UNIVERSIDAD DE DEUSTO

Facultad de Informática

MÉTODO COMPUTACIONAL CAMP DEUSTO DE SIMPLIFICACIÓN DE FUNCIONES BOOLEANAS.

"DESARROLLO E IMPLEMENTACIÓN"

Javier GARCÍA ZUBÍA

BILBAO, FEBRERO 1996

ÍNDICE

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

RESUMEN

La presente tesis se encuadra, dentro del diseño de sistemas digitales, en el campo de la minimización de funciones booleanas, aportando un nuevo método de simplificación: CAMP DEUSTO.

El método CAMP DEUSTO es computacional, heurístico y directo. Se aportan un nuevo enfoque y algoritmo en la obtención de los implicados primos, así como un diferente criterio de discriminación para funciones cíclicas. CAMP DEUSTO es implementado y comparado con otros métodos representativos, resultando más rápido en la obtención de los implicados primos y más exacto en la obtención de la expresión mínima.

La implementación del algoritmo conforma un prototipo, la siguiente etapa de desarrollo será extenderlo a *mainframe.*, y así obtener un entorno avanzado comparable ESPRESSO y McBOOLE.

ABSTRACT

The main goal of this work is to develop a new simplification method for Booleans functions: CAMP DEUSTO.

A whole new philosophy in dealing with the problem of obtaining the prime implicants applied and a different "branch method" for discriminating cyclic functions is used.

The results obtained in the implementation of CAMP DEUSTO are more "near optimal" minimal expressions and have probed to be faster than other methods in obtaining the prime implicants.

CAMP DEUSTO has been implemented as a prototype. The next step in research will be the implementation of this method on a mainframe, obtaining and advanced environment similar to ESPRESSO and McBOOLE.

INTRODUCCIÓN

La tesis se encuadra, dentro del diseño de sistemas digitales, en el campo de la minimización de funciones booleanas, aportando un nuevo método de simplificación de funciones booleanas: CAMP DEUSTO.

El objetivo final de la presente tesis es desarrollar e implementar un entorno propio de minimización de funciones booleanas de carácter avanzado. Los resultados obtenidos conforman un prototipo cuya implementación en una plataforma *mainframe* dará lugar a un entorno de minimización equiparable a ESPRESSO (Universidad de Berkeley) y McBOOLE (Universidad de Montreal).

Históricamente, en los años cincuenta aparecen los primeros métodos no algebraicos de minimización de funciones booleanas, siendo a partir de los años sesenta cuando, para reducir costes, estas técnicas son utilizadas en la producción industrial de circuitos digitales, ya que la función así implementada es mínima. La aparición de ordenadores supone que los métodos desarrollados pueden ser implementados, lo que permite simplificar funciones de gran tamaño que no podían ser procesadas manualmente. Finalmente, y a partir de los años ochenta, comienzan a desarrollarse distintos métodos de simplificación que tienen como única finalidad y posibilidad el ser implementados en ordenadores; estos métodos no son susceptibles de ser aplicados manualmente, como ocurría con los primeros. La tesis se enmarca dentro de esta última tendencia de marcado carácter computacional; pero teóricamente, y frente a los métodos actuales, parte de una nueva perspectiva para desarrollar e implementar un nuevo método.

La tesis se divide en tres capítulos. El primero de ellos describe el estado del arte en el campo de la minimización de funciones mediante la

descripción y clasificación de los distintos métodos actuales. Como resultado se propone el desarrollo de un nuevo método avanzado, comparable a McBOOLE y ESPRESSO, computacional, directo y heurístico. Dicho método se denomina CAMP DEUSTO (*Computer Aided Minimization Procedure DEUSTO*).

Los métodos de obtención de los implicados primos pueden ser combinatorios o algebraicos. CAMP DEUSTO aporta un nuevo enfoque exploratorio, en conjunto más rápido que los anteriores. Por otra parte, CAMP DEUSTO desarrolla para funciones cíclicas un nuevo criterio directo de *'branch mode'*, de una gran optimidad en comparación con los criterios directos utilizados en los métodos actuales.

El método CAMP DEUSTO comprende dos partes:

- obtención de los implicados primos y
- obtención de la expresión mínima,

que son respectivamente el contenido de los capítulos segundo y tercero.

En el capítulo 2, además de desarrollar los algoritmos de obtención de los implicados primos, se analizan las características del CAMP DEUSTO frente a las que poseen los demás métodos en dicha obtención.

Las principales características de un método de obtención de los implicados primos son: rapidez, programación y su carácter computacional o manual. La velocidad de un algoritmo depende del número de minitérminos y del número de variables, siendo el factor predominante en los métodos actuales el número de minitérminos. Sin embargo, para CAMP DEUSTO el principal factor es el número de variables. En cuanto a la implementación, la programación de CAMP DEUSTO es inmediata, frente a la complejidad relativa de los otros métodos. Por otra parte, los métodos actuales pueden ser manuales respecto de la obtención de los implicados primos, mientras que CAMP DEUSTO es estrictamente computacional.

Para obtener los implicados primos, CAMP DEUSTO emplea la matriz MCVK (*Matriz Característica de Veitch-Karnaugh*) que una vez generada contiene ordenados todos los posibles implicados primos de

cualquier función de un determinado número de variables. Esta ordenación se basa en el nuevo concepto de rueda aportado en la tesis.

El capítulo 3 describe el método CAMP DEUSTO de obtención de la expresión mínima de una función. Para esta obtención, los métodos pueden seguir estrategias algebraicas, constructivas o discriminatorias según se basen en la expresión booleana de la función, únicamente en sus minitérminos o en sus implicados primos, respectivamente. CAMP DEUSTO sigue una estrategia discriminatoria.

Dentro de la estrategia discriminatoria, cuando la función está ciclada -funciones complejas- es necesario aplicar el criterio de discriminación o *'branch mode'*. Cada método se distingue por su criterio. Así, en CAMP DESUTO se desarrolla un nuevo criterio llamado DC (*Discriminación Comparativa*), directo y heurístico. En comparación con los otros métodos el CAMP DEUSTO es el más exacto (*more near-optimal*), muy particularmente en funciones cíclicas.

CAPÍTULO 1. PRINCIPALES MÉTODOS DE SIMPLIFICACIÓN DE FUNCIONES BOOLEANAS

1.1 Introducción

Este primer capítulo describe el estado del arte en el campo de la minimización de funciones booleanas, situando el contenido de la tesis y sus conclusiones en su marco de referencia.

Desde el punto de vista teórico la minimización de funciones está totalmente resuelta desde los años cincuenta; no así la resolución e implementación de los distintos métodos de simplificación. En este capítulo desarrollaremos y clasificaremos los principales métodos de minimización de funciones booleanas junto con el aportado en la tesis.

Ciertos métodos parten del conjunto de implicados primos asociados a la función a simplificar, eligiendo cuáles de ellos forman parte de la función simplificada: ***estrategia discriminatoria*** . Otros evitan los problemas de memoria y lentitud que acarrean los anteriores, y bien parten de la expresión booleana original de la función, a la que aplican determinadas reglas algebraicas para construir sobre ella la función simplificada: ***estrategia algebraica.*** O bien parten de sus minitérminos a los que aplican determinadas reglas constructivas: ***estrategia constructiva.***

Asimismo, los métodos de obtención de los implicados primos de una función podrán partir de sus minitérminos asociados y de ciertas tablas:

métodos combinatorios. O podrán partir de la expresión booleana original de la función y de ciertas reglas algebraicas: ***métodos algebraicos***.

El proceso de minimización es dinámico, y así en determinadas situaciones puede tener que decidirse por una entre varias opciones. En esta situación unos métodos deciden *a priori* y mediante un criterio heurístico qué única opción eligen: ***criterios de discriminación directa***. Mientras que otros exploran las posibilidades de todas las opciones y deciden *a posteriori* cuál es el más óptimo: ***criterios de discriminación bifurcativos***.

Los métodos basados en criterios directos son más rápidos, pero no son necesariamente exactos: su calidad depende del criterio. Por contra los basados en criterios bifurcativos son exactos, pero son muy lentos

La calidad de la implementación de los métodos desarrollados depende del software, y éste a su vez de la sencillez del método: métodos sencillos o clásicos y complejos o avanzados. A su vez, los métodos complejos necesitan ordenadores de tipo *mainframes*, mientras que los sencillos se basan en PC's. Por supuesto, los métodos avanzados son más rápidos y exactos que los clásicos.

El principal objetivo y resultado de la tesis es un nuevo método de simplificación de funciones booleanas denominado CAMP DEUSTO. Es un método de tipo selectivo basado en un criterio de discriminación directo.

En la obtención de los implicados primos es aportado un nuevo enfoque y método: ***método exploratorio***. Se basa en el concepto de ***rueda***, el cual conlleva un criterio novedoso de ordenación de los implicados primos.

En el desarrollo del método CAMP DEUSTO se ha considerado el siguiente marco de trabajo: funciones bivaluadas, de una sola salida (*single-output*) y fexpresadas en forma de suma de productos. Queda abierto la aplicación de este método al caso de funciones multivalentes (*multivalued*) y con múltiples salidas (*multiple output*).

1.2 Formulación del problema de obtención de la expresión mínima de una función booleana

El marco teórico del proceso simplificador de una función booleana es muy sencillo y está perfectamente descrito en la literatura clásica (Muroga, 1979), (Kohavi, 1978), (Dietmeyer, 1978), (McCluskey, 1986), (Brayton et al, 1984), etc. En este apartado definiremos los conceptos más importantes, remitiéndonos al apéndice A para aspectos específicos.

Toda función booleana se asocia con:

- Un conjunto de minitérminos, maxitérminos y condiciones libres.
- Una forma normal disyuntiva o conjuntiva.
- Un conjunto de implicados primos.
- Distintas sumas de productos o productos de sumas.
- Al menos una expresión mínima.

En la tesis siempre nos referiremos a la función booleana por su forma disyuntiva, es decir, como suma de productos.

Implicados primos

El conjunto de implicados primos relacionados con una función booleana es único. Para definir implicado es conveniente partir de la siguiente definición:

DEFINICIÓN 1. Una función $f(x_1, x_2, x_3, ..., x_n)$ **incluye** o **cubre** a otra función $g(x_1, x_2, x_3, ..., x_n)$ si para todas las combinaciones de valores de las variables en las que g=1 también es cierto que f=1. También se dice que f **implica** a g.

Basándonos en la relación de inclusión definamos implicado primo:

DEFINICIÓN 2. Un implicado primo de una funcion $f(x_1, x_2, x_3, ..., x_n)$ es un producto de literales $x_{i_1} \cdot x_{i_2} \cdots x_{i_m}, m \leq n$, el cual está incluido en f y tiene como propiedad que si cualquier literal es eliminado del producto

entonces el producto restante no está incluido en f. (x_i^* representa al literal x_i negado o no). (McCluskey, 1986: 206)

TEOREMA DE LOS IMPLICADOS PRIMOS. Una expresión mínima en forma de suma de productos siempre se conforma como una suma de implicados primos. (Quine, 1952)

Este teorema implica que una forma, no la única, de obtener la expresión mínima de una función pasa por obtener previamente su conjunto de implicados primos. Esta metodología es utilizada en la tesis.

Simplificación booleana

El proceso minimizador, según (Dietmeyer, 1978: 609), obtiene la expresión de una función booleana que es óptima según determinado criterio, expresión que no tiene porque ser única. Según los distintos autores:

> Matemáticamente, según (Althoen y Bumcrot, 1988: 99):
> *Una expresión disyuntiva de una función booleana es mínima si, no existe otra expresión disyuntiva con menor número de términos, ni otra con igual número de términos pero con menor número de literales (contando los repetidos).*
>
> Por su implementación, según (Muroga, 1979: 125):
> *Una expresión booleana es mínima si en su implementación se minimiza el número de puertas lógicas y el número de conexiones.*
>
> Por su forma, según (Brayton et al, 1984: 28):
> *Un conjunto C de cubos es mínimo si ningún cubo de C es cubierto por un conjunto de los otros cubos de C.*
>
> Por su coste, según (Kohavi, 1978: 75):
> *Una expresión booleana es mínima si su coste lo es. Coste como suma de literales y/o suma de términos.*

Por sus implicados primos, según (Muroga, 1979: 130):
Una disyunción de implicados primos es mínima si al eliminar cualquiera de ellos la disyunción restante no cubre a la función booleana.

Las anteriores definiciones son muy parecidas entre sí, pero la más adecuada a la metodología de la tesis es la última, por cuanto que explicita el uso del conjunto de implicados primos. Desde este punto de vista, simplificar una función consiste en discriminar entre sus implicados primos.

El marco teórico de simplificación es muy sencillo, ahora bien, las estrategias a seguir para simplificar una función booleana u obtener su conjunto de implicados primos pueden ser muy distintas. De ellas dependerá la calidad de cada método.

1.2.1 Estrategias y fases en el problema de simplificación

La simplificación de una función booleana puede orientarse de tres formas distintas -reflejadas en la figura 1.1- según sean los datos de entrada que utiliza el método:

- Partir del conjunto total de los implicados primos asociados a la función booleana a simplificar. ***Estrategia discriminatoria.***
- Partir de un subconjunto del total de implicados primos asociados a la función booleana a simplificar. ***Estrategia algebraica.***
- Partir del conjunto de minitérminos asociados a la función a simplificar. ***Estrategia constructiva.***

Las dos primeras estrategias -que pueden considerarse una- se basan en el teorema de los implicados primos de Quine y ***reducen*** mediante ***discriminación*** la ***expresión original*** -total o parcial- para encontrar la expresión mínima equivalente a la original.

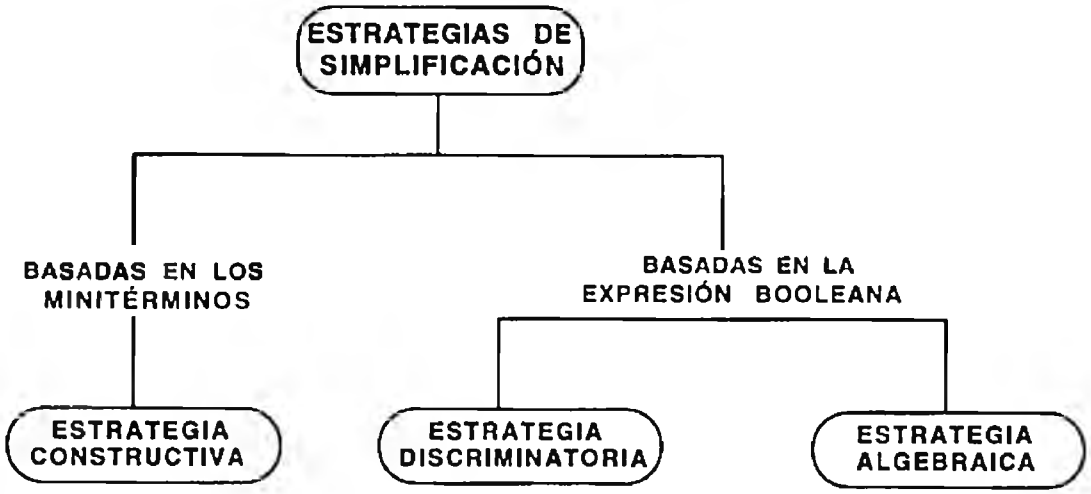


Figura 1.1 Clasificación de las estrategias de simplificación

La tercera estrategia *construye* mediante *determinadas reglas* y a partir de los *minitérminos* los distintos términos que componen la expresión mínima.

El nuevo método aportado en la tesis se encuadra dentro del primer grupo. Con el fin de establecer una comparación tanto teórica como práctica enunciaremos previamente las fases que completa el nuevo método:

1. Obtener el conjunto total de los implicados primos asociados a la función booleana a simplificar.
2. Discriminar entre los implicados primos cuáles de ellos deberán de formar parte de la expresión mínima.

De las tres estrategias sólo desarrollaremos completamente la discriminatoria, mientras que las dos restantes serán comentadas junto con los métodos que las utilizan. Aunque las tres serán observadas como criterio de clasificación en el aprtado 1.2.4.

1.2.2 Fundamentos en la obtención de los implicados primos

Toda función booleana está asociada a un conjunto de implicados primos que es único, su obtención es una labor determinista resuelta tanto

desde el punto de vista teórico como práctico. El nuevo método presentado enfoca esta obtención desde una distinta perspectiva.

Para obtener los implicados primos de una función los métodos parten de distintos datos de entrada y actúan sobre ellos de distinta manera, pudiendo ser clasificados en los siguientes grupos:

- Métodos combinatorios: A partir de los ***minitérminos, combinarlos*** para construir los implicados primos. Quine-McCluskey.
- Métodos algebraicos: A partir de una ***expresión booleana***, aplicarle ***operaciones algebraicas*** para obtener los implicados primos. Consenso Iterativo, McBoole y Espresso.
- Métodos exploratorios: A partir de ***todos los posibles implicados primos*** -matriz MCVK-, ***explorarlos*** frente a los minitérminos para decidir cuáles lo son de la función. CAMP DEUSTO presentado en la tesis.

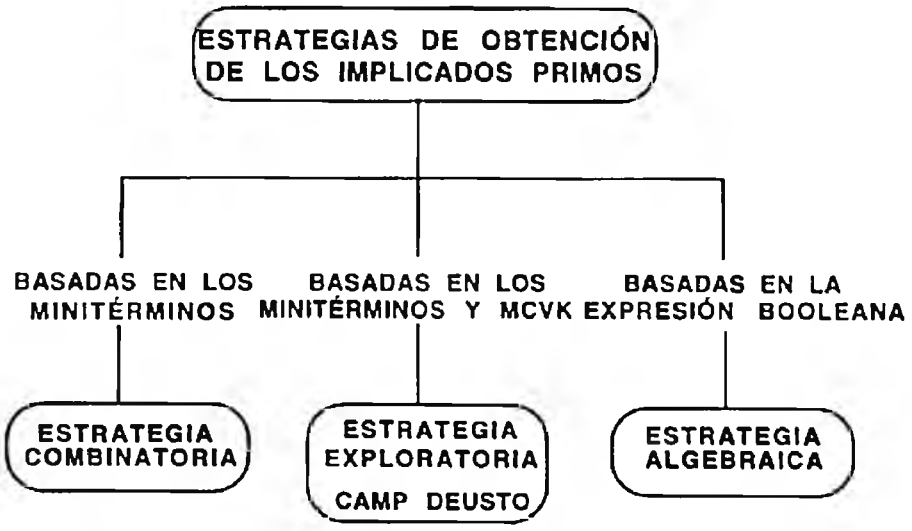


Figura 1.2 Clasificación de estrategias de obtención de los implicados primos

La figura 1.2 clarifica la aportación por parte de la tesis de una nueva línea de trabajo en el campo de la obtención de los implicados primos, desarrollada en el capítulo 2.

1.2.3 Fundamentos en la obtención de la expresión mínima

Las estrategias discriminatorias y algebraicas parten del total de los implicados primos y de la expresión original, respectivamente, y así simplificar supone discriminar qué implicados primos -ya sean del conjunto total o parcial- deben pertenecer a la expresión mínima. Sin embargo, la filosofía de la discriminación algebraica es ajena al nuevo método desarrollado en la tesis, y no es objeto de un estudio detallado en esta sección.

La estrategia discriminatoria exige distinguir entre los implicados primos de la función a simplificar (Muroga, 1979: 130):

- Un ***implicado primo*** es ***esencial*** o principal si alguno o algunos de los minitérminos de la función booleana sólo están cubiertos por él -minitérminos aislados.
- Un ***implicado primo*** es ***no esencial*** o secundario si todos los minitérminos por él cubiertos también lo están por otros implicados primos -minitérminos compartidos.

Frente a la expresión mínima ambos se comportan de distinta forma:

- Un implicado primo esencial debe pertenecer forzosamente a la expresión mínima, pues son los únicos que cubren a los minitérminos aislados. Su obtención es una tarea sencilla.
- Entre los implicados primos no esenciales debemos discriminar mediante un criterio el menor subconjunto de éstos que asegure cubrir los minitérminos no cubiertos por los implicados primos esenciales. Su obtención es una tarea complicada.

El criterio discriminante elegido por cada método es su ***piedra de toque***, calificándolo cualitativa y cuantitativamente, pudiendo tomar dos caminos opuestos entre sí:

- **Discriminación bifurcativa o *'branch mode'***. En este caso el método obtiene todos los posibles subconjuntos de implicados primos no

esenciales que aseguran cubrir los minitérminos restantes, y *a posteriori* elige el menor de ellos.

- **Discriminación directa.** Mediante un **criterio de discriminación heurístico** *-tie-breaking rule-* el método decide ***a priori*** qué implicado primo no esencial debe pasar a la expresión mínima. Los implicados primos son discriminados de uno en uno para formar parte del subconjunto.

1.2.4 Clasificación de los métodos de simplificación

La figura 1.3 muestra una clasificación que tiene como criterio las estrategias de simplificación y los datos de entrada de cada método. La observación conjunta de la figura 1.2 nos permite determinar la posición del nuevo método presentado.

La información contenida en la figura 1.3 queda completada con las tablas del apartado 1.5 que resumen las distintas características de los métodos explicados en los siguientes apartados. Estas tablas permiten establecer una comparación cualitativa entre los distintos métodos -incluido el aportado en la tesis-, aclarando las ventajas y desventajas de cada uno.

1.3 Desarrollo de los métodos de obtención de los implicados primos de una función

Dentro del presente apartado quedan explicados los distintos métodos de obtención de los implicados primos de una función booleana: el método de Quine-McCluskey, el método basado en el consenso iterativo y el método CAMP DEUSTO desarrollado en la tesis. Quedan excluidos los métodos algebraicos implementados en sistemas avanzados como McBoole y Espresso, ya que no permiten una comparación cuantitativa del método.

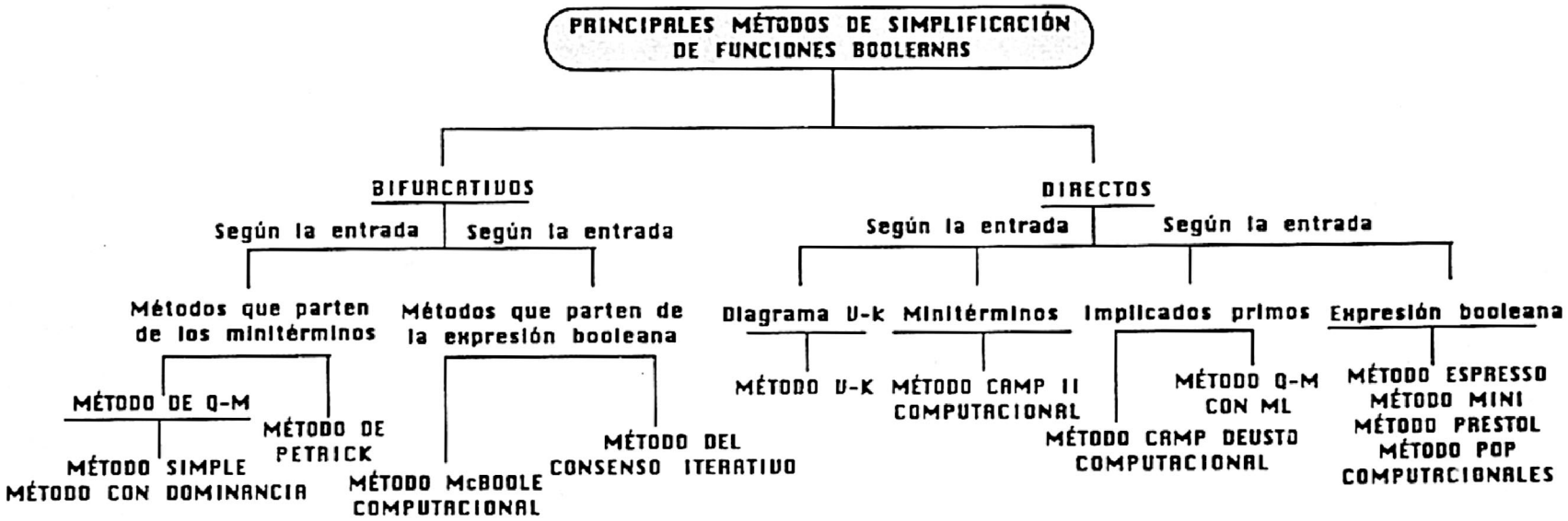


Figura 1.3 Clasificación de los métodos de simplificación

1.3.1 Método de Quine-McCluskey

Desarrollado en los años cincuenta por Quine (1952, 1955) y McCluskey (1956) es un clásico perfectamente conocido. La función booleana objeto de proceso debe estar representada por sus minitérminos en binario puro. A partir de éstos, y mediante sucesivas combinaciones entre ellos, formamos las *'combination tables'* de las cuales obtenemos los correspondientes implicados primos. Como máximo se forman n-1 tablas de combinaciones -donde n es el número de variables-, pudiendo concluir antes el proceso.

Inicialmente debemos expresar los minitérminos de la función en binario puro y acompañados de sus respectivos valores decimales. A continuación el método de Q-M completa las dos siguientes etapas:

Formar la tabla de combinaciones iniciales T_0. T_0 es una lista ordenada con todos los minitérminos agrupados por bloques, formado cada uno de ellos por los minitérminos cuyos respectivos equivalentes en binario puro contengan el mismo número de 1's. Cada minitérmino está expresado en binario puro y acompañado de su valor decimal.

Generar las tablas T_i *i=1, 2, 3,..., n-1.* Los elementos del bloque j de cada tabla se forman emparejando, si es posible, cada elemento del bloque (j+1) de la tabla T_{i-1} con cada uno del bloque j que tenga sólo i dígitos distintos. Al emparejar dos elementos resulta otro nuevo que se anota igual que los anteriores, sustituyendo por guiones los digitos que cambian. Para identificar posteriormente qué minitérminos intervienen en una combinación de i variables se anota al lado de cada elemento de la tabla T_i los valores decimales de los minitérminos que se han combinado, y se marcan con un asterisco todos aquellos que intervienen en posteriores combinaciones para formar la tabla T_{i+1}. Los elementos no marcados de la tabla T_k son implicados primos de (n-k) variables, y el conjunto de los implicados primos estará formado por los elementos no marcados en las tablas T_k, con k=0, 1, 2,..., α, siendo $\alpha \leq$ n-1.

El proceso concluye prematuramente cuando no existen elementos a emparejar en la tabla T_α.

Ejemplo 1.1 Sea F = Σ(0,3,4,5,6,8,10,12,14,18,20,22,25,27,28,29,20,31).

Una vez obtenidas las expresiones binarias de los minitérminos, éstas quedarán ordenadas en bloques. Cada bloque contiene aquellos minitérminos con el mismo número de 1's en su expresión binaria. Los elementos del bloque i-ésimo tienen (i-1) 1's.

T_0	ABCDE	
0	00000	•
4	00100	•
8	01000	•
3	00011	
5	00101	•
6	00110	•
10	01010	•
12	01100	•
18	10010	•
20	10100	•
14	01110	•
22	10110	•
25	11001	•
28	11100	•
27	11011	•
29	11101	•
30	11110	•
31	11111	•

Tabla 1.1 Tabla combinatoria T_0

En la tabla T_0 todos los minitérminos se combinan al menos con otro para formar T_1, excepto el 3.

T_1	ABCDE	
(0,4)	00-00	•
(0,8)	0-000	•
(4,5)	0010-	
(4,6)	001-0	•
(4,12)	0-100	•
(4,20)	-0100	•
(8,10)	010-0	•
(8,12)	01-00	•

Tabla 1.2 Tabla combinatoria T_1

(6,14)	0-110	•
(6,22)	-0110	•
(10,14)	01-10	•
(12,14)	011-0	•
(18,22)	10-10	
(20,22)	101-0	•
(20,28)	1-100	•
(14,30)	-1110	•
(22,30)	1-110	•
(25,27)	110-1	•
(25,29)	11-01	•
(28,29)	1110-	•
(28,30)	111-0	•
(27,31)	11-11	•
(29,31)	111-1	•
(30,31)	1111-	•

Tabla 1.2 *(continuación)*

Para formar T_2 sólo los elementos (4,5) y (18,22) de T_1 no forman ninguna pareja.

T_2	ABCDE	
(0,4,8,12)	0--00	
(0,8,4,12)	0--00	
(4,6,12,14)	0-1-0	•
(4,6,20,22)	-01-0	•
(4,12,6,14)	0-1-0	•
(4,12,20,28)	--100	•
(4,20,6,22)	-01-0	•
(8,10,12,14)	01--0	
(8,12,10,14)	01--0	
(6,14,22,30)	--110	•
(6,22,14,30)	--110	•
(12,14,28,30)	-11-0	•
(20,22,28,30)	1-1-0	•
(25,27,29,31)	11--1	
(25,29,27,31)	11--1	
(28,29,30,31)	111--	
(28,30,29,31)	111--	

Tabla 1.3 Tabla combinatoria T_2

En T_2 se observan muchos elementos repetidos que están incluidos en la tabla al aplicar estrictamente el método. La norma general es que no aparezcan elementos repetidos, tal y como se muestra en la nueva T_2.

T_2	ABCDE	
(0,4,8,12)	0--00	
(4,6,12,14)	0-1-0	
(4,6,20,22)	-01-0	*
(4,12,20,28)	--100	*
(8,10,12,14)	01--0	
(6,14,22,30)	--110	*
(12,14,28,30)	-11-0	*
(20,22,28,30)	1-1-0	*
(25,27,29,31)	11--1	
(28,29,30,31)	111--	

Tabla 1.4 Tabla combinatoria T_2 sin elementos repetidos

En la tabla T_3 se producen varios emparejamientos, que finalmente suponen un único elemento.

T_3	ABCDE
(4,6,20,22,12,14,28,30)	--1-0

Tabla 1.5 Tabla combinatoria T_3

El proceso no puede continuar, y por tanto concluye (como máximo se forma la columna T_{n-1}, donde n es el número de variables).

El conjunto de los implicados primos está formado por los elementos de las distintas T_k que no han formado pareja, es decir, aquellos que no tienen asterisco:

Implicado primo de 5 variables: L1 (3) 0001?
Implicados primos de 4 variables: L2 (4,5) 0010-; L3 (18,22) 10-10
Implicados primos de 3 variables: L4 (0,4,8,12) 0--00; L5 (8,10,12,14) 01--0; L6 (25,27,29,31) 11--1; L7 (28,29,30,31) 111--
Implicados primos de 2 variables: L8 (4,6,20,22,12,14.28,30) --1-0

El conjunto de implicados primos de F es:

F = 00011 + 0010- + 10-10 + 0--00 + 01--0 + 11--1 + 111-- + --1-0

La obtención manual de los implicados primos aplicando el método de Q-M es sencillo, pero es fácil equivocarse. Por contra, el algoritmo correspondiente se implementa fácilmente en un ordenador.

1.3.2 Método basado en el Consenso Iterativo

En este caso partimos de la función booleana representada como suma de productos, y no como suma de minitérminos. Estos productos no tienen porque ser implicados primos. El método presentado a continuación decide cuáles de éstos lo son, cuáles no lo son, y cualés y cómo deben ser modificados para serlo.

Teóricamente el método se basa en el *teorema del consenso* enunciado por Mott (1960) y generalizado por Tison (1967). Los dos teoremas siguientes forman el núcleo del Consenso Iterativo:

Teorema 1
Sea una función booleana expresada como suma de productos, y sean L1 y L2 dos de esos productos. Se dice que L1 cubre a L2, y por tanto L2 puede ser eliminado de la expresión, si L2 puede ser expresado como L1·Lx, donde Lx es cualquier producto. Algebraicamente:

$$L1+L2 = L1+L1\cdot Lx = L1\cdot(1+Lx) = L1$$

Teorema 2. Teorema del Consenso
Sea una función booleana expresada como suma de productos, y sean L1 y L2 dos de esos productos. Se dice que L1 está en consenso con L2 si podemos expresarlos como $L1 = X\cdot L11$ y $L2 = \overline{X}\cdot L22$, donde tanto L11 como L22 son dos productos cualesquiera. El término L11·L22 recibe el nombre de *consenso* entre L1 y L2, y puede ser añadido a la suma L1+L2 sin que ésta varíe. Algebraicamente:

$$L1+L2 = X\cdot L11+\overline{X}\cdot L22 = X\cdot L11+\overline{X}\cdot L22+L11\cdot L22$$

Basados en los dos teoremas anteriores, los pasos que completa el método del consenso iterativo son:

1. Escribir los productos en forma de lista.
2. Comparar cada producto con los que tiene debajo en la lista.
3. Si un producto cubre a otro, borrar éste último de la lista.
4. Si existe consenso entre dos productos añadir el término de consenso, sólo en el caso de que no esté cubierto por algún producto de la lista.
5. Realizar el proceso anterior hasta comparar todos los productos de la lista. Finalmente los productos existentes en la lista son los implicados primos.

Ejemplo 1.2 Obtener mediante la aplicación del método del consenso iterativo los implicados primos de la siguiente función:

$F = \overline{A}\cdot\overline{B} + A\cdot C + B\cdot D + \overline{C}\cdot\overline{D} = 00\text{--} + 1\text{-}1\text{-} + \text{-}1\text{-}1 + \text{--}00 = L1 + L2 + L3 + L4$

En este ejemplo ningún lazo de los cuatro cubre a otro.

La búsqueda de consenso ofrece que:

L1 está en consenso con L2 formando L12=-01- = $\overline{B}\cdot C$
L1 está en consenso con L3 formando L13=0--1 = $\overline{A}\cdot D$
L2 está en consenso con L4 formando L24=1--0 = $A\cdot\overline{D}$
L3 está en consenso con L4 formando L34=-10- = $B\cdot\overline{C}$

Como L12, L13, L24 y L34 han sido añadidos el proceso debe seguir respecto de éstos:

L1 está en consenso con L24 formando L124=-0-0 = $\overline{B}\cdot\overline{D}$
L1 está en consenso con L34 formando L134=0-0- = $\overline{A}\cdot\overline{D}$
L2 está en consenso con L13 formando L213=--11 = $C\cdot D$
L2 está en consenso con L34 formando L234=11-- = $A\cdot B$
L3 está en consenso con L12 formando L312=--11 = $C\cdot D$
L3 está en consenso con L24 formando L324=11-- = $A\cdot B$
L4 está en consenso con L12 formando L412=-0-0 = $\overline{B}\cdot\overline{D}$
L4 está en consenso con L13 formando L413=0-0- = $\overline{A}\cdot\overline{C}$

Los cuatro últimos lazos están repetidos, luego no deben ser añadidos.

Finalmente los implicados primos son: L1, L2, L3, L4, L12, L13, L24, L34, L124, L134, L213 y L234.

$$F = \overline{A}\cdot\overline{B} + A\cdot C + B\cdot D + \overline{C}\cdot\overline{D} + \overline{B}\cdot C + \overline{A}\cdot D + A\cdot\overline{D} + B\cdot\overline{C} + \overline{A}\cdot\overline{C} + \overline{B}\cdot\overline{D} + A\cdot B + C\cdot D$$

1.3.3 Método aportado en la tesis: CAMP DEUSTO

El desarrollo de CAMP DEUSTO (Computer Aided Minimization Procedure DEUSTO) es abordado detalladamente en el capítulo 2. Baste decir por ahora que se basa en generar previamente todos los posibles implicados primos de cualquier función booleana de un determinado número de variables. Partiendo de los posibles implicados primos y de los minitérminos de la función, el método comprende:

- Recorrer todos los posibles implicados primos.
- Decidir para cada posible implicado primo si lo es, o no.

El aspecto crucial del nuevo método reside en la correcta generación de todos los posibles implicados primos.

En el capítulo 2 se comparan cuantitativamente el método Q-M y el DEUSTO: en cuanto a rapidez en la obtención de los implicados primos y en cuanto a qué factores influyen en dicha rapidez.

1.4 Métodos de obtención de la expresión mínima de una función

En los siguientes apartados de esta sección abordaremos los principales métodos de minimización de funciones, su explicación nos ayudará a situar el nuevo método aportado. Los métodos objeto de estudio son:

- El método bifurcativo de Quine-McCluskey. (*)
- El método bifurcativo de Q-M con relación de dominancia. (*)

- El método bifurcativo basado en la proposición de Petrick.
- El método bifurcativo basado en el Consenso Iterativo.
- El método bifurcativo computacional McBOOLE.
- El método directo basado en el mapa de Veitch-Karnaugh.
- El método directo basado en Q-M y el criterio del máximo implicado. (*)
- El método directo CAMP II de Biswas. (*)
- El método directo computacional ESPRESSO.
- El método directo computacional MINI.
- El método directo computacional CAMP DEUSTO. (*)

Sólo los métodos marcados con un asterisco serán desarrollados completamente. De ellos cuatro son clásicos y presentan estrategias discriminantes, mientras que el CAMP II es avanzado, con estrategia constructiva y computacional. Estos cinco métodos permiten su implementación en MATLAB y la correspondiente comparación cuantitativa con el aportado, principalmente frente al CAMP II; mientras que con los métodos restantes la comparación se ceñirá a lo cualitativo. Previamente, y por ser común a todos los métodos a desarrollar, introduciremos el concepto de ***relación de dominancia*** .

1.4.1 Relación de dominancia

La aplicación de la relación de dominancia tiene como objetivo reducir el número de implicados primos y/o el de minitérminos, simplificando de esta forma el proceso de minimización.

La relación de dominancia existe entre implicados primos y entre minitérminos -dominancia entre filas y columnas, respectivamente-, y puede ser aplicada por casi todos los métodos anteriores. Así, dentro de un método se puede aplicar la relación de dominancia o no. En el apéndice B está formalizado el concepto de relación de dominancia.

Relación de dominancia entre implicados primos

Un conjunto de implicados primos domina a otro, si todos los minitérminos cubiertos por el segundo lo están también por el primero. Si el coste de los implicados dominados es mayor que el de los dominadores -caso común-, entonces los primeros pueden ser eliminados del conjunto de implicados primos de la función, simplificando la minimización de la función.

Relación de dominancia entre minitérminos

Un conjunto de minitérminos -o uno sólo- domina a otro, si todos los implicados que cubren al segundo también cubren al primero. El grupo de minitérminos dominador puede ser eliminado del conjunto total de minitérminos de la función a simplificar, simplificando la minimización de la función.

1.4.2 Método bifurcativo de Quine-McCluskey

En los años cincuenta W. V. Quine (1952, 1955) y E. J. McCluskey (1956) enuncian su método. Partiendo de los implicados primos y de los minitérminos que éstos cubren en la función, obtener la expresión mínima consiste en repetir varias veces -al menos una- la creación de una tabla de implicados primos -*'prime implicant table'*- donde se relacionan lazos y minitérminos de la función.

La primera tabla a crear, *'essential prime-implicant table'*, relaciona todos los implicados primos obtenidos con los minitérminos de la función. Si un minitérmino es cubierto por un único implicado primo, éste es esencial y pasa directamente a formar parte de la solución mínima. Puede haber varios implicados primos esenciales.

La segunda tabla, *'selective prime-implicant table'*, se construye eliminando de la primera los implicados primos esenciales -ya están en la expresión mínima- y los minitérminos que cubren -labor facilitada por los números decimales entre paréntesis. Se aplica de nuevo el criterio de la primera tabla, buscando si existe algún minitérmino cubierto por un único

implicado primo. Éste es un implicado primo no esencial, y pasa a formar parte de la expresión mínima. En una misma tabla puede haber varios implicados primos no esenciales.

La tercera y siguientes tablas se forman eliminando los implicados primos no esenciales y sus correspondientes minitérminos, aplicando de nuevo el mismo criterio de selección a los restantes implicados primos no esenciales. El proceso de creación de la tabla y selección se repite hasta que todos los minitérminos quedan cubiertos, despreciándose los implicados primos restantes que no formarán parte de la expresión mínima.

Cuando existen condiciones libres el algoritmo de Quine-McCluskey es idéntico en su procedimiento, con las siguientes acotaciones:

- Las condiciones libres sí aparecen en la tabla de combinaciones para obtener los implicados primos. Hay que tener en cuenta que un implicado primo formado exclusivamente por 'x' no es tal implicado primo, debe contener al menos un '1'. Este hecho debe ser comprobado.
- Las combinaciones que representan a condiciones libres no aparecen en ninguna tabla de implicados primos.

Apliquemos el método para simplificar la función del ejemplo 1.1, partiendo de los implicados primos ya obtenidos.

Formación de la primera tabla:

	0	3	4	5	6	8	10	12	14	18	20	22	25	27	28	29	30	31
L1		x																
L2			x	x														
L3										x		x						
L4	x		x			x		x										
L5						x	x	x	x									
L6													x	x		x		x
L7															x	x	x	x
L8			x		x			x	x		x	x			x		x	

Tabla 1.6 Tabla de implicados primos esenciales

En este caso todos los implicados primos excepto el L7 son esenciales ya que cada uno de ellos cubre al menos un minitérmino que sólo puede ser cubierto por dicho implicado primo.

La elección de L1, L2, L3, L4, L5, L6 y L8 cubre por sí sola todos los minitérminos -incluidos los que cubría L7- por tanto no es necesario plantear una segunda tabla. La solución expresada con literales es:

$$F = \overline{A}\cdot\overline{B}\cdot\overline{C}\cdot D\cdot E + \overline{A}\cdot\overline{B}\cdot C\cdot\overline{D} + A\cdot\overline{B}\cdot D\cdot\overline{E} + \overline{A}\cdot\overline{D}\cdot\overline{E} + \overline{A}\cdot B\cdot E + A\cdot B\cdot E + B\cdot\overline{E}$$

Ejemplo 1.3 Sea la siguiente función $F = \Sigma(3,4,7,9,11,12,13)$

Los implicados primos son:

IMPLICADO PRIMO	MINITÉRMINO
L1: -100	4 y 12
L2: 110-	12 y 13
L3: 1-01	9 y 13
L4: 10-1	9 y 11
L5: -011	3 y 11
L6: 0-11	3 y 7

Tabla 1.7 Tabla de implicados primos y minitérminos

Formación de la primera tabla:

	3	4	7	9	11	12	13
L1		x				x	
L2						x	x
L3				x			x
L4				x	x		
L5	x				x		
L6	x		x				

Tabla 1.8 Primera tabla de implicados primos esenciales

Los minitérminos 4 y 7 generan los dos implicados primos esenciales que pasan a formar parte de la expresión mínima: L1 (-100) y L6 (0-11).

