymbol{w})$ и $g(x, \boldsymbol{w}, \boldsymbol{v})$ представлены соответственно 2^4-компонентным вектором $\boldsymbol{h}$ и 2^5-компонентным вектором $\boldsymbol{g}$ с общепринятым порядком следования компонент:

$\boldsymbol{h}$ = 11010010 01101100,
$\boldsymbol{g}$ = 00110100 11001001 10100101 10101011.

Тогда вычисления сводятся к получению показанной ниже последовательности булевых векторов. При вычислении векторов $\boldsymbol{a}^* = \boldsymbol{h} \times (\boldsymbol{u}, \boldsymbol{w})$, $\boldsymbol{b}^* = \boldsymbol{g}_0 \times (\boldsymbol{w}, \boldsymbol{v})$ и $\boldsymbol{c}^* = \boldsymbol{g}_1 \times (\boldsymbol{w}, \boldsymbol{v})$ сначала находится соответствующий интервал пространства M (он отмечен полужирным шрифтом), а затем в него заносится вектор рассматриваемой функции ($\boldsymbol{h}$, $\boldsymbol{g}_0$ или $\boldsymbol{g}_1$) с сохранением порядка следования компонент:



10001**1**000	**0**0001**1**000	**0**0000**0**000	**1**0000**0**000	**0**0001**1**000	**1**0000**0**000	**1**0001**1**000	**0**0000**0**000	$\boldsymbol{a}^*$
11111111	00001111	00000000	11110000	00001111	11110000	11111111	00000000	$\boldsymbol{a}$
00110100	**11001001**	00000000	00000000	00000000	00000000	00000000	00000000	$\boldsymbol{b}^*$
00110100	11001001	00110100	11001001	00110100	11001001	00110100	11001001	$\boldsymbol{b}$
10100101	**10101011**	00000000	00000000	00000000	00000000	00000000	00000000	$\boldsymbol{c}^*$
10100101	10101011	10100101	10101011	10100101	10101011	10100101	10101011	$\boldsymbol{c}$
00000000	11000000	00110100	00001001	00110000	00001001	00000000	11001001	$\boldsymbol{ab}$
10100101	00001011	00000000	10100000	00000101	10100000	10100101	00000000	$\boldsymbol{ac}$
10100101	11001011	00110100	10101001	00110101	10101001	10100101	11001001	$\boldsymbol{f}$

После получения функции $f(\boldsymbol{x})$ в нее вносится неопределенность путем замены некоторых из ее значений (0 или 1) на символ неопределенности (–). Результат этого преобразования представляется одним троичным вектором $\boldsymbol{f}^-$ либо парой булевых векторов $\boldsymbol{f}^0$ и $\boldsymbol{f}^1$, в которых единицами отмечены значения 0 и 1 функции $f(\boldsymbol{x})$. Например (n = 4):

$\boldsymbol{f}^-$ = 011–0001 –101–110,
$\boldsymbol{f}^0$ = 10001110 00100001,
$\boldsymbol{f}^1$ = 01100001 01010110.

Степень неопределенности функции задается параметром r, принимающим значение из множества {0, 1, 2, …, 31} и определяющим вероятность $r/32$ того, что произвольно выбранная компонента вектора $\boldsymbol{f}^-$ получит значение –.

Внесение неопределенности в функцию $f(\boldsymbol{x})$ осуществляется на основе результатов, полученных в [3], суть которых можно уяснить из следующего примера. Чтобы получить случайный булев вектор с вероятностью 19/32 появления единицы в произвольно выбранной компоненте, достаточно представить число 19 его двоичным кодом 10011, поставить в соответствие значениям 0 и 1 булевы операции $\wedge$ и $\vee$ и, перебирая компоненты кода справа налево, выполнить следующую последовательность операций над полностью случайными (с вероятностью 0,5 появления единицы в любой компоненте) независимыми булевыми векторами $\boldsymbol{c}_1, \boldsymbol{c}_2, \boldsymbol{c}_3, \boldsymbol{c}_4, \boldsymbol{c}_5$:

$$((\mathbf{0} \vee \boldsymbol{c}_1 \vee \boldsymbol{c}_2) \wedge \boldsymbol{c}_3 \wedge \boldsymbol{c}_4) \vee \boldsymbol{c}_5,$$

где **0** – вектор, все компоненты которого равны нулю.

