ИНФОРМАТИКА

2006 **январь-март** **№ 1**

УДК 519.718

П.В. Леончик

МИНИМИЗАЦИЯ СИСТЕМ БУЛЕВЫХ ФУНКЦИЙ В КЛАССЕ ДИЗЪЮНКТИВНЫХ НОРМАЛЬНЫХ ФОРМ

Предлагается программа минимизации систем полностью определенных булевых функций в классе ДНФ. Производится сравнение эффективности разработанного алгоритма, реализованного в программе Tie, и программы Espresso, которая широко используется в настоящее время [1–3]. Приводятся результаты экспериментально-статистических испытаний алгоритмов на псевдослучайных булевых функциях и стандартных примерах Benchmark.

Введение

Минимизация булевых функций является классической задачей, которая широко известна [1–6], однако до сих пор не утратила своей актуальности проблема повышения эффективности алгоритмов минимизации булевых функций как в отношении сокращения времени поиска оптимального решения, так и улучшения качества этого решения. Дело в том, что минимизация систем булевых функций в классе дизъюнктивных нормальных форм (ДНФ) по-прежнему является одним из этапов технологически независимой оптимизации при синтезе структур из библиотечных вентилей (элементов), реализуемых в полузаказных сверхбольших интегральных схемах (СБИС) [7]. При синтезе структур программируемых логических матриц (ПЛМ), входящих в состав заказных СБИС, совместная минимизация систем булевых функций позволяет решать задачу сокращения площади ПЛМ [8].

В данной статье предлагается алгоритм совместной минимизации систем полностью определенных булевых функций в классе ДНФ, который реализован в программе Tie. Приводятся результаты экспериментально-статистических испытаний алгоритма на псевдослучайных системах булевых функций и стандартных примерах Benchmark. Производится сравнение эффективности программы Tie с программой Espresso [1–3].

1. Минимизация булевых функций в классе ДНФ

Задача минимизации системы $F = \{f^1(x_1, x_2, ..., x_n), ..., f^m(x_1, x_2, ..., x_n)\}$ булевых функций в классе ДНФ заключается в поиске такой системы ДНФ для заданной системы булевых функций, содержащей минимальное число конъюнкций, на которых заданы ДНФ всех функций системы. Такую систему ДНФ будем называть *кратчайшей*. При решении этой задачи существенную роль играют понятия импликанты и простой импликанты [4].

Импликантой булевой функции $f(x_1, x_2, ..., x_n)$ называется такая элементарная конъюнкция K литералов $x_i, \bar{x}_i$ булевых переменных x_i, если на любом наборе значений переменных $x_1, x_2, ..., x_n$, на котором K равна 1, значение функции $f(x_1, x_2, ..., x_n)$ также равно 1.

Простой импликантой называется такая импликанта, которая перестает быть таковой при удалении из нее любого символа.

Понятие простой импликанты одной функции обобщается для случая понятия простой импликанты системы функций. Если простая импликанта k является импликантой для некоторого множества функций системы, то это не означает, что k будет являться простой импликантой для каждой функции в отдельности. Элементарная конъюнкция k будет *простой импликантой для системы функций* F тогда и только тогда, когда выполняются следующие условия:

– найдется такая подсистема функций $F^* \subseteq F$, $F^* \neq \varnothing$, что k является импликантой для каждой функции подсистемы F^*;

– k не будет импликантой хотя бы для одной функции подсистемы F^{**}, $F^{**} = F^{*} \cup f^{*}$, где $f^{*} \in F \setminus F^{*}$;

– k не будет импликантой хотя бы для одной функции подсистемы F^{*}, если из k удалить любой символ.

Например, для системы функций

$x_0x_1x_2x_3$	f_1f_2
0 0 1 0	1 1
0 0 1 1	0 1
0 1 1 0	0 1
0 1 1 1	0 1
1 0 1 0	1 0

существуют импликанты *a*, *b*, *c*:

	$x_0x_1x_2x_3$	f_1f_2
a	0 0 1 0	1 1
b	– 0 1 0	1 0
c	0 – 1 –	0 1 .

Очевидно, что импликанта *a* не является простой импликантой для функций f_1 и f_2, если каждую функцию рассматривать отдельно, но она является простой импликантой для системы функций f_1, f_2.