Comoquiera que L1 y L6 no cubren todos los minitérminos hay que crear una segunda tabla, eliminando ambos lazos y sus correspondientes minitérminos (4,12,3,7).

Formación de la segunda tabla:

	9	11	13
L2			x
L3	x		x
L4	x	x	
L5		x	

Tabla 1.9 Segunda tabla de implicados primos selectivos

En este caso vemos que todos los minitérminos están cubiertos al menos por dos lazos, por lo tanto no podemos dilucidar cuál es el implicado primo no esencial (o cuáles son). Se dice entonces que la función está ciclada o entrelazada (función cíclica), o que los lazos están encadenados (función encadenada). Es necesario discriminar entre los implicados primos no esenciales. El método de Quine-McCluskey plantea la bifurcación como solución *'branch mode'*.

1.4.2.1 Bifurcación en Quine-McCluskey: *Branch mode*

Consiste en bifurcar la búsqueda de la solución en tantas vías como implicados primos tenga la tabla que presenta una función cíclica. Es decir, se supone que cualquiera de los lazos forma parte de la solución y se continúa aplicando el algoritmo. Finalmente se comprueba cuál de los caminos tomados ha generado la mínima expresión.

Lo anterior supone que el método Quine-McCluskey no es directo, por cuanto que no es capaz de discernir *a priori* qué camino es el correcto.

En el *ejemplo 1.3* se forman cuatro posibles soluciones:

Solución 1. Suponemos que L2 es un implicado primo no esencial:

	9	11
L3	x	
L4	x	x
L5		x

Tabla 1.10 Primera tabla en *branch mode* de implicados primos selectivos

Vemos que la nueva tabla está ciclada, es necesario volver a bifurcar. Visualmente optaríamos por L4, pero en realidad habría que generar tres

nuevos caminos: S11, S12 y S13, cuyas expresiones finales son: S12=L1+L6+L2+L4, S11=L1+L6+L2+L3+L5 y S13=L1+L6+L2+L5+L3.

Solución 2. Suponemos que L3 es un implicado primo no esencial:

	11
L4	x
L5	x

Tabla 1.11 Segunda tabla en *branch mode* de implicados primos selectivos

La tabla está ciclada de nuevo, sólo existen dos soluciones con igual coste: S21=L1+L6+L3+L4 y S22=L1+L6+L3+L5.

Solución 3. Suponemos que L4 es un implicado primo no esencial:

	13
L2	x
L3	x

Tabla 1.12 Tercera tabla en *branch mode* de implicados primos selectivos

En este caso L5 desaparece con L4. También aparecen dos soluciones con igual coste: S31=L1+L6+L4+L2 y S32=L1+L6+L4+L3.

Solución 4. Suponemos que L5 es un implicado primo no esencial:

	9	13
L2		x
L3	x	x
L4	x	

Tabla 1.13 Cuarta tabla en *branch mode* de implicados primos selectivos

La situación es idéntica a la aparecida en la solución 1. Tendríamos que abrir tres nuevos caminos: S41, S42 y S43, cuyas expresiones finales son: S41=L1+L6+L5+L2+L4, S42=L1+L6+L5+L3 y S43=L1+L6+L5+L2+L4.

Finalmente podemos decir que S21, S22, S31, S32, S12 y S42 son mínimas, y algunas idénticas. El resto de soluciones tiene cinco implicados primos, y por tanto han sido deshechadas.

1.4.2.2 Conclusiones al criterio *branch mode*

El anterior ejemplo es sencillo y sólo contempla cuatro bifurcaciones iniciales, pero muestra claramente que la aplicación del *'branch mode'* aun siendo exacto -comprueba todas las posibles soluciones- tiene un elevado costo en memoria y tiempo de ejecución. Su codificación es relativamente sencilla, siendo un caso típico de recursividad.

Cuantitativamente, el número de bifurcaciones a realizar crece exponencialmente con el número de implicados primos no esenciales:

$$\text{Número de bifurcaciones} = 2^{(N^{\circ}\ \text{Impl}-1)}$$

Por ejemplo, si una función encadenada tiene 20 implicados primos tiene que obtener $2^{(20-1)}=524.288$ expresiones distintas, para elegir entre ellas la mínima.

1.4.3 Método bifurcativo de Quine-McCluskey con dominancia

Se diferencia del anterior método en que aplicaremos en cada una de las sucesivas tablas de implicados primos la relación de dominancia entre los implicados primos y/o minitérminos, lo que reducirá en algunas ocasiones el tamaño de estas tablas. Consiguientemente, funciones que estén cicladas pueden pasar a no estarlo.

Si aplicamos la relación de dominancia entre filas al *ejemplo 1.3* vemos que la segunda tabla de implicados primos se reduce -tabla 1.9-, quedando:

	9	11	13
L3	x		x
L4	x	x	

Tabla 1.14 Segunda tabla de implicados primos selectivos

Tanto L3 como L4 son implicados primos secundarios y esenciales, y por tanto la solución es F=L1+L6+L3+L4. La situación mejora claramente, en este caso no es necesario bifurcar.

Como muestra este ejemplo, la aplicación -computacionalmente costosa- de la dominancia tiene como efecto, en algunos casos, la reducción del coste computacional global. Por último, indicar que la relación de dominancia es aplicable a muchos métodos, aunque en la tesis sólo estudiemos su efecto en los métodos de Quine-McCluskey y CAMP DEUSTO.

1.4.4 Método bifurcativo basado en la proposición de Petrick

El planteamiento de Petrick (1956, 1959 y 1960) difiere del enunciado por Quine-McCluskey sólo en la forma. El método Q-M plantea obtener todas las posibles expresiones una a una para elegir finalmente la menor, mientras que Petrick propone obtener todas las posibles expresiones minimas simultáneamente. Los pasos a dar partiendo de los implicados primos son:

1. Obtener los implicados primos esenciales, pasarlos a la expresión mínima, eliminando los minitérminos correspondientes. Este primer paso no es absolutamente necesario, pero reduce cuantitativamente la proposición de Petrick.
2. Construir la proposición de Petrick:
   - Es un producto de sumas con tantos sumandos como minitérminos.
   - Cada sumando se corresponde con un minitérmino, y es la suma de los implicados que cubren a dicho minitérmino.
3. El producto de sumas se convierte en suma de productos mediante la aplicación de la propiedad distributiva.
4. Aplicamos la propiedad de la idempotencia.
5. Elegimos los productos con menos términos.
6. Elegimos entre los anteriores aquel producto cuyos términos estén compuestos globalemente por menos literales, lo que supone elegir el producto con menor costo.

En el ejemplo 1.2, y tras obtener los implicados primos esenciales L1 y L6, resultaba:

	9	11	13
L2			x
L3	x		x
L4	x	x	
L5		x	

Tabla 1.15 Tabla de implicados primos

La **proposición de Petrick** es: P=(L3+L4)·(L4+L5)·(L2+L3)
que desarrollada queda:
P=L2·L3·L4+L2·L3·L5+L2·L4·L4+L2·L4·L5+L3·L3·L4+L3·L3·L5+L3·L4·L4+L3·L4·L5
Aplicando la idempotencia queda:
P = L2·L3·L4 + L2·L3·L5 + L2·L4 + L2·L4·L5 + L3·L4 + L3·L5 + L3·L4·L5
Elegimos los productos menores: L2·L4, L3·L4 y L3·L5.
Los tres productos menores tienen idéntico coste, luego la expresión mínima se puede formar con cualquiera de los tres:

$$F = L1 + L6 + L2 + L4 = L1 + L6 + L3 + L4 = L1 + L6 + L3 + L5$$

Al aplicar la propiedad distributiva obtenemos todas las expresiones mínimas a la vez. La proposición de Petrick es muy fácil de implementar, pero computacionalmente es muy costosa, más que Q-M, ya que éste último al obtener las expresiones de una en una permite un mejor control del algoritmo.

1.4.5 Método bifurcativo basado en el consenso iterativo

Este método de simplificación (Mott, 1960) y (Tison, 1967) se basa en la búsqueda sucesiva de consenso entre los implicados primos, seguramente obtenidos mediante la aplicación del consenso iterativo. Su núcleo teórico es idéntico al enunciado en el apartado 1.3.2.

Si llamamos LP al conjunto de implicados primos, y LC a un implicado cualquiera de LP candidato a pertenecer a la expresión mínima como implicado primo esencial, entonces los pasos son:

1. Se elimina de LP a LC y se busca el posible consenso dentro de este conjunto.
2. Si existe consenso el algoritmo añadirá a LP el elemento LC, por tanto LC no es un implicado primo esencial.
3. Si no existe consenso quiere decir que LC es un implicado primo esencial, y por tanto pertenece a la expresión mínima.

Una vez que tenemos todos los implicados primos esenciales LE es necesario buscar los posibles implicados primos secundarios LS. El proceso de búsqueda de LS basado en el consenso iterativo es:

1. Ejecutar el algoritmo de consenso en LE. Como resultado pueden haber sido añadidos nuevos elementos a LE, que no tienen porque ser todos implicados primos. Aquellos elementos añadidos que son implicados primos son redundantes LR -lazos redundantes.
2. Eliminamos de LP los lazos de LE y LR. Si no quedan implicados primos en LP el proceso ha terminado, y la expresión mínima sólo tiene implicados primos esenciales LE. En caso contrario hay que obtener los lazos secundarios.
3. Para obtener los lazos secundarios hay que empezar eliminando del LP restante, que llamaremos LPR (LP menos LE y LR), los lazos eclipsados por otros. Para ello tomamos uno a uno los lazos de LPR; cada lazo candidato LC puede eclipsar a aquellos lazos de LPR con igual o mayor costo que LC, que llamaremos LB -lazos borrables. Con cada lazo LC se realiza el siguiente proceso: Se añade LC a LE y se aplica el algoritmo de consenso, de tal forma que si un lazo de LB aparece añadido por el algoritmo debe ser borrado de LPR. El proceso se repite para todos los lazos de LPR.
4. Si en el proceso anterior algún lazo ha sido borrado, entonces LPR pasa a ser LP y volvemos a ejecutar el proceso de búsqueda de implicados primos esenciales, y siguientes procesos.

5. Si ningún lazo ha sido borrado, entonces hay que aplicar el *'branch mode '* como bifurcación por todos los lazos de LPR para continuar el proceso.

El aspecto más destacable y diferenciador del consenso iterativo es que se basa en los lazos y no en los minitérminos, este aspecto no es necesariamente favorable. Sin embargo, cuando la función está ciclada no ofrece ninguna mejora o alternativa al *branch mode* de Quine-McCluskey o a la proposición de Petrick, que es el principal problema teórico y computacional. Lo anterior nos lleva a no implementar este método, y no compararlo en el capítulo 3 con el aportado en la tesis.

1.4.6 Método bifurcativo computacional McBOOLE

El método McBOOLE desarrollado en la Universidad McGill de Montreal por Dagenais, Agarwal y Rumin (1985 y 1986) enfoca computacionalmente el método de Quine-McCluskey. Al igual que Q-M, la expresión mínima por él obtenida es exacta, pero minimiza los tiempos de simplificación.

El método de Q-M es muy lento cuando la función está ciclada y el número de implicados es alto. Esto conlleva un gran número de bifurcaciones que computacionalmente se convierten en recursividades, pudiendo llegar a ser muy profundas. El proceso computacional de simplificación se ralentiza enormemente, incluso no concluye.

La solución propuesta por McBOOLE parte al igual que Q-M de todos los implicados primos, buscando mejorar la implementación computacional del método Q-M, pero no cambiando su aspecto bifurcativo. Las mejoras se producen en dos direcciones:

- La implementación utiliza como estructura de datos los árboles dirigidos que facilitan el manejo de los implicados y de las situaciones bifurcativas.

- Si la función está ciclada McBOOLE subdivide la función en varias, aplicando la bifurcación a cada una de ellas y no al conjunto; para finalmente reunir la expresión mínima correspondiente a cada subfunción. Esto disminuye la profundidad de la bifurcación, y por tanto el tiempo necesario para obtener todas las expresiones mínimas. Aunque hay que reseñar que si la profundidad es muy elevada McBOOLE no concluye la simplificación.

La implementación computacional del método McBOOLE está orientada a *mainframes*, su consumo de memoria es proporcional al número de implicados primos y de variables, mientras que el tiempo de CPU es proporcional al número de implicados primos y al número de funciones cíclicas y su profundidad. McBOOLE consigue simplificar funciones de más de diez variables, que antes no podían serlo mediante las implementaciones clásicas de Q-M.

La tesis no recoge la implementación de este método -ni su consiguiente comparación- por su orientación a *mainframes* y porque se diferencia de Q-M sólo en la rapidez, no en la exactitud.

1.4.7 Método directo basado en el mapa de Veitch-Karnaugh

El diagrama de Veitch-Karnaugh es una forma de representar una función booleana. Definido por Veitch (1952), tiene forma de panel con casillas; y en él comparten el mismo espacio la entrada y la salida. Esta forma de representación favorece la simplificación *visual*, tal y como sistematiza Karnaugh (1953).

En un diagrma de V-K las casillas *próximas* entre sí forman un lazo. Así pues, simplificar supone rodear todas las casillas que tengan '1' con el menor número de lazos, siendo dichos lazos del mayor tamaño posible.

Simplificar en este caso supone una actividad visual, donde se puede establecer una estrategia de formación de lazos que asegure una expresión mínima. Pero generalmente no se aplica esa estrategia, simplemente se

obtiene directamente la disposición *obvia* de lazos que ofrece una expresión mínima. La actividad simplificadora humana frente a un V-K es claramente heurística y artesanal.

El uso de V-K está restringido a pocas variables -nunca más de cinco o seis-, y fundamentalmente se utiliza en el campo de la enseñanza de sistemas digitales. Esta restricción en el número de variables, unida a la propiedad heurística antes enunciada, nos lleva desdeñar su implementación para una posterior comparación.

1.4.8 Método directo basado en Q-M y el criterio del máximo lazo

El método de simplificación básicamente es el aportado por Quine-McCluskey. La diferencia aparece cuando la función es cíclica. En esta versión de Q-M no hay una bifurcación para obtener todos las posibles expresiones mínimas, sino que aplicando su criterio de discriminación elige directamente un único implicado primo por el que seguir, y por tanto la expresión mínima será única. La calidad depende del criterio de discriminación.

1.4.8.1 Criterio directo de discriminación: Máximo lazo

El criterio de máximo lazo pertenece a la categoría de los *'greedy algorithms'* -algoritmos voraces. Su filosofía es: ***Elige aquel camino que ofrezca un mayor beneficio instantáneo***. La suma de beneficios instantáneos no tiene porque producir el beneficio global, en nuestro caso la expresión mínima.

Si todos los implicados primos son no esenciales, es decir, la función es cíclica, el criterio de discriminación **Máximo Lazo** -ML- a aplicar es:

> **Elegir aquel implicado primo que cubra un mayor número de los minitérminos que restan por cubrir, si son varios elegir cualquiera de ellos.**

Ejemplo 1.4 Sea F=Σ(0,1,2,3,4,5,7,8,10,11,12,13,14,15), y habiendo obtenido previamente los implicados primos, la primera tabla es:

	0	1	2	3	4	5	7	8	10	11	12	13	14	15
00--	x	x	x	x										
0-0-	x	x			x	x								
0--1		x		x		x	x							
-0-0	x		x					x	x					
-01-			x	x					x	x				
--00	x				x			x			x			
-10-					x	x					x	x		
1--0								x	x		x		x	
--11				x			x			x				x
-1-1						x	x					x		x
1-1-									x	x			x	x
11--											x	x	x	x

Tabla 1.16 Tabla de implicados primos esenciales

Vemos que la función está ciclada, todos los implicados primos son no esenciales. En este caso Q-M seguiría los doce caminos independientes iniciales -que podrían volver a desdoblarse posteriormente. Ahora bien, si aplicamos el criterio ML eligiríamos directamente cualquiera de los lazos, ya que todos los lazos cubren cuatro minitérminos. Por ejemplo, el primero: 00-- pasa a la expresión mínima, siendo eliminados los minitérminos correspondientes. La segunda tabla quedaría:

	4	5	7	8	10	11	12	13	14	15
0-0-	x	x								
0--1		x	x							
-0-0				x	x					
-01-					x	x				
--00	x			x			x			
-10-	x	x					x	x		
1--0				x	x		x		x	
--11			x			x				x
-1-1		x	x					x		x
1-1-					x	x			x	x
11--							x	x	x	x

Tabla 1.17 Primera tabla de implicados primos selectivos con ML

Vemos que la función continúa ciclada. Aplicamos de nuevo el criterio ML que selecciona a: -10-,1--0,-1-1, 1-1- y 11--; de éstos se debe elegir uno cualquiera. Pero si elegimos el último -la ordenación de los minitérminos en la tabla es libre- la solución final no será mínima, tendría como mínimo cinco términos, y no cuatro como muestra la figura 1.4.

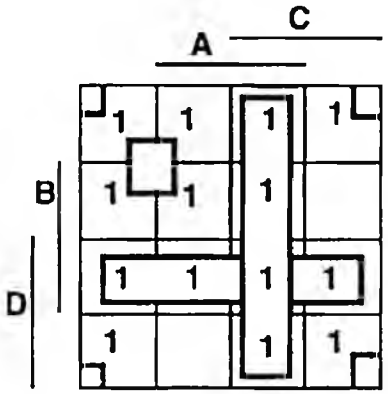


Figura 1.4 Simplificación gráfica del VK correspondiente al ejemplo 1.4

Elijamos sin embargo el primero: -10-, la tercera tabla quedaría:

	7	8	10	11	14	15
0-0-						
0--1	x					
-0-0		x	x			
-01-			x	x		
--00		x				
1--0		x	x		x	
--11	x			x		x
-1-1	x					x
1-1-			x	x	x	x
11--					x	x

Tabla 1.18 Segunda tabla de implicados primos selectivos con ML

La función continúa ciclada, y la aplicación del criterio ML selecciona directamente a: 1-1-. La cuarta tabla quedaría:

	7	8
0-0-		
0--1	x	
-0-0		x
-01-		
--00		x

Tabla 1.19 Tercera tabla de implicados primos selectivos con ML

1--0		x
--11	x	
-1-1	x	
11--		

Tabla 1.19 (*continuación*)

La función está ciclada de nuevo. No es necesario seguir con el proceso sistemáticamente para observar que serán necesarios dos lazos más. Finalmente, la expresión simplificada así obtenida tiene cinco términos, no siendo mínima, sino *casi mínima*.

1.4.8.2 Conclusiones al criterio del máximo lazo

Las conclusiones, ya enunciadas por otros autores, respecto del criterio del máximo lazo las podemos centrar en:

- El criterio ML es heurístico, sencillo y de *sentido común*.
- El criterio ML ofrece expresiones finales que no tienen porque ser mínimas, sino casi mínimas -*'near minimal'*.
- La implementación del algoritmo correspondiente es muy sencilla, y su coste computacional es muy bajo.
- Obviamente el método Q-M con ML es mucho más rápido que con *'branch mode'*.
- Es erróneo utilizar el criterio ML para reducir el campo de la bifurcación -como la relación de dominancia-; que la expresión final así obtenida sea mínima no está asegurado.
- La optimidad del criterio ML depende fuertemente del ordenamiento de los implicados primos, por otra parte, arbitrario.
- La implementación de ML está muy extendida por su sencillez y rapidez, aun siendo poco óptima.

El uso de este criterio está muy extendido, pero poco estudiado; sólo Hayes (Hayes, 1993: 343-347) explicita este criterio heurístico en su método GREEDYCOV. En el capítulo 3 este criterio es comparado con otros en cuanto a rapidez y optimidad.

1.4.9 Método directo CAMP II de Biswas

El método CAMP II desarrollado por Biswas (1984, 1986, 1990), e incluido en su libro (Biswas, 1993: 63-91), encara de una forma distinta al método Q-M la obtención de la expresión mínima de una función booleana. Así, si Q-M partía de los minitérminos para obtener los implicados primos, y con éstos obtener la expresión mínima; el CAMP II parte de los minitérminos para obtener mediante tablas la expresión mínima, y por tanto ni obtiene, ni utiliza los implicados primos. CAMP II es directo por cuanto que no bifurca, sino que en cada paso de la simplificación conforma un lazo que pasará a la expresión mínima. La expresión mínima final no tiene porque serlo, es un método *'near minimal'* (Biswas, 1993: 83), incluso para casos sencillos como mostraremos mediante un ejemplo.

Biswas además de ofrecer esta nueva perspectiva, fundamenta el método teóricamente -aunque sea heúristico- con una gran elegancia. En esta sección respetaremos el uso del término *cubo* que hace Biswas -y otros autores- en vez de lazo.

El procedimiento CAMP II tiene dos etapas diferenciadas:

- Primera etapa. Tiene como objetivo conformar los cubos primos esenciales de la función a simplificar. Esta estapa es cumplimentada una sola vez.
- Segunda etapa. Tiene como objetivo conformar los cubos primos no esenciales. Esta etapa se procesará repetidamente -cada procesado aporta un implicado primo no esencial- hasta cubrir todos los minitérminos de la función.

El método CAMP II (Biswas, 1993: 72-85) se compone de los siguientes pasos:

Primera etapa

1. Por cada minitérmino se generan sus direcciones adyacentes AD *-adjacent address*. El grado de adyacencia DA *-degree of adjacency-* es el número de AD presentes en la función F a simplificar.

Las distintas direcciones adyacentes se generan cambiando sucesivamente cada bit del minitérmino por su valor contrario, anotando su valor decimal. Un minitérmino de n variables tiene n direcciones adyacentes. Por ejemplo, las direcciones adyacentes de 1011 (11) son: 0011 (3), 1111 (15), 1001 (9), 1010 (10).

2. Se genera por cada minitérmino un cubo candidato a formar parte de la expresión simplificada CSC -*candidate solution cube*. Para ello, si la correspondiente dirección de adyacencia a un bit del minitérmino pertenece a F, entonces dicho bit será sustituido por un '2' al pasar al CSC. En caso contrario el bit pasará al CSC con su valor, '0' o '1'.
3. Si el DA de un minitérmino y su correspondiente CSC es cero o uno, automáticamente el CSC se convierte en un EPC -*essential prime cube*-, en QM sería un implicado primo esencial. Es decir, pasa a formar parte de la expresión simplificada.
4. Cada CSC, que no sea ya un EPC, forma su matriz SSM -*set of subsuming minterms*- para ello se sustituyen los '2' del CSC por todas sus posibles combinaciones. Cada combinación del CSC toma un valor decimal que es anotado. Por ejemplo, a un CSC 1202 le corresponde un SSM: 1000 (8), 1001 (9), 1100 (12) y 1101 (13).
   Si todos los elementos de SSM pertenecen a F, entonces el correspondiente CSC también es un EPC.
5. Los minitérminos cubiertos por los EPC's son borrados. Comienza la segunda etapa del algoritmo CAMP.

Segunda etapa

6. Esta segunda etapa comienza por el minitérmino con menor DA y pasaporte, entendido como requisito de paso a esta segunda etapa.

   Pasaporte: si alguna de las direcciones de adyacencia de un minitérmino está cubierta por al menos un EPC, dicho minitérmino tiene pasaporte; en caso contrario no. El pasaporte no es cuantificable por número de AD's cubiertos por EPC's, basta con que una dirección adyacente sea cubierta por un EPC.

   Si varios minitérminos tienen pasaporte y mismo DA se toma cualquiera de llos, por ejemplo el primero.

7. Para el minitérmino seleccionado en (6) y su CSC se genera la matriz MUM *-matrix of uncovered minterms-* compuesta por aquellos minitérminos del CSC -los elementos de SSM- que no están cubiertos por un EPC, y que pertenecen a F. Es decir, aquellos minitérminos de F incluidos en un CSC -posible cubo- y no cubiertos por un EPC.
8. A partir de la matriz MUM formamos la matriz IM *-inclusion màtrix*. Cada fila de MUM genera una fila de IM. Se suma exclusivo ⊕ cada bit del minitérmino original -el elegido en (6)- con cada bit de cada fila de MUM, pero sólo para aquellos bits que tengan un '2' en esa posición del CSC.
9. Retener por columnas. Se retiene aquella columna que tiene al menos un '1'. En esa misma posición y en el CSC se sustituye el '2' por un '3'.
10. Formar el ACSC *-augmented candidate solution cube-* a partir del CSC y las columnas retenidas. Es el CSC, donde quizá algunos '2' se han convertido en '3' en el paso (9).
    El cubo ACSC asegura mediante los '3' cubrir los minitérminos retenidos de MUM.
11. Se forma la matriz MAM *-matrix of absentee minterms*. Está formada por aquellos ininitérminos que estando cubiertos por el ACSC -los elementos de SSM- no pertenecen a F. Es decir, aquellos minitérminos que no pertenecen a F y están representados por ACSC.
12. Formar la matriz EM *-exclusion matrix*. Se suma exclusivo ⊕ cada bit del minitérmino original con cada bit de cada fila de MAM, pero sólo con aquellos bits que tengan un '2' en esa posición del ACSC, no un '3'.
13. Descartar columnas. Si una columna tiene al menos un '1' puede ser descartada, pero no todas las columnas con un '1' deben ser descartadas. Habrá que descartar el mínimo número de columnas que garantice excluir los minitérminos de MAM. Por ejemplo, si una columna A domina a las columnas B y C, bastará con descartar A. La relación de dominancia entre columnas es un criterio en el descarte de columnas.
    Al descartar una columna hay que sustituir el '2' de dicha posición en el ACSC por el valor del bit del minitérmino original. Esto supone

reducir el tamaño del cubo para que no cubra los minitérminos que no pertenecen a F, así pues la reducción debe ser lo menor posible para asegurar una expresión mínima.
En el ejemplo mostraremos que el algoritmo CAMP II no se muestra óptimo en este punto.

14. Formar el VSPC *-valid selective prime cube-* a partir del ACSC y de las columnas descartadas. Es el ACSC, donde quizá algunos '2' han sido cambiados por su valor original.
15. Formar el SPC *-selective prime cube-*, sin más que sustituir los '3' del VSPC por '2'. El cubo finalmente recibe el nombre de SC *-solution cube*.
    Este SPC cubre el mayor número posible de direcciones adyacentes que pertenezcan a F, y no cubre aquellas direcciones adyacentes que no pertenezcan a F. El SPC pasa a la expresión simplificada.
16. Los minitérminos cubiertos por el SPC son borrados.
17. Repetimos el proceso desde (7) con aquel minitérmino que tenga pasaporte y menor DA.

Casos singulares

Las fases anteriores son ejecutadas con el fin de obtener la expresión mínima, ahora bien en este proceso pueden darse excepciones o casos singulares. Algunos de éstos son de una gran importancia teórica:

1. Si una fila de EM tiene todo ceros se descartará el minitérmino objeto de estudio, pasándose al siguiente con pasaporte y menor DA.
   Si una fila de EM tiene todo ceros -bastaría con un '1'- quiere decir que el ACSC correspondiente no tiene forma de no cubrir una casilla que no pertenece a F. Por tanto, dicho ACSC y el minitérmino que lo generó deben ser rechazados.

2. Si una fila IM tiene todo '1', es eliminada sin más. El proceso continúa con normalidad después de haber borrado dicha fila.
   Si una fila tiene todo '1' quiere decir que el minitérmino correspondiente es cubierto por el ACSC cualquiera que sea la columna retenida.

3. Si todas las columnas de IM tienen un '1' al menos, entonces descartamos el minitérmino y pasamos al siguiente.

Si todas las columnas de IM tienen un '1', entonces todos los '2' pasan a ser '3'. Por tanto, el VSPC coincide con el CSC; pero esto es imposible, pues supondría que el correspondiente SSM pertenece a F, y por tanto el CSC hubiera sido un EPC.

4. Si la aplicación del método, teniendo en cuenta las excepciones, no aporta ningún SPC, entonces hay que aplicar el criterio directo de discriminación del CAMP II.

Por otra parte, las *condiciones libres* desde el principio son consideradas como minitérminos que han sido cubiertos. Consecuentemente, participan del cálculo de DA y de SSM, pero no participan en la segunda etapa.

En resumen, las anteriores fases partiendo del minitérmino con pasaporte forman un cubo que cubre el mayor número de casillas con minitérminos; seguidamente este cubo es modificado para que no cubra el mayor número de casillas sin minitérminos. Si las dos acciones anteriores son satisfactorias el CAMP II aporta un nuevo cubo a la expresión mínima, en caso contrario se repite la operación para otro minitérmino. Si esta repetición no aporta un cubo a la expresión mínima quiere decir que la función está encadenada, siendo necesario aplicar el criterio de discriminación directo del CAMP II

1.4.9.1 Criterio directo de discriminación del CAMP II

Si todos los minitérminos que quedan tienen igual DA y no tienen pasaporte, o el CAMP no consigue un SPC, entonces hay que aplicar el criterio de discriminación directo del CAMP II. Este criterio no bifurca, elige una única dirección en la que seguir.

Criterio de discriminación o '*tie-breaking rule*':

CAMP II tomará cualquier minitérmino con menor DA, y se trabajará con él como si tuviera pasaporte.

Para el minitérmino elegido arbitrariamente no tiene sentido generar MUM e IM -todas sus columnas valdrían '1'. Son generadas MAM y EM.

El *ejemplo 1.4*: F=Σ(0,1,2,3,4,5,7,8,10,11,12,13,14,15), nos servirá para aclarar el método CAMP II y mostrar la razón de su no optimalidad. Recordemos que está función es cíclica total, como muestra la figura 1.4.

Partiendo de los minitérminos obtenemos la primera tabla, que nos indicará si hay EPC's o no.

MINITER.		DA	Direcciones	CSC	SSM
0	0000	4	1,2,4,8	2222	0-15
1	0001	3	0,3,5	0222	0-7
2	0010	3	3,0,10	2022	0-3,8-11
3	0011	4	2,1,7,11	2222	0-15
4	0100	3	5,0,12	2202	0,1,4,5,8,9,12,13
5	0101	4	4,7,1,13	2222	0-15
7	0111	3	5,3,15	2221	1,3,5,7,9,11,13,15
8	1000	3	10,12,0	2220	0,2,4,6,8,10,12,14
10	1010	4	11,8,14,2	2222	0-15
11	1011	3	10,15,3	2212	2,3,6,7,10,11,14,15
12	1100	4	13,14,8,4	2222	0-15
13	1101	3	12,15,5	2122	4-7,12-15
14	1110	4	15,12,10,6	2222	0-15
15	1111	4	14,13,11,7	2222	0-15

Tabla 1.20 Primera tabla de CAMP II para obtener EPC's

Primera etapa.

No hay ningún minitérmino con un DA de cero o uno, y ningún SSM pertenece a F. Esta etapa no aporta EPC's.

Segunda etapa.

Tomamos del grupo de minitérminos con DA igual a tres, uno cualquiera, por ejemplo el primero: M1. Obtenemos la tabla:

M1	0	0	0	1
AD	9	5	3	0
Presente		•	•	•
CSC	0	2	2	2
SSM	0-7			

Tabla 1.21 Tabla de CAMP II donde M1 es descartado

MUM	2	0	0	1	0
	3	0	0	1	1
	4	0	1	0	0
	5	0	1	0	1
	7	0	1	1	1
IM	2	-	0	1	1
	3	-	0	1	0
	4	-	1	0	1
	5	-	1	0	0
	7	-	1	1	0
Retener			•	•	•
ACSC					
MAM					
EM					
Descartar					
VSPC					
SC					

Tabla 1.21 (*continuación*)

Todas las columnas de IM tienen al menos un '1'. Hay que descartar el minitérmino y probar con el siguiente con DA igual a tres: M2.

M2		0	0	1	0
AL		10	6	0	3
Presente		•		•	•
CSC		2	0	2	2
SSM		0-3,8-11			
MUM	0	0	0	0	0
	1	0	0	0	1
	3	0	0	1	1
	8	1	0	0	0
	10	1	0	1	0
	11	1	0	1	1
IM	0	0	-	1	0
	1	0	-	1	1
	3	0	-	0	1
	8	1	-	1	0
	10	1	-	0	0
	11	1	-	0	1
Retener		•		•	•
ACSC					

Tabla 1.22 Tabla de CAMP II donde M2 es descartado

MAM	
EM	
Descartar	
VSPC	
SC	

Tabla 1.22 (*continuación*)

Todas las columnas de IM tienen al menos un '1'. Hay que descartar el minitérmino y continuar con el siguiente. Realmente todos los minitérminos son descartados, la función está ciclada, hay que aplicar el criterio directo de discriminación de CAMP II. Consiste en tomar el primer minitérmino -en realidad cualquiera- con DA igual a tres - M1- y operar con él, exceptuando MUM e IM.

M1		0	0	0	1
AD		9	5	3	0
Presente			•	•	•
CSC		0	2	2	2
SSM		0-7			
MUM					
IM					
Retener					
ACSC		0	2	2	2
MAM	6	0	1	1	0
EM		-	1	1	1
Descartar			•	•	•
VSPC		0	0	2	2
SC		0	0	2	2

Tabla 1.23 Tabla de M1 en *branch mode*, sin MU ni IM

Hay que elegir qué columnas descartar. Por ejemplo, el cubo 0022 es suficiente para no cubrir la casilla 6, también valdrían 0202 ó 0221.

Elegimos el SPC 0022 para pasar a la expresión mínima, que cubre los minitérminos: 0-3. Quedan por cubrir: M4 (3), M5 (4), M7 (3), M8 (3), M10 (4), M11 (3), M12 (4), M13 (3), M14 (4) y M15 (4), con el DA entre paréntesis.

Entre los minitérminos con DA igual a tres tienen pasaporte: M4, M7, M8 y M11. Elegimos el primero: M4.

M4		0	1	0	0
AD		12	0	6	5
Presente		•	•		•
CSC		2	2	0	2
SSM		0,1,4,5,8,9,12,13			
MUM	5	0	1	0	1
	8	1	0	0	0
	12	1	1	0	0
	13	1	1	0	1
IM	5	0	0	-	1
	8	1	1	-	0
	12	1	0	-	0
	13	1	0	-	1
Retener		•	•		•
ACSC					
MAM					
EM					
Descartar					
VSPC					
SC					

Tabla 1.24 Tabla de CAMP II donde M4 es descartado

Todas las columnas de IM tienen al menos un '1'. Descartamos el minitérmino M4, pasamos al M7.

M7		0	1	1	1
AD		15	3	5	6
Presente		•	•	•	
CSC		2	2	2	1
SSM		1,3,5,7,9,11,13,15			
MUM	5	0	1	0	1
	11	1	0	1	1
	13	1	1	0	1
	15	1	1	1	1
IM	5	0	0	1	-
	11	1	1	0	-
	13	1	0	1	-
	15	1	0	0	-
Retener		•	•	•	
ACSC					

Tabla 1.25 Tabla de CAMP II donde M7 es descartado

MAM	
EM	
Descartar	
VSPC	
SC	

Tabla 1.25 (*continuación*)

Todas las columnas de IM tienen al menos un '1'. Hay que descartar el minitérmino. De hecho, la función está ciclada de nuevo -como se puede comprobar continuando con los otros minitérminos. Hay que aplicar de nuevo el criterio directo de discriminación de CAMP II, tomamos como minitérmino el M4.

M4		0	1	0	0
AD		12	0	6	5
Presente		•	•		•
CSC		2	2	0	2
SSM		0,1,4,5,8,9,12,13			
MUM					
IM					
Retener					
ACSC		2	2	0	2
MAM	9	1	0	0	1
EM	9	1	1	-	1
Descartar		•	•	-	•
VSPC		2	2	0	0
SC		2	2	0	0

Tabla 1.26 Tabla de M4 en *branch mode*, sin MU ni IM

Análisis del ejemplo: Discusión de la no optimalidad del CAMP II.

Los cubos resultantes y los minitérminos que cubren, según sea la columna descartada son: 0202 (0,1,4,5), 2102 (4,5,12,13) y 2200 (0,4,8,12). Los tres cubos tienen el mismo coste, y según Biswas cualquiera puede ser elegido. Pero la elección del primero es errónea. El VK de la figura 1.5.a muestra que eligiendo el primer cubo la simplificación tiene cinco cubos, en vez de cuatro (trazo grueso los aportados a EM y en fino los restantes necesarios).

Podemos observar que 0202 cubre dos minitérminos de F sin cubrir: M4 y M5; el 2102 cubre cuatro: M4, M5, M12 y M13; y 2200 cubre tres: M4, M8 y M12. Podríamos pensar que habiendo cubos de igual tamaño debemos elegir aquel que cubre más minitérminos restantes de F. Es decir, elegir 2102; pero el VK de la figura 1.5.b muestra que la expresión mínima final también tendría cinco cubos.

Tanto si elegimos el primer cubo 0202, como si elegimos el 2102 que cubre un mayor número de casillas la supuesta expresión mínima tiene cinco lazos. El resultado es incorrecto y, por tanto, el método CAMP II presentado por Biswas no es óptimo, sino *'near minimal'*.

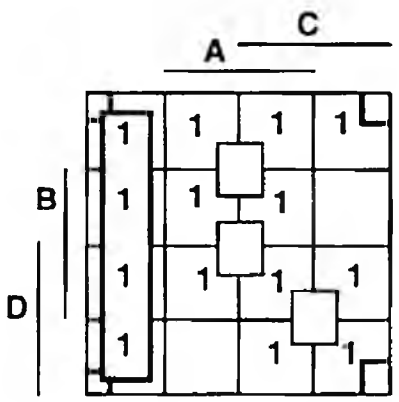


Figura 1.5.a

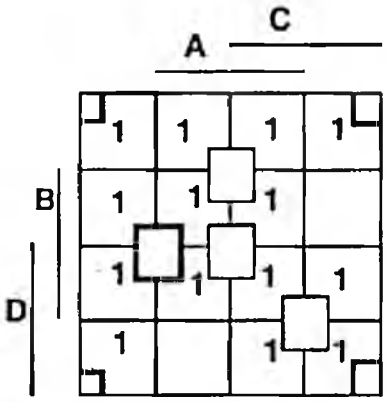


Figura 1.5.b

La resolución gráfica muestra la no optimalidad del método CAMP II

La elección correcta es 2200, que cubre; M0, M4, M8 y M12. Quedan por cubrir los minitérminos: 5 (4), 7 (3), 10 (4), 11 (3), 13 (3), 14 (4) y 15 (4) - el DA entre paréntesis. Empezamos con M7.

M7		0	1	1	1
AD		15	3	5	6
Presente		*	*	*	
CSC		2	2	2	1
SSM		1,3,5,7,9,11,13,15			
MUM	5	0	1	0	1
	11	1	0	1	1
	13	1	1	0	1
	15	1	1	1	1

Tabla 1.27 Tabla de CAMP II donde M7 es descartado

IM	5	0	0	1	-
	11	1	1	0	-
	13	1	0	1	-
	15	1	0	0	-
Retener		•	•	•	
ACSC					
MAM					
EM					
Descartar					
VSPC					
SC					

Tabla 1.27 (*continuación*)

Todas las columnas de IM tienen al menos un '1', hay que descartar a M7. Si tomamos M11y M13 obtendremos el mismo resultado: la función sigue ciclada. Hay que aplicar de nuevo el criterio de directo de discriminación de CAMP II, por ejemplo a M7.

M7		0	1	1	1
AD		15	3	5	6
Presente		•	•	•	
CSC		2	2	2	1
SSM		1,3,5,7,9,11,13,15			
MUM					
IM					
Retener					
ACSC		2	2	2	1
MAM	9	1	0	0	1
EM	9	1	1	1	-
Descartar		•	•	•	
VSPC					
SC					

Tabla 1.28 Tabla de M7 en *branch mode*, sin MU ni IM

De nuevo podemos descartar cualquiera de las tres columnas, generando cubos de igual tamaño: 0221, 2121 y 2211. Pero, al igual que antes, no todos obtienen una expresión mínima.

El cubo 0221 cubre a M5 y M7; 2121 cubre a: M5, M7, M13 y M15; y 2211 cubre a: M7, M11 y M15. El primer y tercer cubo no ofrecen una expresión minima final -como muestra la figura 1.6. Por contra, el cubo 2121 que cubre

cuatro minitérminos es la elección correcta -cabe recordar que en el anterior paso el cubo que cubría cuatro minitérminos era incorrecto. Quedan por cubrir los minitérminos: 10 (4), 11 (3) y 14 (4). Tomamos el M11.