2. Метод поиска пригодного разбиения по следу

В основе методов декомпозиции булевых функций лежит решение следующих задач.

Задача 1. Для заданных функции $f(\boldsymbol{x})$ и разбиения $\boldsymbol{u}/\boldsymbol{v}$ выяснить, декомпозируется ли $f(\boldsymbol{x})$ по $\boldsymbol{u}/\boldsymbol{v}$, т. е. существует ли композиция $g(h(\boldsymbol{u}, \boldsymbol{w}), \boldsymbol{w}, \boldsymbol{v})$, эквивалентная функции f, и, может быть, найти функции g и h.

Если такая композиция существует, назовем разбиение $\boldsymbol{u}/\boldsymbol{v}$ *пригодным*.

Задача 2. Для заданной функции $f(\boldsymbol{x})$ найти пригодное разбиение.

Вторая задача более трудная. Очевидно, что перебор всех разбиений с целью проверки их на пригодность практически неосуществим при больших n, поскольку их число аппроксимируется величиной 3^n. Существенно более эффективным является предложенный в [1] метод поиска пригодного разбиения по следу. Он основан на следующих определениях и утверждениях.

Пусть заданы два разбиения $\boldsymbol{u}/\boldsymbol{v}$ и $\boldsymbol{u}^*/\boldsymbol{v}^*$, связанные отношениями $\boldsymbol{u}^* \subseteq \boldsymbol{u}$ и $\boldsymbol{v}^* \subseteq \boldsymbol{v}$. Будем говорить, что разбиение $\boldsymbol{u}^*/\boldsymbol{v}^*$ *подчиняется* разбиению $\boldsymbol{u}/\boldsymbol{v}$, или является его *следом*.

Утверждение 2. *Если булева функция $f(\boldsymbol{x})$ декомпозируется по разбиению $\boldsymbol{u}/\boldsymbol{v}$, то она декомпозируется и по его следу $\boldsymbol{u}^*/\boldsymbol{v}^*$.*

Следствие. *Если функция $f(\boldsymbol{x})$ не декомпозируется по разбиению $\boldsymbol{u}^*/\boldsymbol{v}^*$, то она не декомпозируется и по $\boldsymbol{u}/\boldsymbol{v}$.*

Положим, что $|\boldsymbol{u}| = k$ и $|\boldsymbol{v}| = m$. Разбиение с $k = 2$ и $m = 1$ назовем *триадой*. Оно является простейшим из разбиений, по которому может иметь место нетривиальная декомпозиция.

Утверждение 3. *Булева функция $f(\boldsymbol{x})$ не декомпозируема, если она не декомпозируется ни по какой из триад.*

Поэтому будем начинать поиск пригодного разбиения $\boldsymbol{u}/\boldsymbol{v}$ с поиска его следов на множестве триад, т. е. с нахождения пригодной триады. Положим, что поиск заключается в случайной выборке серии из q триад, завершающейся нахождением пригодной. В работе [1] показано, что практически (при $n > 10$) любая пригодная триада подчиняется искомому разбиению $\boldsymbol{u} / \boldsymbol{v}$, другими словами, побочные решения отсутствуют. При этом предположении справедливо

Утверждение 4. *Математическое ожидание $M(q)$ величины q равно $C_n^2(n-2) / C_k^2\, m$.*

Таким образом, поиск пригодной триады производится довольно быстро. Например, представляющее величину $M(q)$ отношение можно аппроксимировать более простой формулой $3^3 = 27$ (при $k = m = n/3$) и $2^3 = 8$ (при $k = m = n/2$).

Найдя пригодную триаду, можно последовательно расширять множества $\boldsymbol{u}$ и $\boldsymbol{v}$, испытывая различные переменные на возможность их включения в одно из этих множеств (включение возможно, если расширенное разбиение остается пригодным).

3. Программирование в макрооперациях

Фрагменты разбиения $\boldsymbol{u} / \boldsymbol{v}$. Любое нестрогое разбиение $\boldsymbol{u} / \boldsymbol{v}$ делит 2^n-компонентный троичный вектор $\boldsymbol{f}^-$ на 2^{n-p-q} частей, представляющих коэффициенты дизъюнктивного разложения функции $f(\boldsymbol{x})$ по переменным множества $\boldsymbol{w}$. Каждую из этих частей можно представить соответствующим *фрагментом* – так назовем троичную матрицу размером 2^p на 2^q, строки которой соответствуют различным наборам значений переменных из $\boldsymbol{u}$, а столбцы – наборам значений переменных из $\boldsymbol{v}$.