Классические методы решения поставленной задачи [4, 5] основаны на предварительном получении множества всех простых импликант заданной системы булевых функций с последующим выделением из него некоторого подмножества, которое и будет представлять искомое решение. Нахождение кратчайшей системы ДНФ можно свести к поиску безызбыточных (тупиковых) систем ДНФ, из которых выбирается кратчайшая система ДНФ. *Безызбыточной* или *тупиковой ДНФ* называется такая дизъюнкция простых импликант, если она представляет данную функцию и не содержит импликант, покрываемых другими импликантами. Нахождение кратчайшей системы ДНФ может привести к слишком громоздким вычислениям, поэтому в разработанном алгоритме осуществляется поиск меньшей по числу конъюнкций тупиковой системы ДНФ среди найденных решений.

Все недостающие определения из области минимизации булевых функций в классе ДНФ можно найти в работах [4, 5].

2. Алгоритм минимизации

Предполагается, что функции исходной системы заданы в виде совершенных ДНФ, поэтому если система функций задана в другом виде, то первоначально ее надо привести к системе совершенных ДНФ (СДНФ). Алгоритм минимизации системы полностью определенных булевых функций состоит из трех этапов:

1. Поиск всех простых импликант исходной системы СДНФ булевых функций.
2. Построение матрицы покрытия и ее сокращение.
3. Поиск тупиковых систем ДНФ.

На *первом этапе* для получения всех простых импликант исходной системы СДНФ булевых функций $f_i(x_1, x_2, ..., x_n)$ разобьем все переменные исходной системы на две части:

$$x_1, x_2, ..., x_k \; ; \;\; x_{k+1}, x_{k+2}, ..., x_n \, ,$$

где $k = \dfrac{n}{2}$, если n четно, и $k = \dfrac{n+1}{2}$, если n нечетно.

Сгруппируем все исходные наборы, соответствующие полным элементарным конъюнкциям ДНФ исходной системы функций, по классам $C_0^{\bar{x}_1,\bar{x}_2,\ldots,\bar{x}_k}, C_1^{\bar{x}_1,\bar{x}_2,\ldots,x_k},\ldots,C_{2^k}^{x_1,x_2,\ldots,x_k}$ так, что в каждый класс войдут только элементарные конъюнкции, для которых значения одноименных переменных $x_1,x_2,...,x_k$ совпадают. Назовем переменные $x_1,x_2,...,x_k$ *базисом* классов, а переменные $x_{k+1},x_{k+2},...,x_n$ *внутренними переменными* классов.

Каждый класс можно рассматривать как систему булевых функций $f_i(x_{k+1},x_{k+2},...,x_n)$, $i=1,\ldots,m$, для которой, применив алгоритм Квайна – МакКласки [5], получим все импликанты внутри образованных классов.

Далее производится поиск импликант между классами. Для межклассового поиска новых импликант следует рассматривать только классы, отличающиеся на одну переменную из базиса классов, тем самым сокращая перебор возможных вариантов поиска. Например, при $n=5$, $k=3$ ищем новые импликанты из классов C_0^{000} и C_1^{001}, но не ищем из классов C_0^{000} и C_3^{011}. Операция склеивания импликант между классами представляет собой операцию *обобщенного склеивания* импликант по методу Блейка – Порецкого [5], при этом переменные $x_1,x_2,...,x_k$ из базиса классов образуют новый класс. Следует отметить, что внутренние переменные классов $x_{k+1},x_{k+2},...,x_n$ не должны находиться в отношении *ортогональности*, так как переменные, по которым смежны импликанты из разных классов, находятся среди переменных $x_1,x_2,...,x_k$ из базиса классов. Операция обобщенного склеивания производится до тех пор, пока нельзя будет создать ни одного нового класса.

После поиска импликант между классами и удаления поглощенных импликант получим все простые импликанты исходной системы булевых функций.

Проиллюстрируем метод поиска простых импликант на примере системы функций f_1,f_2,f_3 от $n=4$ переменных x_0,x_1,x_2,x_3 (табл. 1). Первоначально, для приведения данной системы функций к СДНФ, исключим из рассмотрения исходный набор с номером 11, так как все значения функций на данном наборе равны 0.