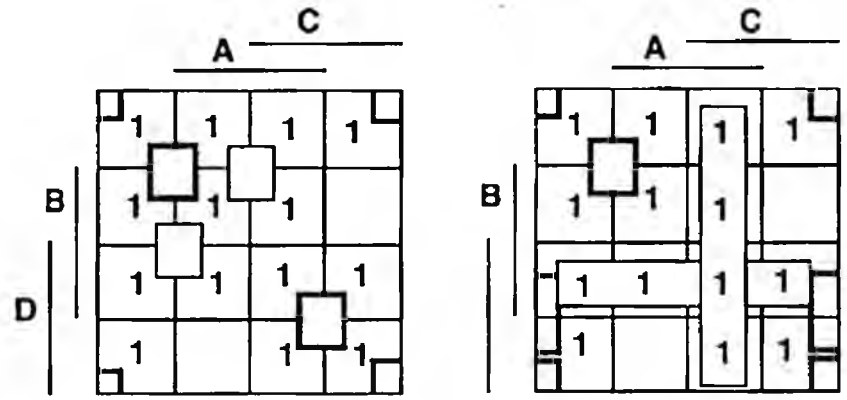


Figura 1.6 La resolución gráfica en ambos casos muestra la incorrección del método CAMP II

M11		1	0	1	1
AD		3	15	9	10
Presente		•	•		•
CSC		2	2	1	2
SSM		2,3,6,7,10,11,14,15			
MUM	10	1	0	1	0
	14	1	1	1	0
IM	10	0	0	-	1
	14	0	1	-	1
Retener			•		•
ACSC		2	3	1	3
MAM	6	0	1	1	0
EM	6	1	-	-	-
Descartar		•			
VSPC		1	3	1	3
SC		1	2	1	2

Tabla 1.29 Tabla de CAMP II correspondiente a M11

Este cubo completa el proceso, la función simplificada queda:

$$F = \overline{A}\cdot\overline{B} + \overline{C}\cdot\overline{D} + B\cdot D + A\cdot C$$

1.4.9.2 Conclusiones al método CAMP II

Las principales conclusiones podemos agruparlas en los siguientes puntos:

- Biswas aporta en CAMP II una orientación diferente.
- El CAMP II aunque sistemático y teórico es heurístico, por cuanto que queda sin justificar el artificio del pasaporte.
- La expresión final obtenida por CAMP II no tiene porque ser necesariamente mínima. Es un método *'near minimal*, como ha quedado aclarado mediante un simple ejemplo de cuatro variables.
- El criterio directo de discriminación no admite mejoras.
- El método es manualmente aplicable, y fácilmente implementable.
- Su coste computacional no es despreciable dado el alto número de tablas a generar.
- El método CAMP II no admite la bifurcación para asegurar la expresión mínima como alternativa al criterio de discriminación.

El método CAMP II será implementado, y en el capítulo 3 lo compararemos cuantitativamente con el resto, en cuanto a rapidez y optimidad de las expresiones mínimas.

1.4.10 Método directo computacional ESPRESSO

El método ESPRESSO ha sido desarrollado en la Universidad de Berkeley por Brayton (1982 y 1984), Hachtel, McMullen, Sangiovanni-Vincentelli y Rudell (1985) para simplificar computacionalmente funciones con un gran número de variables. Su criterio de discriminación es directo, su estrategia heurística y el método es estrictamente computacional y principalmente orientado a *mainframes*. El método es teóricamente no-óptimo, pero en la práctica su optimalidad es total en muchos casos, como ha podido ser comprabado mediante el método óptimo McBOOLE.

ESPRESSO es el método de simplificación más potente, rápido y exacto actualmente.

Para simplificar una función booleana ESPRESSO parte de su expresión booleana y le que aplica las operaciones *expand, reduce, irredundant* y *essential primes,* evitando el uso de minitérminos y del conjunto de implicados primos. La estrategia resumida en (Rudell et al, 1985) consiste en reducir y expandir la expresión booleana original, eliminando durante el proceso los términos redundantes y pasando a la expresión mínima los términos esenciales. Esta secuencia se repite continuamente, refinando así la expresión original hasta obtener la expresión mínima.

El método es muy sólido desde el punto de vista teórico (Brayton et al, 1984). Sin embargo, la heurística es fundamental tanto al elegir el orden en que deben ser reducidos los términos, como para desbloquear el proceso cuando el refinamiento no produce mejoras. Dichas estrategias heurísticas que son cruciales en el desarrollo del método no están suficientemente explicitadas, sobre todo en el caso de funciones cicladas.

La falta de una descripción completa del método, su complejidad, su orientación básica a *mainframes,* así como el tipo de situaciones hacia las que está orientado nos lleva a no implementar el ESPRESSO. Cabe recordar que el objetivo de la tesis es desarrollar un prototipo en MATLAB que siente las bases de desarrollo de un sistema como ESPRESSO.

1.4.11 Método directo computacional MINI

El método directo MINI fue desarrollado en IBM a partir de los años setenta por Hong, Cain y Ostapko (1974). Tiene las mismas características principales que ESPRESSO, destacando su aspecto computacional y heurístico, pero su optimidad y rapidez son inferiores a las presentadas por ESPRESSO.

El método MINI conceptualmente busca implementar un algoritmo que siga la estrategia de simplificación humana frente a un diagrama de Veitch-Karnaugh. Para ello parte de la entrada como expresión booleana de la función a simplificar -evitando los minitérminos y el conjunto de implicados primos- a la que le aplica tres operaciones fundamentales: *reduction, reshap y expand*. Cada procesado de las tres operaciones reduce el tamaño de la expresión; la secuencia se repite refinando la expresión original hasta obtener la mínima posible.

En primer lugar los términos o cubos son **reducidos** *-reduction-* en su tamaño sin dejar de cubrir los minitérminos que le son propios; seguidamente los cubos son tomados por pares para **recombinarlos** *-reshap-* reduciendo el tamaño de uno y aumentando el del otro, pero manteniendo el mismo conjunto de minitérminos de los dos; finalmente cada cubo es **aumentado** *-expand-* de tamaño tanto como sea posible, lo que puede suponer que algunos de los cubos queden cubiertos y sean eliminados. Los procesos *reduction* y *expand* reducen el número de cubos de la expresión, mientras que *reshap* reduce el tamaño de la expresión.

El orden en que son procesados los cubos de la expresión es determinante en la calidad final de la expresión mínima obtenida. Dicho orden es establecido mediante un criterio heurístico.

El método PRESTOL de simplificación, desarrollado en los años ochenta (Bartholomeus y Man, 1985) puede ser observado como una mejora computacional del MINI, sin variaciones excesivas en la estrategia. Y ambos métodos pueden ser considerados a su vez los precedentes teóricos y algorítmicos en los que se basa el método ESPRESSO. En cualquier caso no son ni implementados, ni comparados al quedar fuera del entorno de la tesis por las mismas razones que las aducidas para ESPRESSO.

1.4.12 Método directo computacional CAMP DEUSTO

Este método será desarrollado detenidamente en los capítulos 2 y 3, a la vez que comparado cuantitativamente con los métodos anteriormente expuestos.

Las dos grandes diferencias frente al resto de métodos estriban en:

- La obtención de los implicados primos.
- El criterio directo de discriminación.

Obtención de los implicados primos

El primer paso es obtener todos los posibles implicados primos con los que se relacionan todas las funciones booleanas de un determinado número de variables. Para cada número de variables tendremos una matriz MCVK que contiene todos los posibles implicados primos, las casillas con las que se relacionan e informaciones complementarias.

Obtener los implicados primos de una función particular de un determinado número de variables consiste en recorrer todas las filas de la correspondiente MCVK comprobando cuáles de los posibles implicados primos finalmente lo son.

Criterio directo de discriminación de CAMP DEUSTO

El criterio directo y heurístico de discriminación -**Discriminación Comparativa: algoritmo DC**- desarrollado tiene como filosofía:

> **Elegir aquel implicado primo que cubra el mayor número de determinados minitérminos, que a su vez son cubiertos por el menor número de determinados implicados primos.**

Antes de pasar a desarrollar estos dos aspectos del nuevo método aportado en la tesis, clasificaremos el conjunto de métodos según distintos criterios, y con esta visión global cualitativa se comprenderá mejor la aportación del nuevo método y su interés frente a los métodos actuales.

1.5 Clasificación y comparación de los métodos de simplificación

La clasificación se realizará según el proceso de obtención de:

- Implicados primos.
- Expresión mínima.

La clasificación finalmente obtenida nos permite observar qué lugar ocupa el método aportado en la tesis, mientras que en los capítulos 2 y 3 centraremos el análisis en los aspectos cuantitativos.

1.5.1 Comparación según el proceso de obtención de los implicados primos

Cada método encara de distinta manera la obtención de los implicados primos de una función. La diferencia es radical respecto de los datos de entrada que utiliza cada método.

	Datos de entrada	Implementación del método	Algoritmo e implement.	Dependencia en rápidez	Estrategia de obtención
Método de Q-M	Minitérminos	Manual	Combinatorio Normal	Exponencial Nº miniter.	Combinatoria
Consenso Iterativo	Expresión booleana SOP	Computacional	Combinatorio Normal	Exponencial Nº términos	Algebraica
McBOOLE ESPRESSO	Expresión booleana SOP	Estrictamente Computacional	Algebraico Muy Complejo	Exponencial Nº términos	Algebraica
CAMP DEUSTO	Matriz MCVK y minitérminos	Estrictamente Computacional	Exploratorio Muy Sencillo	Exponencial Nº variables	Exploratoria

Tabla 1.30 Clasificación de las técnicas de obtención de implicados primos

En el capítulo 2 implementamos y comparamos cuantitativamente el método de Q-M con el aportado en la tesis.

1.5.2 Comparación según el proceso de obtención de la expresión mínima

La tabla 1.31 caracteriza los distintos métodos de obtención de la expresión mínima.

	Fundamento Teórico	Datos de Entrada	Criterio de Discriminación	Optimidad de la expresión	Aplicación del método
Método Q-M	No heurístico	Implicados Primos	Bifurcativo	Total	Manual
Proposición de Petrick	No heurístico	Implicados Primos	Bifurcativo	Total	Manual
Consenso Iterativo	No heurístico	Implicados Primos	Bifurcativo	Total	Manual
MCBOOLE	No heurístico	Expresión Booleana	Bifurcativo	Total	Específicamente Computacional
Método V-K	No heurístico	Diagrama de V-K	Directo	Total	Específicamente Manual
Método Q-M criterio ML	Heurístico	Implicados Primos	Directo	Casi Mínima	Manual
CAMP II	Heurístico	Minitérminos	Directo	Casi Mínima	Computacional
ESPRESSO	Heurístico	Expresión Booleana	Directo	Casi Mínima	Específicamente Computacional
MINI	Heurístico	Expresión Booleana	Directo	Casi Mínima	Específicamente Computacional
CAMP DEUSTO	Heurístico	Implicados Primos	Directo	Casi Mínima	Específicamente Computacional

Tabla 1.31 Clasificación de las técnicas de obtención de la expresión mínima

Complementando a la tabla cabe reseñar que:

- Los métodos directos son rápidos pero no son exactos, los métodos bifurcativos son exactos pero son lentos.
- El método más parecido cualitativamente al CAMP DEUSTO es el CAMP II de Biswas.

- Los métodos calificados como estrictamente computacionales no son reimplementables en otros entornos.

La diferencia principal queda establecida entre:

- los métodos avanzados: CAMP II, McBOOLE, ESPRESSO y MINI,
- los métodos clásicos: Veitch-Karnaugh, Quine-McCluskey con sus modificaciones y Consenso Iterativo.

Los métodos avanzados computacionales son de carácter comercial, rápidos, exactos y fuertemente dependientes de la calidad de su muy complicada implementación. Debido a su carácter comercial su explicación es limitada para no permitir posibles imitaciones. De todos ellos el más potente y respetado para un gran número de variables -del orden de veinte- es ESPRESSO por la rapidez propia de su carácter heurístico, y también por la exactitud conseguida por su criterio directo. Para un número de variables menor -del orden de diez- el McBOOLE aporta la exactitud de todo método bifurcativo, y también una gran rapidez gracias a una correcta implementación.

Los métodos clásicos son sencillos en su planteamiento y fácilmente implementables. El proceso de simplificación asociado a cada método puede llegar a ser rápido, mientras que su exactitud en el caso de criterios directos suele ser escasa. Ahora bien los métodos avanzados se basan en estos, es por lo tanto necesario conocerlos e implementarlos para desarrollar uno nuevo.

El método CAMP DEUSTO desarrollado en la tesis es un prototipo que debe servir como base para desarrollar un método con prestaciones elevadas. Se basa en los implicados primos, es de tipo directo y computacional. Su exactitud y rapidez serán estudiadas en el capítulo 3, donde CAMP DEUSTO será comparado cuantitativamente con: Quine-McCluskey, Quine-McCluskey con relación de dominancia, Quine-McCluskey con máximo lazo y CAMP II.

CAPÍTULO 2. EL NUEVO MÉTODO DIRECTO COMPUTACIONAL: OBTENCIÓN DE LOS IMPLICADOS PRIMOS

2.1 Introducción

Según el primer capítulo, las fases en la simplificación de una función booleana son:

- Obtención de los implicados primos de la función.
- Discriminación de aquellos implicados primos que forman parte de la expresión simplificada.

En este segundo capítulo nos centraremos en la obtención de los implicados primos de una función booleana, aportando un nuevo enfoque y algoritmo.

Para obtener los implicados primos Quine-McCluskey se basa en combinar los minitérminos de la función a simplificar. El algoritmo desarrollado para este propósito es **manualmente aplicable**, de tipo **combinatorio, fácilmente implementable y su rapidez depende directamente del número de minitérminos**, e inversamente del número final de implicados primos.

El método propuesto en la tesis para obtener los implicados primos de una función se basa en generar previamente todos los posibles implicados primos asociados a toda función booleana de un determinado número de variables. Partiendo de todos los posibles implicados primos en general, para obtener los implicados de una función en particular basta con explorarlos y decidir cuáles se convierten en implicados primos de la función a

simplificar. Así pues, el método aportado sólo es **aplicable computacionalmente, es exploratorio, muy fácil de implementar** y su **rapidez sólo depende del número de variables**, no del número de minitérminos.

El objetivo global del presente capítulo es presentar el nuevo método de obtención de los implicados primos de una función booleana. Dicho capítulo consta de las siguientes secciones:

1. Conceptos fundamentales del nuevo método.
2. Fases en la obtención de los implicados primos.
3. Definición y creación de la matriz MCVK.
4. Aspectos de la implementación en MATLAB de MCVK.
5. Obtención de los implicados primos.
6. Conclusiones.

En la primera sección presentaremos los conceptos de rueda y lazo, así como la relación entre ambos. En la segunda desarrollaremos las distintas fases, a través de las cuales se obtienen los implicados primos. La tercera se centra en los algoritmos que obtienen la matriz MCVK -cuyos nombres de variables aparecen en el apéndice C. Esta matriz es la base de la quinta sección para obtener los implicados primos. Los algoritmos desarrollados son comparados con el método de Q-M para obtener las conclusiones en la última sección.

2.2 Conceptos fundamentales del nuevo método

Previamente al desarrollo del método de simplificación de funciones es necesario concretar ciertos aspectos nuevos y básicos, como es el caso de la rueda y la clasificación que supone. Los aspectos de la disciplina del álgebra booleana utilizados son los comunes.

2.2.1 Conceptos relacionados con los posibles implicados primos

En la tesis utilizaremos indistintamente el término posible implicado primo o el término lazo; éste último puede ser más expresivo en algunas ocasiones. Cualquiera que sea el término utilizado se asocia con los siguientes conceptos:

- Identificador, tipo, tamaño y conjunto de lazos.
- Lazos cubiertos y lazos adyacentes.

1. *Identificador de un lazo, su tipo y tamaño*

Todo lazo queda definido por su identificador general IDL y el activo IDA:

Identificador general, IDL. Está formado por NV dígitos de valor 0, 1 ó 2 que indican respectivamente que está negada, sin negar o eliminada la letra que le correspondería en el caso de utilizar un identificador con letras. Los dígitos de valor 0 ó 1 se consideran bits activos.

Identificador activo, IDA. Es la combinación binaria formada por los bits activos del IDL en orden inverso. El número de bits activos de un lazo es su tipo (TIPOL).

Tamaño de un lazo. Es el número de minitérminos cubiertos por un lazo, y es igual a $2^{(NV-TIPOL)}$.

Ejemplo 2.1 El lazo cuyo IDL es 11020 se corresponde con el literal $A \cdot B \cdot \overline{C} \cdot \overline{E}$, y su IDA es 0011 (3) -entre paréntesis su valor decimal. Su tipo es cuatro y su tamaño dos.

2. *Conjunto de posibles implicados primos o lazos*

Está formado por todos los posibles implicados primos o lazos de una función booleana cualquiera de NV variables.

Ejemplo 2.2 Conjunto de lazos para dos variables = {02, 12, 20, 21, 00, 10, 01 y 11}

Una función booleana siempre puede ser expresada como la disyunción de determinados implicados primos, subconjunto del total de posibles implicados primos.

3. *Lazo cubierto y lazo adyacente*

Lazo cubierto. Decimos que un lazo cubre a otro si los minitérminos de éste último son cubiertos por el primero. El tipo del lazo cubierto debe ser mayor que el del lazo que lo cubre. Todos los bits inactivos del lazo cubierto deben aparecer en el lazo que lo cubre, sin embargo algunos o todos los bits activos del lazo cubierto deben aparecer como inactivos en el lazo que lo cubre.

Ejemplo 2.3 El lazo 1102 cubre a los lazos 1101 y 1100.

Lazo adyacente. Dos lazos son adyacentes cuando se distinguen en un solo bit activo que ocupa la misma posición: los lazos adyacentes son del mismo tipo. Un lazo tiene tantos lazos adyacentes como su tipo.

Ejemplo 2.4 El lazo 11022 es adyacente a 11122, y viceversa.

A la vista de estas definiciones se puede establecer la siguiente relación entre lazo e implicados primos:

- Si un lazo IDL se convierte en implicado primo de una función no lo serán aquéllos cubiertos por él.
- Si un lazo IDL es un implicado primo de una función no lo será ninguno de sus adyacentes. En efecto, supongamos que dos lazos adyacentes, IDL1 y IDL2, son implicados primos de una función, entonces lo sería también IDL1+IDL2. Pero al cubrir esta suma a IDL1 y IDL2 niega la supuesta condición de implicados primos de IDL1 y IDL2.

Ejemplo 2.5 Si el lazo 1121 es un implicado primo entonces no lo son ni 1111, ni 1101. Si 002 es un implicado primo no lo son sus adyacentes 012 y 102.

2.2.2 Concepto de rueda y términos relacionados

Además de los conceptos relativos a la ordenación de las ruedas entre sí y de los lazos dentro de una rueda, y de los relativos a las posiciones absoluta y realtiva de un lazo, el concepto de rueda tiene asociados ciertos términos que le caracterizan, tales como su identificador, número, tipo -que representaremos por IDR, NR y TIPOR, respectivamente- y su tamaño igual a 2^{TIPOR}.

Para facilitar la comprensión de estos conceptos, y de los presentados anteriormente, la tabla 2.1 muestra para cuatro variables:

- Los posibles implicados primos o lazos para cuatro variables, agrupados por ruedas -delimitadas por líneas horizontales.
- Las características IDR, NR y TIPOR de las ruedas resultantes.

Para mayor claridad junto al IDR se indica el correspondiente con letras, y para el IDA su valor decimal. También se han indicado el valor decimal de los minitérminos cubiertos por cada lazo.

	Relación ordenada de ruedas y lazos para cuatro variables					
POS.	TIPOR	NR	IDR	LAZO	IDA	MINITÉRMINOS
1	1	1	-222 A	0222	0	0, 4, 2, 6, 1, 5, 3, 7
2				1222	1	8, 12, 10, 14, 9, 13, 11, 15
3	1	2	2-22 B	2022	0	0, 8, 2, 10, 1, 9, 3, 11,
4				2122	1	4, 12, 6, 14, 5, 13, 7, 15
5	1	3	22-2 C	2202	0	0, 8, 4, 12, 1, 9, 5, 13
6				2212	1	2, 10, 6, 14, 3, 11. 7, 15
7	1	4	222- D	2220	0	0, 8, 4, 12, 2, 10, 6, 14
8				2221	1	1, 9, 5, 13, 3, 11, 7, 15
9	2	12	--22 AB	0022	0	0, 2, 1, 3
10				1022	1	8, 10, 9, 11
11				0122	2	4, 6, 5, 7
12				1122	3	12, 14, 13, 15
13	2	13	-2-2 AC	0202	0	0, 4, 1, 5
14				1202	1	8, 12, 9, 13
15				0212	2	2, 6, 3, 7
16				1212	3	10, 14, 11, 15
17	2	14	-22- AD	0220	0	0, 2, 4, 6
18				1220	1	8, 12, 10, 14
19				0221	2	1, 5, 3, 7
20				1221	3	9, 13, 11, 15

Tabla 2.1 Tabla de ruedas, lazos y casillas para cuatro variables

21	2	23	2--2 BC	2002	0	0, 8, 1, 9
22				2102	1	4, 12, 5, 13
23				2012	2	2, 10, 3, 11
24				2112	3	6, 14, 7, 15
25	2	24	2-2-BD	2020	0	0, 8, 2, 10
26				2120	1	4, 12, 6, 14
27				2021	2	1, 9, 3, 11
28				2121	3	5, 13, 7, 15
29	2	34	22-- CD	2200	0	0, 8, 4, 12
30				2210	1	2, 10, 6, 14
31				2201	2	1, 9, 5, 13
32				2211	3	3, 11, 7, 15
33	3	123	---2 ABC	0002	0	0, 1
34				1002	1	8, 9
35				0102	2	4, 5
36				1102	3	12, 13
37				0012	4	2, 3
38				1012	5	10, 11
39				0112	6	6, 7
40				1112	7	14, 15
41	3	124	--2- ABD	0020	0	0, 2
42				1020	1	8, 10
43				0120	2	4, 6
44				1120	3	12, 14
45				0021	4	1, 3
46				1021	5	9, 11
47				0121	6	5, 7
48				1121	7	13, 15
49	3	134	-2-- ACD	0200	0	0, 4
50				1200	1	8, 12
51				0210	2	2, 6
52				1210	3	10, 14
53				0201	4	1, 5
54				1201	5	9, 13
55				0211	6	3, 7
56				1211	7	11, 15
57	3	234	2--- BCD	2000	0	0, 8
58				2100	1	4, 12
59				2010	2	2, 10
60				2110	3	6, 14
61				2001	4	1, 9
62				2101	5	5, 13
63				2011	6	3, 11
64				2111	7	7, 15
65	4	1234	---- ABCD	0000	0	0
66				1000	1	8
67				0100	2	4
68				1100	3	12
69				0010	4	2
70				1010	5	10
71				0110	6	6

Tabla 2.1 (*continuación*)

72				1110	7	14
73				0001	8	1
74				1001	9	9
75				0101	10	5
76				1101	11	13
77				0011	12	3
78				1011	13	11
79				0111	14	7
80				1111	15	15

Tabla 2.1 (*continuación*)

Tal y como podemos observar en esta tabla, los posibles implicados primos para un determinado NV se pueden agrupar según el tipo de sus respectivos identificadores generales, y subagrupar cada grupo en subgrupos según el orden creciente del número formado por las posiciones decimales de los bits activos que tienen estos identificadores. Finalmente, los lazos de cada subgrupo se ordenan en el sentido creciente del valor decimal de sus identificadores activos.

La subagrupación de los lazos de mismo tipo en subgrupos ordenados entre sí permite introducir el concepto de rueda, y consecuentemente los términos relacionados con ella, y que utilizaremos a lo largo del capítulo.

La figura 2.1 muestra gráficamente la distribución de las ruedas y sus correspondientes lazos para una función booleana de tres variables. La figura 2.1.a relaciona cada rueda con aquellas que contienen lazos cubiertos por lazos pertenecientes a la primera. La figura 2.1.b es idéntica a la anterior, donde además aparecen para cada rueda sus lazos ordenados.

Concepto de rueda y términos relacionados

Una rueda es un conjunto ordenado de lazos con el mismo número total de bits activos que ocupan las mismas posiciones decimales en los identificadores generales de estos lazos. A cada rueda le están asociados los siguientes términos:

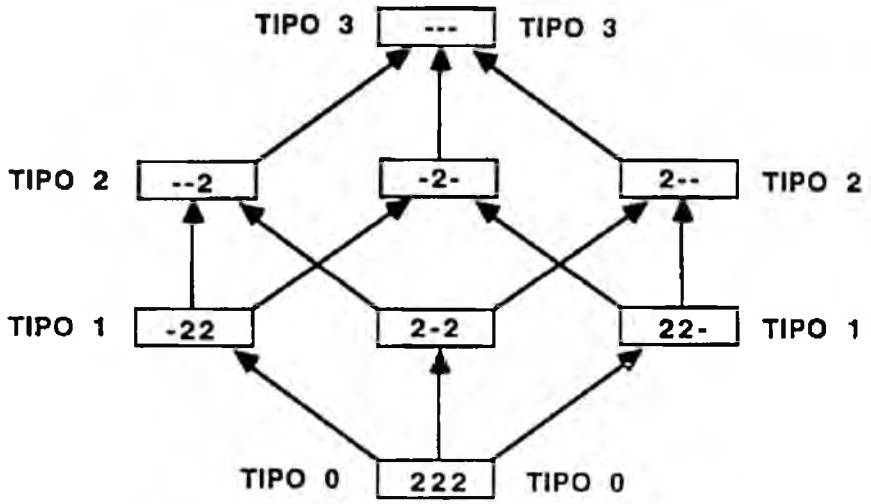


Figura 2.1.a Relación entre ruedas de tres variables

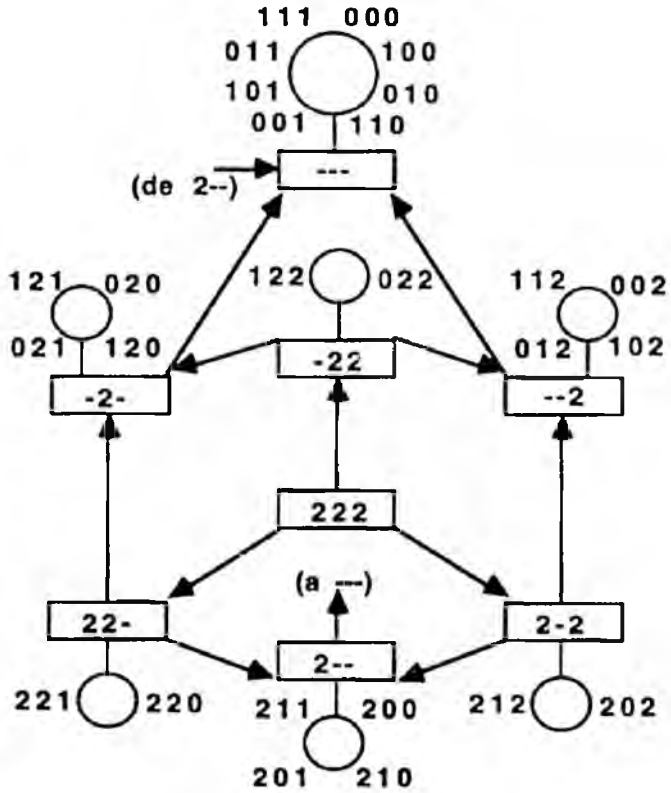


Figura 2.1.b Relación entre las ruedas y los lazos de tres variables

Identifcador de rueda, IDR. Se obtiene sustituyendo por guiones los bits activos de los lazos de una misma rueda. A cada IDR le puede corresponder un identificador de literales, compuesto por las letras cuyas posiciones tienen guiones.

Movimiento dentro de una rueda. La siguiente posición a la última dentro de una rueda es la primera, como muestra la figura 2.2.

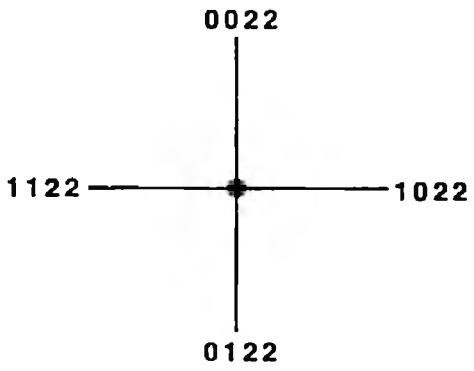


Figura 2.2 Ejemplo de rueda de cuatro elementos

Número de rueda, NR. Está formado por los índices de las posiciones de IDR con un guión. Es decir, NR es el número formado por las posiciones decimales de los bits activos de los lazos que componen la rueda.

Tipo de rueda, TIPOR. Es el número de guiones que aparecen en su identificador. El tamaño de una rueda, o número de lazos que contiene, es igual a 2^{TIPOR}.

Ejemplo 2.6 La rueda --22 (AB) y NR=13 es de tipo 2 y tiene cuatro lazos.

El concepto de rueda supone un ordenamiento de los lazos y de las ruedas entre sí. En efecto, las ruedas se clasifican por orden creciente de su tipo y de su NR -para las de igual tipo-; y los lazos dentro de cada rueda se clasifican por su IDA. Atendiendo a este ordenamiento, podemos referirnos a un lazo por su posición absoluta dentro del conjunto de lazos o por su posición relativa dentro de una rueda.

La posición absoluta de un lazo es la que ocupa su identificador IDL en el conjunto de lazos, y así IDL(i, j) representa al bit j-ésimo del identificador de posición i. Mientras que la posición relativa de un lazo se refiere a una rueda, así IDL(i, j, k) representa al bit k-ésimo del lazo que ocupa la posición j

dentro de la rueda de posición i. En el desarrollo de la tesis utilizaremos indistintamente un direccionamiento u otro.

Para finalizar esta sección, la tabla 2.2 relaciona para distintos números de variables el tipo de cada rueda con el número de lazos que contiene cada rueda y con el número de minitérminos que cubre cada lazo.

RUEDA		NÚMERO DE MINITÉRMINOS CUBIERTOS POR CADA LAZO								
Tipo	Tamaño	NV=2	NV=3	NV=4	NV=5	NV=6	NV=7	NV=8	NV=9	NV=10
1	2	2	4	8	16	32	64	128	256	512
2	4	1	2	4	8	16	32	64	128	256
3	8		1	2	4	8	16	32	64	128
4	16			1	2	4	8	16	32	64
5	32				1	2	4	8	16	32
6	64					1	2	4	8	16
7	128						1	2	4	8
8	256							1	2	4
9	512								1	2
10	1024									1

Tabla 2.2 Relación entre un VK de un número de variables con el tamaño y tipo de cada una de sus ruedas y con el tamaño de los lazos que pertenecen a las ruedas

2. 3 Fases en la obtención de los implicados primos

Para obtener los implicados primos de una función los métodos del capítulo 1 procesan todos los minitérminos cubiertos por la función. Por tanto, el esfuerzo para obtener los implicados primos de una función es claramente dependiente del número de minitérminos que cubre dicha función.

El nuevo método supone un planteamiento global distinto en la obtención de los implicados primos de una función. En este caso partiremos de todos los posibles implicados primos y seleccionaremos cuáles de ellos pertenecen a la función. El esfuerzo para obtener los implicados primos es siempre el mismo ya que, en principio, es necesario comprobar todos los posibles implicados primos. La labor de investigar todos los posibles implicados primos conlleva muchas operaciones, afirmando el carácter computacional del método.

Comparando con los otros métodos el esfuerzo es aparentemente mayor en el nuevo método. No obstante el esfuerzo requerido para comprobar un posible implicado primo en el nuevo método es mucho menor que el realizado por los otros métodos para obtener cada implicado primo a partir de los minitérminos, como se verá posteriormente en un estudio cuantitativo (sección 2.6.3).

La obtención de los implicados primos de una función mediante el nuevo método consta de las dos siguientes fases:

Primera fase: Crear MCVK (Matriz Característica de Veitch-Karnaugh)

Dicha matriz contiene toda la información referida a los posibles implicados primos, ordenada según el criterio de la rueda. Para cada número de variables, NV, existe una única matriz MCVK con:

$$\sum_{i=1}^{NV} 2^i \cdot C_i^{NV} \text{ número de filas}$$

Cada fila se refiere a un lazo formado por seis campos que consecutivamente contienen:

- Un índice de la fila.
- Un identificador.
- Los minitérminos que cubre cada lazo.
- Las posiciones en MCVK de los lazos cubiertos.
- Las posiciones en MCVK de sus lazos adyacentes.

Así pues, la tarea de crear la matriz MCVK consiste en generar esta información, conformada por las submatrices ID, CS, BA y BD que permiten obtener respectivamente el contenido de los sucesivos campos, partiendo únicamente de NV.

Segunda fase: Procesar MCVK

Presentamos un método de obtención de los implicados primos de una función procesando la información contenida en el MCVK correspondiente a su NV. Para ello se tiene en cuenta la siguiente secuencia:

- Si un lazo no ha sido borrado y cumple el criterio de selección es un implicado primo.

  Criterio de selección de un implicado primo: Un posible implicado primo L de MCVK, se convierte en un implicado primo de la función a procesar si todas las casillas por él cubiertas se corresponden con minitérminos o condiciones libres de la función -no siendo todo condiciones libres.

- Si un lazo se convierte en implicado primo, éste debe borrar los lazos por él cubiertos y sus lazos adyacentes.

Como se puede ver la matriz MCVK es el núcleo teórico-práctico en el marco de la obtención de los implicados primos que desarrollamos en esta tesis. Después de crearla y desarrollar el método propuesto para obtener los implicados primos, se hará un análisis comparativo de este método con los presentados en el capítulo anterior.

2.4 Generación de MCVK

Tal y como hemos señalado en la sección anterior la creación de MCVK exige la formación de las submatrices ID, CS, BD y BA. Previamente se necesitan ciertas matrices y vectores auxiliares que sirven para cuantificar las relaciones existentes entre las ruedas y los lazos dentro de MCVK.

El apéndice C relaciona los distintos nombres de variables y matrices con sus definiciones a modo de glosario.

2.4.1 Matrices, vectores y variables auxiliares previos a la creación de MCVK

Estas son las matrices RSML y NLB, los vectores NR, NL, LA y TLB, y las variables LT y RT.

2.4.1.1 Matriz RSML

Es una matriz cuadrada de NVxNV, triangular superior izquierda con $\frac{NVx(NV+1)}{2}$ elementos distintos de cero, siendo unos los elementos de la diagonal principal. Es el triángulo de Tartaglia o Pascal con una mera transformación formal. Cada elemento indica el número de ruedas seguidas de un mismo tipo cuyos IDR tienen el primer guión en la misma posición, o tienen la misma primera letra. Así, el elemento RSML(i, j) indica el número de ruedas seguidas de tipo i que tienen el primer guión en la posición j, es decir, que empiezan por la letra correspondiente a la posición j.

Algoritmo 2.1 *Generación de RSML*

La matriz es obtenida aplicando el siguiente algoritmo:

1. RSML(1,i)=1, con i=1,2....NV
2. RSML(i,j)=RSML(i,j+1)+RSML(i-1,j+1)
   con i=2,3..,NV y j=(NV-(i-1)),..2,1

Las siguientes matrices y vectores auxiliares se basan directa o indirectamente en RSML.

2.4.1.2 Vector NR

Es un vector con NV elementos no nulos. Cada elemento de este vector indica el número de ruedas de un determinado tipo. Así, el elemento NR(i) indica cuantas ruedas de tipo i va a tener la matriz MCVK.

Algoritmo 2.2 *Generación de NR*

El vector es obtenido aplicando el siguiente algoritmo:

$$NR(i) = \sum_{j=1}^{NV-(i-1)} RSML(i,j), \text{ con } i=1,2..NV$$

2.4.1.3 Vector RA

Es un vector con NV elementos no nulos. Cada elemento es el vector NR acumulado por posiciones. Así, el elemento RA(i) indica cuantas ruedas tiene MCVK cuyo tipo va desde 1 hasta i incluido.

Algoritmo 2.3 *Generación de RA*

El vector es obtenido aplicando el siguiente algoritmo:

1. RA(1)=NR(1)
2. RA(i)=RA(i-1)+NR(i), con i=2,3..,NV

2.4.1.4 Vector NL

Es un vector con NV elementos no nulos. Cada elemento indica el número de lazos de un determinado tipo. Así, el elemento NL(i) indica cuántos lazos de tipo i tiene la matriz MCVK.

Algoritmo 2.4 *Generación de NL*

La matriz es obtenida aplicando el siguiente algoritmo :

$$NL(i) = C_{NV}^{i} \cdot 2^{i}, \text{ con } i=1,2...NV$$

2.4.1.5 Vector LA

Es un vector con NV elementos distintos de cero. Cada elemento es el vector NL acumulado por posiciones. Así, el elemento LA(i) indica cuántos lazos tiene la matriz MCVK cuyo tipo va desde 1 hasta i incluido.

Algoritmo 2.5 *Generación de LA*

El vector es obtenido aplicando el siguiente algoritmo:

1. LA(1)=NL(1)
2. LA(i)=LA(i-1)+NL(i), con i=2,3..,NV

2.4.1.6 Variable LT

Esta variable indica el número de posibles implicados primos que puede tener una función booleana de NV variables -excepto el lazo que cubre 2^{NV} minitérminos.

Algoritmo 2.6 *Generación de LT*

La variable LT pude obtenerse de dos maneras:

1. $LT=3^{NV}-1$
2. LT=LA(NV)

2.4.1.7 Variable RT

Esta variable indica el número de ruedas de MCVK en las que se agrupan ordenadamente todos los lazos.

Algoritmo 2.7 *Generación de RT*

La variable RT puede obtenerse de dos maneras:

1. $RT=2^{NV}-1$
2. RT=RA(NV)

2.4.1.8 Matriz NLB

Es una matriz cuadrada de (NV-1)x(NV-1) con $\frac{(NV-1)xNV}{2}$ elementos distintos de cero y triangular superior. Cada elemento indica cuántos lazos de un determinado tipo son cubiertos. Así, el elemento NLB(i, j) indica

cuantos lazos de tipo j ,con j>i, cubre un lazo de tipo i. Cabe recordar que si un lazo se convierte en implicado primo, no pueden serlo los cubiertos por él, y así cada elemento de esta matriz indica cuántos lazos de determinado tipo deben ser borrados.

Algoritmo 2.8 *Generación de NLB*

La matriz es obtenida aplicando el siguiente algoritmo:

$$NLB(i,j)=C_{(NV-i)}^{(NV-j)} \cdot 2^{(j-i)}, \text{ con } i=1,2..,NV-1 \text{ y } j=i..,NV-1$$

2.4.1.9 Vector TLB

Es un vector con NV-1 elementos no nulos. Cada elemento representa a la matriz NLB acumulada por filas. Así, el elemento TLB(i) indica cuántos lazos de cualquier tipo deben ser borrados si un lazo de tipo i se convierte en implicado primo.

Algoritmo 2.9 *Generación de TLB*

El vector es obtenido aplicando el siguiente algoritmo:

$$TLB(i)=\sum_{j=1}^{NV} NLB(i,j), \text{ con } i=1,2...NV$$

Globalmente debemos destacar respecto de las matrices y vectores auxiliares que no tienen en cuenta el lazo que representa a un VK con todo unos: f=1. En principio puede parecer un caso particular, pero se trata del lazo que compone la rueda de tipo 0 y cubre 2^{NV} minitérminos. Si se considera también este lazo, las matrices: NR, RA, NL, LA, RT y LT tendrían una posición más, de índice 0 y valor 1. Por tanto, todos los valores acumulados quedarían incrementados en 1. Este aspecto particular se tiene en cuenta en la sección 2.6.2, aunque no es necesario ni en la creación y en el procesado de MCVK.

LEYENDA INTERPRETATIVA:

La columna TIPO indica el tipo de la rueda o lazo referido.

Las letras A, B..... de RSML indican por qué letra comienzan las ruedas referidas.

Los números 1, 2... de NLB indican cuántos lazos de ese tipo son borrados.