Разбиение $\boldsymbol{u} / \boldsymbol{v}$ оказывается пригодным, если пригоден каждый фрагмент разбиения $\boldsymbol{u} / \boldsymbol{v}$, а фрагмент пригоден, если частичную булеву функцию $f(\boldsymbol{x})$ можно доопределить так, что он будет содержать не более двух значений булевых строк. Другими словами, граф ортогональности строк каждого фрагмента должен быть бихроматичным [4]. Проверка этого условия упрощается для частного случая разбиения – триады $(a, b) / c$. В этом случае размер графа ортогональности равен 4×2.

Проверка триады на пригодность. Граф ортогональности строк фрагмента триады содержит четыре вершины, следовательно, он бихроматичен, если в нем не найдется цикла длины три. Рассмотрим две строки фрагмента триады, соответствующие значениям $(a, b) = (0, 0)$ и $(a, b) = (1, 1)$.

Если такой цикл существует, то, по крайней мере, одна из выбранных вершин будет ему принадлежать. Следовательно, достаточно проверить каждую из этих двух вершин на вхождение в цикл длины три. Если такое вхождение не будет обнаружено, граф бихроматичен и, значит, триада пригодная. Необходимое и достаточное условие вхождения вершины, т. е. соответствующей ей строки, в цикл длины три сформулируем так: среди строк, ортогональных данной, существуют взаимно ортогональные строки.

Предложенный способ реализуется следующим алгоритмом, который замечателен тем, что он проверяет на пригодность одновременно все фрагменты, соответствующие заданной триаде, и тем самым проверяет пригодность триады в целом. Анализируемая частичная булева функция $f(\boldsymbol{x})$ задается парой булевых векторов $\boldsymbol{f}^0$ и $\boldsymbol{f}^1$.

В этом алгоритме наряду с введенной ранее операцией $\boldsymbol{f} - k$ используется операция $\boldsymbol{f} + k$, определяемая аналогично: присвоением значения 1 аргументу x_k. Обе операции легко реализуются в булевом пространстве на парах элементов, соседних по переменной x_i. В алгоритм входит также операция

$$\mathrm{S}_k^{*}\boldsymbol{f} = (\boldsymbol{f} - k)*(\boldsymbol{f} + k)$$

симметрирования вектор-функции $\boldsymbol{f}$ по переменной x_k и булевой операции $* \in \{\vee, \wedge, \oplus\}$. В дальнейшем изложении используется обобщенная операция $\mathrm{S}_{\boldsymbol{t}}^{*}\boldsymbol{f}$, эквивалентная операции $\mathrm{S}_k^{*}\boldsymbol{f}$, которая выполняется для всех переменных, образующих подмножество $\boldsymbol{t} = (t_1, t_2, \ldots, t_m) \subseteq \boldsymbol{x}$.

Проиллюстрируем введенные операции следующими примерами, в которых $\boldsymbol{u} = (x_2, x_5)$:

$$\begin{aligned}
\boldsymbol{f} &= 10010010\ 01111000\ 01100001\ 11100011,\\
\boldsymbol{f} - 2 &= 10010010\ 10010010\ 01100001\ 01100001,\\
\boldsymbol{f} + 2 &= 01111000\ 01111000\ 11100011\ 11100011,\\
\boldsymbol{f} - \boldsymbol{u} &= 11000011\ 11000011\ 00110000\ 00110000,\\
\mathrm{S}_2^{\vee}\boldsymbol{f} &= 11111010\ 11111010\ 11100011\ 11100011,\\
\mathrm{S}_2^{\oplus}\boldsymbol{f} &= 11101010\ 11101010\ 10000010\ 10000010,\\
\mathrm{S}_{\boldsymbol{u}}^{\oplus}\boldsymbol{f} &= 00111111\ 00111111\ 11000011\ 11000011.
\end{aligned}$$