Таблица 1

Таблица истинности системы функций

Номер исходного набора	Класс	Исходная система функций	
		$x_0x_1x_2x_3$	$f_1f_2f_3$
0 *	C_0^{00}	0 0 0 0	0 0 1
1		0 0 0 1	0 1 1
2		0 0 1 0	1 1 0
3 *		0 0 1 1	0 1 0
4 *	C_1^{01}	0 1 0 0	1 0 0
5 *		0 1 0 1	1 1 0
6		0 1 1 0	0 1 1
7 *		0 1 1 1	1 1 0
8 *	C_2^{10}	1 0 0 0	1 0 1
9		1 0 0 1	1 1 1
10 *		1 0 1 0	1 0 1
11		1 0 1 1	0 0 0
12	C_3^{11}	1 1 0 0	0 1 1
13 *		1 1 0 1	0 0 1
14 *		1 1 1 0	0 0 1
15		1 1 1 1	1 0 1

Разобьем все исходные наборы, соответствующие полным элементарным конъюнкциям, входящим в СДНФ функций, на классы C_0^{00}, C_1^{01}, C_2^{10}, C_3^{11}, как показано в табл. 1. Внутри каждого класса переменные x_0 и x_1 не изменяют свои значения, т. е. они не принимают участия в формировании импликант внутри классов. Следовательно, каждый класс исходной системы функций однозначно определяется значениями переменных x_0 и x_1 и его можно рассматривать как самостоятельную систему функций от $n-k$ переменных. Например, для класса C_0^{00} самостоятельной системой функций является следующая:

x_2x_3	$f_1f_2f_3$
0 0	0 0 1
0 1	0 1 1
1 0	1 1 0
1 1	0 1 0 .

Таким образом, можно осуществлять поиск импликант по методу Квайна – МакКласки для каждого класса отдельно. Символом * помечены импликанты, которые принимали участие в склеивании с другими импликантами и при этом оказались поглощенными. Результат поиска импликант внутри классов представлен в табл. 2. Для импликант, являющихся продуктами склеивания, справа указаны номера склеенных импликант.

Таблица 2

Поиск импликант внутри классов

Номер импликанты	Класс	$x_0x_1x_2x_3$	$f_1f_2f_3$	Пары склеенных импликант
1^	C_0^{00}	0 0 0 1	0 1 1	
2		0 0 1 0	1 1 0	
16^		0 0 0 –	0 0 1	0, 1
17^		0 0 – 1	0 1 0	1, 3
18^		0 0 1 –	0 1 0	2, 3
6	C_1^{01}	0 1 1 0	0 1 1	
19		0 1 0 –	1 0 0	4, 5
20		0 1 – 1	1 1 0	5, 7
21^		0 1 1 –	0 1 0	6, 7
9	C_2^{10}	1 0 0 1	1 1 1	
22		1 0 0 –	1 0 1	8, 9
23		1 0 – 0	1 0 1	8, 10
12	C_3^{11}	1 1 0 0	0 1 1	
15		1 1 1 1	1 0 1	
24*		1 1 0 –	0 0 1	12, 13
25*		1 1 – 0	0 0 1	12, 14
26*		1 1 – 1	0 0 1	13, 15
27*		1 1 1 –	0 0 1	14, 15
28		1 1 – –	0 0 1	24, 27

Следующим шагом первого этапа работы алгоритма является поиск новых импликант за счет импликант, находящихся в разных классах. Операция склеивания импликант между классами представляет собой операцию обобщенного склеивания импликант, при этом переменные x_0, x_1 образуют новый класс. Например, при обобщенном склеивании импликант из классов C_0^{00} и C_1^{10} будем получать импликанты, которые входят в класс C_5^{-0}. Например, склеивая импликанты с номерами 2 и 23, получим импликанту с номером 35:

	$x_0x_1x_2x_3$	$f_1f_2f_3$
C_0^{00}	0 0 1 0	1 1 0
C_2^{10}	1 0 – 0	1 0 1
C_5^{-0}	– 0 1 0	1 0 0.

Переменная, по которой смежны вышеприведенные импликанты, находится среди переменных из базиса классов x_0x_1, а внутренние переменные классов x_2x_3 не ортогональны. Образованная импликанта системы функций является импликантой только функции f_1, так как импликанты с номерами 2 и 23 являются импликантами только функции f_1.