La columna seis comparte dos variables, la primera posición es LT y la segunda RT, ambas separadas por una línea

MATRICES AUXILIARES DE MCVK PARA NV=2

	RSML		NR	RA	NL	LA	LT y RT	NLB	TLB
TIPO	A	B						2	
1	1	1	2	2	4	4	8	2	2
2	1		1	3	4	8	3		
3									

MATRICES AUXILIARES DE MCVK PARANV=3

	RSML			NR	RA	NL	LA	LT y RT	NLB		TLB
TIPO	A	B	C						2	3	
1	1	1	1	3	3	6	6	26	4	4	8
2	2	1		3	6	12	18	7		2	2
3	1			1	7	8	26				

Tabla 2.3.a Matrices y vectores auxiliares de MCVK para NV=2 y NV=3

MATRICES AUXILIARES DE MCVK PARA NV=4

	RSML				NR	RA	NL	LA	LT y RT	NLB			TLB
TIPO	A	B	C	D						2	3	4	
1	1	1	1	1	4	4	8	8	80	6	12	8	26
2	3	2	1		6	10	24	32	15		4	4	8
3	3	1			4	14	32	64				2	2
4	1				1	15	16	80					

Tabla 2.3.b Matrices y vectores auxiliares de MCVK para NV=4

MATRICES AUXILIARES PARA NV=5

	RSML					NR	RA	NL	LA	LT y RT	NLB				TLB
TIPO	A	B	C	D	E						2	3	4	5	
1	1	1	1	1	1	5	5	10	10	243	8	24	16	32	80
2	4	3	2	1		10	15	40	50	31		6	12	8	26
3	6	3	1			10	25	80	130				4	4	8
4	4	1				5	30	80	210					2	2
5	1					1	31	32	242						

Tabla 2.3.c Matrices y vectores auxiliares de MCVK para NV=5

MATRICES AUXILIARES DE MCVK PARA NV=6

	RSML						NR	RA	NL	LA	LT y RT	NLB					TLB
TIPO	A	B	C	D	E	F						2	3	4	5	6	
1	1	1	1	1	1	1	6	6	12	12	728	10	40	80	80	32	242
2	5	4	3	2	1		15	21	60	72	63		8	24	16	32	80
3	10	6	3	1			20	41	160	232				6	12	8	26
4	10	4	1				15	56	240	472					4	4	8
5	5	1					6	62	192	664						2	2
6	1						1	63	64	728							

Tabla 2.3.d Matrices y vectores auxiliares de MCVK para NV=6

MATRICES AUXILIARES DE MCVK PARA NV=7

	RSML							NR	RA	NL	LA	LT y RT	NLB						TLB
TIPO	A	B	C	D	E	F	G						2	3	4	5	6	7	
1	1	1	1	1	1	1	1	7	7	14	14	2186	12	60	160	240	192	64	728
2	6	5	4	3	2	1		21	28	84	98	127		10	40	80	80	32	242
3	15	10	6	3	1			35	63	280	378				8	24	16	32	80
4	20	10	4	1				35	98	560	938					6	12	8	26
5	15	5	1					21	119	672	1610						4	4	8
6	6	1						7	126	448	2058							2	2
7	1							1	127	128	2186								

Tabla 2.3.e Matrices y vectores auxiliares de MCVK para NV=7

MATRICES AUXILIARES DE MCVK PARA NV=8

	RSML								NR	RA	NL	LA	LT y RT	NLB							TLB
TIPO	A	B	C	D	E	F	G	H						2	3	4	5	6	7	8	
1	1	1	1	1	1	1	1	1	8	8	16	16	6560	14	84	280	560	672	448	128	2186
2	7	6	5	4	3	2	1		28	36	112	128	255		12	60	160	240	192	64	728
3	21	15	10	6	3	1			56	92	448	576				10	40	80	80	32	242
4	35	20	10	4	1				70	162	1120	1696					8	24	16	32	80
5	35	15	5	1					56	218	1792	3488						6	12	8	26
6	21	6	1						28	246	1792	5280							4	4	8
7	7	1							8	254	1024	6304								2	2
8	1								1	255	256	6560									

Tabla 2.3.f Matrices y vectores auxiliares de MCVK para NV=8

MATRICES AUXILIARES DE MCVK PARA NV=9

	RSML									NR	RA	NL	LA	LT y RT	NLB								TLB
TIPO	A	B	C	D	E	F	G	H	I						2	3	4	5	6	7	8	9	
1	1	1	1	1	1	1	1	1	1	9	9	18	18	19682	16	112	448	1120	1792	1792	1024	256	6560
2	8	7	6	5	4	3	2	1		36	45	144	162	511		14	84	280	560	672	448	128	2186
3	28	21	15	10	6	3	1			84	129	672	834				12	60	160	240	192	64	728
4	56	35	20	10	4	1				126	255	2016	2850					10	40	80	80	32	242
5	70	35	15	5	1					126	381	4032	6882						8	24	16	32	80
6	56	21	6	1						84	465	5376	12258							6	12	8	26
7	28	7	1							36	501	4608	16866								4	4	8
8	8	1								9	510	2304	19170									2	2
9	1									1	511	512	19682										

Tabla 2.3.g Matrices y vectores auxiliares de MCVK para NV=9

MATRICES AUXILIARES DE MCVK PARA NV=10

	RSML										NR	RA	NL	LA	LT y RT	NLB									TLB
TIPO	A	B	C	D	E	F	G	H	I	J						2	3	4	5	6	7	8	9	10	
1	1	1	1	1	1	1	1	1	1	1	10	10	20	20	59048	18	144	672	2016	4032	5376	4608	2304	512	19682
2	9	8	7	6	5	4	3	2	1		45	55	180	200	1023		16	112	448	1120	1792	1792	1024	256	6560
3	36	28	21	15	10	6	3	1			120	175	960	1160				14	84	280	560	672	448	128	2186
4	84	56	35	20	10	4	1				210	385	3360	4520					12	60	160	240	192	64	728
5	126	70	35	15	5	1					252	637	8064	12584						10	40	80	80	32	242
6	126	56	21	6	1						210	847	13440	26024							8	24	16	32	80
7	84	28	7	1							120	967	15360	41384								6	12	8	26
8	36	8	1								45	1012	11520	52904									4	4	8
9	9	1									10	1022	5120	58024										2	2
10	1										1	1023	1024	59048											

Tabla 2.3.h Matrices y vectores auxiliares de MCVK para NV=10

2.4.2 Formación de las submatrices que integra MCVK

Toda vez que tenemos las matrices y vectores auxiliares de MCVK, queda generar la información propiamente dicha de MCVK: ID, CS, BD Y BA. Para ello contamos únicamente con NV, las matrices y los vectores auxiliares.

2.4.2.1 Matriz ID

La matriz ID contiene ordenados los identificadores generales de los lazos. Las ruedas se ordenan crecientemente por su tipo, y dentro de ellas sus lazos se ordenan crecientemente por su identificador activo.

Para generar la matriz ID previamente obtendremos los identificadores ordenados de las ruedas -matriz IR-, para finalmente asignar a cada rueda los identificadores generales de sus lazos.

1. *Obtención de la submatriz IR*

La submatriz IR es auxiliar por cuanto que su creación es previa y necesaria para obtener ID. Contiene las posiciones dei IDR de cada rueda que tienen un guión. Es decir, no contiene explícitamente cada IDR, sino la posición de los guiones en IDR, siendo el resto doses.

La tabla 2.4 muestra el contenido de IR para NV=4 y NV=5, las líneas horizontales muestran el cambio en el tipo de las ruedas; además aparecen explícitamente los IDR con el fin de aclarar la clasificación de las ruedas.

IR NV=4		
1		-222 A
2		2-22 B
3		22-2 C
4		222- D
1	2	--22 AB
1	3	-2-2 AC
1	4	-22- AD
2	3	2--2 BC
2	4	2-2- BD
3	4	22-- CD

IR NV=5		
1		-2222 A
2		2-222 B
3		22-22 C
4		222-2 D
5		2222- E
1	2	--222 AB
1	3	-2-22 AC
1	4	-22-2 AD
1	5	-222- AE
2	3	2--22 BC

Tabla 2.4 Matriz IR para NV=4 y NV=5

1	2	3		—2 ABC
1	2	4		–2- ABD
1	3	4		-2-- ACD
2	3	4		2— BCD
1	2	3	4	---- ABCD

2	4				2-2-2 BD
2	5				2-22- BE
3	4				22–2 CD
3	5				22-2- CE
4	5				222-- DE
1	2	3			—22 ABC
1	2	4			–2-2 ABD
1	2	5			–22- ABE
1	3	4			-2–2 ACD
1	3	5			-2-2- ACE
1	4	5			-22-- ADE
2	3	4			2—2 BCD
2	3	5			2–2- BCE
2	4	5			2-2-- BDE
3	4	5			22— CDE
1	2	3	4		—2 ABCD
1	2	3	5		—2- ABCE
1	2	4	5		–2-- ABDE
1	3	4	5		-2— ACDE
2	3	4	5		2--- BCDE
1	2	3	4	5	22222 ABCDE

Tabla 2 4 *(continuación)*

Observando la tabla 2.4, el elemento IR(i, j, k) indica cuál es la posición del k-ésimo guión de la rueda de posición j de tipo i. La matriz consta de NV bloques, cada bloque contiene NR(i) ruedas, con i=1,..,NV, y cada rueda tiene i elementos.

Ejemplo 2.7 Para cuatro variables IR(2, 2, 1)=1, indica que el primer guión de la segunda rueda de tipo 2 está en la primera posición, como se puede comprobar en la tabla 2.1.

Algoritmo 2.9 *Generación de IR*

El algoritmo de creación de IR es el siguiente:

```
1.  IR(1,i,1)=i, con i=1,2..,NR(1)
2.  para i=2,3,..,NV
3.  para j=1,2,..,NR(i)
4.  IR(i,k,m)=IR(i-1,j,m),
    con k=1,2,..(NV-IR(i-1,j,i-1)) y m=1,2,..,i-1
5.  IR(i,k,i)=k+IR(i-1,j,i-1),
    con k=1,2,..(NV-IR(i-1,j,i-1))
6.  fin de j
7.  fin de i
```

2. ***Obtención de la submatriz ID***

Partiendo de IR, donde tenemos ordenados los identificadores de ruedas, queda generar ordenadamente para cada identificador de rueda los identificadores de los lazos que componen dicha rueda. Como ha quedado enunciado en 2.2.2 los lazos de una rueda se obtienen sustituyendo los guiones de IDR por 0 ó 1. El orden de sustitución fue enunciado en base al IDA del lazo.

El IDA de cada lazo es fundamental en la ordenación de los lazos dentro de cada rueda, pero observamos en la tabla 2.2 que este orden gráficamente supone *'encajar una tabla de verdad'* de TIPOR variables en los guiones, donde el bit de la izquierda es el que varía su valor más rápidamente -es una tabla de verdad con los bits de cada fila invertidos de posición.

Ejemplo 2.8 El IDR de la tercera rueda de tipo 3 (número 13) es: -2--, entonces debe encajarse la tabla de verdad ordenada de tres variables: 000, 100, 010, 110, 001, 101, 011 y 111. Los guiones son sustituidos ordenadamente por cada terna, resultando que el contenido de ID para la rueda 13 es: 0200, 1200, 0210, 1210, 0201, 1201, 0211 y 1211, como podemos observar en la tabla 2.1.

Toda vez que la matriz haya sido creada, el elemento ID(i, j, k) contiene el bit k-ésimo del identificador general del lazo de posición j en la rueda i. Esta matriz tiene RT ruedas, cada una de ellas tiene 2^{TIPOR} lazos, teniendo cada uno de ellos NV bits identificadores.

Algoritmo 2.10 *Generación de ID*

El algoritmo de creación de ID supone:

1. para i=1,2,..,NV
2. para j=i,i+1,..,NV
3. para k=1,2,..,NR(j)
4. para m=1, 2, ..$2^{(j-i)}$
5. ID(NR(j-1)+k, (m-1)*2^{i}+n, IR(j,k,i))=0,
   con n=1,2,..,$2^{(i-1)}$

6. ID(NR(j-1)+k,(m-1)*2^i+n,IR(j,k,i))=1,
   con n=$2^{(i-1)}$+1, $2^{(i-1)}$+2,..,2^i
7. fin de m
8. fin de k
9. fin de j
10. fin de i

2.4.2.2 Matriz CS

Cada lazo cubre determinados minitérminos o casillas. La matriz CS contiene los minitérminos cubiertos por cada lazo de MCVK. Así, el elemento CS(i, j, k) contiene el valor del k-ésimo minitérmino cubierto por el lazo de posición j dentro de la rueda i. La matriz contiene RT ruedas, cada rueda contiene 2^{TIPOR} lazos, y cada lazo cubre $2^{(NV-TIPOR)}$ minitérminos.

Para obtener las casillas cubiertas por un lazo hay que sustituir los 2 de su identificadorpor todas las posibles combinaciones de 0 y 1, en cualquier orden, calculando finalmente el valor decimal de cada combinación resultante. De nuevo supone *'encajar una tabla de verdad'* de (NV-TIPOR) variables en las posiciones ocupadas por doses en cada identificador general. La tabla 2.2 muestra las casillas asociadas a cada lazo para cuatro variables.

Ejemplo 2.9 Al lazo 0122 le corresponde una tabla de verdad de dos variables: 00, 10, 01 y 11 -el orden es indiferente-, sustituyendo los doses obtenemos: 0100 (4), 0110 (6), 0101 (5) y 0111 (7); así las casillas cubiertas son: 4, 6, 5 y 7, como se puede comprobar en la tabla 2.1.

Algoritmo 2.11 *Generación de CS*

En el algoritmo utilizaremos una variable de trabajo ,VTRAB(i), que contendrá el identificador del lazo donde los 2 se van convirtiendo en 0 ó 1. El algoritmo supone los siguientes pasos:

1. para i=1,2,..,RT
2. para j=1,2,..,2^{TIPOR}
3. Si ID(i,j,m)=2,⇒ POS_2(k)=m

con k=1,2,..,(NV-TIPOR) y m=1, 2,..,NV

4. VTRAB(k)=ID(i,j,k), con k=1,2,..,NV
5. VTRAB(POS_2(k))=0, con k=1,2,..,(NV-TIPOR)
6. para k=1,2,..,$2^{(NV-TIPOR)}$
7. para m=1,2,..,(NV-TIPOR)
8. si ceil($\frac{k}{2^{(m-1)}}$) es par $\Rightarrow$ VTRAB(POS_2(m))=1
9. si ceil($\frac{k}{2^{(m-1)}}$) es impar $\Rightarrow$ VTRAB(POS_2(m))=0
10. fin de m
11. $CS(i, j, k)=\sum_{m=1}^{NV} VTRAB(m)\cdot 2^{(NV-m)}$
12. fin de k
13. fin de j
14. fin de i

2.4.2.3 Matriz BD

Recordemos que si un lazo se convierte en implicado primo sus adyacentes no lo podrán ser. Por otra parte, los lazos adyacentes entre sí se encuentran en la misma rueda. Es necesario que cada lazo conozca la posición de sus lazos adyacentes dentro de la propia rueda.

La obtención de BD es laboriosa. El centro teórico-práctico vuelve a ser la rueda; sobre todo que la siguiente posición a la última sea la primera. Primeramente, obtendremos para cada tipo de rueda la forma de relacionar cada posición de una rueda con sus adyacentes. Esta relación sólo depende del tipo de la rueda y de sus posiciones (IDA), y no de los identificadores generales que contiene, como veremos más adelante. Finalmente, y toda vez que tenemos el patrón de cada rueda, le asignamos su contenido a cada rueda de MCVK. El esfuerzo algorítmico se concentra en la primera fase, siendo la segunda fase una mera transformación de posiciones.

1. *Obtención de DR y DE*

La matriz DR contiene para cada tipo de rueda y lazo los desplazamientos a sumar a la posición del lazo para relacionarle con las posiciones de sus lazos adyacentes.

En la figura 2.3 se muestran ruedas de tipo 1, 2, 3 y 4 -de tamaño 2, 4, 8 y 16- cada posición se distingue por su IDA. Cada posición tiene asignados entre paréntesis los desplazamientos que hay que sumar a dicha posición para obtener las respectivas de sus lazos adyacentes. Si el resultado supera el tamaño de la rueda se le resta dicho tamaño, pues la siguiente posición a la última es la primera.

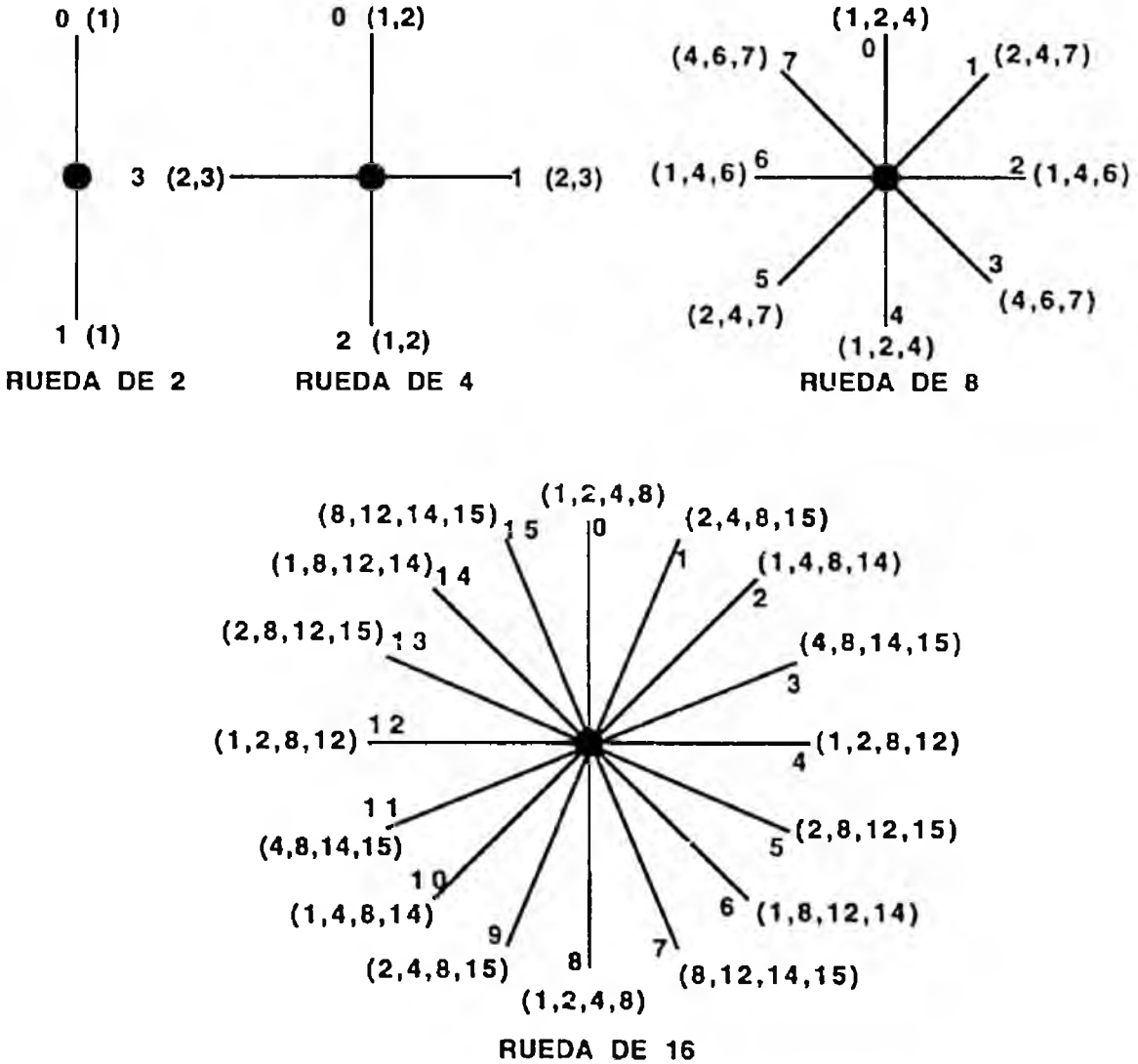


Figura 2.3. Ruedas de 2, 4, 8 y 16 posiciones

Ejemplo 2.10 Los lazos adyacentes al lazo de posición 6 en una rueda de tipo 4 son: 6+1=7, 6+8=14, 6+12=18 y 6+14=20; es necesario corregir los dos últimos resultados: 18-16=2 y 20-16=4.
Finalmente, las posiciones de los lazos adyacentes al sexto de una rueda de dieciséis son 7, 14, 2 y 4. Este resultado es fácilmente comprobable en la tabla 2.1.

Queda claro que todo se reduce a encontrar la ley de formación de dichos desplazamientos para almacenarlos en DR. El elemento DR(i, j, k) contiene el desplazamiento a sumar a j para obtener el k-ésimo lazo adyacente al de posición j dentro de la rueda de tipo i. La matriz tiene NV bloques, cada uno de éstos contiene 2^i lazos ,con i=1,..,NV, y a su vez cada lazo contiene i desplazamientos.

Pero la obtención de DR no es directa. Estudiando las ruedas presentadas en la figura 2.3, y otras de mayor tamaño, sistematizamos su ley de formación en los siguientes pasos:

- A cada lazo de la rueda le corresponden TIPOR desplazamientos.
- Generaremos los desplazamientos de la primera mitad de la rueda, porque el resto se repite.
- Asignaremos a la posición 0 los desplazamientos 2^0, 2^1, 2^2,....2^{n-1}.
- Los desplazamientos de las siguientes posiciones se obtienen quitando uno o varios de los desplazamientos de la posición 0, sustituyéndolos por la diferencia del tamaño de la rueda menos el elemento quitado:

  NUEVO DESPLAZAMIENTO=2^n-DESPLAZAMIENTO QUITADO

- Los desplazamientos a quitar en cada posición son aquellos cuyo valor coincide con el de la tabla 2.5. La tabla cubre ruedas cuyo tamaño no sobrepase los 64 lazos; aunque podría extenderse fácilmente:

DESPLAZAMIENTOS A ELIMINAR DE LA POSICIÓN 0 DE LA RUEDA PARA CADA LAZO				
1				
2				
2	1			
4				
4	1			
4	2			
4	2	1		
8				
8	1			
8	2			
8	2	1		
8	4			
8	4	1		
8	4	2		
8	4	2	1	
16				
16	1			
16	2			
16	2	1		
16	4			
16	4	1		
16	4	2		
16	4	2	1	
16	8			
16	8	1		
16	8	2		
16	8	2	1	
16	8	4		
16	8	4	1	
16	8	4	2	
16	8	4	2	1
....	....	....	....	....

Tabla 2.5 Matriz DE

Para una mejor comprensión de las anteriores fases, en la tabla adjunta se muestra cada una de las fases en la creación de DR para cuatro variables partiendo de las siete primeras filas de la tabla 2.5; las líneas horizontales delimitan el cambio de rueda:

INICIO	FASE 1	FASE 2	FASE 3	FASE 4
IDA de cada lazo	Desplaz. Iniciales	Desplaz. a Eliminar	Nuevo Desplaz.	Simetría Desplaz.
0	1		1	1
1				1

Tabla 2.6 Fases de obtención y contenido de DR

0	1, 2		1, 2	1, 2
1	1, 2	1	2, 3	2, 3
2				1, 2
3				2, 3
0	1, 2, 4		1, 2, 4	1, 2, 4
1	1, 2, 4	1	2, 4, 7	2, 4, 7
2	1, 2, 4	2	1, 4, 6	1, 4, 6
3	1, 2, 4	2, 1	4, 6, 7	4, 6, 7
4				1, 2, 4
5				2, 4, 7
6				1, 4, 6
7				4, 6, 7
0	1, 2, 4, 8		1, 2, 4, 8	1, 2, 4, 8
1	1, 2, 4, 8	1	2, 4, 8, 15	2, 4, 8, 15
2	1, 2, 4, 8	2	1, 4, 8, 14	1, 4, 8, 14
3	1, 2, 4, 8	2, 1	4, 8, 14, 15	4, 8, 14, 15
4	1, 2, 4, 8	4	1, 2, 8, 15	1, 2, 8, 15
5	1, 2, 4, 8	4, 1	2, 8, 12, 15	2, 8, 12, 15
6	1, 2, 4, 8	4, 2	1, 8, 12, 14	1, 8, 12, 14
7	1, 2, 4, 8	4, 2, 1	8, 12, 14, 15	8, 12, 14, 15
8				1, 2, 4, 8
9				2, 4, 8, 15
10				1, 4, 8, 14
11				4, 8, 14, 15
12				1, 2, 8, 15
13				2, 8, 12, 15
14				1, 8, 12, 14
15				8, 12, 14, 15

Tabla 2.6 (*continuación*)

Así pues, obtener DR supone obtener una nueva matriz que denominaremos DE. El elemento DE(i, j) contiene el j-ésimo desplazamiento a sustituir en el lazo de posición i dentro de la correspondiente rueda de DR.

Algoritmo 2.12 *Generación de DE*

El algoritmo de creación de DE utiliza tres índices, con una interpretación puramente gráfica, y supone los siguientes pasos:

1. DE(1,1,1)=1
2. para i=2,3,..,(NV-1)
3. DE(i,1,j)=$2^{(i-1)}$, con j=1,2,..,$2^{(i-1)}$
4. DE(i,j,k)=DE(p,m,n),
   con j=2,3,..,$2^{(i-1)}$, k=2,3,.., p=1,2,..,(i-1),
   m=1,2,..,$2^{(i-1)}$-1 y n=1,2,..
5. fin de i

Algoritmo 2.13 *Generación de DR*

La matriz DE ha sido creada con tres índices, pero la usaremos con dos -la transformación correspondiente se encuentra en 2.5.1. El algoritmo de creación de DR supone los siguientes pasos:

1. para i=1,2,..,NV
2. DR(i,1,j)=2^k, con k=0,1,..,(i-1) y j=1,2,..,i
3. DR(i,1+$2^{(i-1)}$,j)=2^k, con k=0,1,..,i-1 y j=1,2,..,i
4. VTRAB(j)=DR(i,1,j), con j=1,2,..,i
5. para j=2, 3, ..$2^{(i-1)}$
6. Si VTRAB(m)=DE(j,k) ⇒ PE(k)=m con k=1,2,.. y m=1,2,..,i
7. VTRAB(PE(k))=2^i-DE(j,k), con k=1,2,..
8. DR(i,j,k)=VTRAB(k), con k=1,2,..,i
9. DR(i,j+$2^{(i-1)}$,k)=VTRAB(k), con k=1,2,..,i
10. fin de j
11. fin de i

2. *Creación de la submatriz BD*

La matriz DR contiene para cada tipo de rueda el desplazamiento a sumar a la posición de cada lazo para obtener las posiciones dentro de la rueda de sus lazos adyacentes. Por tanto, resta asignar a todas las ruedas de cada tipo de MCVK el contenido de la rueda de mismo tipo en DR, es decir, simplemente repetir el contenido. Además, en vez de asignar los desplazamientos -dirección implícita-, sumaremos dichos desplazamientos a cada posición para obtener la posición explícita de los lazos adyacentes. El anteúltimo campo de la tabla 2.15 es BD para cuatro variables.

Cada elemento BD(i, j, k) contiene la posición del j-ésimo lazo adyacente al lazo de posición j dentro de la rueda i. La matriz tiene RT ruedas, cada una de éstas tiene 2^{TIPOR} lazos, y cada uno de éstos direcciona TIPOR lazos adyacentes.

Ejemplo 2.11 Para cuatro variables cada lazo de una rueda de tipo 3 se relaciona con tres lazos adyacentes. BD(12, 2, 2)=4 indica que el segundo lazo adyacente al segundo lazo de la duodécima

rueda, que es de tipo 3, se encuentra en la cuarta posición de dicha rueda.
Este dato se obtiene como la suma de la posición del lazo dentro de la rueda mas el desplazamiento correspondiente dentro de una rueda de tipo 3: poslazo+DR(3,2,2)=2+2=4.

Algoritmo 2.14 Generación de BD

El algoritmo supone los siguientes pasos:

1. para i=1,2,..,RT
2. para j=1,2,..,2^{TIPOR}
3. VTRAB(k)=j+DR(TIPOR,j,k), con k=1,2,..,TIPOR
4. Si VTRAB(k)>2^{TIPOR}+1 $\Rightarrow$ VTRAB(k)=2^{TIPOR}-VTRAB(k), con k=1,2,..,TIPOR
5. BD(i,j,k)=VTRAB(k), con k=1,2,..,TIPOR
6. fin de j
7. fin de i

En el cuarto paso la cantidad '+1' se hace necesaria para ajustar el ordenamiento de la rueda de 1 a 2^n, en vez de el 0 a 2^n-1 utilizado en la ley de formación de BD.

2.4.2.4 Matriz BA

Recordemos que si un lazo se convierte en implicado primo entonces los lazos por él cubiertos deben ser borrados. La matriz BA contiene para cada lazo las posiciones de los lazos por él cubiertos.

La obtención de BA se divide en dos fases:

1. Identificar para cada rueda las ruedas en las que se encuentran los lazos cubiertos por los lazos de la primera: Matriz PRA
2. Identificar dentro de la rueda obtenida en la fase 1 las posiciones donde se encuentran los lazos cubiertos: Matriz BA

La primera fase concentra el esfuerzo algorítmico, siendo la segunda una mera transformación de posiciones.

1. *Obtención de PRA*

Cada rueda es un conjunto de lazos; la matriz PRA indica para cada rueda en qué rueda se encuentran los lazos cubiertos por lazos que pertenecen a la primera rueda. Más sencillamente podemos decir que PRA indica qué *ruedas son cubiertas* por cada una de ellas.

Para ilustrar la estructura de la matriz PRA se han formado en la tabla 2.7 las matrices PRA para NV igual a dos hasta seis. La matriz tiene NV columnas y está particionada en (NV-1) bloques-fila correspondientes sucesivamente a los distintos tipos de rueda que resultan para un determinado valor de NV. Si consideramos el i-ésimo bloque-fila:

- Los elementos de la primera columna son todas las ruedas de tipo i, ocupando cada una de ellas una posición j. Cada elemento se anota (i, j).
- Los elementos de las restantes columnas representan las ruedas de tipo (i+1) hasta NV que *cubre* la rueda (i, j). Para señalar qué ruedas de tipo i+k , con k=1, 2, ... NV-1, *están cubiertas* se anota el valor i+k delante de cada corchete -aunque no es necesario- y dentro del corchete se anotan las posiciones de las ruedas.

Por ejemplo, para cuatro variables las entradas (2,3), 3[2,3] y 4[1] de la séptima fila indican respectivamente que la tercera rueda de tipo 2 -paréntesis- *cubre* a la segunda y tercera rueda de tipo 3 -primer corchete- y a la primera rueda de tipo 4 -segundo corchete. Dicho en otras palabras, todos los lazos de la tercera rueda de tipo 2 *cubren* lazos contenidos en la segunda y tercera rueda de tipo 3, y en la primera de tipo 4

O sea, la matriz PRA tiene NV-1 bloques-fila -i=1,..,NV-1- con NR(i) ruedas cada uno de ellos, cada rueda se relaciona con j=1,..,NV-i bloques-columna y cada uno de estos contiene las posiciones de NR(k+i) ruedas. No existe el bloque-fila NV ya que una rueda de tipo NV no cubre a ninguna otra rueda.

La estructura de PRA mostrada en las tablas 2.7 queda implementada en una matriz de cuatro índices. Así PRA(i, j, k, m) indica que la rueda m-ésima de tipo (k+i) está cubierta por la j-ésima rueda de tipo i, o también que la rueda de posición j dentro de las de tipo i cubre a la m-ésima rueda dentro de las de tipo (k+i).

Observando la matriz PRA respecto de la tabla 2.7 resulta que el elemento PRA(i, j, k, m)=posrueda se relaciona con el corchete situado en el i-ésimo bloque-fila, dentro de éste en la fila j y (k+1)-ésimo bloque columna. Dentro de este corchete la posición m-ésima es el valor posrueda.

Ejemplo 2.12 Para cuatro variables PRA(2, 5, 1, 2)=4 indica que todos los lazos de la quinta rueda de tipo 2 -dos primero índices- *cubren* a lazos contenidos en la cuarta rueda de tipo 3 -i+k=3-, además de otros lazos en otras ruedas.

Si leemos en la tabla 2.7.c nos referimos a la novena fila: (2,5) 3[2,4] y 4[1], y dentro de ella a cuarta rueda de tipo 3 -en negrita.

Lo anterior puede ser comprobado en la tabla 2.1. La quinta rueda de tipo 2 se corresponde con las filas 21 a 24, si tomamos cualquiera de estos cuatro lazos veremos que los lazos por él cubiertos se encuentran en la cuarta rueda de tipo 3 -filas 57 a 64.

DOS VARIABLES	
POSICIÓN DEL LAZO	POSICIÓN BORRAR
(1,1)	2[1]
(1,2)	2[1]

Tabla 2.7.a Matriz PRA para NV=2

TRES VARIABLES		
POSICIÓN DE LAZO	POSICIÓN PARA BORRAR EN LA RUEDA	
(1,1)	2[1,2]	3[1]
(1,2)	2[1,3]	3[1]
(1,3)	2[2,3]	3[1]
(2,1)	3[1]	
(2,2)	3[1]	
(2,3)	3[1]	

Tabla 2.7.b Matriz PRA para NV=3

CUATRO VARIABLES			
POSICIÓN DE L	POSICIÓN PARA BORRAR EN LA RUEDA		
(1,1)	2[1,2,3]	3[1,2,3]	4[1]
(1,2)	2[1,4,5]	3[1,2,4]	4[1]
(1,3)	2[2,4,6]	3[1,3,4]	4[1]
(1,4)	2[3,5,6]	3[2,3,4]	4[1]
(2,1)	3[1,2]	4[1]	
(2,2)	3[1,3]	4[1]	
(2,3)	3[2,3]	4[1]	
(2,4)	3[1,4]	4[1]	
(2,5)	3[2,4]	4[1]	
(2,6)	3[3,4]	4[1]	
(3,1-4)	4[1]		

Tabla 2.7.c Matriz PRA para NV=4

CINCO VARIABLES				
POSICIÓN DE L	POSICIÓN PARA BORRAR EN LA RUEDA			
(1,1)	2[1,2,3,4]	3[1,2,3,4,5,6]	4[1,2,3,4]	5[1]
(1,2)	2[1,5,6,7]	3[1,2,3,7,8,9]	4[1,2,3,5]	5[1]
(1,3)	2[2,5,8,9]	3[1,4,5,7,8,10]	4[1,2,4,5]	5[1]
(1,4)	2[3,6,8,10]	3[2,4,6,7,9,10]	4[1,3,4,5]	5[1]
(1,5)	2[4,7,9,10]	3[3,5,6,8,9,10]	4[2,3,4,5]	5[1]
(2,1)	3[1,2,3]	4[1,2,3]	5[1]	
(2,2)	3[1,4,5]	4[1,2,4]	5[1]	
(2,3)	3[2,4,6]	4[1,3,4]	5[1]	
(2,4)	3[3,5,6]	4[2,3,4]	5[1]	
(2,5)	3[1,7,8]	4[1,2,5]	5[1]	
(2,6)	3[2,7,9]	4[1,3,5]	5[1]	
(2,7)	3[3,8,9]	4[2,3,5]	5[1]	
(2,8)	3[4,7,10]	4[1,4,5]	5[1]	
(2,9)	3[5,8,10]	4[2,4,5]	5[1]	
(2,10)	3[6,9,10]	4[3,4,5]	5[1]	
(3,1)	4[1,2]	5[1]		
(3,2)	4[1,3]	5[1]		
(3,3)	4[2,3]	5[1]		
(3,4)	4[1,4]	5[1]		
(3,5)	4[2,4]	5[1]		
(3,6)	4[3,4]	5[1]		
(3,7)	4[1,5]	5[1]		
(3,8)	4[2,5]	5[1]		
(3,9)	4[3,5]	5[1]		
(3,10)	4[4,5]	5[1]		
(4,1-5)	5[1]			

Tabla 2.7.d Matriz PRA para NV=5

SEIS VARIABLES					
POS L	POSICIÓN PARA BORRAR EN LA RUEDA				
(1,1)	2[1,2,3,4,5]	3[1,2,3,4,5,6,7,8,9,10]	4[1,2,3,4,5,6,7,8,9,10]	5[1,2,3,4,5]	6[1]
(1,2)	2[1,6,7,8,9]	3[1,2,3,4,11,12,13,14,15,16]	4[1,2,3,4,5,6,11,12,13,14]	5[1,2,3,4,6]	6[1]
(1,3)	2[2,6,10,11,12]	3[1,5,6,7,11,12,13,17,18,19]	4[1,2,3,7,8,9,11,12,13,15]	5[1,2,3,5,6]	6[1]
(1,4)	2[3,7,10,13,14]	3[2,5,8,9,11,14,15,17,18,20]	4[1,4,5,7,8,10,11,12,14,15]	5[1,2,4,5,6]	6[1]
(1,5)	2[4,8,11,13,15]	3[3,6,8,10,12,14,16,17,19,20]	4[2,4,6,7,9,10,11,13,14,15]	5[1,3,4,5,6]	6[1]
(1,6)	2[5,9,12,14,15]	3[4,7,9,10,13,15,16,18,19,20]	4[3,5,6,8,9,10,12,13,14,15]	5[2,3,4,5,6]	6[1]
(2,1)	3[1,2,3,4]	4[1,2,3,4,5,6]	5[1,2,3,4]	6[1]	
(2,2)	3[1,5,6,7]	4[1,2,3,7,8,9]	5[1,2,3,5]	6[1]	
(2,3)	3[2,5,8,9]	4[1,4,5,7,8,10]	5[1,2,4,5]	6[1]	
(2,4)	3[3,6,8,10]	4[2,4,6,7,9,10]	5[1,3,4,5]	6[1]	
(2,5)	3[4,7,9,10]	4[3,5,6,8,9,10]	5[2,3,4,5]	6[1]	
(2,6)	3[1,11,12,13]	4[1,2,3,11,12,13]	5[1,2,3,6]	6[1]	
(2,7)	3[2,11,14,15]	4[1,4,5,11,12,14]	5[1,2,4,6]	6[1]	
(2,8)	3[3,12,14,16]	4[2,4,6,11,13,14]	5[1,3,4,6]	6[1]	
(2,9)	3[4,13,15,16]	4[3,5,6,12,13,14]	5[2,3,4,6]	6[1]	
(2,10)	3[5,11,17,18]	4[1,7,8,11,12,15]	5[1,2,5,6]	6[1]	
(2,11)	3[6,12,17,19]	4[2,7,9,11,13,15]	5[1,3,5,6]	6[1]	
(2,12)	3[7,13,18,19]	4[3,8,9,12,13,15]	5[2,3,5,6]	6[1]	
(2,13)	3[8,14,17,20]	4[4,7,10,11,14,15]	5[1,4,5,6]	6[1]	
(2,14)	3[9,15,18,20]	4[5,8,10,12,14,15]	5[2,4,5,6]	6[1]	
(2,15)	3[10,16,19,20]	4[6,9,10,13,14,15]	5[3,4,5,6]	6[1]	
(3,1)	4[1,2,3]	5[1,2,3]	6[1]		
(3,2)	4[1,4,5]	5[1,2,4]	6[1]		
(3,3)	4[2,4,6]	5[1,3,4]	6[1]		
(3,4)	4[3,5,6]	5[2,3,4]	6[1]		
(3,5)	4[1,7,8]	5[1,2,5]	6[1]		
(3,6)	4[2,7,9]	5[1,3,5]	6[1]		
(3,7)	4[3,8,9]	5[2,3,5]	6[1]		
(3,8)	4[4,7,10]	5[1,4,5]	6[1]		
(3,9)	4[5,8,10]	5[2,4,5]	6[1]		
(3,10)	4[6,9,10]	5[3,4,5]	6[1]		
(3,11)	4[1,11,12]	5[1,2,6]	6[1]		
(3,12)	4[2,11,13]	5[1,3,6]	6[1]		
(3,13)	4[3,12,13]	5[2,3,6]	6[1]		
(3,14)	4[4,11,14]	5[1,4,6]	6[1]		
(3,15)	4[5,12,14]	5[2,4,6]	6[1]		
(3,16)	4[6,13,14]	5[3,4,6]	6[1]		
(3,17)	4[7,11,15]	5[1,5,6]	6[1]		
(3,18)	4[8,12,15]	5[2,5,6]	6[1]		
(3,19)	4[9,13,15]	5[3,5,6]	6[1]		
(3,20)	4[10,14,15]	5[4,5,6]	6[1]		
(4,1)	5[1,2]	6[1]			
(4,2)	5[1,3]	6[1]			
(4,3)	5[2,3]	6[1]			
(4,4)	5[1,4]	6[1]			
(4,5)	5[2,4]	6[1]			
(4,6)	5[3,4]	6[1]			
(4,7)	5[1,5]	6[1]			
(4,8)	5[2,5]	6[1]			
(4,9)	5[3,5]	6[1]			

Tabla 2.7.e Matriz PRA para NV=6

(4,10)	5[4,5]	6[1]			
(4,11)	5[1,6]	6[1]			
(4,12)	5[2,6]	6[1]			
(4,13)	5[3,6]	6[1]			
(4,14)	5[4,6]	6[1]			
(4,15)	5[5,6]	6[1]			
(5,1-6)	6[1]				

Tabla 2.7.e *(continuación)*

Algoritmo 2.15 *Generación de PRA*

Para obtener la matriz PRA de NV variables es necesaria la misma matriz para NV-1 variables, a la que llamaremos PRBV, y que habrá sido previamente almacenada.

El algoritmo de creación de PRA supone los siguientes pasos:

1. PRA(1,1,1,i)=i, con i=1,2,..,RSML(2, 1)
2. PRA(1,i+1,,1,1)=i, con i=1,2,..,RSML(2, 1)
3. PRA(1,i+1,1,j)=PRBV(1,i,1,j-1)+RSML(1+1,1),
   con i=1,2,..,RSML(2,1) y j=2,3,..,NV-1
4. para i=2,3,..,NV-1
5. PRA(1,1,1,j)=j, con j=1,2,..RSML(i+1,1)
6. PRA(1,j,i,k)=PRBV(1,j-1,i-1,k),
   con j=2,3,..,RSML(2,1) y k=1,2,..,RSML(j, 2)
7. PRA(1,j,i,k)=PRBV(1,j-1,i,k-RSML(j,2)),
   con j=2,3,..,RSML(2,1) y k=RSML(j,2),..,RSML(j,1)
8. fin de i
9. para i=2,3,..,NV-1
10. PRA(i,j,1,k)=PRBV(i-1,j,1,k),
    con j=1,2,..,RSML(i,1) y k=1,2,..,nv-i
11. PRA(i,RSML(i,1)+j,1,1)=j, con j=1,2,..,NR(i)
12. PRA(i,RSML(i,1)+j,1,k)=PRBV(i,j+1,1,k),
    con j=1,2,..,RSML(i+1,1) y k=2,3,..,NV-i
13. para j=2, 3, ..NV-i
14. PRA(i,k,j,m)=PRBV(i-1,k,j,m),
    con k=1,2,..,RSML(i,1) y m=1,2,..,RSML(i+1,j)
15. PRA(i,RSML(i,1)+k,j,m)=PRBV(i,k,j-1,m),
    con k=1,2,..,RSML(i+1,1) y m=1,2,..,RSML(i,j)
16. PRA(i,RSML(i+1,1)+k,j,m)=PRBV(i,k,j,m)+RSML(i+j, 1),

```
      con k=1,2,..,RSML(i+1,1) y m=1,2,..,RSML(i,j+1)
17.  fin de j
18.  fin de i
```

2. *Creación de* BA

La matriz PRA relaciona ruedas con ruedas, pero el objetivo final es relacionar lazos con lazos. Comoquiera que cada lazo pertenece a una rueda, y que la matriz PRA indica cuáles son las ruedas que contienen lazos cubiertos por el primero, sólo resta conocer la posición de los lazos cubiertos dentro de esas ruedas. Pasar de PRA a BA supone cambiar el direccionado por ruedas al direccionado por lazos.

El elemento BA(i, j, k, m) contiene para m=2 la posición dentro de la rueda indicada por m=1 donde se encuentra el k-ésimo lazo cubierto; donde el lazo *que cubre* se encuentra en la posición j de la rueda i. La matriz BA tiene i=1, ..., RA(NV-1) ruedas de distinto tipo: cada rueda tiene 2^{TIPOR} lazos y cada lazo cubre TLB(i) lazos, definidos cada uno por dos posiciones.