Проверка триады производится сначала по первым строкам фрагментов, образующим в совокупности начальный коэффициент f^{-} разложения функции $f(\boldsymbol{x})$ по переменным a и b. Строки, ортогональные начальной строке, отмечаются значением 1 в вычисляемом векторе $\boldsymbol{g}$ и проверяются далее на совместность (неортогональность). С этой целью вычисляется пара векторов $\boldsymbol{h}^0$ и $\boldsymbol{h}^1$. Так же, как и исходные векторы $\boldsymbol{f}^0$ и $\boldsymbol{f}^1$, это булевы векторы с 2^n компонентами:

$\boldsymbol{h}^0 := (\boldsymbol{f}^0 - a) - b;$
$\boldsymbol{h}^1 := (\boldsymbol{f}^1 - a) - b$ – получение начального коэффициента f^{-};
$\boldsymbol{g} := \mathrm{S}_c^{\vee}(\boldsymbol{h}^0\boldsymbol{f}^1 \vee \boldsymbol{h}^1\boldsymbol{f}^0)$ – выделение строк, ортогональных начальной строке;

$\boldsymbol{h}^0 := \mathrm{S}_a^{\vee}(\mathrm{S}_b^{\vee}(\boldsymbol{f}^0\boldsymbol{g})$;
$\boldsymbol{h}^1 := \mathrm{S}_a^{\vee}(\mathrm{S}_b^{\vee}(\boldsymbol{f}^1\boldsymbol{g})$ – проверка их на совместность.

Если оказывается, что $\boldsymbol{h}^0\boldsymbol{h}^1 \neq \boldsymbol{0}$, то триада признается непригодной. В противном случае производится проверка по последним строкам, образующим конечный коэффициент f^{+} (при этом символ «–» в первых двух строках алгоритма заменяется на «+»). Если оказывается, что триада все-таки непригодна, испытываются другие триады, пока не будет найдена пригодная.

Поиск разбиения по следу. Если рассматриваемая триада $(a, b)/c$ оказалась пригодной, можно предположить, что она является следом искомого пригодного разбиения. В этом случае последнее можно найти, двигаясь по следу, т. е. используя полученное на предыдущем этапе значение вектора $\boldsymbol{g}$ и последовательно расширяя множества $\boldsymbol{u}$ и $\boldsymbol{v}$ с начальными значениями $\boldsymbol{u} = (a, b)$ и $\boldsymbol{v} = (c)$.

Начнем с множества $\boldsymbol{v}$. Перебирая последовательно все элементы s из множества $\boldsymbol{x} \setminus (\boldsymbol{u} \cup \boldsymbol{v})$, будем находить среди них такие, при включении которых в множество $\boldsymbol{v}$ разбиение $\boldsymbol{u}/\boldsymbol{v}$ остается пригодным. С этой целью для каждого элемента s выполняются три операции

$$\begin{aligned}
\boldsymbol{e} &:= \mathrm{S}_s^{\vee}\boldsymbol{g}\ ;\\
\boldsymbol{h}^0 &:= \mathrm{S}_{\boldsymbol{u}}^{\vee}(\boldsymbol{f}^0\boldsymbol{e})\ ;\\
\boldsymbol{h}^1 &:= \mathrm{S}_{\boldsymbol{u}}^{\vee}(\boldsymbol{f}^1\boldsymbol{e}),
\end{aligned}$$

и если $\boldsymbol{h}^0\boldsymbol{h}^1 = \boldsymbol{0}$, то s включается в $\boldsymbol{v}$, что реализуется операциями

$$\boldsymbol{v} := \boldsymbol{v} \cup \{s\},\ \boldsymbol{g} := \boldsymbol{e}.$$

Затем находится максимальное расширение множества $\boldsymbol{u}$. Если известно, что искомое разбиение является строгим, можно положить $\boldsymbol{u} = \boldsymbol{x} \setminus \boldsymbol{v}$. В противном случае следует проверять все элементы из текущего значения множества $\boldsymbol{x} \setminus (\boldsymbol{u} \cup \boldsymbol{v})$ и, если это возможно, включать их в множество $\boldsymbol{u}$.