Операция обобщенного склеивания производится до тех пор, пока нельзя будет создать ни одного нового класса. В табл. 3 представлен результат обобщенного склеивания импликант с появлением новых классов C_4^{0-}, C_5^{-0}, C_6^{1-}, C_7^{-1}. Символом ^ помечены те импликанты, которые покрываются другими импликантами.

Таблица 3

Поиск импликант между классами

Номер импликанты	Класс	$x_0x_1x_2x_3$	$f_1f_2f_3$	Пары склеенных импликант
29^	C_4^{0-}	0 – 0 1	0 1 0	1, 20
30^		0 – 1 0	0 1 0	2, 6
31		0 – – 1	0 1 0	17, 20
32^		0 – 1 1	0 1 0	17, 21
33		0 – 1 –	0 1 0	18, 21
34	C_5^{-0}	– 0 0 1	0 1 1	1, 9
35		– 0 1 0	1 0 0	2, 23
36		– 0 0 –	0 0 1	16, 22
37^		– 0 0 0	0 0 1	16, 23
38	C_6^{-1}	– 1 1 0	0 0 1	6, 28
39		– 1 1 1	1 0 0	14, 20
40^	C_7^{1-}	1 – 0 1	0 0 1	9, 28
41^		1 – 0 0	0 0 1	12, 22
42		1 – 0 –	0 0 1	22, 28
43		1 – – 0	0 0 1	23, 28

По завершении поиска все непокрытые импликанты из табл. 2 и 3 (не отмеченные символами * и ^) являются простыми импликантами.

Построение булевой матрицы A происходит на *втором этапе* алгоритма, где строки соответствуют тем наборам значений аргументов, на которых функции f_1,f_2,f_3 принимают значение 1, а столбцам соответствуют простые импликанты системы функций. Если на некотором наборе значений аргументов несколько функций принимают значение 1, то такому набору будут соответствовать столько строк булевой матрицы A, сколько функций принимают значение 1 на данном наборе. Например, для набора

$x_0x_1x_2x_3$	$f_1f_2f_3$
0 0 0 1	0 1 1

в матрице A будут присутствовать строки

0 0 0 1	0 1 0
0 0 0 1	0 0 1.

В *i*-м столбце *j*-й строки ставится 1, если *i*-я простая импликанта покрывает *j*-ю полную элементарную конъюнкцию системы СДНФ. В таком случае будем говорить, что *i*-й столбец покрывает *j*-ю строку.

Сокращаем матрицу покрытия по следующим правилам:

1. Если в некоторой строке присутствует только одна единица, то столбец, в котором находится эта единица, обязательно должен быть включен в покрытие. Такой столбец удаляется из матрицы со всеми покрываемыми им строками, в которых он содержит единицы.

2. Если столбец k имеет единицы везде, где имеет единицы столбец d, то столбец k удаляется из матрицы.

3. Если строка k имеет единицы везде, где имеет единицы строка d, то строка d удаляется из матрицы.

На *третьем этапе* производим поиск тупиковых систем ДНФ, который представляет собой задачу нахождения кратчайшего столбцового покрытия матрицы A. Данная задача имеет многочисленные применения [4], представляет самостоятельный интерес и поэтому в данной статье не рассматривается.

3. Экпериментально-статистические испытания алгоритма

Сравнение эффективности программ Espresso и Tie проводилось на ПК с процессором Intel Pentium 4, 1,7 ГГц, 1024 Мб оперативной памяти. Программа Tie создана на языке Си и протестирована на операционной системе Windows 98/2000/Xp. Объем исполняемого файла около 1 Мб.

Программа Tie, реализующая вышеописанный алгоритм, была подвергнута многократным испытаниям на известных примерах серии Benchmark [9] и на псевдослучайных булевых функциях, полученных с помошью параметрически управляемых генераторов [10].

Для испытаний программы Tie использовалось несколько модификаций программы Espresso:

Espresso – программа Espresso без ключей;

Espresso-Qm – программа Espresso с ключем –Dqm, которая позволяет найти лучшее приближенное решение, чем программа без ключей;

Espresso-Exact – программа Espresso с ключем –Dexact, которая позволяет найти точное решение.