El contenido de la matriz BA para cuatro variables aparece en el último campo de la tabla 2.15.

La tabla 2.8 muestra cómo está relacionado el lazo 2020 con las matrices necesarias para determinar qué lazos están cubiertos por él. El ejemplo se basa en el lazo 2020 perteneciente a la quinta rueda de tipo 2 -IR=2-2-. La primera y segunda columna indican las *ruedas cubiertas* por 2-2-; el lazo o lazos cubiertos se encuentran en dichas ruedas y pueden ser buscados ***secuencialmente***. Por contra, la cuarta columna direcciona ***directamente*** mediante la matriz BA la posición del lazo o lazos cubiertos.

Ejemplo 2.13 La matriz BA utiliza posiciones absolutas de ruedas, y así la quinta rueda de tipo 2 es la novena del conjunto global de ruedas. El elemento BA(9, 1, 1, 1)=12 indica que el primer lazo de la novena rueda cubre a un primer lazo en la duodécima rueda, mientras que BA(9, 1, 1, 2)=1 indica que ese primer lazo cubierto se encuentra en la primera posición de la duodécima rueda.

La tabla 2.1 nos muestra que la primera posición de la duodécima rueda -filas 41 a 48- corresponde a 0020, lazo cubierto por 2020.

Ejemplo 2.14 Para cuatro variables BA(9, 1, 6, 1)=15 y BA(9, 1, 6, 2)=2 indica que el primer lazo de la novena rueda cubre al segundo lazo de la decimoquinta rueda, siendo éste el sexto lazo que cubre, como se puede observar en la tabla 2.1.

PRA para TIPO=2 RUEDA=5	RUEDAS BORRADAS POR EL PRIMER LAZO	LAZOS ELEGIDOS DE LA RUEDA BORRADA	CONTENIDO DE BORRARBAJO
PRA(2,5,1,1)=2	TIPO=3 RUEDA=2	0020	BA(9,1,1,1)=12 BA(9,1,1,2)=1
		1020	BA(9,1,2,1)=12 BA(9,1,2,2)=2
PRA(2,5,1,2)=4	TIPO=3 RUEDA=4	2000	BA(9,1,3,1)=14 BA(9,1,3,2)=1
		2010	BA(9,1,4,1)=14 BA(9,1,4,2)=3
PRA(2,5,2,1)=1	TIPO=4 RUEDA=1	0000	BA(9,1,5,1)=15 BA(9,1,5,2)=1
		1000	BA(9,1,6,1)=15 BA(9,1,6,2)=2
		0010	BA(9,1,7,1)=15 BA(9,1,7,2)=5
		1010	BA(9,1,8,1)=15 BA(9,1,8,2)=6

Tabla 2.8 Ejemplo de creación de BA

Algoritmo 2.16 *Generación de BA*

Este algoritmo utiliza de nuevo el concepto de *encajar una tabla de verdad* en determinadas posiciones; utilizaremos la variable TVERDAD(i, j, k), que representa al bit k-ésimo de la fila j de una tabla de verdad de i variables. Además, en el algoritmo la asignación de pesos a los bits es contraria a la normal, el de más peso está a la derecha.

El algoritmo de creación de BA supone los siguientes pasos:

1. `para i=1,2,..,RA(NV-1)`
2. `para j=1,2,..,2^TIPOR`
3. `para k=i+1,i+2,..,NV`
4. `para l=1,2,..,RSML(k,TIPOR)`

5. BA(i,j,m,1)=RA(k-1)+PRA(TIPOR,j,k-TIPOR,1), con m=1,2,..,NLB(TIPOR,k)
6. para m=1,2,..,NLB(TIPOR,k)
7. VTRAB(n)=0, con n=1,2,..,k
8. Si ID(i,j,p)≠2 ⇒ POS_10(n)=p con p=1,2,..,NV y n=1,2,..,TIPOR
9. Si ID(i,j,p)=2 ⇒ POS_2(n)=p con p=1,2,..,NV y n=1,2,..,TIPOR
10. VTRAB(POS_10(n))=ID(i,j,POS_10(n)), con n=1,2,..,TIPOR
11. VTRAB(POS_2(n))=TVERDAD(NV-TIPOR,m,n),con n=1,2,..,(NV-TIPOR)
12. $BA(i,j,m,2)=\sum_{p=1}^{TIPOR} VTRAB(p)\cdot 2^{(TIPOR-p)}$
13. fin de m
14. fin de k
15. fin de j
16. fin de i

2.5 Problemática de la implementación en MATLAB de MCVK

En la sección 2.4 hemos desarrollado los algoritmos que generan las distintas submatrices de MCVK, partiendo únicamente de NV. En ellos, y para referirnos a las submatrices hemos utilizado tres o cuatro índices con el fin de asegurarnos una mejor comprensión del algoritmo. Sin embargo, la implementación de estos algoritmos puede exigir el uso de matrices de dos dimensiones -como es el caso del entorno MATLAB elegido en la tesis. El uso de matrices de dos índices conlleva:

- La explicación de la nueva estructura.
- La conversión entre los índices.
- Un estudio teórico del tamaño y densidad de las matrices.
- Comparar los resultados teóricos con su implementación en MATLAB.

Por otra parte, y observando la figura 2.1, queda claro que la matriz MCVK puede ser implementada utilizando árboles binarios y OBDD's -según (Bryant, 1986), (Liaw y Lin, 1992) y (Chakravarty, 1993). Una comparación entre ambas técnicas -la utilizada y la orientada a árboles binarios- determinará su idoneidad.

2.5.1 Nueva estructura de MCVK y conversión entre índices

En este apartado enunciaremos el nuevo significado de cada submatriz y cómo convertir cada una de ellas con tres o cuatro índices en la misma submatriz, pero con dos índices.

En una matriz de dos índices cada lazo es distinguido por su posición en el conjunto total de lazos, y no por la posición que ocupa dentro de una determinada rueda -caso de los algoritmos-; asimismo cada rueda es distinguida por su posición en el conjunto total de ruedas, y no por la posición que ocupa entre las de su tipo. La conversión consiste entonces en transformar dichos direccionamientos.

1. Matriz IR

Tiene tantas filas como ruedas y cada fila tiene como máximo NV columnas. El contenido de cada fila es la posición de los guiones dentro del identificador de la rueda. La conversión se realiza como sigue:

```
IR(i,j)=RA(k-1)+IR(k,m,n)
donde i=RA(k-1)+m
con i=1,2,..,RT y j=1,2,..,NV
con k=1,2,..,NV, m=1,2,..,NR(i) y n=1,2,..,TIPOR
```

2. Matriz ID

Tiene tantas filas como posibles implicados primos y cada fila tiene NV columnas. El contenido de cada fila es el identificador general correspondiente al posible implicado primo. La conversión consiste en:

```
ID(i,j)=ID(k,m,j)
donde i=LA(TIPOR-1)+m
con i=1,2,..,LT y j=1,2,..,NV
```

con k=1,2,..,RT, m=1,2,..,2^{TIPOR} y j=1,2,..,NV

3. Matriz CS

Tiene tantas filas como posibles implicados primos y cada fila tiene 2^{NV-1} columnas como máximo. El contenido de cada fila son los minitérminos cubiertos por el correspondiente posible implicado primo. La conversión se realiza como sigue:

CS(i,j)=CS(k,m,n)
donde i=LA(TIPOR-1)+m
con i=1,2,..,LT y j=1,2,..,2^{NV-1}
con k=1,2,..,RT, m=1,2,..,2^{TIPOR} y n=1,2,..,$2^{(NV-TIPOR)}$

4. Matriz DR

Tiene 2^{NV+1}-2 filas y NV columnas como máximo. Cada fila contiene el desplazamiento a sumar a la posición del posible implicado primo dentro de una rueda para obtener la posición de su adyacente en la misma rueda. La conversión se realiza como sigue:

DR(i,j)=DR(k,n,p)
donde $i=\sum_{m=1}^{k-1} 2^m + p$
con $i=\sum_{m=1}^{NV} 2^m$ y j=1, 2, ..NV
con k=1,2,..,NV y n=1,2,..,2^k y p=1,2,..,k

5. Matriz DE

Tiene 2^{NV}-1 filas y cada fila tiene como máximo NV columnas. Cada columna contiene el desplazamiento a eliminar de los desplazamientos de la primera posición. La conversión se realiza como sigue:

DE(i,j)=DE(k,n,p)
donde $i=\sum_{m=0}^{k-2} 2^m + p$
con i=1,2,..,2^{NV}-1 y j=1,2,..,NV
con k=1,2,..,NV, n=1,2,..,$2^{(i-1)}$ y p=1,2...

6. Matriz BD

Tiene tantas filas como posibles implicados primos y cada fila tiene NV columnas como máximo. Cada fila contiene las posiciones de sus lazos adyacentes. La conversión se realiza como sigue:

BD(i,j)=LA(TIPOR-1)+BD(k,m,n)
donde i=LA(TIPOR-1)+m
con i=1,2,..,LT y j=1,2,..,NV
con k=1,2,..,RT, m=1,2,..,2^{TIPOR} y n=1,2,..,2^{TIPOR}

7. Matriz PRA

Tiene tantas filas como ruedas y cada fila tiene tantas columnas como ruedas *cubre*. Cada fila contiene el número de rueda cubierta por la correspondiente rueda. La conversión se realiza:

PRA(i,j)=RA(k+m-1)+PRA(k,q,m,n)
donde i=RA(k-1)+q y j= $\sum_{p=2}^{m+k-1}$ RSML(p,k)+n

con i=1,2,..,RA(nv-1) y j=1,2..$\sum_{p=2}^{NV}$ RSML(1,p)

k=1,2,..,RA(NV-1), q=1,2,..,NR(k), m=1,2,..,(NV-k) y
n=1,2,..,RSML(m+1,k)

8. Matriz BA

Tiene tantas filas como posibles implicados primos pueden cubrir lazos y tiene como máximo $3^{(NV-1)}-1$. Cada fila contiene las posiciones de los lazos cubiertos por el correspondiente implicado primo. La conversión se realiza:

BA(i,j)=LA(TR_M-1)+BA(k,1,m,n)
donde i=LA(TR_M-1)+q y j= $\sum_{p=1}^{TR\_M-1}$ NLB(TR_K,p) + n

con i=1,2,..,LA(NV-1) y j=1,2,..,TLB(1)
con k=1,2,..,RA(NV-1), q=1,2,..,2^{TR_K}, m=1,2,..,(NV-TR_K)
y n=1,2,..,RSML(TR_M,TR_K)
donde TR_K y TR_M son el tipo de las ruedas k y m, respectivamente.

2.5.2 Estudio teórico del tamaño y densidad de MCVK

Desde una aproximación teórica abordaremos el estudio del tamaño de las principales matrices de MCVK, y también cuántos de estos elementos son distintos de cero en cada matriz.

Tamaño y número de elementos de cada submatriz

En la tabla 2.9 aparecen las expresiones teóricas del número de filas y columnas de cada matriz de MCVK implementada en MATLAB. El número total de posiciones es el producto de ambas expresiones.

	FILAS	COLUMNAS
MATRIZ ID	$3^{NV}-1$	NV
MATRIZ CS	$3^{NV}-1$	$2^{(NV-1)}$
MATRIZ BD	$3^{NV}-1$	NV
MATRIZ BA	$3^{NV}-1-2^{NV}$	$3^{(NV-1)}-1$

Tabla 2.9 Tamaño en filas y columnas de las submatrices de MCVK

Número de elementos distintos de cero y densidad de cada matriz

La tabla 2.10 contiene las expresiones que indican cuántos elementos de cada matriz son distintos de cero. En el siguiente apartado aparece un estudio cuantitativo de los resultados de esta tabla.

La matriz ID aunque tiene elementos iguales a cero éstos aportan la misma información que los restantes unos y doses. Es decir, desde el punto de vista teórico no son casillas libres. Pero contienen ceros, y por lo tanto desde un punto de vista práctico podemos calcular su número.

La densidad teórica de cada matriz se obtiene como el cociente entre el número de elementos distintos de cero y el número total de posiciones de la matriz. Es por tanto el cociente entre las tablas 2.10 y 2.9.

	NÚMERO DE ELEMENTOS DISTINTOS DE CERO
MATRIZ ID	$(3^{NV}-1)\cdot NV-\sum_{i=1}^{NV}\frac{2^i\cdot i}{2}$
MATRIZ CS	$\sum_{i=1}^{NV} NL(i)*2^{(NV-i)}$
MATRIZ BD	$\sum_{i=1}^{NV} NL(i)*i$
MATRIZ BA	$\sum_{i=1}^{NV-1} NL(i)\cdot TLB(i)$

Tabla 2.10 Número de elementos distintos de cero de cada submatriz de MCVK

2.5.3 Estudio práctico del tamaño y densidad de MCVK en MATLAB

El MATLAB es capaz de crear y manejar matrices con densidad inferior a uno -*matriz sparse*-, con la ventaja que esto supone respecto del tamaño de la matriz. Como desventaja resulta que el acceso a dichas matrices es más lento. Y además un elemento de una *matriz sparse* ocupa 12 bytes, frente a los 8 bytes de una matriz normal, por tanto el uso de *matrices sparse* es sólo aconsejable si la densidad de la matriz a crear es menor de 2/3.

En este apartado cuantificaremos el tamaño y densidad de cada matriz de MCVK (García y Kahoraho, 1995), tanto teóricamente -según las tablas 2.9 y 2.10- como su implementación en MATLAB para los distintos valores de NV. La información es presentada en forma de tablas, las tres primeras líneas son de tipo teórico y se corresponden con las tablas 2.9 y 2.10, mientras que las cuatro últimas reflejan el resultado de la implementación en MATLAB.

La generación de las matrices MCVK se hace a partir de cuatro variables, el cebo del algoritmo -introducido como dato fijo- son los datos correspondientes a la matriz de tres variables. Los datos aportados en las tablas y correspondientes a dos y tres variables son puramente teóricos.

Los datos presentados son fuertemente dependientes de la máquina utilizada. Así, para nueve variables es necesario subdividir las matrices por

lo que no aparece el tiempo de generación en la columna correspondiente, mientras que para diez variables la máquina no concluye el proceso. Como respuesta a estos datos cabe decir que la mayoría del software de simplificación está orientado a *'mainframes'* y que el método desarrollado en la tesis es un prototipo.

En las tablas la densidad es expresada en tanto por uno u absoluta, el tamaño en Kbytes y el tiempo en segundos. La máquina utilizada es un PC-486 equipado con un procesador de 66 MHz y 16 Mbytes de RAM.

Matriz ID

	NV=2	NV=3	NV=4	NV=5	NV=6	NV=7	NV=8	NV=9	NV=10
Matriz dos índices	8*2	26*3	80*4	243*5	728*6	2186*7	6560*8	19682*9	59048*10
Número total	16	78	320	1210	4368	15302	52480	177138	590480
Distintos de cero	10	51	212	805	2910	10199	34984	118089	393650
Densidad	0,625	0,6538	0,6625	0,6653	0,6662	0,6665	0,6666	0,6666	0,6667
Tamaño Matlab			2,560	9,680	34,944	122,416	419,840	1417,068	
Densidad Matlab			0,6625	0,6653	0,6662	0,6665	0,6666	0,6666	
Tiempo Matlab			0,39	1,48	9,62	93,98	1279		

Tabla 2.11 Tamaño, densidad y tiempo de creación de ID

Matriz CS

	NV=2	NV=3	NV=4	NV=5	NV=6	NV=7	NV=8	NV=9	NV=10
Matriz dos índices	8*2	26*4	80*8	243*16	728*32	2186*64	6560*128	19682*256	59048*512
Número total	16	104	640	3888	23296	139904	839680	5038592	30232576
Distintos de cero	12	56	240	992	4032	16256	65280	261632	1047552
Densidad	0,75	0,538	0,375	0,255	0,173	0,116	0,078	0,052	0,035
Tamaño Matlab			2,744	11,608	47,768	193,816	780,824	313,958	
Densidad Matlab			0,3531	0,2485	0,1704	0,1153	0,0774	0,052	
Tiempo Matlab			1,2	6,21	34,82	254,74	1734,62		

Tabla 2.12 Tamaño, densidad y tiempo de creación de CS

Matriz BD

	NV=2	NV=3	NV=4	NV=5	NV=6	NV=7	NV=8	NV=9	NV=10
Matriz dos índices	8*2	26*3	80*4	243*5	728*6	2186*7	6560*8	19682*9	59048*10
Número total	16	78	320	1215	4368	15302	52480	177138	590480
Distintos de cero	12	54	216	810	2916	10206	34992	118098	393660
Densidad	0,75	0,692	0,675	0,667	0,667	0,667	0,667	0,667	0,667
Tamaño Matlab			2,608	9,740	35,016	122,500	419,936	1417,176	
Densidad Matlab			0,675	0,669	0,6676	0,667	0,6668	0,667	
Tiempo Matlab			0,66	2,36	10,43	68,100	520,32		

Tabla 2.13 Tamaño, densidad y tiempo de creación de BD

Matriz BA

	NV=2	NV=3	NV=4	NV=5	NV=6	NV=7	NV=8	NV=9	NV=10
Matriz dos índices	4*2	18*8	64*26	211*80	664*242	2058*728	6304*2186	19170*6560	58024*19682
Número total	8	144	1664	16880	160688	1498224	13780544	125760000	1142000000
Distintos de cero	8	72	464	2640	14168	73752	377504	1913760	9647528
Densidad	1	0,5	0,278	0,156	0,088	0,049	0,027	0,0152	0,0085
Tamaño Matlab			5,672	32	170,984	887,936	4547,536	22965,12	
Densidad Matlab			0,223	0,1364	0,0804	0,0463	0,0263	0,0152	
Tiempo Matlab			31,36	154,12	771,65	5718,3	85040		

Tabla 2.14 Tamaño, densidad y tiempo de creación de BA

Podemos extraer distintas conclusiones de las anteriores tablas:

- La densidad teórica y la obtenida por MATLAB son casi idénticas y permite manejar matrices de gran tamaño. Por ejemplo, la matriz BA para ocho variables tendría un tamaño de casi 182 Mbytes en vez de los 4,5 Mbytes utilizados.
- El tamaño de las matrices crece exponencialmente con el número de variables, al igual que el número de elementos distintos de cero.
- La densidad de las matrices BD e ID tiende al valor constante 2/3, y por tanto su implementación en *matrices sparse* no es aconsejable.

- La densidad de CS es decreciente frente al aumento de NV. La relación entre las densidades para NV-1 y NV tiende a 1,5. Así podemos interpolar la densidad para NV partiendo de la correspondiente a NV-1: `DENSIDAD(NV)=DENSIDAD(NV-1)/1,5`.
- La densidad de BA es decreciente frente al aumento de NV. La relación entre las densidades para NV-1 y NV tiende a 1,8. Así podemos interpolar la densidad para NV partiendo de la correspondiente a NV-1: `DENSIDAD(NV)=DENSIDAD(NV-1)/1,8`.
- El tiempo de creación de las matrices crece exponencialmente con el número de variables. Además, en MATLAB el uso de grandes matrices de densidad no nulas ralentiza enormemente su proceso, por ejemplo la creación de BA para ocho variables supone un día de proceso en la máquina citada. Este no es un gran problema por cuanto que la creación de MCVK es única en la vida del algoritmo.

2.6 Obtención de los implicados primos de una función

Toda vez que la matriz MCVK ha sido creada en las anteriores secciones, resta procesarla para obtener los implicados primos de una función. En esta sección explicaremos de forma operativa qué supone procesar MCVK.

Para clarificar la explicación utilizaremos la matriz MCVK para cuatro variables. En la tabla 2.15 cada lazo contiene distintos campos separados por '#' y ordenados según: su índice (posición), su identificador general (ID), los minitérminos cubiertos (CS), los índices de los lazos adyacentes (BD) y los índices de los lazos cubiertos (BA). Además a cada lazo le corresponde un indicador de borrado inicialmente a '0', que pasará a '-1' si el lazo es borrado.

	MATRIZ CARACTERÍSTICA PARA CUATRO VARIABLES	
1	0222	0,4,2,6,1,5,3,7 # 2 # 9,11,13,15,17,19,33,35,37,39,41,43,45,47,49,51,53,55,65,67,69,71,73,75,77,79
2	1222	8,12,10,14,9,13,11,15 # 1 # 10,12,14,16,18,20,34,36,38,40,42,44,46,48,50,52,54,56,66,68,70,72,74,76,78,80
3	2022	0,8,2,10,1,9,3,11 # 4#9,10,21,23,25,27,33,34,37,38,41,42,45,46,57,59,61,63,65,66,69,70,73,74,77,78
4	2122	4,12,6,14,5,13,7,15 # 3 # 11,12,22,24,26,28,35,36,39,40,43,44,47,48,58,60,62,64,67,68,71,72,75,76,79,80
5	2202	0,8,4,12,1,9,5,13 # 6 # 13,14,21,22,29,31,33,34,35,36,49,50,53,54,57,58,61,62,65,66,67,68,73,74,75,76
6	2212	2,10,6,14,3,11,7,15 # 5 # 15,16,23,24,30,32,37,38,39,40,51,52,55,56,59,60,63,64,69,70,71,72,77,78,79,80
7	2220	0,8,4,12,2,10,6,14 # 8 # 17,18,25,26,29,30,41,42,43,44,49,50,51,52,57,58,59,60,65,66,67,68,69,70,71,72
8	2221	1,9,5,13,3,11,7,15 # 7 # 19,20,27,28,31,32,45,46,47,48,53,54,55,56,61,62,63,64,73,74,75,76,77,78,79,80
9	0022	0,2,1,3 # 10,11 # 33,37,41,45,65,69,73,77
10	1022	8,10,9,11 # 9,12 # 34,38,42,46,66,70,74,78
11	0122	4,6,5,7 # 12,9 # 35,39,43,47,67,71,75,79
12	1122	12,14,13,15 # 11,10 # 36,40,44,48,68,72,76,80
13	0202	0,4,1,5 # 14,15 # 33,35,49,53,65,67,73,75
14	1202	8,12,9,13 # 13,16 # 34,36,50,54,66,68,74,76
15	0212	2,6,3,7 # 16,13 # 37,39,51,55,69,71,77,79
16	1212	10,14,11,15 # 15,14 # 38,40,52,56,70,72,78,80
17	0220	0,2,4,6 # 18,19 # 41,43,49,51,65,67,69,71
18	1220	8,12,10,14 # 17,20 # 42,44,50,52,66,68,70,72
19	0221	1,5,3,7 # 20,17 # 45,47,53,55,73,75,77,79
20	1221	9,13,11,15 # 19,18 # 46,48,54,56,74,76,78,80
21	2002	0,8,1,9 # 22,23 # 33,34,57,61,65,66,73,74
22	2102	4,12,5,13 # 21,24 # 35,36,58,62,67,68,75,76
23	2012	2,10,3,11 # 24,21 # 37,38,59,63,69,70,77,78
24	2112	6,14,7,15 # 23,22 # 39,40,60,64,71,72,79,80
25	2020	0,8,2,10 # 26,27 # 41,42,57,59,65,66,69,70
26	2120	4,12,6,14 # 25,28 # 43,44,58,60,67,68,71,72
27	2021	1,9,3,11 # 28,25 # 45,46,61,63,73,74,77,78
28	2121	5,13,7,15 # 27,26 # 47,48,62,64,75,76,79,80
29	2200	0,8,4,12 # 30,31 # 49,50,57,58,65,66,67,68
30	2210	2,10,6,14 # 29,32 # 51,52,59,60,69,70,71,72
31	2201	1,9,5,13 # 32,29 # 53,54,61,62,73,74,75,76
32	2211	3,11,7,15 # 31,30 # 55,56,63,64,77,78,79,80
33	0002	0,1 # 34,35,37 #65,73
34	1002	8,9 # 33,36,38 # 66,74
35	0102	4,5 # 36,33,39 # 67,75
36	1102	12,13 # 35,34,40 # 68,76
37	0012	2,3 # 38,39,33 # 69,77
38	1012	10,11 # 37,40,34 # 70,78
39	0112	6,7 # 40,37,35 # 71,79
40	1112	14,15 # 39,38,36 # 72,80
41	0020	0,2 # 42,43,45 # 65,69
42	1020	8,10 # 41,44,46 # 66,70
43	0120	4,6 # 44,41,47 # 67,71
44	1120	12,14 # 43,42,48 # 68,72
45	0021	1,3 # 46,47,41 # 73,77
46	1021	9,11 # 45,48,42 # 74,78
47	0121	5,7 # 48,45,43 # 75,79
48	1121	13,15 # 47,46,44 # 76,80
49	0200	0,4 # 50,51,53 # 65,67
50	1200	8,12 # 49,52,54 # 66,68
51	0210	2,6 # 52,49,55 # 69,71
52	1210	10,14 # 51,50,56 # 70,72
53	0201	1,5 # 54,55,49 # 73,75
54	1201	9,13 # 53,56,50 # 74,76
55	0211	3,7 # 56,53,51 # 77,79
56	1211	11,15 # 55,54,52 # 78,80
57	2000	0,8 # 58,59,61 # 65,66
58	2100	4,12 # 57,60,62 # 67,68
59	2010	2,10 # 60,57,63 # 69,70
60	2110	6,14 #59,58,64 # 71,72
61	2001	1,9 # 62,63,57 # 73,74

Tabla 2.15 Matriz MCVK de cuatro variables

62	2101	5,13 # 61,64,58 # 75,76
63	2011	3,11 # 64,61,59 # 77,78
64	2111	7,15 # 63,62,60 # 79,80
65	0000	0 # 66,67,69,73 #
66	1000	8 # 65,68,70,74 #
67	0100	4 # 68,65,71,75 #
68	1100	12 # 67,66,72,76 #
69	0010	2 # 70,71,65,77 #
70	1010	10 # 69,72,66,78 #
71	0110	6 # 72,69,67,79 #
72	1110	14 # 71,70,68,80 #
73	0001	1 # 74,75,77,65 #
74	1001	9 # 73,76,78,66 #
75	0101	5 # 76,73,79,67 #
76	1101	13 # 75,7480,68 #
77	0011	3 # 78,79,73,69 #
78	1011	11 # 77,80,74,70 #
79	0111	7 # 80,77,75,71 #
80	1111	15 # 79,78,76,72 #

Tabla 2.15 (*continuación*)

La creación de MCVK ha concentrado el esfuerzo teórico-práctico para que el algoritmo de obtención de los implicados primos de una función booleana -almacenada en la matriz VK que contiene un 1, 0 ó x para cada casilla- sea muy sencillo. Obtener los implicados primos consiste en procesar MCVK frente a VK, **seleccionando de todos los posibles implicados primos aquellos que cumplan determinado criterio** -enunciado en 2.3- respecto de VK. Las fases que conforman el sencillo algoritmo son:

1. Procesar todos los posibles implicados primos.
2. Si un lazo ya ha sido borrado anteriormente pasar directamente a la fase 7, si no pasar a la fase 3.
3. Si todas las casillas cubiertas por un lazo contienen 1 ó X en VK, no siendo todas X, entonces dicho lazo es un implicado primo de la función representada por VK. Si es un implicado primo procesar las fases 4, 5 y 6, en caso contrario pasar a la fase 7.
4. Almacenar el identificador general del implicado primo y las casillas por él cubiertas.
5. Borrar todos los lazos adyacentes al convertido en implicado primo.
6. Borrar todos los lazos cubiertos por él convertido en implicado primo.
7. Procesar el siguiente posible implicado primo.

El proceso se reduce a comprobar cada lazo, a repetir un gran número de veces una sencilla comprobación. Este proceso es claramente computacional, su aplicación manual sería tediosa y absurda. Tiene como

entradas a MCVK, VK y B_IP -esta última indica si cada lazo ha sido borrado-, y como salidas a las matrices IP y CS_IP, que contienen a los implicados primos y a las casillas cubiertas por ellos, respectivamente.

2.6.1 Algoritmos de obtención de los implicados primos

El organigrama de la figura 2.4 se corresponde con el algoritmo OB_IP de obtención de los implicados primos de una función booleana representada por VK. El algoritmo *explora* todas las filas de MCVK, para cada una el algoritmo IMPLICADOPRIMO decide si es implicado de VK o no, en caso afirmativo pasa a las matrices IP y CS_IP de implicados primos. El organigrama está orientado a MATLAB, donde el símbolo ':' debe leerse como *todas las filas* o *todas las columnas*.

El organigrama de la figura 2.5 conforma al algoritmo IMPLICADOPRIMO - rombo sombreado en OB_IP. Éste simplemente comprueba si las casillas asociadas a la fila de MCVK procesada por OB_IP se corresponden con '1' o 'x' en VK, no siendo todo 'x', en caso afirmativo la fila procesada de MCVK es un implicado primo de VK.

2.6.2 Ejemplo de obtención de los implicados primos

Los organigramas anteriores, dada su sencillez, tienen un detalle a nivel de instrucción. Mediante el siguiente ejemplo aclararemos el algoritmo de búsqueda de los implicados primos de una función.

Ejemplo 2.15 Obtendremos, utilizando para ello la matriz MCVK de la tabla 2.15, los implicados primos de la siguiente función de cuatro variables:

$$f=\sum(3,\ 11,\ 13,\ 15) \text{ y } X=\sum(4,\ 6,\ 12,\ 14)$$

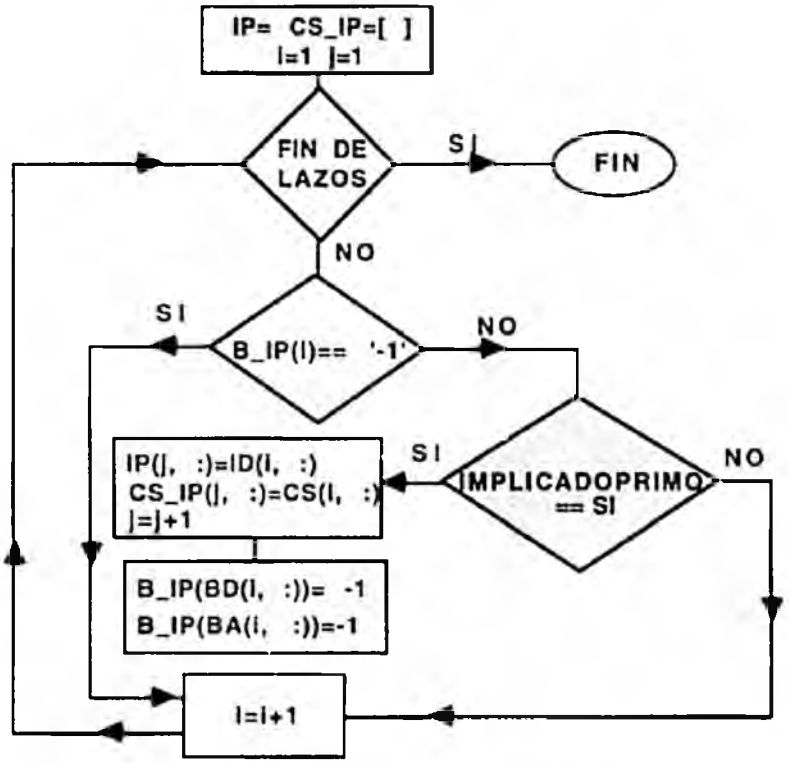


Figura 2.4. Organigrama del algoritmo OB_IP

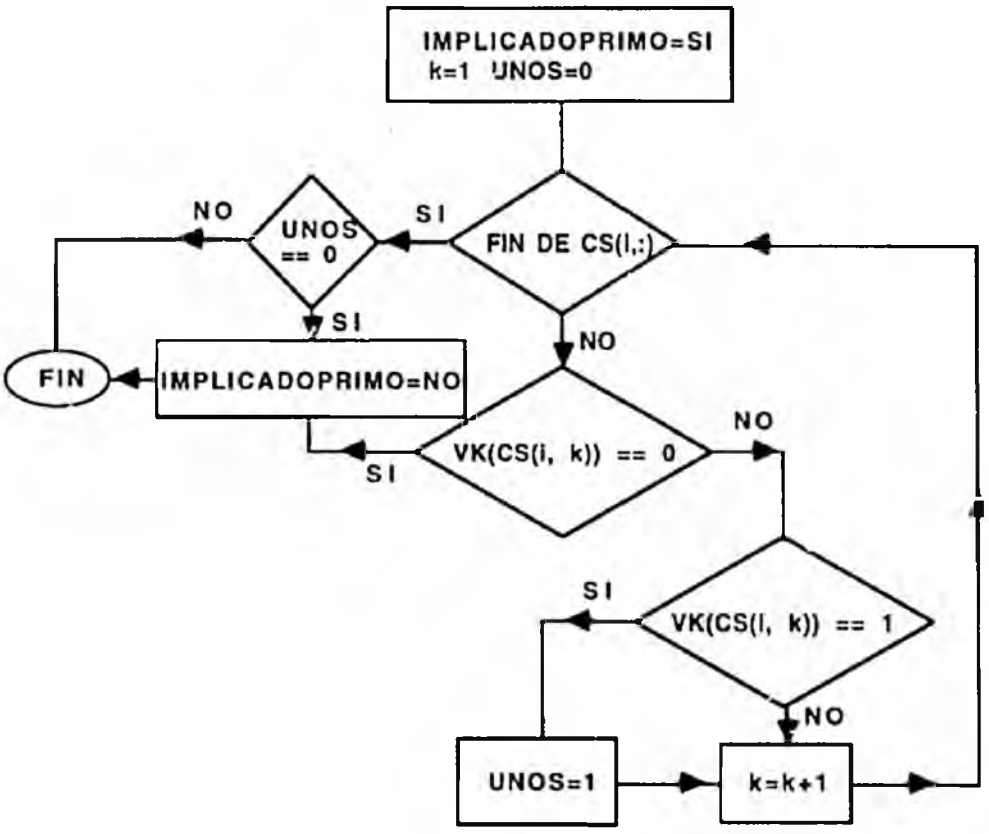


Figura 2.5 Organigrama del algoritmo IMPLICADOPRIMO

Los momentos más destacables del proceso de MCVK frente a VK son:

1. El lazo de posición 12 e identificador general 1122 cubre a las casillas 12, 13, 14 y 15 y se convierte, por tanto, en implicado primo de f.
   Marca como borrados, para no ser procesados, los lazos: 11, 10, 36, 40, 44, 48, 68, 72, 76 y 80. Los dos primeros por ser adyacentes a 1122, y los restantes por estar cubiertos por éste.
2. El lazo 2020 no se convierte en implicado primo, pues cubre sólo condiciones libres.
3. El lazo de posición 56 e identificador general 1211 cubre a las casillas 11 y 15, se convierte, por tanto, en implicado primo de f.
   Marca como borrados, para no ser procesados, los lazos: 55, 54, 52, 78 y 80. Los tres primeros por ser adyacentes a 1211, y los restantes por estar cubiertos por éste.
4. El lazo de posición 63 e identificador general 2011 cubre a las casillas 3 y 11, se convierte, por tanto, en implicado primo de f.
   Marca como borrados, para no ser procesados, los lazos: 64, 61, 59, 77 y 78. Los tres primeros por ser adyacentes a 2011, y los restantes por estar cubiertos por éste.

2.6.3 Análisis cualitativo y cuantitativo del proceso de borrado

Cuando un lazo se convierte en implicado primo, otros, debido a esto, no pueden serlo. Este apartado estudia el efecto del ***borrado hacia abajo*** y el ***borrado dentro de la rueda***, tanto cualitativo como cuantitativo.

Las razones de un borrado y otro son muy distintas. Así, cuando un lazo se convierte en implicado primo sus lazos adyacentes no van a serlo, como vimos en 2.2.1. Es decir, si los procesaramos frente a VK nunca cumplirían el criterio de selección. Por tanto, el *borrado dentro de la rueda* evita estas comprobaciones, acelerando el proceso de búsqueda; pero remarcamos que no es necesario borrarlos. Sin embargo, los lazos cubiertos por el implicado primo deben ser borrados necesariamente, ya que de lo contrario el criterio de selección los convertiría en implicados primos. Por tanto, el *borrado hacia abajo* es absolutamente necesario. Refrendamos que cualitativamente el *borrado hacia abajo* es absolutamente necesario,

mientras que el *borrado dentro de la rueda* supone una mejora del algoritmo de búsqueda, no siendo estrictamente necesario.

Tras las anteriores indicaciones cualitativas resta estudiar el efecto cuantitativo de ambos tipos de borrado.

En primer lugar, y brevemente, un lazo *borra dentro de la rueda* tantos lazos como su propio tipo. La relación entre los lazos borrados y el tamaño de la rueda es $\frac{TIPOR}{2^{TIPOR}}$. Esta función es decreciente y cabe decir que el efecto respecto de la rueda es mayor cuanto menor es el tipo; pero no hay que olvidar que respecto del conjunto de lazos el efecto del borrado es mayor cuanto mayor es el tipo.

Pero, realmente, el interés del estudio cuantitativo se centra en el *borrado hacia abajo*. Previamente, recordemos -sección 2.4.1- que en MCVK no hemos considerado el lazo que representa al VK con todo unos -f=1-, su efecto se traduce en que: `LT=LT+1` y `LA(i)=LA(i)+1`. Además, cabe considerar que el lazo convertido en implicado primo queda borrado, por cuanto que junto a los lazos cubiertos forma una clase, su efecto se traduce en que: `NLB(i)=NLB(i)+1`. La tabla 2.16 muestra la relación entre los lazos a *borrar hacia abajo* y el total de los lazos teniendo en cuenta el tipo de los lazos, según la siguiente expresión:

$$\frac{NLB(TIPO)+1}{LT+1}$$

Por ejemplo, si un lazo de tipo 1 en un VK de cuatro variables se convierte en su implicado primo *borra hacia bajo* 26+1 lazos, la relación es: $\frac{26+1}{80+1} = \frac{1}{3}$.

La tabla 2.16 muestra el conjunto de estas relaciones:

TIPO	NV=2	NV=3	NV=4	NV=5	NV=6	NV=7	NV=8	NV=9	NV=10
1	$\frac{1}{3}$	$\frac{1}{3}$	$\frac{1}{3}$	$\frac{1}{3}$	$\frac{1}{3}$	$\frac{1}{3}$	$\frac{1}{3}$	$\frac{1}{3}$	$\frac{1}{3}$
2		$\frac{1}{9}$	$\frac{1}{9}$	$\frac{1}{9}$	$\frac{1}{9}$	$\frac{1}{9}$	$\frac{1}{9}$	$\frac{1}{9}$	$\frac{1}{9}$
3			$\frac{1}{27}$	$\frac{1}{27}$	$\frac{1}{27}$	$\frac{1}{27}$	$\frac{1}{27}$	$\frac{1}{27}$	$\frac{1}{27}$
4				$\frac{1}{81}$	$\frac{1}{81}$	$\frac{1}{81}$	$\frac{1}{81}$	$\frac{1}{81}$	$\frac{1}{81}$
5					$\frac{1}{243}$	$\frac{1}{243}$	$\frac{1}{243}$	$\frac{1}{243}$	$\frac{1}{243}$

Tabla 2.16 Efecto cuantitativo porcentual global del '*borrado hacia abajo*'

6						$\frac{1}{729}$	$\frac{1}{729}$	$\frac{1}{729}$	$\frac{1}{729}$
7							$\frac{1}{2187}$	$\frac{1}{2187}$	$\frac{1}{2187}$
8								$\frac{1}{6569}$	$\frac{1}{6569}$
9									$\frac{1}{19683}$

Tabla 2.16 (*continuación*)

La tabla 2.16 nos permite afirmar que el porcentaje de *lazos borrados hacia abajo* depende únicamente del tipo de la rueda a la que pertenezca el lazo convertido en implicado primo; no depende en absoluto del número de variables. El efecto del *borrado hacia abajo* es igual $\frac{1}{3^{TIPOR}}$. Así por ejemplo, si un lazo de tipo 1 se convierte en implicado primo borra un 1/3 de los lazos, que no deberá procesar, con la rapidez que esto supone en el algoritmo de búsqueda de los implicados primos.

La tabla 2.16 muestra el aspecto más relevante del *borrado hacia abajo*, pero actuando en puridad no deberíamos obtener la relación respecto de LT, sino respecto del total de lazos de las ruedas restantes, no contabilizando los lazos de la propia rueda. El resultado obtenido aparece en la tabla 2.17.

TIPO	NV=2	NV=3	NV=4	NV=5	NV=6	NV=7	NV=8	NV=9	NV=10
1	$\frac{2}{4}$	$\frac{8}{20}$	$\frac{26}{72}$	$\frac{80}{232}$	$\frac{242}{716}$	$\frac{728}{2172}$	2186/6544	6560/19664	19682/59028
2		$\frac{2}{8}$	$\frac{8}{48}$	$\frac{26}{192}$	$\frac{80}{656}$	$\frac{242}{2088}$	728/6432	2186/19520	6560/58848
3			$\frac{2}{16}$	$\frac{8}{112}$	$\frac{26}{496}$	$\frac{80}{1808}$	242/5984	728/18848	2186/57888
4				$\frac{2}{32}$	$\frac{8}{256}$	$\frac{26}{1248}$	80/4864	242/16832	728/54528
5					$\frac{2}{64}$	$\frac{8}{576}$	26/3072	80/12800	242/46464
6						$\frac{2}{128}$	8/1280	26/7424	80/33024
7							2/256	8/2816	26/17664
8								2/512	8/6144
9									2/1024

Tabla 2.17 Efecto cuantitativo porcentual del '*borrado hacia abajo*' referido a los lazos restantes por comprobar

El resultado obtenido de 2.17 es muy parecido al anterior; aunque en este caso el porcentaje depende del tipo de la rueda y también del número de variables, de tal forma que para un mismo tipo de rueda el porcentaje disminuye al aumentar el número de variables. Por ejemplo, si aparece un implicado primo en una rueda de tipo 4, éste borrará hacia abajo un 6,25% de los lazos restantes de menor tamaño si el número de variables fuera cinco; sin embargo si el número de variables fuera 10 el porcentaje se reduciría al 1,335%.