Проверку очередного элемента s будем производить эвристическим алгоритмом, действия которого ограничиваются тем, что он рассматривает начальный коэффициент f^- разложения функции f по текущему значению множества $\boldsymbol{u}$, отыскивает ортогональные ему коэффициенты, проверяя их на совместимость, и в случае совместимости включает элемент s в множество $\boldsymbol{u}$ без дальнейшей проверки:

$$\begin{aligned}
&\boldsymbol{e} := \boldsymbol{u} \cup \{s\}\ ;\\
&\boldsymbol{h}^0 := \boldsymbol{f}^0 - \boldsymbol{e}\ ;\\
&\boldsymbol{h}^1 := \boldsymbol{f}^1 - \boldsymbol{e}\ ;\\
&\boldsymbol{g} := \mathrm{S}_{\boldsymbol{v}}^{\vee}(\boldsymbol{h}^0\boldsymbol{f}^1 \vee \boldsymbol{h}^1\boldsymbol{f}^0)\ ;\\
&\boldsymbol{h}^0 := \mathrm{S}_{\boldsymbol{u}}^{\vee}(\boldsymbol{f}^0\boldsymbol{g})\ ;\\
&\boldsymbol{h}^1 := \mathrm{S}_{\boldsymbol{u}}^{\vee}(\boldsymbol{f}^1\boldsymbol{g})\ .
\end{aligned}$$

Если $\boldsymbol{h}^0\boldsymbol{h}^1 = \boldsymbol{0}$, то s включается в $\boldsymbol{u}$, что реализуется операцией $\boldsymbol{u} := \boldsymbol{e}$. Так находится множество $\boldsymbol{u}$ и, следовательно, искомое разбиение $\boldsymbol{u}/\boldsymbol{v}$ в целом.

4. Экспериментальная система

Для оценки эффективности разработанной эвристической программы декомпозиции булевых функций и определения границ ее практического применения разработана экспериментальная система генерирования случайных примеров и их решения.

Генерирование примеров управляется следующими входными параметрами: n – число аргументов генерируемой функции $f(\boldsymbol{x})$; p и q – числа переменных в множествах $\boldsymbol{u}$ и $\boldsymbol{v}$ разбиения $\boldsymbol{u}\,/\,\boldsymbol{v}$ на множестве переменных $\boldsymbol{x}$; r – степень неопределенности функции $f(\boldsymbol{x})$; g – номер примера (порядковый номер элемента квазислучайной последовательности, используемый при генерировании примера). Определяется также тип разбиения: строгое – D (disjunctive) или нестрогое – ND (non-disjunctive).

Результаты решения примеров фиксируются значениями выходных параметров: Nt – число рассмотренных триад при поиске пригодной; Tc, Tt, Tu, Tv и Tw – время (в секундах), затраченное соответственно на подготовку примера, на нахождение пригодной триады, на расширение множества $\boldsymbol{u}$, на расширение множества $\boldsymbol{v}$ и на решение примера в целом (исключая Tc). Кроме того, оценивается качество решения в целом – сообщается, совпадают ли найденные разбиение $\boldsymbol{u}\,/\,\boldsymbol{v}$ и функции $h(\boldsymbol{u}, \boldsymbol{w})$ и $g(x, \boldsymbol{w}, \boldsymbol{v})$ с заданными.

Для облегчения процесса экспериментального исследования программы в систему введен «режим разреза», при котором генерируется и решается серия примеров с совпадающими значениями всех параметров, кроме некоторого выбранного параметра, для которого задаются разность между соседними значениями и их число.

5. Экспериментальные оценки эффективности программы и границ ее практического применения

Серия проведенных экспериментов (на компьютере Pentium IV, 2.8 GHz) показывает, что описанная эвристическая программа функционирует достаточно надежно, находя правильные решения, если $8 < n < 28$ и $r < 30$. Ошибки могут возникать за пределами этого диапазона. Ниже приводятся результаты решения некоторых конкретных примеров.