В табл. 4 приводятся результаты минимизации функций из примеров Benchmark программами Espresso и Tie. При этом используются следующие обозначения: n – число переменных; m – число функций в системе; j – число конъюнкций в исходном представлении функций; k – число конъюнкций, полученных в результате работы программ Espresso и Tie для соответствующих примеров; k_{min} – число конъюнкций в точном решении; t – время (с), затраченное на поиск решения.

Для 11 из 30 примеров серии Benchmark (табл. 4) программа Tie находит лучшее решение, чем программа Espresso, в большинстве случаев затрачивая на его поиск меньшее время. На других 19 примерах из серии Benchmark были получены одинаковые решения с программой Espresso, соответствующие точному решению.

Таблица 4

Сравнение эффективности программ Tie и Espresso на стандартных примерах серии Benchmark

Пример	n	m	j	k_{min}	Espresso		Tie	
					k	t, с	k	t, с
max512	9	6	512	133	145	0,12	134	0,388
max1024	10	6	1024	261	274	0,57	263	0,239
ex5	8	63	256	67	74	0,09	66	1,164
z5xp1	7	10	128	63	65	0,03	63	0,014
z9sym	9	1	420	84	86	0,07	84	0,198
addm4	9	8	480	189	200	0,12	189	0,167
dist	8	5	255	120	123	0,06	120	0,026
mlp4	8	8	225	121	128	0,06	125	0,035
pope.rom	6	48	64	59	62	0,11	59	0,020
max128	7	24	128	78	89	0,09	78	0,028
sqr6	6	12	63	47	49	0,01	48	0,005

Эффективность предложенного метода поиска всех простых импликант показана в табл. 5, где приводятся результаты сравнения времени поиска всех простых импликант программами Tie и Espresso-Exact на примерах серии Benchmark и псевдослучайных булевых функций. В столбце n_{imp} дается число всех простых импликант.

Таблица 5

Сравнение скорости поиска всех простых импликант программами Tie и Espresso-Exact

Пример	n	m	j	n_{imp}	Espresso-Exact	Tie
					t, с	t, с
ex5	8	63	256	2532	1,20	0,065
f12_4_60	12	4	4096	21470	23,90	0,901
f12_16_60	12	4	4096	98078	1129,79	7,658
f14_10_50	14	10	16384	215089	3189,0	22,617
f16_8_50	16	8	65536	819668	30498,0	185,62

Следует обратить особое внимание на некоторые результаты из серии примеров Benchmark (табл. 6).

Таблица 6

Примеры Benchmark, для которых программой Tie было найдено покрытие, меньшее, чем k_{best}

Пример	n	m	j	k_{best}	Espresso-Exact		Espresso-Qm		Tie	
					k	t, с	k	t, с	k	t, с
max1024	10	6	1024	261	259	474582,0	264	1,01	260	33,69
ex5	8	63	256	67	65	818504,0	68	8,79	65	25,07

Особенность приведенных выше примеров заключается в том, что с помощью программы Espresso-Exact (точного алгоритма минимизации) удалось найти точные решения для примеров *max1024* и *ex5*, хотя для них в Benchmark указаны другие лучшие решения. Программа Tie для примера *ex5* нашла точное решение значительно быстрее, чем программа Espresso-Exact. Для примера *max1024* найденное решение отличается от точного лишь на одну конъюнкцию, но при этом на его поиск затрачено значительно меньше времени.

В табл. 7 представлены результаты испытаний программ Espresso и Tie на псевдослучайных системах булевых функций. Генератор, создающий систему СДНФ, управляется следующими параметрами: n – число переменных, m – число функций в системе, p – процентное соотношение количества единиц в значениях данной системы булевых функций. Далее используются следующие обозначения: j – число конъюнкций в исходном представлении функций; k – число конъюнкций, полученных в результате работы программ Espresso и Tie; t – время (с), затраченное на поиск решения.