2.6.4 Comparación con otros métodos de obtención de implicados primos

En esta sección, y mediante ejemplos, compararemos el tiempo de búsqueda de los implicados primos de una función según los métodos: tabular de Quine (1952 y 1955) y McCluskey (1956) y el aportado en la tesis basado en MCVK -denominado CAMP DEUSTO.

Los resultados aquí mostrados están condicionados por:

- El ordenador utilizado es un PC-486 a 66 MHz con 16 Mb de RAM.
- Los tiempos están indicados en segundos.
- Las 141 funciones booleanas han sido generadas aleatoriamente en su mayoría.
- Los algoritmos han sido implementados en MATLAB 4.2 bajo WINDOWS.
- MATLAB es un lenguaje interpretado y por lo tanto lento. Así pues los tiempos ofrecidos deben ser observados sólo comparativamente, y nunca como valores absolutos. El objetivo es comparar los dos métodos entre sí, y no el ofrecer una estadística exhaustiva.
- Las tablas aportan de forma resumida los resultados referentes a las 282 funciones simuladas.
- Las gráficas aportan la expresividad necesaria para recalcar las conclusiones, no buscan representar valores exactos.

La tabla 2.18 contiene los *tiempos medios* de obtención de los implicados primos de una función booleana *según su número de variables*, e independientemente del número de minitérminos y condiciones libres de la función.

Número de Variables	CAMP DEUSTO	Q-M	Número de Funciones
4	0,3496	0,5879	47
5	1,1879	2,8617	47
6	4,1345	17,4149	47
7	15,1047	141,0145	47
8	56,2630	1044,2462	47
9	207,2821	10056,5157	47

Tabla 2.18 Tiempos medios de obtención de los implicados primos en función del número de variables

Las siguientes tablas contienen los *tiempos medios* de obtención de los implicados primos; pero no sólo en *función del número de variables*, sino también en *función del número de minitérminos y condiciones libres*, expresado en %. De esta forma mostramos la posible dependencia de los tiempos medios de la tabla 2.18 respecto del número de variables y/o del porcentaje de minitérminos y condiciones libres.

	Número de Variables = 4			Número de Variables = 5		
Porcentaje (%)	CAMP DEUSTO	Q-M	Nº fun	CAMP DEUSTO	Q-M	Nº fun
0-12,5	0,3012	0,1037	8	0,8633	0,1467	3
12,5-25	0,2625	0,1625	4	1,245	0,220	2
25-37,5	0,3467	0,2733	3	1,1227	0,5609	11
37,5-50	0,3455	0,3553	9	1,1167	1,0917	6
50-62,5	0,3627	0,5691	11	1,1182	2,2319	11
62,5-75	0,4320	0,8480	5	1,557	4,3910	10
75-87,5	0,4083	1,530	6	1,1133	11,130	3
87,5-100	0,220	2,470	1	0,490	19,770	1

Tabla 2.19 Tiempos medios de obtención de los implicados primos según el número de variables y porcentaje de minitérminos y condiciones libres

	Número de Variables = 6			Número de Variables = 7		
Porcentaje (%)	CAMP DEUSTO	Q-M	Nº fun	CAMP DEUSTO	Q-M	Nº fun
0-12,5	2,530	0,270	1	11,1375	0,55	4
12,5-25	3,0729	0,7514	7	-----	-----	0
25-37,5	4,1933	2,0	9	11,5944	8,1055	9
37,5-50	2,7633	3,6433	3	8,3875	15,60	4
50-62,5	4,3933	11,5793	15	14,7914	66,41	14
62,5-75	5,6314	22,20	7	22,0286	132,0325	8
75-87,5	4,3625	68,4375	4	19,9371	404,08	7
87,5-100	1,480	181,2	1	4,60	1675,570	1

Tabla 2.20 Tiempos medios de obtención de los implicados primos según el número de variables y porcentaje de minitérminos y condiciones libres

	Número de Variables = 8			Número de Variables = 9		
Porcentaje (%)	CAMP DEUSTO	Q-M	Nº fun	CAMP DEUSTO	Q-M	Nº fun
0-12,5	46,705	2,4767	6	142,30	7,51	4
12,5-25	38,344	8,2620	5	-----	-----	0
25-37,5	53,078	57,8460	5	124,5857	280,1186	7
37,5-50	55,184	109,730	5	208,6557	583,5771	7
50-62,5	59,1712	450,6744	16	198,1006	2954,2453	17
62,5-75	93,6725	1118,2975	4	355,555	4927,560	2
75-87,5	58,932	3912,816	5	302,03	25853,40	9
87,5-100	15,010	16935,56	1	43,28	173822,440	1

Tabla 2.21 Tiempos medios de obtención de los implicados primos según el número de variables y porcentaje de minitérminos y condiciones libres

Observando las tablas anteriores y las gráficas del Apéndice D podemos extraer las siguientes conclusiones principales respecto del tiempo de obtención de los implicados primos de una función:

- El método propuesto dentro de CAMP DEUSTO es más rápido que el de Quine-McCluskey -tabla 2.18.
- El método de Quine-McCluskey es más rápido sólo si el número de minitérminos y condiciones libres es pequeño -tablas 2.19 a 2.21.
- El método propuesto en CAMP DEUSTO depende fuertemente del número de variables de la función, y no del número de minitérminos y condiciones libres -gráficas D.1 a D.5.

- El método de Quine-McCluskey depende fuertemente del número de minitérminos y condiciones libres de la función, y no del número de variables -gráficas D.17 a D.22.
- El método propuesto en CAMP DEUSTO implica manejar matrices de gran tamaño.

Las anteriores conclusiones principales quedan complementadas con las siguientes:

- El método aportado es tanto más rápido que el de Quine-McCluskey cuanto mayor es el número de variables y el tamaño de la función. La relación crece exponencialmente -gráficas D.23 a D.28, D.35 a D.37 y D.41 a D.43.
- Si la escala de tiempos es logarítmica podemos observar una interpolación lineal de los tiempos medios de ambos métodos -figuras D.35 a D.37.
- El método aportado se ralentiza si la función tiene condiciones libres -figuras D.1 a D.6 y D.13 a D.18-, debido a que el algoritmo es más complejo.
- El método de Quine-McCluskey presenta un crecimiento exponencial respecto del tamaño de la función, observable en las escalas logarítmicas del tiempo.
- Para el método aportado y un determinado número variables el tiempo medio desciende si el tamaño de la función es muy elevado -últimas filas de las tablas 2.19 a 2.21 y gráficas D.7 a D.12-, recordando que el efecto del *borrado hacia abajo* para implicados de tipo 1 es del 33%.

Otros muchos aspectos destacables podrían ser obtenidos, pero no ayudarían necesariamente a reafirmar los aspectos principales de la comparación.

2.7 Conclusiones

En este capítulo hemos desarrollado un nuevo método de búsqueda de implicados primos como primera parte del método CAMP DEUSTO. Los métodos clásicos presentados en el capítulo 1 obtenían sólo los implicados primos correspondientes a cada función, y así los implicados primos iban *tomando forma* durante la aplicación del método. Sin embargo, el método aportado en la tesis parte de todos los posibles implicados primos, seleccionando cuáles de ellos lo son. Conclusión de lo anterior es que los algoritmos clásicos son ***combinatorios***, mientras que el aportado es ***exploratorio***. Y mientras que Q-M y Consenso Iterativo concentran sus esfuerzos en la propia metodología, en la tesis nos concentramos en el ordenamiento de todos los posibles aspectos de un VK. Los primeros aportan métodos susceptibles de ser implementados y aplicados manualmente, no así el método presentado, que sólo puede ser utilizado computacionalmente.

Los aspectos más destacables del nuevo método aportado son:

- La aportación de un nuevo enfoque y de una nueva línea de trabajo.
- La obtención de todos los posibles implicados primos -MCVK- agrupados y ordenados en torno al nuevo concepto de rueda.
- La cuantificación de todos los aspectos relacionados con MCVK.
- El desarrollo de un nuevo y sencillo algoritmo de obtención de los implicados primos de una función.
- El método aportado es más rápido que el de Quine-McCluskey.
- El método aportado necesita disponer de una memoria amplia.

CAPÍTULO 3. EL NUEVO MÉTODO DIRECTO COMPUTACIONAL: OBTENCIÓN DE LA EXPRESIÓN MÍNIMA

3.1 Introducción

La expresión mínima de una función booleana está formada por algunos de sus implicados primos -teorema de los implicados primos de Quine-, aquellos que aseguran cubrir todos los minitérminos con el menor número de ellos. Así pues, el proceso de simplificación puede asimilarse a discriminar qué implicados primos aseguran una expresión mínima.

En la primera sección presentamos el organigrama global de simplificación y la distinción entre implicados primos esenciales y no esenciales. La tercera y cuarta sección aportan el algoritmo IPE de obtención de los implicados primos esenciales y el algoritmo DC de discriminación entre los implicados primos no esenciales, respectivamente. Ambos algoritmos serán aclarados mediante varios ejemplos en la quinta sección.

La discriminación entre los implicados primos no esenciales concentra el esfuerzo de los distintos autores. El algoritmo DC aportado en este capítulo viene a sumarse a los anteriores.

Por último compararemos los resultados obtenidos con el método aportado en la tesis frente a algunos de los métodos clásicos enunciados en el capítulo 1.

3.2 Organigrama global del proceso de simplificación

En esta sección nos centraremos en los aspectos operativos de conjunto del nuevo método: el organigrama global de obtención de la expresión mínima de una función booleana a partir de sus implicados primos.

En el capítulo 2 hemos obtenido a partir de MCVK y de un sencillo algoritmo los implicados primos de una función y los minitérminos que cubren cada uno de ellos: IP y CS_IP, respectivamente. Resta obtener EM (Expresión Mínima); que en el nuevo método requiere completar las siguientes fases -figura 3.1:

1. Elegir uno o varios implicados primos como pertenecientes a EM .
2. Eliminar de la función booleana los minitérminos asociados a los implicados primos elegidos.
3. Si quedan minitérminos por cubrir en la función booleana volver a la fase 1, en caso contrario el proceso ha terminado, siendo EM la expresión mínima.

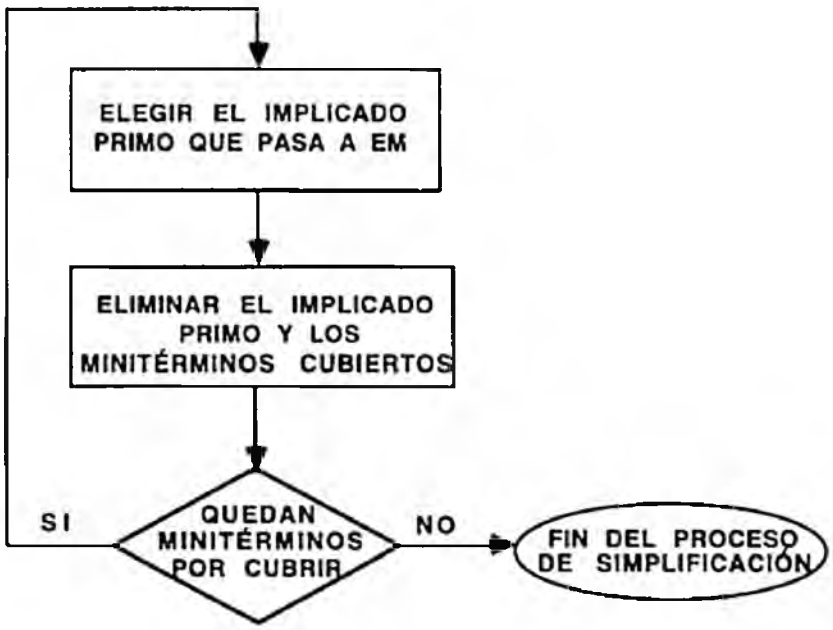


Figura 3.1 Organigrama global del proceso de simplificación

El aspecto más destacable en la operativa anterior es que la función booleana y su VK muestran un aspecto dinámico, al igual que el conjunto de los implicados primos IP. El proceso de simplificación supone ir borrando minitérminos del VK e implicados primos de IP para pasarlos a EM: **la simplificación es un proceso dinámico.**

3.2.1 Implicados primos esenciales y no esenciales

Como ya se ha indicado, simplificar implica discriminar entre implicados primos. Recordando brevemente el capítulo 1, éstos pueden ser de dos tipos:

Implicado primo esencial: al menos uno de los minitérminos por él cubiertos sólo está cubierto por él.

Implicado primo no esencial: todos los minitérminos por él cubiertos lo están al menos por otro implicado primo.

Desde el punto de vista de los minitérminos, éstos pueden ser:

Minitérmino aislado: aquel que está cubierto por un único implicado primo. Un implicado primo es esencial si cubre al menos un minitérmino aislado.

Minitérmino compartido: aquel que está cubierto por más de un implicado primo. Un implicado primo es no esencial si todos los minitérminos que cubre están compartidos.

Un implicado primo esencial siempre pertenece a EM puesto que sólo él asegura cubrir los minitérminos aislados. Mientras que no todos los implicados primos no esenciales deben pertenecer a EM; habrá que elegir el menor subconjunto de ellos que asegure cubrir los minitérminos no cubiertos por los implicados primos esenciales.

Observando la anterior clasificación es obvio que la discriminación se reduce al conjunto de los implicados primos no esenciales. La calidad y rapidez de esta discriminación calificarán la optimidad del método de simplificación en su conjunto

3.2.2 Organigrama global del método de simplificación aportado

Teniendo en cuenta lo dicho, el proceso de simplificación consta de dos fases:

1. Selección de todos los implicados primos esenciales, que pasan a EM.
2. Discriminación de aquellos implicados primos no esenciales que unidos a los anteriores aseguran una expresión mínima en EM.

La figura 3.2 destaca esta serialización en el proceso. En el método aportado la primera fase es resuelta por el algoritmo IPE, mientras la segunda lo es mediante el algoritmo DC.

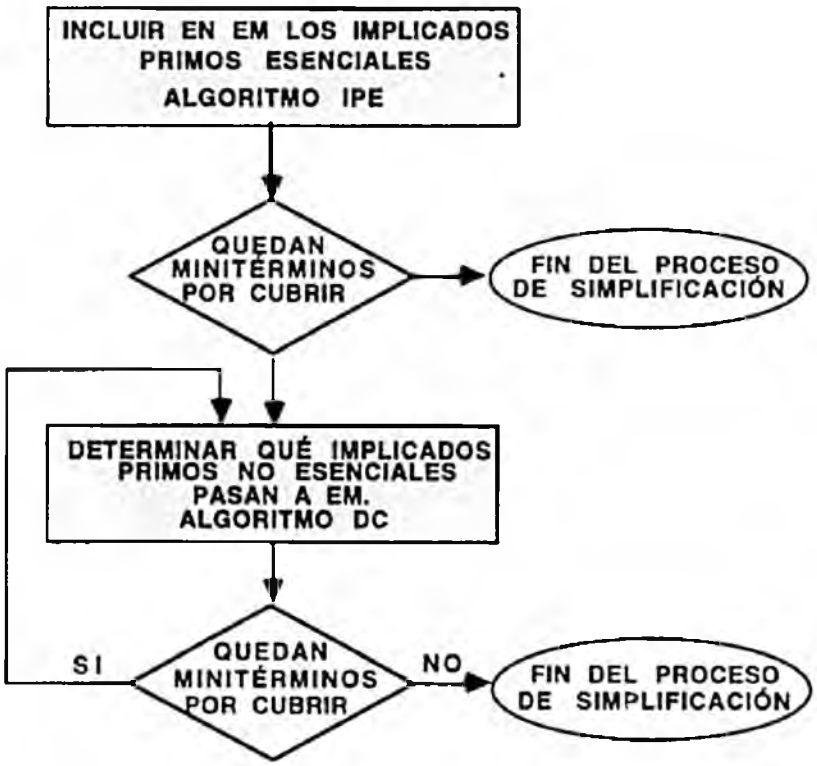


Figura 3.2 Fases principales del proceso de simplificación

La figura 3.3 completa el anterior organigrama con la aplicación de la relación de dominancia. Esta relación elimina de IP aquellos implicados primos dominados por otros, reduciendo el conjunto de implicados primos para facilitar la selección de los implicados primos.

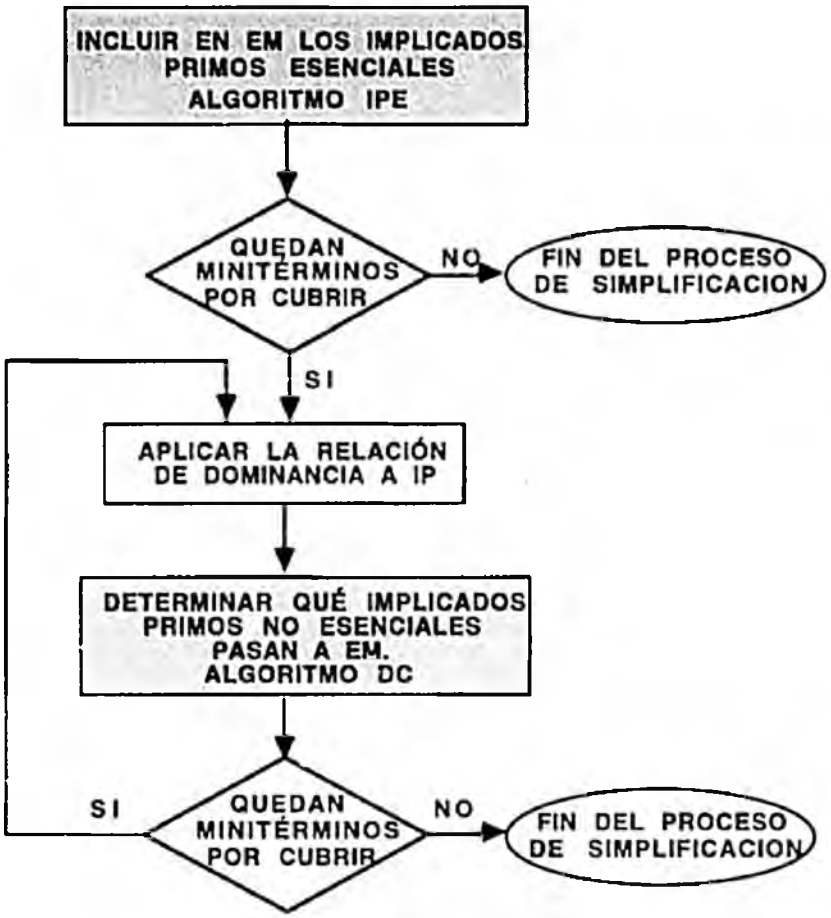


Figura 3.3 Fases principales del proceso de simplificación con relación de dominancia

3.3 Obtención de los implicados primos esenciales

La elección de los implicados primos esenciales es muy sencilla, recordando que es esencial por cuanto que al menos uno de los minitérminos es únicamente cubierto por él. Y por tanto la única forma de cubrir ese minitérmino es mediante su correspondiente implicado primo esencial.

El algoritmo encargado de la obtención de los implicados primos esenciales es el IPE.

Algoritmo 3.1 *Algoritmo IPE*

Partiendo de IP, CS_IP y VK -definidos en el capítulo 2- el proceso de obtención de los implicados primos esenciales por el algoritmo IPE comprende las siguientes fases:

1. Indicar para cada minitérmino de VK cuántos implicados primos le cubren. Submatriz VK_IP
2. Para cada minitérmino de VK_IP que tiene un 1 -minitérmino aislado- obtener el identificador del implicado primo esencial que lo cubre. Submatriz IPE
3. Los identificadores de IPE pasan a EM.
4. Los identificadores de IPE y sus correspondientes minitérminos son borrados de IP y CS_IP.
5. Los minitérminos cubiertos por los implicados de IPE son borrados de VK.

El organigrama de la figura 3.4 muestra las fases anteriores:

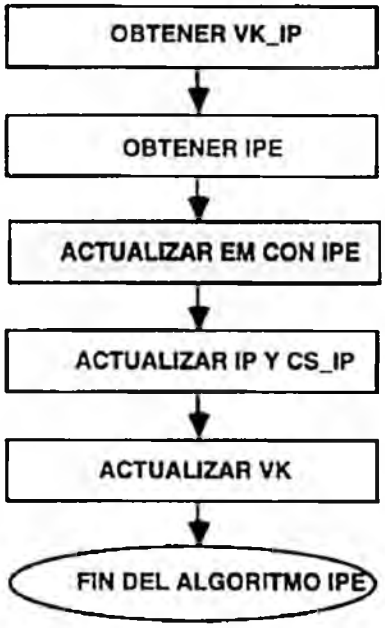


Figura 3.4 Organigrama del algoritmo IPE

Al completar estas fases EM contiene todos los implicados primos esenciales -si los hubiera-, que a su vez han sido eliminados de IP. Además son borrados de VK los correspondientes minitérminos.

Es posible que los implicados primos esenciales cubran la totalidad de los minitérminos de VK, no siendo necesario por tanto continuar el proceso de simplificación; en caso contrario es necesario elegir mediante el algoritmo DC qué implicados primos no esenciales forman parte de EM.

Ejemplo 3.1

Simplificar la función booleana representada por el VK de la figura 3.5.

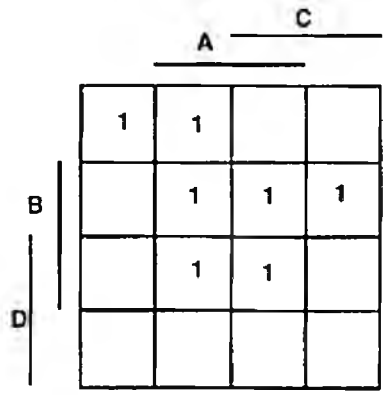


Figura 3.5 Diagrama VK correspondiente al ejemplo 3.1

Aplicando el algoritmo de obtención de los implicados primos (Capítulo 2) obtenemos las matrices IP y CS_IP:

IMPLICADOS PRIMOS: IP	MINITÉRMINOS: CS_IP
1122	12, 13, 14 y 15
2110	6 y 14
1200	8 y 12
2000	0 y 8

Tabla 3.1 Matrices IP y CS_IP del ejemplo 3.1

En la siguiente tabla mostramos cuántos implicados primos cubren a cada minitérmino, en caso de ser uno indicamos cuál es el implicado primo esencial correspondiente.

MINITÉRMINOS	VK_IP	MINITÉRMINOS CUBIERTOS POR ESENCIALES	IMPLICADOS PRIMOS QUE PASAN A EM: IPE
0	1	•	2000
6	1	•	2110
8	2		
12	2		

Tabla 3.2 Obtención de los implicados primos esenciales del ejemplo 3.1

13	1	*	1122
14	2		
15	1	*	1122

Tabla 3.2 *(continuación)*

Los implicados primos esenciales de EM y sus correspondientes minitérminos son: 2000 (0 y 8), 2110 (6 y 14) y 1122 (12, 13, 14 y 15). En este ejemplo todos los minitérminos de VK quedan cubiertos por los implicados primos esenciales, y no es necesario aplicar el algoritmo DC. Así resulta que EM contiene a: 2000, 2110 y 1122.

Ejemplo 3.2

Simplificar la función booleana representada por el VK de la figura 3.6.

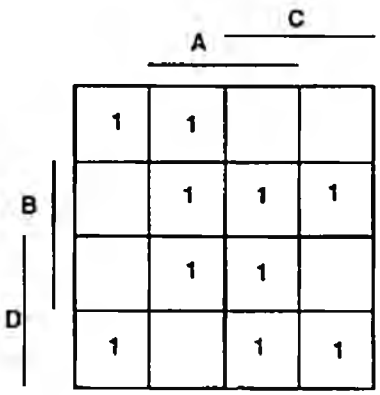


Figura 3.6 Diagrama VK correspondiente al ejemplo 3.2

Al igual que en el ejmplo anterior se forman las matrices IP y CS_IP:

IMPLICADOS PRIMOS: IP	MINITÉRMINOS: CS_IP
1122	12, 13, 14 y 15
1200	8 y 12
2000	0 y 8
2110	6 y 14
1211	11 y 15
2011	3 y 11
0002	0 y 1
0021	1 y 3

Tabla 3.3 Matrices IP y CS_IP del ejemplo 3.2

E indicamos en la tabla 3.4 cuántos implicados primos cubren a cada minitérmino, en caso de ser uno marcamos cuál es el implicado primo esencial correspondiente.

MINITÉRMINOS	VK_IP	MINITÉRMINOS CUBIERTOS POR ESENCIALES	IMPLICADOS PRIMOS QUE PASAN A EM: IPE
0	2		
1	2		
3	2		
6	1	•	2110
8	2		
11	2		
12	2		
13	1	•	1122
14	2		
15	2		

Tabla 3.4 Obtención de los implicados primos esenciales del ejemplo 3.2

Los implicados primos esenciales de EM y sus minitérminos son: 2110 (6 y 14) y 1122 (12, 13, 14 y 15). Faltan por cubrir: 0, 1, 3, 8 y 11, siendo necesario obtener qué implicados primos no esenciales de: 1200 (8 y 12), 2000 (0 y 8), 1211 (11 y 15), 2011 (3 y 11), 0002 (0 y 1) y 0021 (1 y 3) pasan a EM. Esta decisión corresponde al algoritmo DC, a desarrollar y valorar en las siguientes secciones.

3.4 Discriminación entre los implicados primos no esenciales

En la primera fase del proceso de simplificación obteníamos los implicados primos esenciales, pasando todos ellos a la expresión mínima EM. El resto son implicados primos no esenciales: aquellos cuyos minitérminos son cubiertos por más de un implicado primo.

En el ejemplo 3.2, y tras obtener los implicados primos esenciales, quedaban los implicados primos no esenciales junto con los minitérminos que quedan por cubrir entre paréntesis: 1200 (8), 2000 (0 y 8), 1211 (11), 2011 (3 y 11), 0002 (0 y 1) y 0021 (1 y 3). Como se puede observar todos los minitérminos son cubiertos por más de un lazo. El algoritmo DC debe decidir cuáles de los seis implicados primos no esenciales deben pasar a EM.

Antes de desarrollar el algoritmo DC distingueremos entre los implicados primos no esenciales distintas categorías, según su naturaleza.

3.4.1 Clasificación de los implicados primos no esenciales y de las funciones cíclicas

Un implicado primo no esencial puede ser de dos tipos:

- **Implicado primo no esencial completo.** Todos y cada uno de los minitérminos por él cubiertos no están cubiertos por ningún implicado primo perteneciente a EM.
- **Implicado primo no esencial incompleto.** Al menos un minitérmino de los cubiertos por él ya está cubierto por un implicado primo perteneciente a EM.

En el ejemplo 3.2 el implicado primo no esencial 1200 está incompleto ya que *'su'* minitérmino 12 está cubierto por el implicado primo 1122 de EM. Por contra, el implicado primo no esencial 2000 está completo ya que *'sus'* minitérminos 0 y 8 todavía no han sido cubiertos por ningún implicado de EM.

En base a los implicados primos que contiene una función ésta puede ser:

- **Función no cíclica.** Aquella que contiene al menos un implicado primo esencial.
- **Función cíclica total.** Aquella que contiene sólo implicados primos no esenciales completos.
- **Función cíclica parcial.** Aquella que contiene sólo implicados primos no esenciales, y al menos uno está incompleto.

Es necesario acotar la anterior clasificación. Si en un VK los minitérminos se distribuyen en subconjuntos (subfunciones) cuyos implicados primos son disjuntos entre sí -no comparten minitérminos- entonces cada subfunción puede ser de cualquiera de los tres tipos

anteriores. No estaríamos ante una función sino ante la disyunción de varias, donde cada una de ellas puede comportarse de cualquier forma.

Recordemos que el proceso de simplificación es tal que la función booleana a simplificar pasa por diversos estados. Así inicialmente puede ser no cíclica, y, quizá, el traspaso de sus implicados primos no esenciales a EM la convierta en cíclica total o parcial. A su vez el traspaso sucesivo de implicados primos no esenciales a EM modificará de nuevo el aspecto del VK, y así sucesivamente. En el método presentado en la tesis, durante el proceso de simplificación un VK sólo puede pasar de no cíclico a cíclico, o ser siempre cíclico -como quedó claro en el organigrama de la figura 3.2.

Por ejemplo, la figura 3.7 muestra tres momentos de la simplificación de la función booleana representada por el primer VK. El aspecto de la función pasa de no cíclica a cíclica total, y finalmente a cíclica parcial. Cada figura se corresponde con el paso de un implicado primo, esencial o no, a EM.

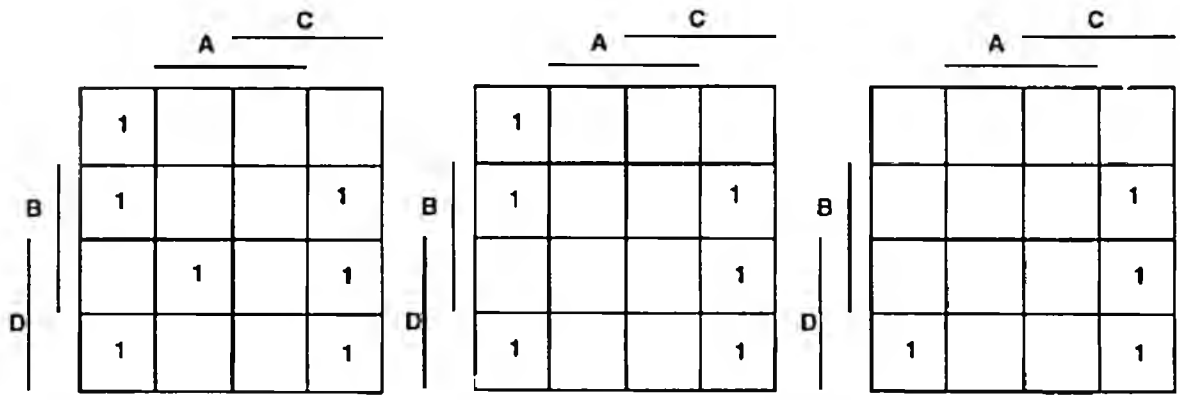


Figura 3.7 Diversos aspectos de un VK en su proceso de simplificación

Visualmente en el VK de una función cíclica total podemos recorrer todos sus minitérminos sin *saltar* entre implicados, volviendo al minitérmino original que puede ser cualquiera -gráficamente recuerda a una cadena o anillo. En una función cíclica parcial este recorrido no puede volver al minitérmino original, pero no hay saltos en su recorrido -gráficamente recuerda a una cadena rota.

3.4.2 Filosofía y objetivos globales del algoritmo de discriminación DC

El método de simplificación aportado en la tesis tiene como hito novedoso en este tercer capítulo el algoritmo de Discriminación Comparativa -DC-, encargado de discriminar qué implicados primos no esenciales son necesarios para, unidos a los esenciales, conformar la mínima expresión de la función booleana original.

El algoritmo DC se procesa cuando el VK presenta un estado cíclico o encadenado -segunda fase del organigrama de la figura 3.2. Tiene como datos de entrada a IP y CS_IP, y como salida obtiene qué implicado primo no esencial debe formar parte de la expresión mínima EM.

El algoritmo DC es: **heurístico, directo, no-óptimo, computacional y rápido.**

La calidad del algoritmo DC califica globalmente al método de simplificación. Su estudio se centra en dos aspectos:

- **Cualitativo.** Desarollo del algoritmo DC y de su *'tie-breaking rule'*.
- **Cuantitativo.** Comparar la optimidad y rapidez del nuevo método con los clásicos aportados en el capítulo 1.

'tie breaking rule' **del algoritmo DC:**

> **Elegir aquel implicado primo que cubra el mayor número de determinados minitérminos, que a su vez son cubiertos por el menor número de determinados implicados primos.**
>
> O también:
>
> ***Cruzar* minitérminos con implicados, y en base a este cruce *filtrar* ambos conjuntos según un criterio de máximo y mínimo. Repetir el proceso con los conjuntos obtenidos del filtrado hasta obtener el implicado primo no esencial que debe pasar a EM.**

Las dos operaciones principales para implementar la *'tie-breaking rule'* del algoritmo de DC son ***cruzar y filtrar***: Sean IPXX y CSXX dos matrices

con algunos implicados primos y con algunos minitérminos, respectivamente, entonces:

- Cruzar las matrices IPXX y CSXX tiene como resultado una matriz que refleja cuántos minitérminos cubre cada implicado, o viceversa: cuántos implicados cubren a cada minitérmino.
- Filtrar una matriz supone retener parte de los elementos de ella, eliminando otros, según un determinado criterio. El filtrado de una matriz es precedido por un cruce entre dos.

En los apartados 3.4.3 y 3.4.4 desarrollamos e implementamos mediante el algoritmo DC, basándonos en las operaciones *cruzar* y *filtrar*, la '*tie-breaking rule*' enunciada. La explicación participa de un enfoque '*top-down*', el primer apartado presenta las fases del algoritmo y el segundo las detalla.

3.4.3 Fases del algoritmo DC

El algoritmo DC se compone de tres fases y una previa, según refleja la figura 3.8. Los objetivos de cada una de las fases son:

FASE PREVIA Esta fase obtiene las matrices que son la base de las siguientes: IP0 que contiene los implicados primos no esenciales y completos, y CS0 que contiene los minitérminos no cubiertos todavía por EM.

DECISIÓN: Si al menos un implicado primo no esencial está completo entonces se procesa la segunda fase, en caso contrario -si IP0 está vacía- la fase primera.

FASE PRIMERA Obtenemos los minitérminos de CS0 cubiertos por el menor número de implicados de IP. Pasamos a EM aquel implicado de IP que cubra el mayor número de los anteriores minitérminos.

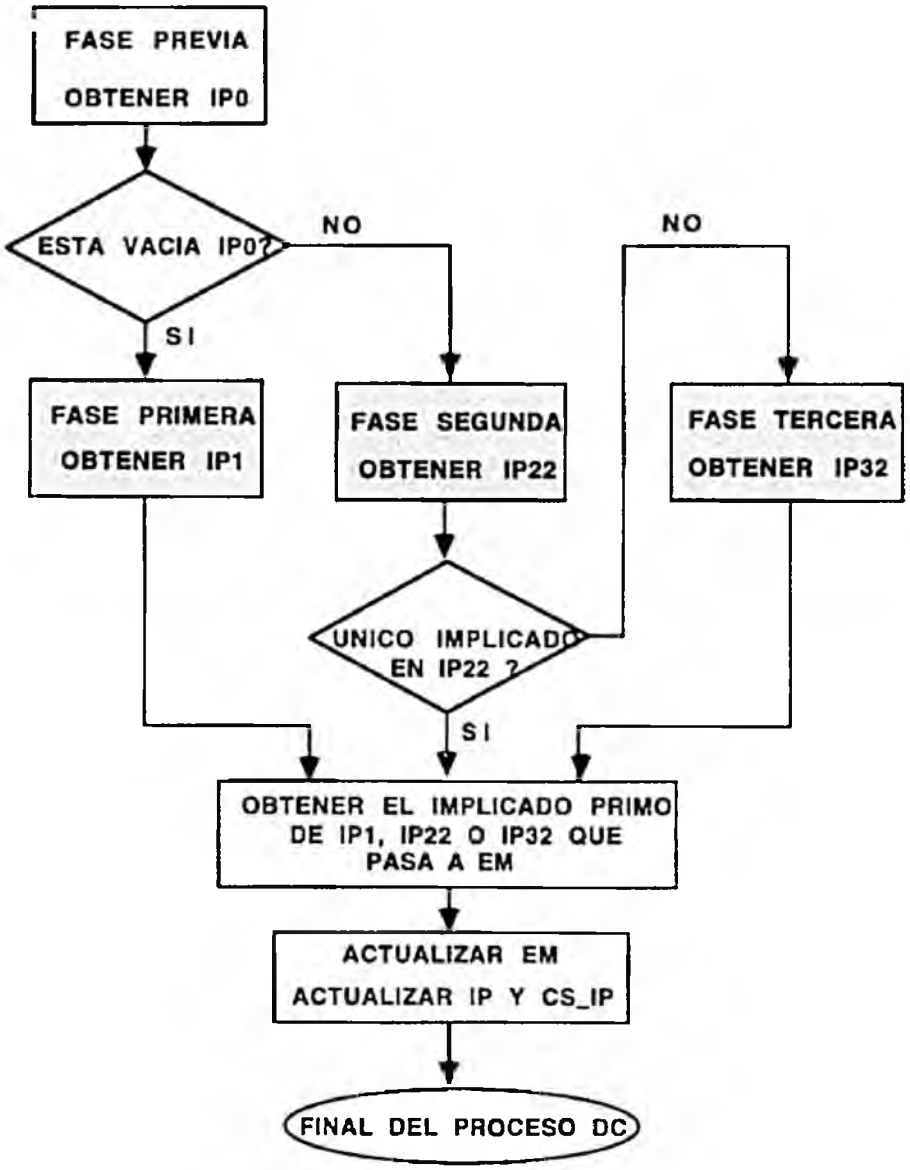


Figura 3.8 Fases del algoritmo DC

FASE SEGUNDA En el caso de existir implicados primos no esenciales y completos obtenemos qué minitérminos de CS0 son cubiertos por el menor número de éstos, entonces seleccionamos qué implicados primos no esenciales de IP -o sea, completos o incompletos- cubren a un mayor número de los minitérminos anteriores. Si la selección anterior aporta más de un implicado primo entonces elegimos entre ellos aquel que cubre un mayor número de minitérminos de CS0 -o sea, cualquier minitérmino.

DECISIÓN: Si la última selección de la fase segunda produjera un único implicado primo éste pasaría a EM, en caso contrario completamos la tercera y última parte, partiendo de este conjunto final de implicados primos.

FASE TERCERA La tercera parte repite el proceso de la segunda cambiando las matrices a relacionar. Por tanto, esta tercera fase supone volver a discriminar dentro del conjunto de implicados y minitérminos obtenidos por la segunda fase. Esta nueva discriminación obtiene un nuevo conjunto de implicados primos del que cualquiera -no todos- puede pasar a EM.

3.4.4 Organigrama detallado del algoritmo DC

Las tres fases de la sección anterior se basan en *cruzar* y *filtrar* distintas matrices, operaciones detalladas a continuación:

FASE PREVIA

Obtener CS0 minitérminos que quedan por cubrir, no condiciones libres.

Obtener IP0 aquellos implicados que sólo cubren minitérminos de CS0 o condiciones libres -implicados primos no esenciales y completos.

DECISIÓN: Si la matriz IP0 está vacía quiere decir que todos los implicados primos de IP están incompletos, procesando la primera fase. En caso contrario se procesa la segunda fase.

FASE PRIMERA

Cruzar CS0 e IP la matriz CS0_IP indica cuántos implicados de IP cubren a cada minitérmino de CS0.

Filtrar CS0 la matriz CM1 contiene los minitérminos de CS0 cubiertos por un menor número de implicados de IP.

Cruzar IP y CM1 la matriz IP_CM1 indica cuántos minitérminos de CM1 cubre cada implicado de IP.

Filtrar IP la matriz IP1 contiene los implicados de IP que cubren un mayor número de minitérminos de CM1.

Pasar a EM cualquiera - pero no todos- de los implicados primos no esenciales e incompletos de IP1 puede pasar a EM.

FASE SEGUNDA

Cruzar CS0 e IP0 la matriz **CS0_IP0** indica cuántos implicados de IP0 cubren a cada minitérmino de **CS0.**

Filtrar CS0 la matriz **CM2** contiene los minitérminos de CS0 cubiertos por un menor número de implicados de **IP0.**

Cruzar IP y CM2 la matriz **IP_CM2** indica cuántos minitérminos de CM2 cubre cada implicado de IP.

Filtrar IP la matriz **IP21** contiene los implicados de IP que cubren un mayor número de minitérminos de CM2.

Cruzar IP21 y CS0 la matriz **IP21_CS0** indica cuántos minitérminos de CSO cubre cada implicado de IP21.

Filtrar IP21 la matriz **IP22** contiene los implicados de IP21 que cubren un mayor número de casillas de CM2.

DECISIÓN: Si IP21 tiene un único elemento, éste pasa a EM. Si por el contrario IP22 contiene varios implicados procesamos la tercera fase casi idéntica a la segunda.

FASE TERCERA

Cruzar CM2 e IP22 la matriz **CM2_IP22** indica cuántos implicados de IP22 cubren a cada minitérmino de CM2.

Filtrar CM2 la matriz **CM3** contiene los minitérminos de CM2 cubiertos por un menor número de implicados de **IP22.**

Cruzar IP22 y CM3 la matriz **IP22_CM3** indica cuántos minitérminos de CM3 cubre cada implicado de IP22.

Filtrar IP22 la matriz **IP31** contiene los implicados de IP22 que cubren un mayor número de minitérminos de CM3.

Cruzar IP31 y CS0 la matriz **IP31_CS0** indica cuántos minitérminos de CS0 cubre cada implicado de IP31.

Filtrar IP31 la matriz **IP32** contiene los implicados de IP31 que cubren un mayor número de minitérminos de CS0.

Pasar a EM cualquiera -pero no todos- de los implicados primos no esenciales de IP32 puede pasar a EM.

A simple vista puede parecer que la tercera fase es idéntica a la segunda, cambiando únicamente las matrices de entrada -como muestra el organigrama de la figura 3.9. Así IP0 se convertiría en IP22 y CS0 en CM2. Pero hay que hacer notar que el algoritmo en la segunda fase *cruza* IP con CM2, y no IP0 con CM2 como sería en caso de ser identicas las dos fases. Es totalmente imprescindible para el buen funcionamiento del método -ver ejemplos 3.3 a 3.6- este cruce y no otro, aunque cruzar IP0 y CM2 parezca más *natural*.