Оценка верхней границы по числу переменных n. Приведем результаты эксперимента над серией примеров, в которых число переменных n меняется с 22 по 28, а значения остальных параметров фиксированы:

Type	*n*	*p*	*q*	*r*	*g* |	Nt	Tc	Tt	Tv	Tu	Tw	Result
D	22	11	11	20	1	1	0,34	0,16	1,56	0,00	2,08	OK
D	23	12	11	20	1	1	0,69	0,33	3,33	0,00	4,45	OK
D	24	12	12	20	1	5	1,47	1,90	7,20	0,00	10,79	OK
D	25	13	12	20	1	2	3,12	1,99	15,19	0,00	20,70	OK
D	26	13	13	20	1	2	6,18	4,04	34,07	0,00	45,20	OK
D	27	14	13	20	1	2	12,52	8,43	74,24	0,00	99,01	OK
D	28	14	14	20	1	2	24,97	31,22	1938,61	0,00	2096,32	OK

Результаты эксперимента положительны (OK – найденное разбиение совпадает с исходным) и показывают, что при данных параметрах программа находит решение довольно быстро (в пределах минуты), кроме случая $n = 28$, когда длина вектора $\boldsymbol{f}$ становится слишком большой для используемой оперативной памяти, что приводит к резкому увеличению времени Tv. Видно также, что при $n < 28$ время увеличивается в два раза при каждом увеличении числа переменных n на единицу. Заметим, что Tu = 0, поскольку рассматривается пример задачи со строгим разбиением.

Оценка времени поиска пригодной триады. Результаты предыдущего эксперимента показывают, что при больших p и q это время (Tt) относительно мало. Положение меняется при малых значениях этих параметров, когда сильно возрастает число Nt триад, просматриваемых в поисках пригодной, и как следствие растет время решения задачи в целом, которое почти полностью тратится на этот поиск:

Type	*n*	*p*	*q*	*r*	*g* |	Nt	Tc	Tt	Tv	Tu	Tw	Result
ND	14	2	1	0	1	491	0,00	0,04	0,00	0,00	0,04	OK
ND	15	2	1	0	1	603	0,00	0,09	0,00	0,00	0,09	OK
ND	16	2	1	0	1	1250	0,01	0,36	0,01	0,00	0,37	OK
ND	17	2	1	0	1	1618	0,00	0,93	0,00	0,02	0,95	OK
ND	18	2	1	0	1	2055	0,02	2,37	0,01	0,05	2,43	OK
ND	19	2	1	0	1	2978	0,04	9,82	0,05	0,11	9,98	OK
……												
ND	27	2	2	0	1	2649	10,28	6812,28	48,64	95,09	6956,09	OK

Заметим, что величина Nt характеризуется большой дисперсией. Например, для серии 12 случайных примеров при фиксированных значениях параметров $(n, p, q, r) = (16, 2, 1, 0)$ получены следующие результаты:

Nt =	1250	596	2799	498	2315	1598	4834	3797	2362	1637	3990	700
Tt =	0,36	0,18	0,80	0,14	0,66	0,46	1,39	1,08	0,67	0,47	1,15	0,21

Оценка верхней границы по степени неопределенности r. С ростом степени неопределенности булевой функции растет вероятность получения ошибочного решения при ее декомпозиции. Был проведен эксперимент над серией случайных булевых функций, полученных в результате композиции по строгому разбиению с равными (по возможности) значениями параметров p и q. Определялась максимальная степень неопределенности r_{max} функции от n переменных, при которой находилось правильное решение. Результаты показаны ниже:

n =	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27
r_{max} =	0	1	6	8	14	19	19	21	22	23	25	25	26	26	28	27	29	29	29	30	30	30

С целью уточнения этих данных был проведен эксперимент над десятью случайными примерами с параметрами (Type, *n, p, q*) = (D, 24, 12, 12). При $r = 28$ все десять примеров были решены правильно, при $r = 29$ правильные решения были получены лишь для семи примеров, а при $r = 30$ все решения оказались ложными.

Оценка нижней границы по числу переменных n. При малом числе переменных многие из пригодных триад не являются истинными следами разбиения, как это вытекает из следующих экспериментальных данных, полученных при строгом разбиении с равными (по возможности) значениями параметров p и q и значениями параметров $n = 5, 6, \ldots, 16$ и $r = 15$:

Число переменных n	5	6	7	8	9	10	11	12	13	14	15	16
Общее число триад	30	60	105	168	252	360	495	660	858	1092	1365	1680
Число пригодных триад	28	44	80	78	54	90	77	90	126	147	196	224
Число следов разбиения	6	9	18	24	40	50	75	90	126	147	196	224