Для создания псевдослучайных систем булевых функций использовался генератор Парка – Миллера:

$$R_{j+1} = aR_j (\mathrm{mod}\, m),\ a = 7^5 = 16807,\ m = 2^{31} - 1.$$

Здесь следует иметь в виду, что для языков высокого уровня применяется следующее значение:

$$m = a \times q + r,\ q = 12773,\ r = 2836.$$

Из табл. 7 видно, что с помощью программы Tie получают решения, лучшие, чем те, которые получают с помощью программы Espresso, при этом быстродействие программы Tie часто превосходит быстродействие программы Espresso в десятки раз, что особенно ощутимо на функциях от 14 переменных и более. В табл. 7 представлены примеры с числом переменных $n < 18$, потому что программа Espresso не смогла найти решение ни для одной системы функций $f(x_1, x_2, ..., x_n)$, где $n > 17$.

Таблица 7

Сравнение эффективности программ Tie и Espresso на псевдослучайных системах булевых функций

Пример	*n*	*m*	*j*	*p*	Espresso		Tie	
					k	*t*, с	*k*	*t*, с
f9_8_50	9	8	512	50	453	0,78	431	0,265
f12_8_50	12	8	4096	50	3366	3134,0	3198	14,446
f16_8_50	16	8	65536	50	50844	18365,0	48894	399,62
f12_4_25	12	4	4096	25	1834	4,00	1771	0,79
f12_4_50	12	4	4096	50	2100	9,65	1970	3,11
f12_4_75	12	4	4096	75	1633	9,26	1559	12,04
f14_4_50	14	4	16384	50	7954	202,26	7385	21,784
f14_10_50	14	10	16384	50	14689	778,62	13939	253,55
f14_16_50	14	16	16384	50	16299	668,96	16196	99,58
f10_8_60	10	8	1024	60	833	1,43	811	1,355
f11_11_30	11	11	2048	30	1949	4,48	1921	0,880
f11_48_8	11	48	2048	8	2027	4,07	2027	0,078
f12_4_60	12	4	4096	60	2024	11,25	1917	2,886
f12_90_5	12	90	4096	5	4085	32,10	4085	0,218
f14_10_75	14	10	16384	75	11449	909,85	10784	596,38
f15_12_40	15	12	32768	40	31594	4317,0	30814	99,51
f17_1_50	17	1	131072	50	15721	2193,0	14869	242,87
f17_2_60	17	2	131072	60	29135	6805,0	26871	676,41
f18_6_25	18	6	262144	25	–	–	138597	189,48
f19_8_15	19	8	524288	15	–	–	282990	1588,0
f22_4_20	22	4	4194304	20	–	–	1351632	4538,0

Отличительной особенностью программы Tie от программы Espresso является то, что во время ее работы можно сохранять промежуточные результаты минимизации. Программа Espresso, найдя некоторое решение, завершает свою работу. В свою очередь, программа Tie, найдя некоторое решение, позволяет продолжить поиск и найти еще более качественное решение.

На рис. 1 показана динамика улучшения качества решения программой Tie с течением времени. Решение программы Espresso представлено одной точкой.