El organigrama de la figura 3.9 detalla todas las fases y las acciones que las conforman.

Desde el punto de vista de la implementación las subrutinas asociadas a ***cruzar*** y ***filtrar*** son muy sencillas de codificar, máxime en un entorno orientado a matrices como MATLAB.

3.4.5 Versión modificada del algoritmo DC

La tercera fase del algoritmo DC concluye pasando cualquiera de los implicados de IP32 a EM, pero también podemos repetir la tercera fase: donde IP32 se convierte en IP22 y CM3 en CM2. Este proceso, computacionalmente caro, se repite hasta que IP32 tenga un único elemento, o hasta que IP32 no haya experimentado cambio en la repetición de la tercera fase. Es decir, *refinamos* el contenido de IP32 mientras sea posible hasta dejar un único implicado que pasará a EM. En principio este planteamiento parece adecuado y casi inexcusable, pero en la experimentación con el algoritmo observamos que EM es tan óptima como sin refinamiento -reafirmando el carácter heurístico del algoritmo DC.

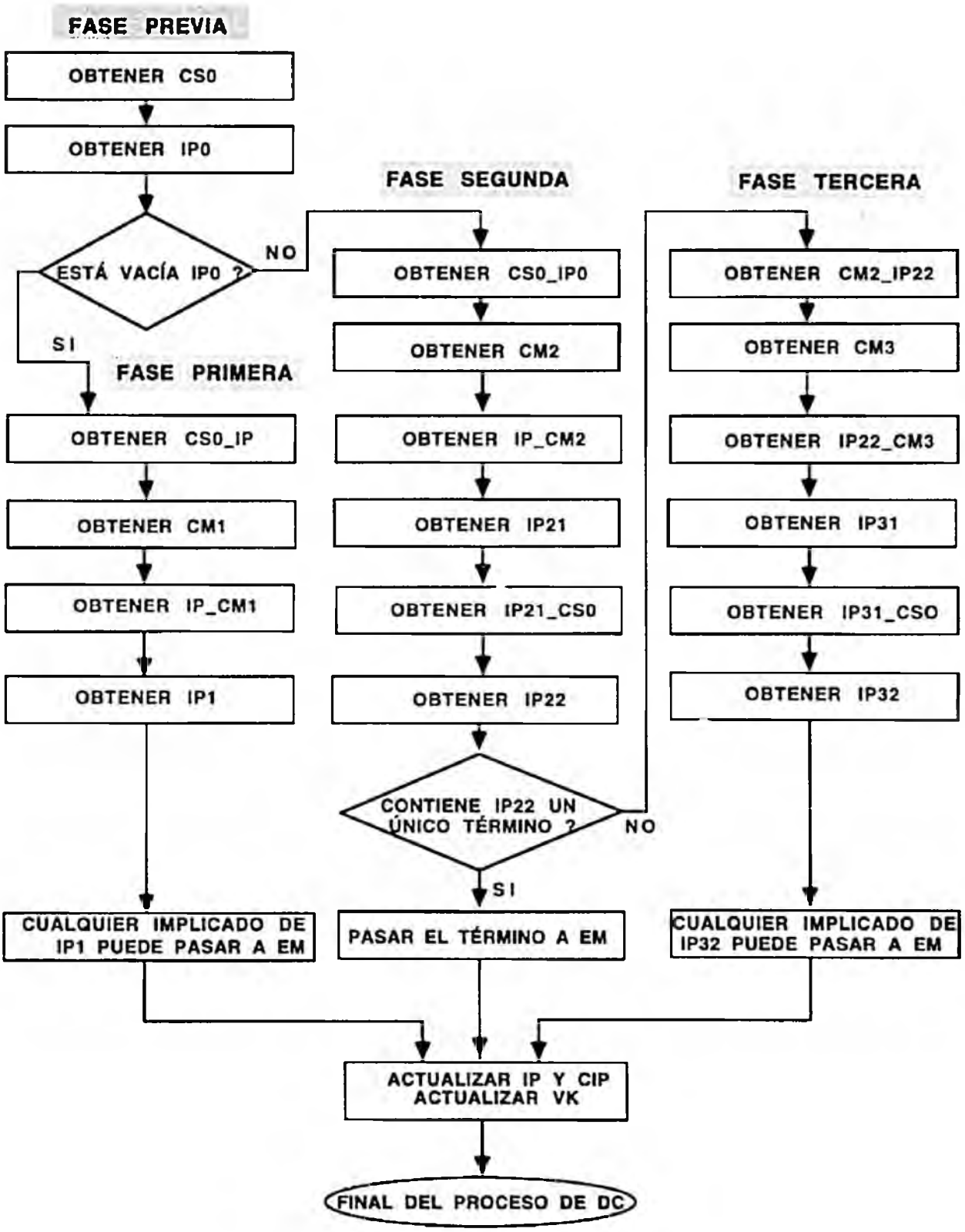


Figura 3.9 Organigrama general del algoritmo DC

La figura 3.10 representa la estructura modificada de la tercera fase de la figura 3.9, las restantes columnas de proceso permanecen inalteradas.

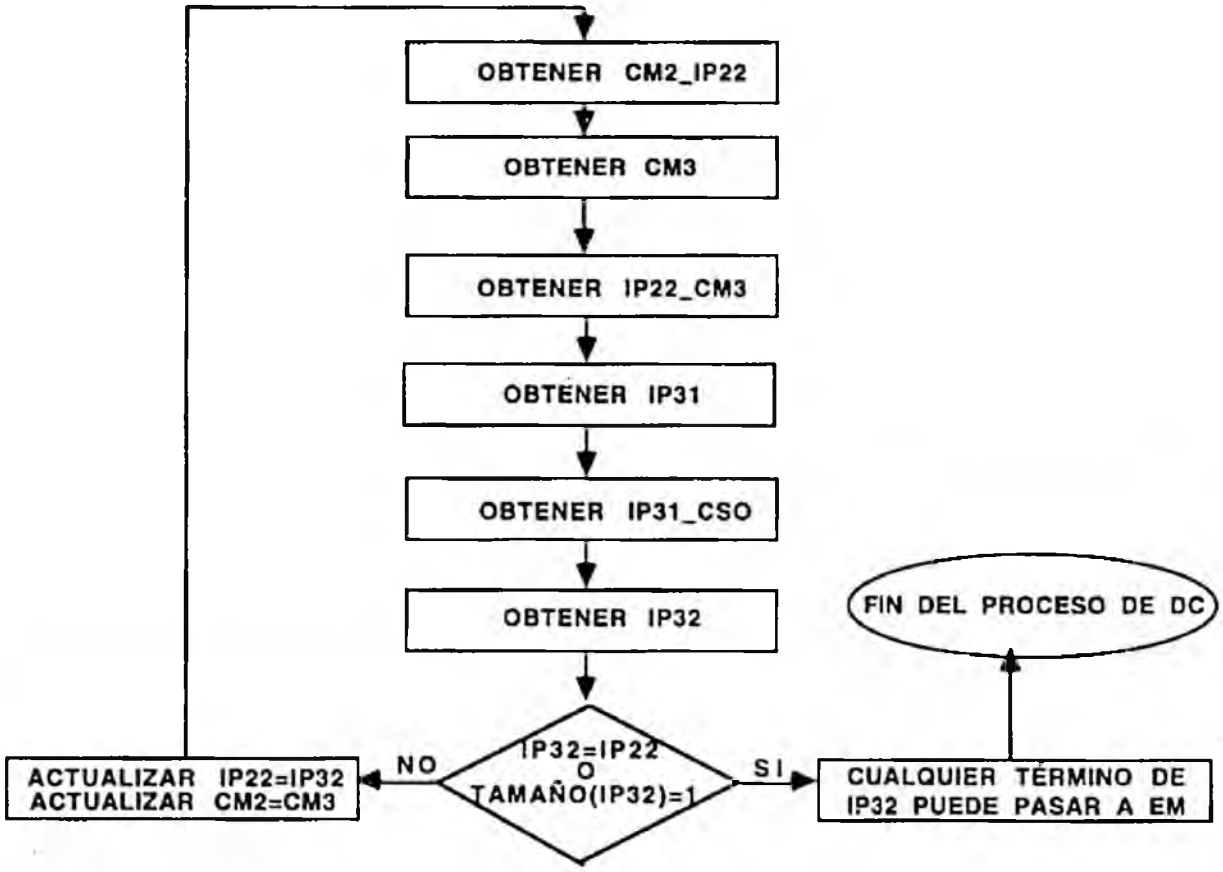


Figura 3.10 Estructura alternativa de la tercera fase del algoritmo DC

3.4.6 Ejemplos ilustrativos del algoritmo DC

Los tres ejemplos 3.3, 3.4 y 3.5 de la figura 3.11 son originalmente funciones cíclicas completas o encadenadas. Su proceso de simplificación está detallado en las tablas 3.5 a 3.12 de las correspondientes hojas apaisadas. Cada línea horizontal gruesa es una *pasada* del algoritmo simplificador, y las columnas representan a cada una de las fases de DC. Cada *pasada* obtiene **directamente** qué implicado primo pasa a la expresión mínima EM. Hemos obviado representar la búsqueda de los implicados primos esenciales en las tablas correspondientes a los ejemplos 3.3, 3.4 y 3.5, ya que en estos ejemplos

las funciones booleanas son cíclicas totales,y no tienen implicados primos esenciales.

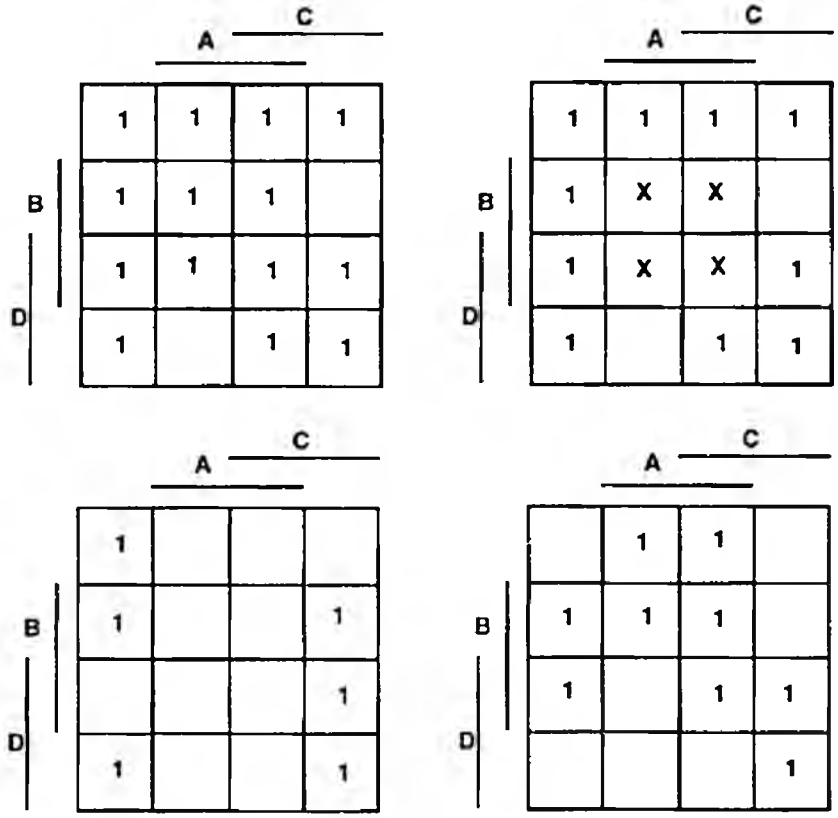


Figura 3.11 Funciones booleanas a simplificar de los ejemplos 3.3, 3.4, 3.5 y 3.6

En los siguientes párrafos destacamos aquellos aspectos fundamentales de los ejemplos que ayudan a comprender y valorar el método de simplificación aportado en la tesis, principalmente su algoritmo DC:

- En una función cíclica total el primer implicado primo seleccionado para pasar a EM es cualquiera del conjunto de implicados primos.
- En el ejemplo 3.3, en la segunda iteración del método y dentro de la tercera columna de la segunda fase -matriz IP_CM2- debemos remarcar que es IP la matriz utilizada. Esto conlleva la selección final de 2200 o 2211. Ambos implicados primos no pertenecen a IP0, con lo que si hubieramos cruzado IP0 con CM2 éstos no habrían sido elegidos. Es mas, si hubiesemos utilizado IP0 la expresión mínima final hubiese sido incorrecta. La matriz IP_CM2 es fundamental en DC.

EJEMPLO 3.3

IMPLICADO PRIMO	MINTERMINOS CUBIERTOS
0022	0, 1, 2 y 3
1122	12, 13, 14 y 15
0202	0, 1, 4 y 5
1212	10, 11, 14 y 15
1220	8, 10, 12 y 14
0221	1, 3, 5 y 7
2102	4, 5, 12 y 13
2012	2, 3, 10 y 11
2020	0, 2, 8, y 10
2121	5, 7, 13 y 15
2200	0, 4, 8 y 12
2211	3, 7, 11 y 15

Tabla 3.5 Implicados primos y minitérminos corespondientes al ejemplo 3.3

	ALG. IPE	ALGORITMO DC															
ENTR.		FASE PREVIA		PRIMERA FASE	SEGUNDA FASE						TERCERA FASE						SALIDA
IP		CS0	IP0		CS0_IP0	CM2	IP_CM2	IP21	IP21_CS0	IP22	CM2_IP22	CM3	IP22_CM3	IP31	IP31_CS0	IP32	EM
0022		0	0022		4	1	2	0022	4	0022	3	1	2	0022	4	0022	0022
1122		1	1122		3	2	2	1122	4	1122	3	2	2	1122	4	1122	
0202		2	0202		3	4	2	0202	4	0202	3	4	2	0202	4	0202	
1212		3	1212		4	7	2	1212	4	1212	3	7	2	1212	4	1212	
1220		4	1220		3	8	2	1220	4	1220	3	8	2	1220	4	1220	
0221		5	0221		4	11	2	0221	4	0221	3	11	2	0221	4	0221	
2102		7	2102		3	13	2	2102	4	2102	3	13	2	2102	4	2102	
2012		8	2012		3	14	2	2012	4	2012	3	14	2	2012	4	2012	
2020		10	2020		4		2	2020	4	2020			2	2020	4	2020	
2121		11	2121		3		2	2121	4	2121			2	2121	4	2121	
2200		12	2200		4		2	2200	4	2200			2	2200	4	2200	
2211		13	2211		3		2	2211	4	2211			2	2211	4	2211	
		14			3												
		15			4												

Tabla 3.6 Proceso de simplificación del ejemplo 3.3

1122 0202 1212 1220 0221 2102 2012 2020 2121 2200 2211		4 5 7 8 10 11 12 13 14 15	1122 1212 1220 2102 2121		1 2 1 1 2 1 3 3 3 3	4 7 8 11	0 1 1 1 1 1 1 1 2 2	2200 2211	3 3	2200 2211	1 1 1 1	4 7 8 11	2 2	2200 2211	3 3	2200 2211	0022 2200
1122 0202 1212 1220 0221 2102 2012 2020 2121 2211		5 7 10 11 13 14 15	1212 2121		1 1 1 1 1 1 2	5 7 10 11 13 14	2 1 3 2 2 2 2 1 3 2	1212 2121	4 4	1212 2121	1 1 1 1 1 1	5 7 10 11 13 14	3 3	1212 2121	4 4	1212 2121	0022 2200 1212
1122 0202 1220 0221 2102 2012 2020 2121 2211		5 7 13		2121													0022 2200 1212 2121

Tabla 3.6 (*continuación*)

EJEMPLO 3.4

IMPLICADO PRIMO	MINTERMINOS CUBIERTOS
0022	0, 1, 2 y 3
0202	0, 1, 4 y 5
1212	10, 11, 14 y 15
1220	8, 10, 12 y 14
0221	1, 3, 5 y 7
2102	4, 5, 12 y 13
2012	2, 3, 10 y 11
2020	0, 2, 8, y 10
2121	5, 7, 13 y 15
2200	0, 4, 8 y 12
2211	3, 7, 11 y 15

Tabla 3.7 Implicados primos y minitérminos corespondientes al ejemplo 3.4

	ALG. IPE	ALGORITMO DC														
ENTR.		FASE PREVIA		PRIMERA FASE	SEGUNDA FASE						TERCERA FASE						SALIDA
IP		CS0	IP0		CS0_IP0	CM2	IP_CM2	IP21	IP21_CS0	IP22	CM2_IP22	CM3	IP22_CM3	IP31	IP31_CS0	IP32	EM
0022		0	0022		4	1	2	0022	4	0022	3	4	0	0202	4	0202	0202
0202		1	0202		3	2	2	0202	4	0202	3	7	1	0221	4	0221	
1212		2	1212		3	4	1	0221	4	0221	1	8	1	2012	4	2012	
1220		3	1220		4	7	1	2012	4	2012	1	11	1	2020	4	2020	
0221		4	0221		3	8	2	2020	4	2020	1		1				
2102		5	2102		4	11	1	2200	3		1						
2012		7	2012		3		2	2211	3								
2020		8	2020		3		2										
2121		10	2121		4		1										
2200		11	2200		3		2										
2211			2211				2										
0022		2	1212		1	2	1	2020	3	2020							0202
1212		3	1220		2	7	0										2020
1220		7	2012		1	8	1										
0221		8	2211		1		1										
2102		10			3		0										
2012		11			3		1										
2020							2										
2121							1										
2200							1										
2211							1										

Tabla 3.8 Proceso de simplificación del ejemplo 3.4

0022 1212 1220 0221 2102 2012 2121 2200 2211		3 7 11	2211		1 1 1	3 7 11	1 1 0 2 0 2 1 0 3	2211									0202 2020 2211

Tabla 3.8 (*continuación*)

EJEMPLO 3.5

IMPLICADO PRIMO	MINTERMINOS CUBIERTOS
0002	0 y 1
0112	6 y 7
0120	4 y 6
0021	1 y 3
0200	0 y 4
0211	3 y 7

Tabla 3.9 Implicados primos y minitérminos corespondientes al ejemplo 3.5

	ALG. IPE	ALGORITMO DC															
ENTR.		FASE PREVIA		PRIMERA FASE	SEGUNDA FASE						TERCERA FASE						SALIDA
IP		CS0	IP0		CS0_IP0	CM2	IP_CM2	IP21	IP21_CS0	IP22	CM2_IP22	CM3	IP22_CM3	IP31	IP31_CS0	IP32	EM
0002		0	0002		2	0	2	0002	2	0002	2	0	2	0002	2	0002	0002
0112		1	0112		2	1	2	0112	2	0112	2	1	2	0112	2	0112	
0120		3	0120		2	3	2	0120	2	0120	2	3	2	0120	2	0120	
0021		4	0021		2	4	2	0021	2	0021	2	4	2	0021	2	0021	
0200		6	0200		2	6	2	0200	2	0200	2	6	2	0200	2	0200	
0211		7	0211		2	7	2	0211	2	0211	2	7	2	0211	2	0211	
0112		3	0112		1	3	0	0120	2	0120	1	3	1	0120	2	0120	0002
0120		4	0120		1	4	1	0021	1	0211	1	4	1	0211	2	0211	0120
0021		6	0211		2		1	0200	1								
0200		7			2		1	0211	2								
0211							1										
0112		3	0211		1	3	1	0211	2	0211							0002
0021		7			1	7	1										0120
0200							0										0211
0211							2										

Tabla 3.10 Proceso de simplificación correspondiente al ejemplo 3.5

EJEMPLO 3.6

IMPLICADO PRIMO	MINTERMINOS CUBIERTOS
1220	8, 10, 12 y 14
2100	4 y 12
0102	4 y 5
1112	14 y 15
0211	3 y 7
0121	5 y 7
2111	7 y 15

Tabla 3.11 Implicados primos y minitérminos corespondientes al ejemplo 3.6

	ALGOR. IPE	ALGORITMO DC													
ENTR.		FASE PREVIA		PRIMERA FASE				SEGUNDA FASE						TERCERA FASE	SALIDA
IP	IPE	CS0	IP0	CS0_IP	CM1	IP_CM1	IP1	CS0_IP0	CM2	IP_CM2	IP21	IP21_CS0	IP22		EM
1220 2100 0102 1112 0121 2111 0211	1220 0211														1220 0211
2100 0102 1112 0121 2111		4 5 15	0102					1 1 0	4 5	1 2 0 1 0	0102	2	0102		1220 0211 0102
2100 1112 0121 2111		15		2	15	0 1 0 1	1112 2111								1220 0211 0102 1112

Tabla 3.12 Proceso de simplificación correspondiente al ejemplo 3.6

- En el ejemplo 3.3 en la última iteración no existe IP0, el implicado primo que pasa a EM es seleccionado en la primera fase explicada en el ejemplo 3.6. Baste ahora decir que la elección de 2121 es obvia por cuanto que cubre todos los minitérminos restantes.
- En el ejemplo 3.4, en la segunda iteración del método queda claro el uso de las condiciones libres en la formación de IP0. Así, el implicado primo 1212 pertenece a IP0 aun cubriendo dos condiciones libres -casillas 11 y 14. Remarcamos que IP0 se forma tanto a partir de CS0 -minitérminos por cubrir- como a partir de las condiciones libres.
- En algunas situaciones observamos que no es necesaria la creación de determinadas matrices que nada aportan, que no es necesario seguir cruzando minitérminos e implicados primos. Esto reafirma el carácter computacional del método, pues el no detener el proceso de simplificación tiene como contrapartida una sencilla y clara estructura -organigrama de la figura 3.9- de implementación.

3.5 Análisis cuantitativo del nuevo método

El presente apartado comparará la optimidad del método CAMP DEUSTO propuesto frente a los métodos: Quine-McCluskey con y sin Dominancia, Quine-McCluskey con Máximo Lazo y CAMP II de Biswas -(Quine, 1952 y 1955), (McCluskey, 1956) y (Biswas, 1993). Resaltaremos especialmente la comparación con el método CAMP II, por cuanto que siendo de carácter avanzado es implementable, para validar el prototipo CAMP DEUSTO.

El estudio a realizar -presentado en (García y Kahoraho, 1995)- se concentra en la optimidad del método y no en su rapidez, ya que ésta última depende mucho de la implementación y del entorno utilizado. Además, el carácter de prototipo del CAMP DEUSTO nos hace considerar la rapidez como una propiedad inicialmente secundaria; no así la optimidad.

Diseño del experimento

Los resultados se presentan de forma gráfica -figuras 2.12 a 2.43- y agrupados para permitir un rápido análisis visual. Las funciones a simplificar por los distintos métodos tienen de cuatro a ocho variables y son diversas:

- Funciones generadas aleatoriamente que contienen sólo minitérminos. Veinte funciones generadas por cada número de variables.
- Funciones generadas aleatoriamente que contienen minitérminos y condiciones libres. Veinte funciones generadas por cada número de variables.
- Funciones cíclicas totales creadas *ad hoc*. En este caso el valor del índice '*minimization complexity*' de Biswas (1993, p. 86) es máximo.

Ahora bien dentro de los dos primeros tipos de funciones, éstas pueden ser de dos tipos:

- Formadas únicamente por implicados primos no esenciales.
- Formadas por implicados primos esenciales y no esenciales. Funciones cíclicas.

El primer grupo de funciones carece de interés pues todos los métodos de simplificación son totalmente óptimos frente a este tipo de funciones. El interés se centra en las funciones cíclicas, ya sean parciales o totales, y por tanto los resultados presentados se refieren sólo a estas funciones, habiéndose omitido los correspondientes al primer grupo.

Las funciones simplificadas en el experimento se dividen en tres grupos, todos ellos con implicados primos no esenciales:

- El primer grupo está compuesto por funciones con sólo minitérminos.
- El segundo grupo está compuesto por funciones con minitérminos y condiciones libres.

- El tercer grupo está formado por funciones cíclicas totales, para justamente resaltar las diferencias entre los métodos en esta situación.

Índice de optimidad

La optimidad de un método viene determinada por la diferencia existente entre el número de términos de la expresión exacta obtenida por Quine-McCluskey y el número de términos de la expresión por él obtenida. Dicha diferencia puede ser expresada en términos absolutos y relativos, siendo más clara la primera y más adecuada la segunda, y así para cada función y método podemos establecer dos medidas:

- Diferencia absoluta = Nº términos de Q-M - Nº términos del método.
- Diferencia relativa = $\frac{\text{Nº términos de Q-M - Nº términos del método}}{\text{Nº términos de Q-M}}$

El índice de optimidad es muy sencillo, pero totalmente adecuado. Un método es tanto más óptimo cuanto mas cerca de cero está su diferencia relativa, y es tanto menos óptimo cuanto mas cerca está del valor uno.

Los programas han sido codificados en entorno MATLAB 4.2 bajo Windows, el ordenador es un PC-486 DX2 a 66 MHz con 16 MBytes de RAM.

Análisis de los resultados

La siguiente tabla muestra el valor medio de las diferencias relativas de cada método, distinguiendo el tipo de función y sin distinguir el número de variables.

	CAMP DEUSTO DOMINANCIA	CAMP DEUSTO	Q-M con MÁXIMO LAZO	CAMP II
MINITÉRMINOS	0,0008	0,0725	0,0823	0,0437
COND. LIBRES	0,0050	0,0509	0,0718	0,2066
FUN. CÍCLICAS	0	0	0,4043	0,1129

Tabla 3.13 Comparación entre la optimidad de los distintos métodos según el índice de diferencia relativa

Observando las gráficas del Apéndice E y la tabla 3.13 podemos establecer las siguientes conclusiones:

- El método CAMP DEUSTO con Dominancia es el más óptimo.
- La exactitud de CAMP DEUSTO con Dominancia es casi total. Su diferencia absoluta máxima es de un término y tiene carácter puntual -tres funciones.
- La exactitud de CAMP DEUSTO, con Dominancia o no, es total para funciones cíclicas totales.
- La exactitud de Q-M con Máximo Lazo es muy pobre para funciones cíclicas.
- La exactitud de CAMP II de Biswas es pobre para funciones cíclicas totales o con condiciones libres.
- Para nueve variables -gráficas E.25 a E.32- las diferencias son muy acusadas. Ningún método es más óptimo que el CAMP DEUSTO.

Aun cuando la rapidez no es determinante para un prototipo podemos extraer las siguientes conclusiones del experimento:

- El método más rápido es el Q-M con Máximo Lazo.
- El método Q-M, con Dominancia o sin ella, es muy lento.
- El método CAMP II es poco más rapido que CAMP DEUSTO, excepto para funciones cíclicas totales en las que es mucho más rápido CAMP DEUSTO.
- La rapidez de CAMP II depende directamente del número de minitérminos de la función y de su posible condición cíclica.
- La rapidez de CAMP DEUSTO depende del número de implicados primos de la función, no le afecta su posible condición cíclica.
- La rapidez de Q-M con Máximo Lazo depende únicamente del número de implicados primos de la función.
- La aplicación de la relación de dominancia consume bastante tiempo.

3.6 Conclusiones respecto del proceso de minimización

En este capítulo hemos desarrollado un nuevo método de simplificación basado en los implicados primos de la función a simplificar. El método CAMP DEUSTO es directo, computacional y su criterio de discriminación DC -Discriminación Comparativa- es heurístico. La optimidad de CAMP DEUSTO es muy alta, y en cualquier caso superior a los métodos Q-M con Máximo Lazo y CAMP II de Biswas para funciones de cuatro a ocho variables. Los resultados obtenidos nos lleva a confirmar la validez del prototipo como base para un desarrollo más ambicioso y completo, comparable a los métodos ESPRESSO y McBOOLE.

CONCLUSIONES

A través de la tesis se ha desarrollado un prototipo sobre PC de un entorno de simplificación de funciones booleanas, implementado en MATLAB. El núcleo de este prototipo es el método CAMP DEUSTO presentado en los capítulos 2 y 3.

Las principales características de este nuevo método son:

- su *enfoque exploratorio* en la fase de obtención de los implicados primos,
- su *criterio de discriminación* directo en la fase de obtención de la expresión minima y,
- el uso d*el concepto de rueda*, introducido en la tesis, a la hora de generar *la matriz MCVK*.

Los otros métodos son de carácter combinatorio o algebraico, y utilizan un criterio bifurcativo o directo. Una ventaja adicional del CAMP DEUSTO es el hecho de que el método está imbricado con la clasificación de todos los posibles implicados primos en MCVK.

Los resultados de la implementación, reflejados en las figuras y tablas de los capítulos 2 y 3, han permitido comparar cuantitativamente CAMP DESUTO con los métodos de Quine-McCluskey, Quine-McCluskey con Máximo Lazo y CAMP II, estableciéndose las siguientes conclusiones:

Rapidez:

- La rapidez de CAMP DEUSTO es superior en su conjunto a la de los demás métodos de obtención de los implicados primos. El factor determinante en la rapidez es el número de variables en CAMP DEUSTO y el número de minitérminos en Quine-McCluskey.

Optimidad:

- La optimidad de la expresión mínima obtenida por CAMP DEUSTO es superior en todos los casos, muy particularmente para funciones cíclicas totales, a la observada en los otros métodos. La rapidez de obtención de la expresión mínima por CAMP DEUSTO es mayor que la de CAMP II y Quine-McCluskey, y algo inferior a la de Quine-McCluskey con Máximo Lazo.
- En la evaluación del método CAMP II de Biswas (capítulo 1) se constata mediante un ejemplo su escasa optimidad, contrariamente a la opinión del autor. Además, la optimidad de CAMP II se degrada bastante para funciones con condiciones libres.

Finitud:

- La ejecución de los algoritmos inducidos por CAMP DEUSTO, CAMP II y Quine-McCluskey con Máximo Lazo es finita, mientras que la de Quine-McCluskey puede resultar no finita, porque las múltiples recursividades asociadas pueden provocar una explosión combinatoria.

El universo de funciones booleanas contemplado por CAMP DEUSTO abarca a funciones bivaluadas, con una sola salida y expresadas en forma de suma de productos. Este método puede extenderse al caso de funciones multivalentes y con múltiples salidas, así como a funciones expresadas en forma de producto de sumas.

PRINCIPALES APORTACIONES

La metodología seguida en la tesis para resolver el problema de simplificación de funciones booleanas comprende la obtención de los implicados primos y de la expresión mínima, utilizando éstos. Los métodos actuales de simplificación son de dos tipos: algebraicos y combinatorios. La expresión mínima se obtiene aplicando métodos algebraicos, constructivos o discriminatorios: la calidad de un método discriminatorio depende de su rapidez y optimidad. Las dos propiedades son prácticamente imposibles simultáneamente, de ahí que existan dos estrategias discriminatorias: bifurcativa y directa. La bifurcativa prima la optimidad frente a la rapidez, y la directa lo contrario. La optimidad de la estrategia directa depende íntimamente de la calidad del criterio de discriminación desarrollado.

En el desarrollo de la tesis se han aportado un nuevo enfoque y método exploratorio para la obtención de los implicados primos, y un criterio de discriminación directo, distinto de los actuales.

Las principales aportaciones que se derivan del presente trabajo son:

1. Clasificación y evaluación de los principales métodos de simplificación actuales desde un enfoque cuantitativo en base a la rapidez y optimidad del método. Validación de la optimidad del método CAMP II de Biswas.
2. Desarrollo de un nuevo método computacional CAMP DEUSTO con un enfoque exploratorio para la obtención de los implicados primos, cuya implementación es directa.
3. Ordenación de todos los posibles implicados primos que conforman la matriz MCVK utilizando el nuevo concepto de rueda, introducido en la tesis. Dicha ordenación se realiza una sola vez en la utilización del CAMP DEUSTO.

4. Desarrollo de un nuevo criterio de discriminación directo DC para obtener expresiones mínimas en el caso de funciones cíclicas, totales o no.
5. Desarrollo de un entorno de simplificación de funciones booleanas a nivel de prototipo, en plataforma PC y bajo MATLAB.

Entre los métodos actuales de simplificación de funciones booleanas desatacan ESPRESSO y McBOOLE: métodos directo y bifurcativo, respectivamente. Ambos métodos son computacionalmente eficaces y están orientados a *mainframe*.

Analizados los resultados obtenidos del prototipo CAMP DEUSTO, consideramos que una investigación posterior puede situar al CAMP DEUSTO al nivel de ESPRESSO y McBOOLE. Dicha investigación se concentrará en dos principales líneas:

1. Desarrollar en una plataforma *mainframe* el prototipo desarrollado en PC, optimizando los recursos.
2. La estructura de la matriz MCVK desarrollada en la tesis es bidimensional, de gran tamaño y rapidez. Generar la matriz MCVK utilizando como estructura de datos los árboles binarios podrá resultar en una reducción del tamaño, preservando su rapidez. La reformulación e implementación de los algoritmos generadores de MCVK, aplicando la estrategia basada en árboles binarios, y su evaluación comparativa con CAMP DEUSTO es otra línea de investigación.

APÉNDICE A: DEFINICIONES

En este apéndice enunciaremos ciertos conceptos y definiciones básicas que son utilizadas en el desarrollo de la tesis.

Premisa

- Toda función booleana está relacionada con un conjunto de minitérminos, y quizá otro de condiciones libres.

Términos relacionados con lazo

- Un lazo es el producto de distintas variables booleanas negadas o no, y cubre determinados minitérminos.
- Toda función booleana puede ser representada como una disyunción entre lazos.

Términos relacionados con los implicados primos

- Sean F y G dos funciones booleanas; se dice que F cubre a G -$G \subseteq F$- si cuando G toma el valor 1, F también lo toma. Es decir, F cubre al menos los mismos minitérminos que G.
- Sean F y G dos funciones booleanas; si F cubre a G se dice que G implica a F, o que G es un implicado de F.
- Sea F una función booleana y L un producto de literales (lazo); se dice que L es un implicado primo de F si y sólo si implica a F, y además ningún subproducto obtenido de L -de menor coste pues tiene menos literales- es un implicado de F. Dicho de otra manera, el conjunto de implicados primos de una función F son todos los posibles lazos que

contiene la función booleana, excepto aquellos lazos que son totalmente cubiertos por otros de mayor tamaño.

Términos relacionados con la expresión mínima

- Una función F es mínima si es irreducible. Es decir, si no existe otra con menos términos (lazos), u otra que teniendo el mismo número de términos tenga menos elementos en al menos uno de los términos.
- La expresión mínima no tiene porque ser única.
- Una función mínima como suma de productos siempre es una suma de implicados primos. Teorema de los implicados primos de Quine.

Acotación de trabajo

- En la tesis trabajaremos siempre con minitérminos para obtener expresiones mínimas como suma de productos -SOP (*Sum of Products*). Fácilmente se puede interpretar desde los maxitérminos y POS (*Products of Sum*).

APÉNDICE B: RELACIÓN DE DOMINANCIA

La aplicación de la relación de dominancia tiene como objetivo reducir el número de implicados primos y/o el de minitérminos, simplificando de esta forma el proceso de minimización.

La relación de dominancia existe entre implicados primos -dominancia entre filas- y entre minitérminos -dominancia entre columnas-, y puede ser aplicada por casi todos los métodos de simplificación. Así, dentro de un método se puede aplicar la relación de dominancia o no.

Para explicar la relación de dominancia utilizaremos unas tablas cuyas filas representan a los implicados primos de la función y las columnas a los minitérminos que quedan por cubrir en la función. Una 'x' en una casilla indica que el implicado primo de esa fila cubre al minitérmino de esa columna.

B.1 Relación de dominancia entre implicados primos

Si un implicado primo es dominado por otro, el primero puede ser eliminado del conjunto de implicados de la función. Plantearemos en este apartado las condiciones necesarias para identificar y eliminar a estos implicados primos. Hecho que supone la simplificación del proceso de minimización de la función booleana, por cuanto que reduce la entrada.

Definición de dominancia simple

Sean L_i y L_j dos implicados primos, y $\{M_i\}$ y $\{M_j\}$ los minitérminos cubiertos por cada implicado primo, respectivamente. Se dice que L_i ***domina*** a L_j si $\{M_j\}$ es un subconjunto de $\{M_i\}$: $\{M_j\} \subseteq \{M_i\}$. Respecto de la tabla de

simplificación se dice que una fila i domina a j si todas las 'x' de la fila j también lo son de la fila i. Si una fila L_i domina a L_j, entonces se puede borrar la fila L_j de la tabla de simplificación sólo si L_i tiene menor o igual costo que L_j. Es decir, si L_i tiene menor o igual número de literales que L_j.

La dominancia es transitiva. Si L_i domina a L_j y L_j domina a L_k; entonces L_i domina a L_k. Por tanto, se pueden borrar L_j y L_k. El razonamiento se puede extender a un mayor número de filas.

Definición de dominancia múltiple

Un conjunto de implicados primos $\{L_i\}$ relacionado con un conjunto de mintérminos $\{M_i\}$ domina a otro conjunto $\{L_j\}$ relacionado con $\{M_j\}$, si $\{M_j\} \subseteq \{M_i\}$. El conjunto $\{L_j\}$ puede ser eliminado sólo si el elemento de menor coste de $\{L_i\}$ es menor o igual que el correspondiente de $\{L_j\}$.

Claramente, la implementación de un algoritmo para establecer la dominancia entre filas supone una mejora por cuanto que reduce el conjunto de implicados primos durante la simplificación. La codificación del algoritmo de dominancia simple es sencilla. Su tiempo de ejecución es elevado porque se deben comparar todos los implicados primos entre sí. Por contra, la dominancia múltiple es costosa en cuanto a codificación y ejecución porque plantea la dominancia de varios con varios.

B.2 Relación de dominancia entre minitérminos

Del mismo que en el apartado anterior eliminábamos determinados implicados primos -filas-, en este apartado plantearemos las condiciones y efectos de la eliminación de determinados minitérminos -columnas.

Definición de dominancia simple

Sean M_i y M_j dos minitérminos, y $\{L_i\}$ y $\{L_j\}$ los conjuntos de lazos que cubren a cada minitérmino, respectivamente. Se dice que M_i ***domina*** a M_j si $\{L_j\}$ es un subconjunto de $\{L_i\}$: $\{L_j\} \subseteq \{L_i\}$. Respecto de la tabla de simplificación se dice que una columna i domina a j si todas las 'x' de la

columna j también lo son de i. El enunciado de la dominancia entre columnas es el recíproco de la dominancia entre filas.

Si un minitérmino M_i domina a M_j, entonces la columna de la tabla de simplificación correspondiente a M_i puede ser borrada.

La dominancia es transitiva. Si M_i domina a M_j, y M_j domina a M_k; entonces M_i domina a M_k. Por tanto, se pueden borrar M_i y M_j. El razonamiento se puede extender a un mayor número de columnas.

La dominancia entre columnas está poco implementada en los algoritmos de simplificación.

Al igual que con la dominancia entre filas, con la dominancia entre columnas reducimos la función a minimizar, haciéndo más sencillo y rápido el proceso de simplificación.

APÉNDICE C: NOMBRES DE MATRICES Y VARIABLES DE MCVK

En este apéndice presentamos ordenados alfabéticamente los distintos nombres de las variables y matrices utilizadas en la tesis acompañados de una breve definición.

BA *Borrar Abajo*
Matriz que contiene las posiciones de los posibles implicados primos que deben ser *borrados hacia abajo* por cada posible implicado primo convertido en tal.

BD *Borrar Dentro*
Matriz que contiene las posiciones de los posibles implicados primos que deben ser *borrados dentro de la rueda* por cada posible implicado primo convertido en tal.

CS *Casillas*
Matriz que contiene las casillas con las que se relaciona cada uno de los posibles implicados primos.

DE *Desplazamientos a Eliminar*
Matriz que contiene los desplazamientos a eliminar para obtener DR.

DR *Desplazamiento Relativo*
Matriz que contiene el desplazamiento a sumar dentro de una rueda para encontrar el posible implicado primo adyacente.

ID *Identificador*
Matriz que contiene los identificadores de todos los posibles implicados primos.

IR *Identificador de Rueda*
Matriz que contiene los identificadores de todas las ruedas.

LA *Lazos Acumulados*
Es el vector NL acumulado por posiciones.

LT *Lazos Totales*
Variable que indica el número total de posibles implicados primos relacionado con un número de variables.

NL *Número de Lazos*
Vector que indica cuántos posibles implicados primos hay de cada tipo.

NLB *Número de Lazos Borrados*
Matriz que indica cuántos posibles implicados primos de un determinado tipo son borrados por aquel que se convierte en implicado primo.

NR *Número de Ruedas*
Vector que indica cuántas ruedas hay de cada tipo.

PRA *Posición de las Ruedas Abajo*
Matriz que contiene las direcciones de las ruedas *cubiertas* por cada rueda.

RA *Ruedas Acumuladas*
Es el vector NR acumulado por posiciones.

RSML *Ruedas Seguidas con la Misma Letra*
Matriz que indica cuántas ruedas seguidas comienzan por la misma primera letra.

RT *Ruedas Totales*
Variable que indica el número total de ruedas relacionado con un número de variables.