Ниже показано, насколько часто это приводит к ложному решению. В проводимых экспериментах решались серии из 100 случайных примеров с фиксированными значениями параметров Type, n, p, q, r и для каждой серии подсчитывалось число ложных решений Falsh:

Type	n	p	q	r	Falsh	Type	n	p	q	r	Falsh
D	4	2	2	0	62	ND	4	2	2	0	62
D	5	3	2	0	76	ND	5	2	2	0	82
D	6	3	3	0	64	ND	6	2	2	0	55
D	7	4	3	0	50	ND	7	3	2	0	21
D	8	4	4	0	35	ND	8	3	3	0	6
D	9	5	4	0	6	ND	9	3	3	0	0
D	10	5	4	0	0	ND	10	4	3	0	0

О нахождении функций $h(\boldsymbol{u}, \boldsymbol{w})$ *и* $g(x, \boldsymbol{w}, \boldsymbol{v})$. Главной целью декомпозиции булевой функции является нахождение пригодного разбиения $\boldsymbol{u}/\boldsymbol{v}$, позволяющего сократить число входных полюсов в блоках композиции $g(h(\boldsymbol{u}, \boldsymbol{w}), \boldsymbol{w}, \boldsymbol{v})$. Некоторый интерес представляет нахождение самих функций $h(\boldsymbol{u}, \boldsymbol{w})$ и $g(x, \boldsymbol{w}, \boldsymbol{v})$. Очевидно, что после получения этих функций однозначно определяется и разбиение $\boldsymbol{u}/\boldsymbol{v}$. Однако из нахождения разбиения не следует, что находятся и данные функции.

Это подтверждается экспериментальными данными при решении задачи декомпозиции на сотне случайных примеров с параметрами (Type, n, p, q) = (D, 10, 5, 5) и r, принимающими значения от 0 до 26 включительно. Через Nr обозначено число угаданных разбиений, через Nf – число угаданных функций:

r	0	1	…	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26
Nr	100	98	…	98	98	98	97	95	92	86	75	63	47	28	16	13	10	6	6	3	1	0
Nf	100	98	…	98	97	97	96	94	88	82	64	52	33	15	7	4	2	1	1	0	0	0

Заключение

Программно реализован и экспериментально исследован оригинальный эвристический алгоритм последовательной двухблочной декомпозиции частичных булевых функций от n переменных по нестрогому разбиению на множестве аргументов. Показано, что реализующая алгоритм программа функционирует достаточно надежно, находя правильные решения, если $8 < n < 28$ и функция определена не менее чем на 29/32 части булева пространства. Приводятся результаты более детального исследования программы на границах этого диапазона и за его пределами.

Работа была поддержана Белорусским республиканским фондом фундаментальных исследований (проект Ф07МС-034).

Список литературы

1. Закревский, А.Д. Последовательная декомпозиция булевой функции – поиск подходящего разбиения на множестве аргументов / А.Д. Закревский // Доклады НАН Беларуси. – 2007. – Т. 51, № 1. – С. 7–11.

2. Zakrevskij, A.D. Parallel operations over neighbors in Boolean space / A.D. Zakrevskij // Proceedings of the Sixth International Conference CAD DD-07. – Minsk, 2007. – Vol. 2. – P. 6–13.

3. Закревский, А.Д. Реализация случайных событий с заданной вероятностью / А.Д. Закревский // Труды СФТИ. – 1965. – Вып. 47. – С. 56–59.

4. Закревский, А.Д. Декомпозиция частичных булевых функций – поиск подходящего разбиения / А.Д. Закревский // Информатика. – 2007. – № 2. – С. 45–52.

Поступила 11.04.08

Объединенный институт проблем
информатики НАН Беларуси,
Минск, Сурганова, 6
e-mail: zakr@newman.bas-net.by

A.D. Zakrevskij, N.R. Toropov

IMPLEMENTATION OF A HEURISTIC METHOD FOR DECOMPOSITION OF BOOLEAN FUNCTIONS

An original heuristic algorithm of successive two-block decomposition of partial Boolean functions is suggested. The key combinatorial task investigated in the paper is the finding suitable partition on the set of arguments that is a partition on which the function is separable. It is shown that the search of suitable partition is substantially accelerated by preliminary detection of its traces. The efficiency of the algorithm is examined in the framework of an experimental system. The limits of its practical application are determined.