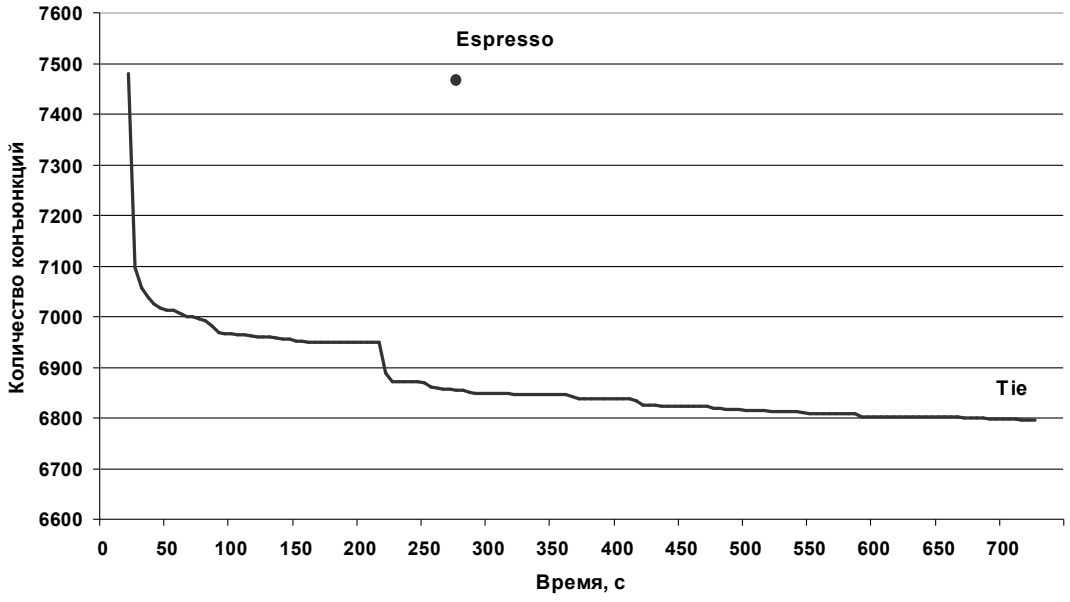

Рис. 1. График изменения качества решения с течением времени работы программ

Заключение

Программная реализация разработанного алгоритма и испытания на компьютере подтверждают высокую практическую эффективность алгоритма как на серии примеров Benchmark, так и на псевдослучайных системах булевых функций. Это особенно заметно, если исходная система булевых функций представлена в виде таблицы истинности или СДНФ. На основании проведенных испытаний можно сделать вывод, что разработанный алгоритм целесообразно применять, если исходная система булевых функций представлена в виде таблицы истинности или СДНФ, а число n входных переменных не превышает 22.

Программу Espresso свободно можно получить по адресам в Интернете:
http://www.csc.uvic.ca/~csc485c/espresso/espresso.exe,
http://www.ecse.rpi.edu/Courses/CStudio/Espresso/espresso.exe,
http://www1.cs. columbia.edu/~cs4861/sis/ .

Список литературы

1. Logic Minimization Algorithms for VLSI synthesis / R.K. Brayton, G.D. Hatchel, C.T. McMullen et al. – Boston: Kluwer Academic Publishers, 1984.

2. ESPRESSO-II: a New Logic Minimizer for PLA / R.K. Brayton, G.D. Hatchel, C.T. McMullen et al. // IEEE Pros. Custom Integrated Circuits Conf. – Rochester, 1984.

3. Rudel R., Sangiovanni Vincentelli A. Multiple-Valued Minimization for PLA Optimization // IEEE Trans. Computer-Aided Des. –1987. – Vol. CAD-6. – № 5. – P. 727–751.

4. Закревский А.Д. Логический синтез каскадных схем. – М.: Гл. ред. физ.-мат. лит., 1981.

5. Закревский А.Д., Поттосин Ю.В., Черемисинова Л.Д. Основы логического проектирования. Оптимизация в булевом пространстве. Кн. 2. – Мн.: ОИПИ НАН Беларуси, 2004.

6. Сапоженко А.А., Чухров И.П. Минимизация булевых функций в классе дизъюнктивных нормальных форм // Итоги науки и техники. – М., 1987. – Т. 25. – С. 68–116.

7. Бибило П.Н. Синтез логических схем с использованием языка VHDL. – М.: Солон-Р, 2002. – 384 с.

8. Система «Custom Logic» автоматизированного проектирования управляющей логики заказных цифровых СБИС / П.Н. Бибило, И.В. Василькова, С.Н. Кардаш и др. // Микроэлектроника. – 2003. – Т. 32. – № 5. – С. 379–398.

9. Yang S. Logic synthesis and optimization benchmarks user guide version 3.0. Technical Report. – Microelectronics Center of North Carolina, 1991.

10. Закревский А.Д., Торопов Н.Р. Генераторы псевдослучайных логико-комбинаторных объектов // Логическое проектирование: сб. науч. тр. – Мн.: Ин-т техн. кибернетики НАН Беларуси, 1999. – С. 49–63.

Поступила 22.11.05

Объединенный институт проблем
информатики НАН Беларуси,
Минск, Сурганова, 6
e-mail: pleonchik@gmail.com

P.V. Leonchik

MINIMIZATION OF BOOLEAN FUNCTION SYSTEMS IN THE CLASS OF DISJUNCTIVE NORMAL FORMS

The paper presents the DNF-minimization programme of systems of fully specified Boolean functions. The comparative analysis of the effectiveness of the developed algorithm realized in the programme Tie and the programme Espresso is presented. The results of statistical experimental testing of the algorithms on pseudorandom Boolean functions and on a wide set of Benchmark examples are given.