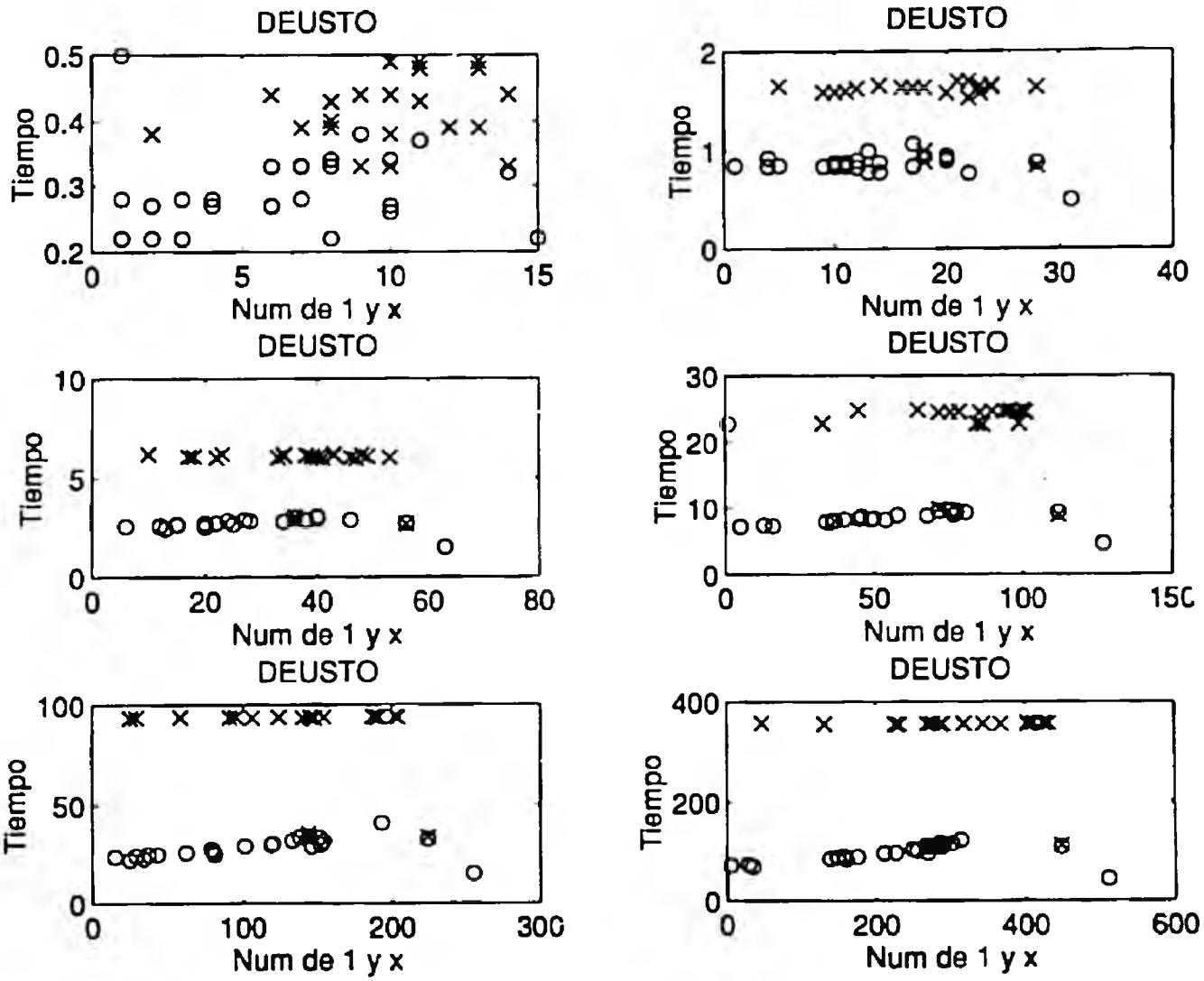
TLB *Total de Lazos a Borrar*
Vector que indica cuántos posibles implicados primos en total son borrados por aquel qeu se convierte en implicado primo.

TR *Tipo de una Rueda*
Vector que contiene el tipo de una rueda.

APÉNDICE D: GRÁFICAS COMPARATIVAS DE LA OBTENCIÓN DE IMPLICADOS PRIMOS

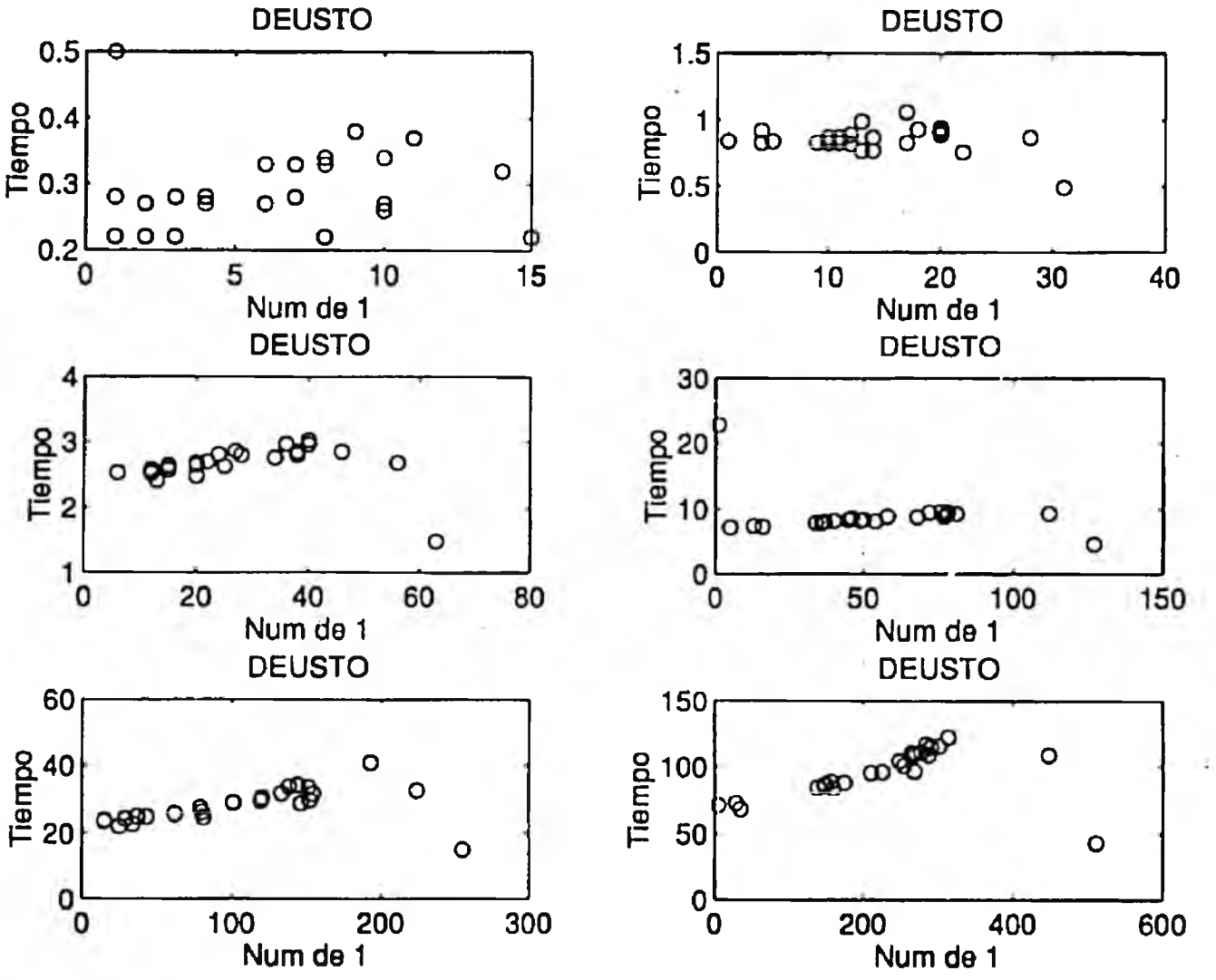
Las siguientes figuras permiten comparar los métodos CAMP DEUSTO y Quine-McCLuskey en cuanto a la rapidez de obtención de los implicados primos de una función booleana.

Las figuras D.1 a D.6 -de arriba abajo y de izquierda a derecha- representan los tiempos de obtención de los implicados primos según el nuevo método incluido en CAMP DEUSTO. La figura D.1 se refiere a funciones booleanas de cuatro variables, la D.2 a funciones de cinco variables, etc. Los 'o' representan los tiempos para funciones booleanas con sólo minitérminos, mientras que las 'x' se corresponden con funciones con condiciones libres. Esta distinción nos permite observar la dependencia del tiempo respecto de la existencia o no de condiciones libres en la función a simplificar.



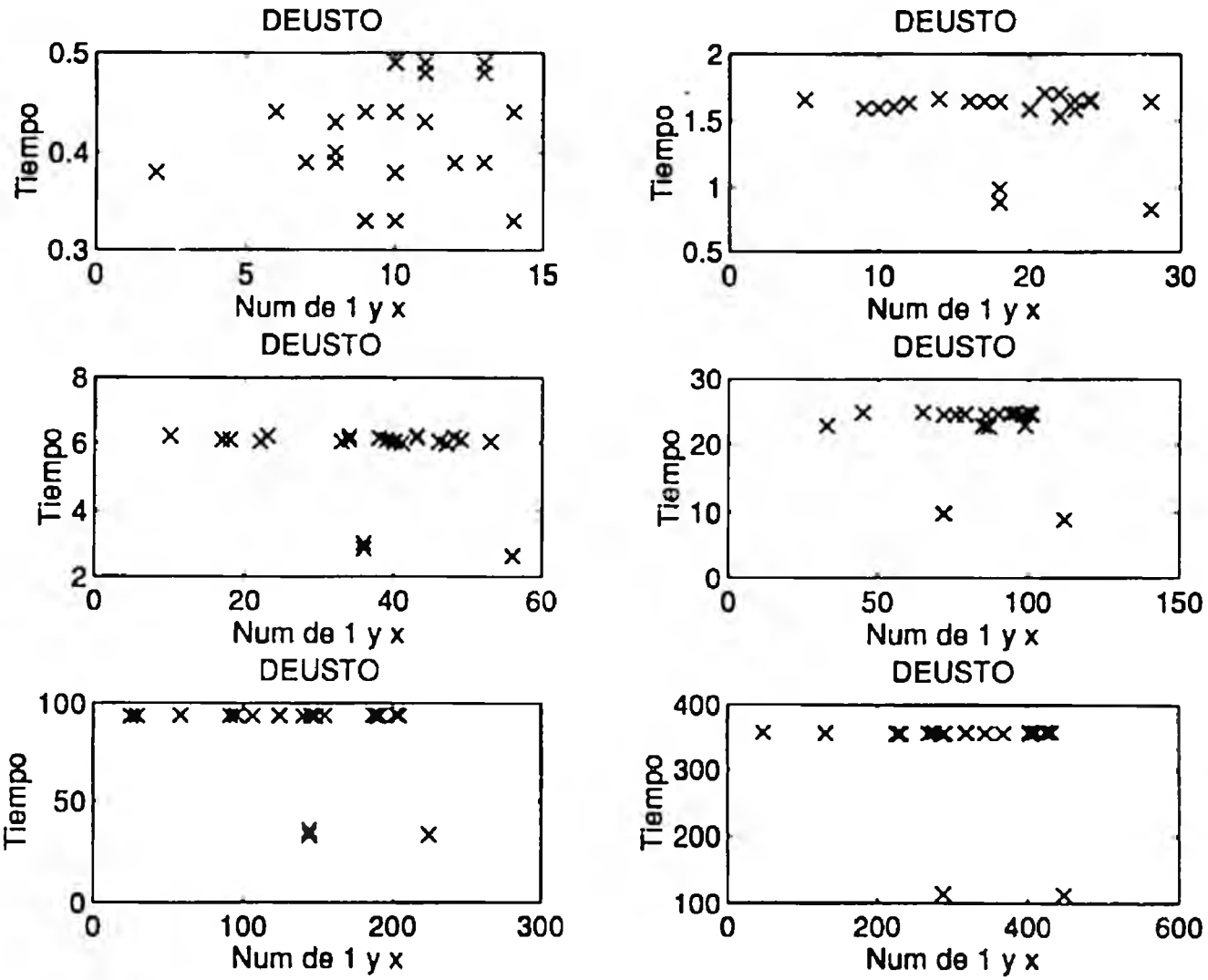
Figuras D.1 a D.6. Método CAMP DEUSTO: Tiempos de obtención de los implicados primos de una función según su número de variables

Las figuras D.7 a D.12 -de arriba abajo y de izquierda a derecha- representan los tiempos de obtención mediante CAMP DEUSTO de los implicados primos de una función sin condiciones libres, es decir son los 'o' de las figuras D.1 a D.6 aislados. Cada gráfica se corresponde con un número de variables, de cuatro a nueve.



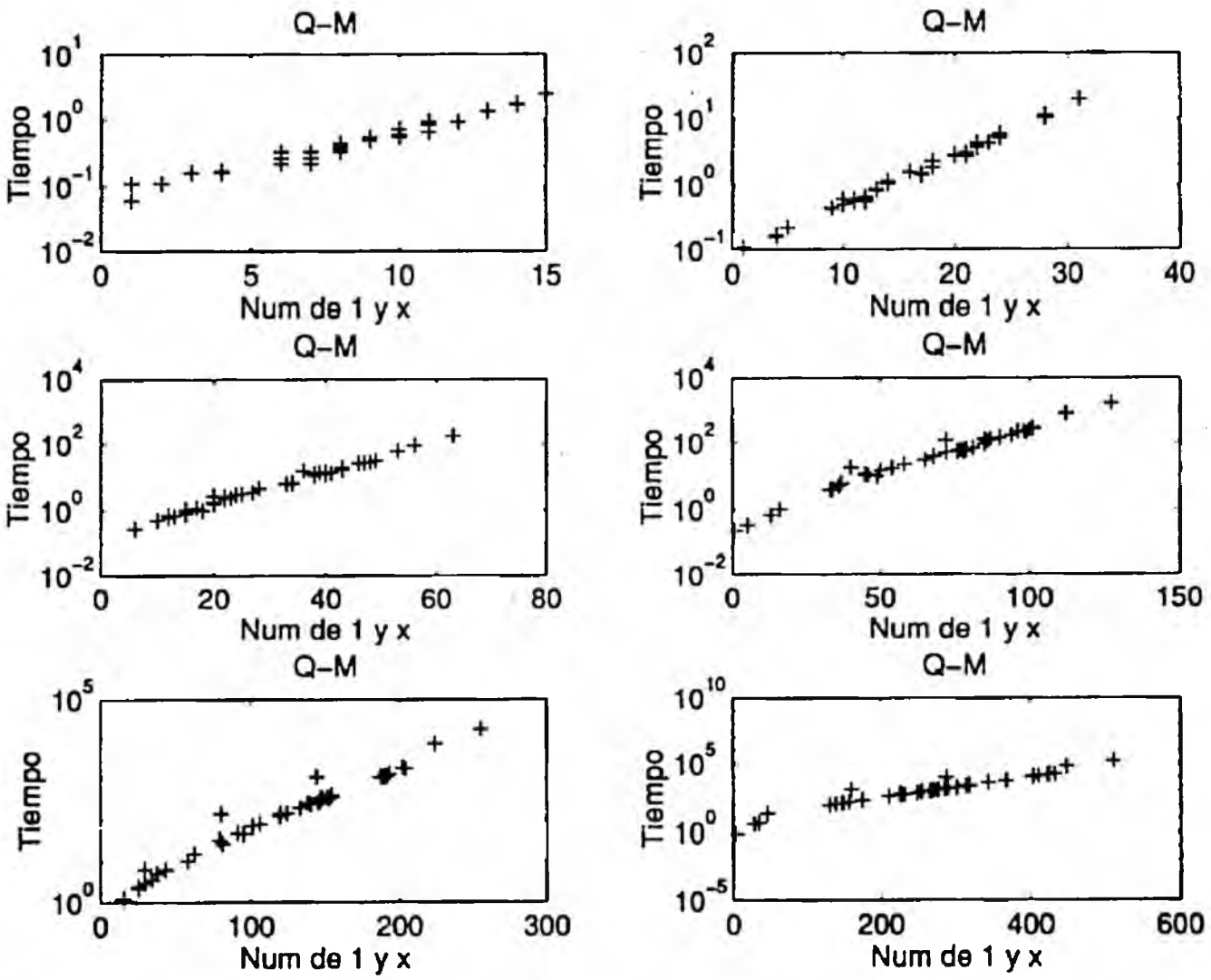
Figuras D.7 a D.12. Método CAMP DEUSTO: Tiempos de obtención de los implicados primos de una función sin condiciones libres según su número de variables

Las figuras D.13 a D.18 -de arriba abajo y de izquierda a derecha- representan los tiempos de obtención mediante CAMP DEUSTO de los implicados primos de una funación con condiciones libres, es decir, son las 'x' de las figuras D.1 a D.6 aisladas. Cada gráfica se corresponde con un número de variables, de cuatro a nueve. Respecto de las gráficas D.7 a D.12 cabe destacar el cambio de valores en las escalas de tiempo.



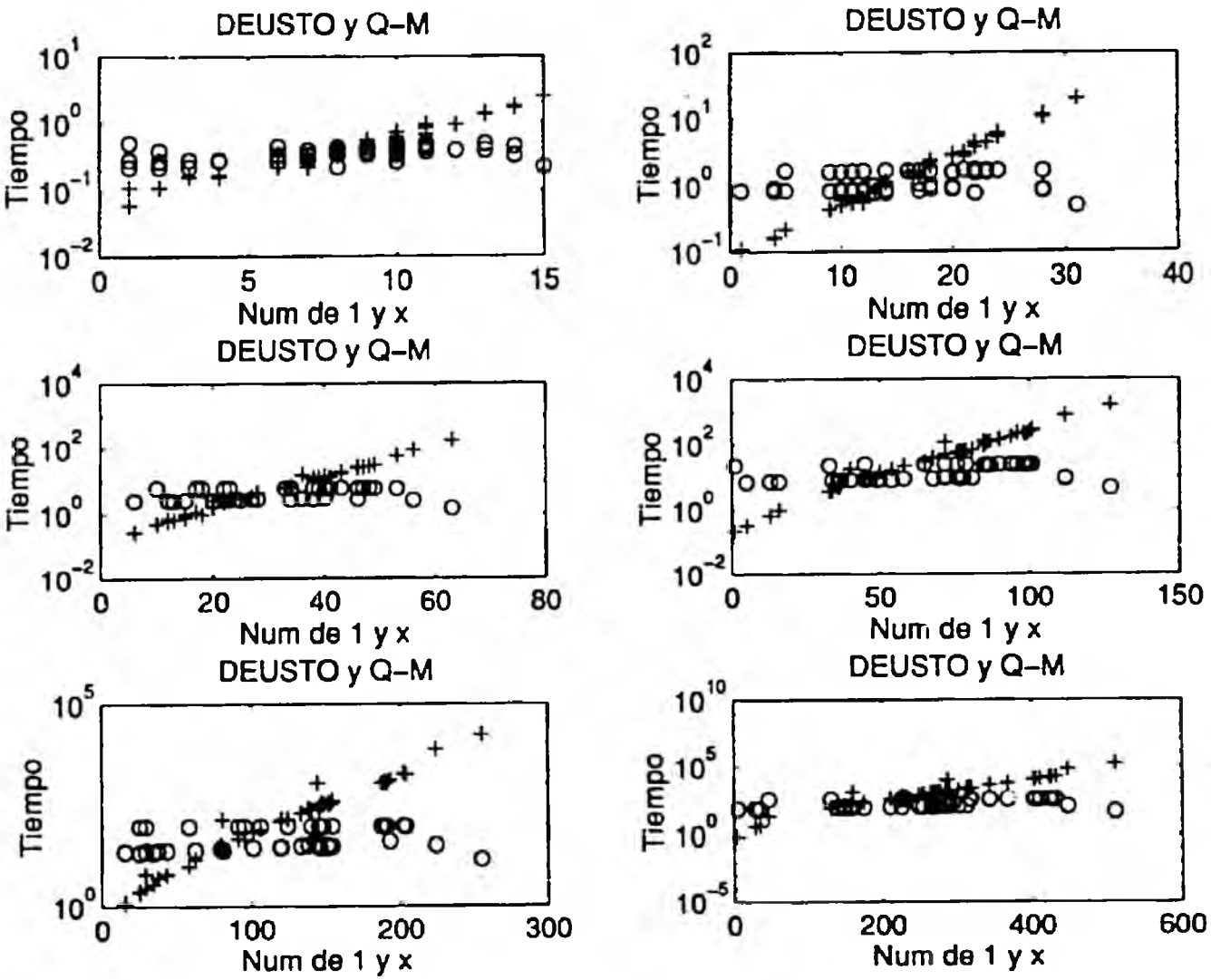
Figuras D.13 a D.18. Método CAMP DEUSTO: Tiempos de obtención de los implicados primos de una función con condiciones libres según su número de variables

Las figuras D.19 a D.24 -de arriba abajo y de izquierda a derecha- representan los tiempos de obtención de los implicados primos de una función booleana según el método de Quine-McCluskey. Cada gráfica se corresponde con un número de variables. En este caso no tiene sentido distinguir las funciones con condiciones libres y sin ellas. La escala de tiempo es necesariamente logarítmica, y en algunos casos -fig. D.23 y D.24- su rango es muy elevado.



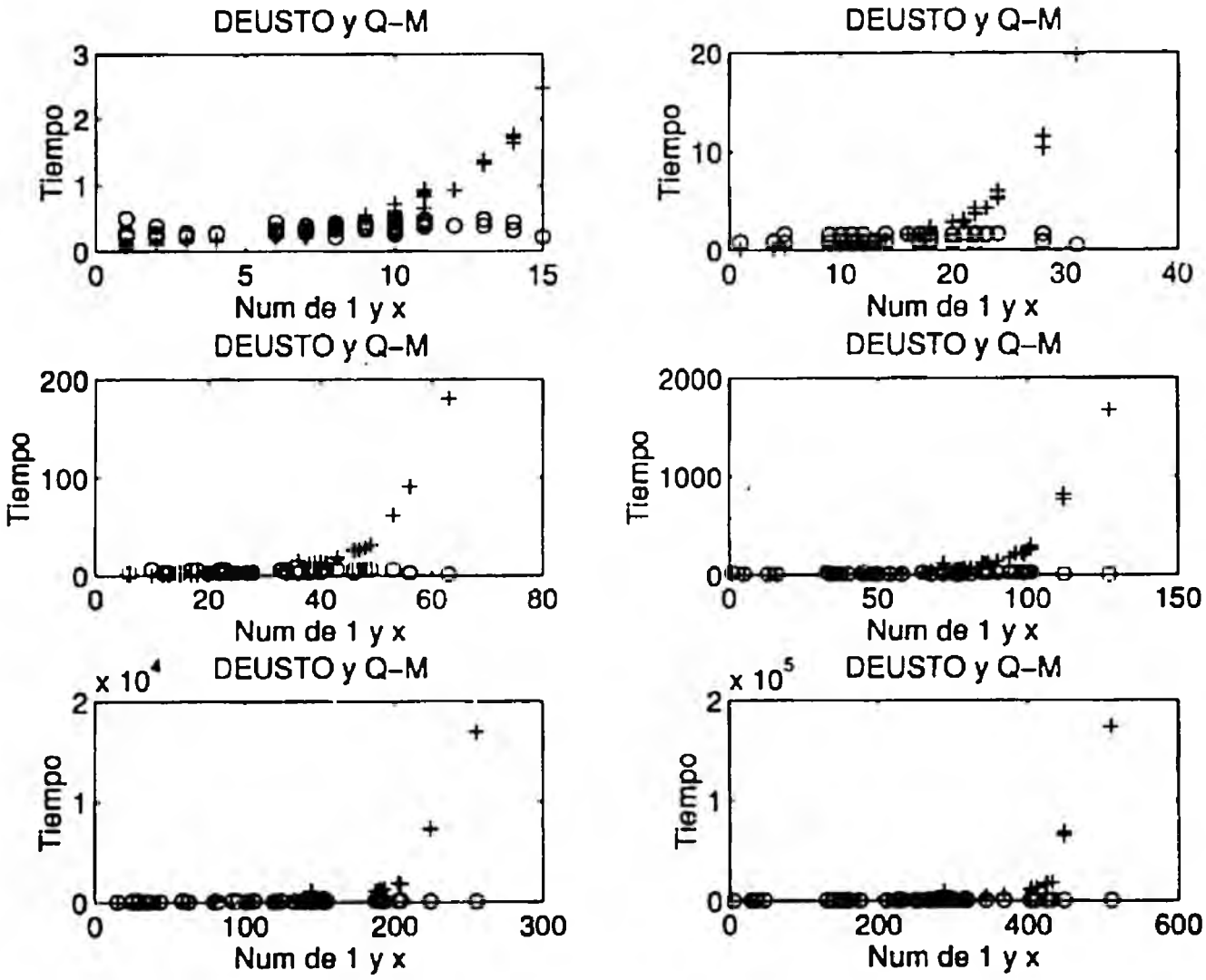
Figuras D.19 a D.24. Método Q-M: Tiempos de obtención de los implicados primos de una función según su número de variables

Las figuras D.25 a D.30 -de arriba abajo y de izquierda a derecha- comparan los tiempos de obtención de los implicados primos de una función según los métodos de CAMP DEUSTO 'o' y de Quine-McCluskey '+'. Cada gráfica se corresponde con un número de variables, de cuatro a nueve. La escala de tiempo es logarítmica.



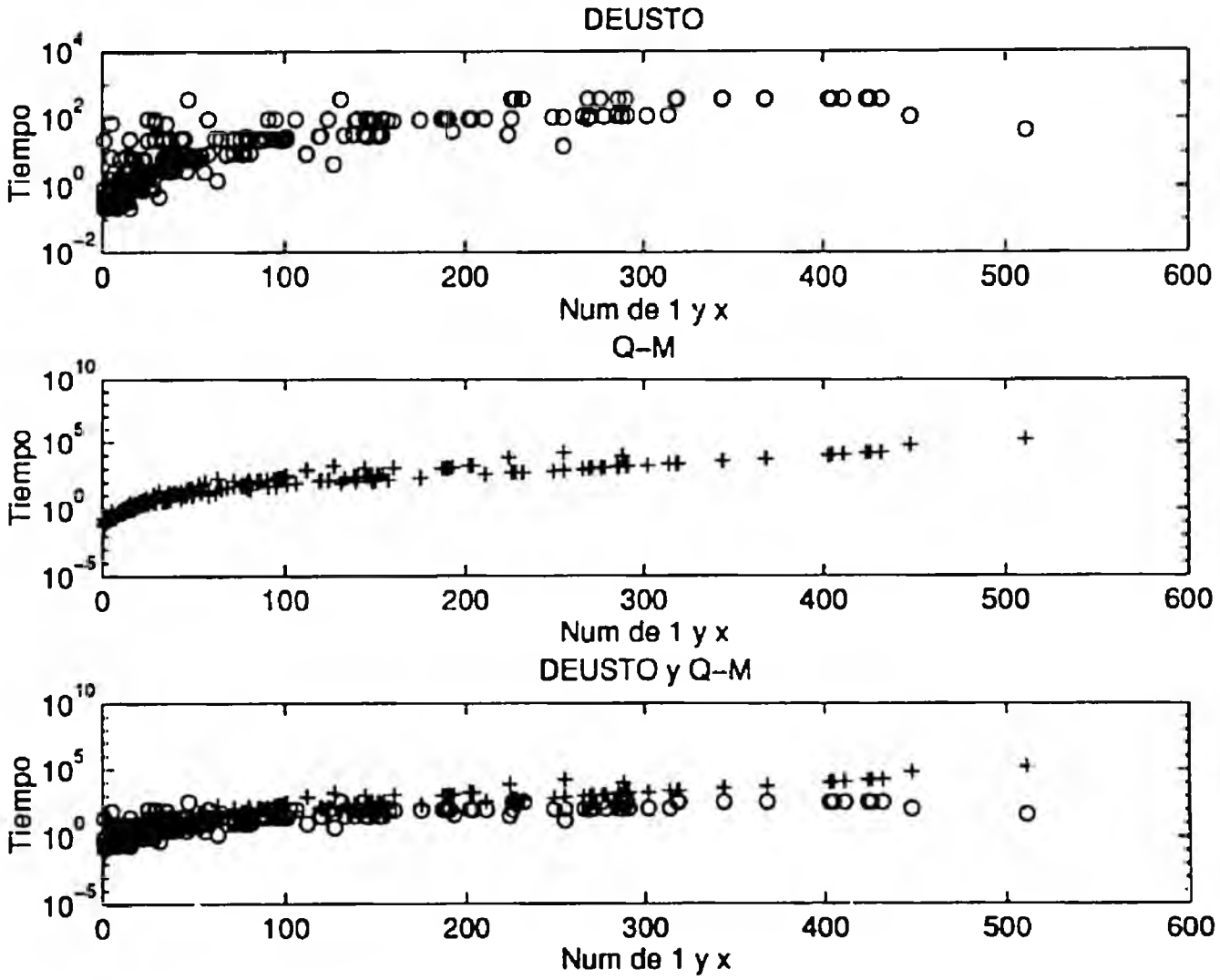
Figuras D.25 a D.30. Comparación CAMP DEUSTO vs Q-M según el número de variables de una función

Las figuras D.31 a D.36 -de arriba abajo y de izquierda a derecha- vuelven a comparar los métodos CAMP DEUSTO y Quine-McCluskey, pero en este caso las gráficas se expresan con escalas lineales de tiempo.



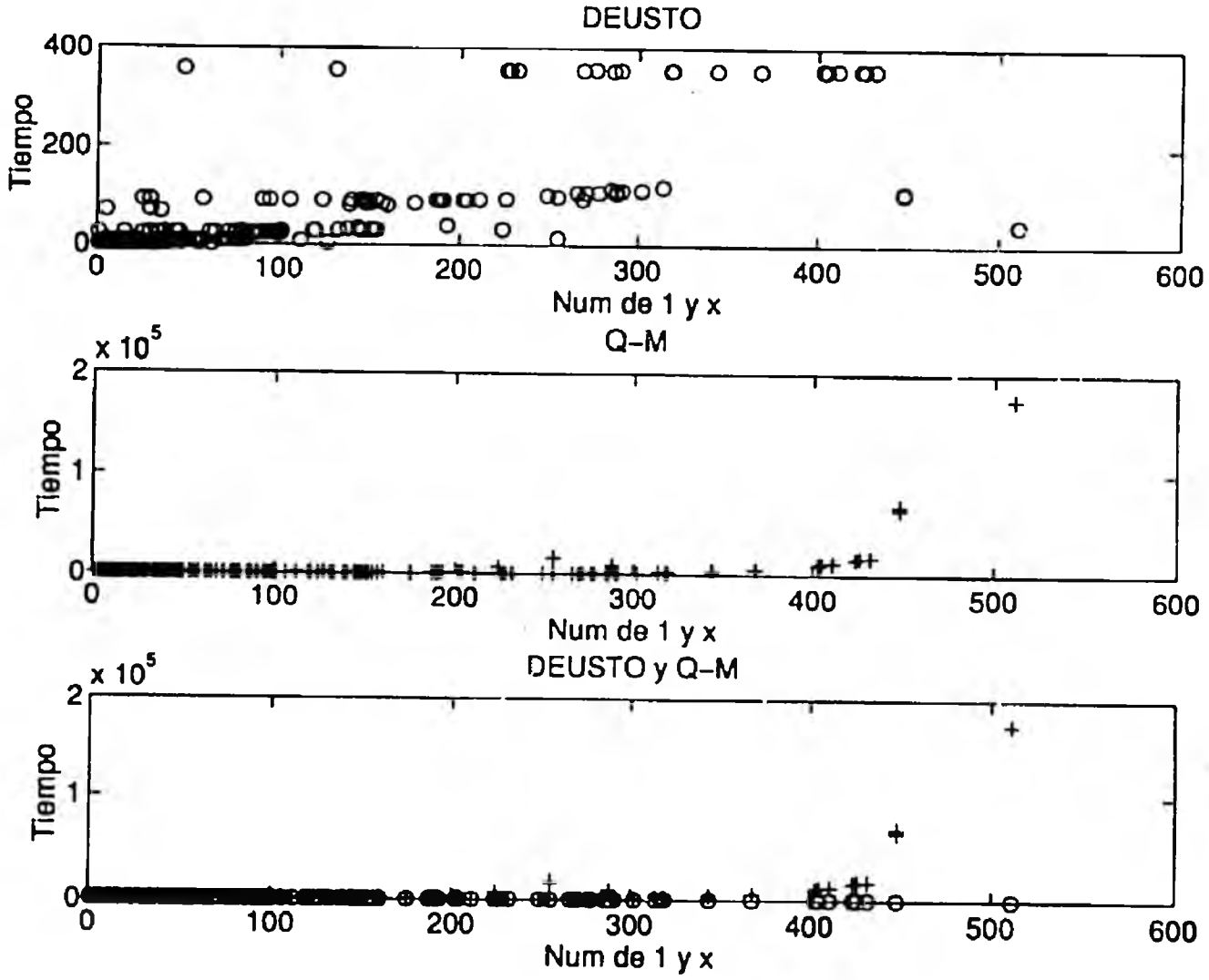
Figuras D.31 a D.36. Comparación CAMP DEUSTO vs Q-M según el número de variables de la función

Las figuras D.37 a D.39 -de arriba abajo y de izquierda a derecha- representan los tiempos de obtención de los implicados primos de una función según los métodos de CAMP DEUSTO y Quine-McCluskey, juntos y por separado. En este caso las gráficas no distinguen entre número de variables, sólo respecto del número de minitérminos y condiciones libres. La escala de tiempo es logarítmica, y de muy distinto rango según cada gráfica.



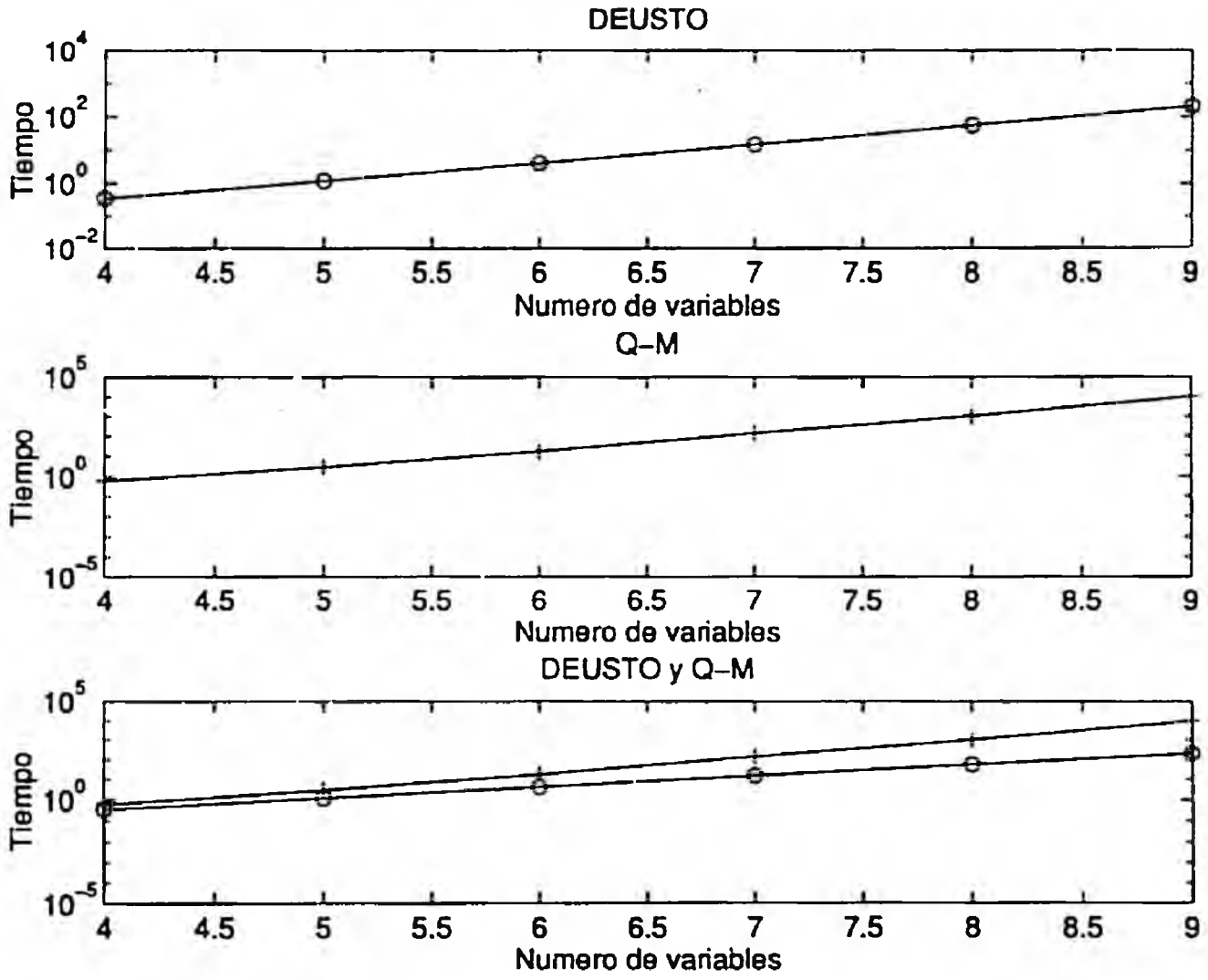
Figuras D.37 a D.39. Comparación CAMP DEUSTO vs Q-M según el número de minitérminos y condiciones libres de la función, sin tener en cuenta el número de variables

Las figuras D.40 a D.42 -de arriba abajo y de izquierda a derecha- son idénticas a las gráficas D.37 a D.39, pero las escalas de tiempo son lineales, y de muy distinto rango según cada gráfica.



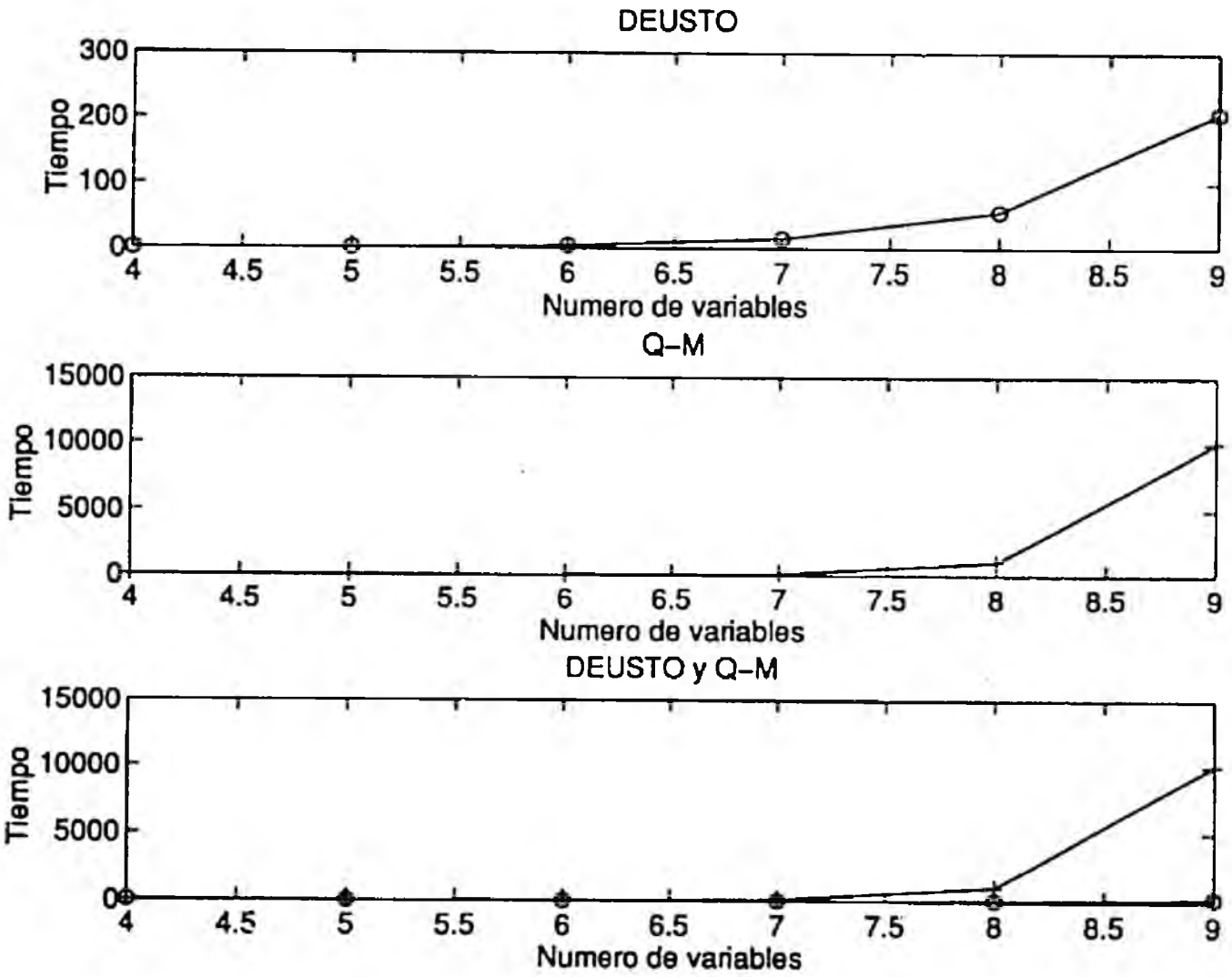
Figuras D.40 a D.42. Comparación CAMP DEUSTO vs Q-M según el número de minitérminos y condiciones libres de la función, sin tener en cuenta el número de variables

Las figuras D.43 a D.45 -de arriba abajo y de izquierda a derecha- representan los tiempos medios de obtención de los implicados primos de una función según su número de variables respecto de los métodos CAMP DEUSTO y Quine-McCluskey, tanto juntos como por separado. La escala de tiempo es logarítmica, y de muy distinto rango según cada gráfica. El trazo continuo ayuda a observar la tendencia de los métodos.



Figuras D.43 a D.45. Comparación CAMP DEUSTO vs Q-M según los tiempos medios y el número de variables de la función

Las figuras D.46 a D.48 -de arriba abajo y de izquierda a derecha- son idénticas a las figuras D.43 a D.45, pero las escalas de tiempo son lineales y de muy distinto rango entre sí. Las rectas ayudan a observar la tendencia de los métodos.

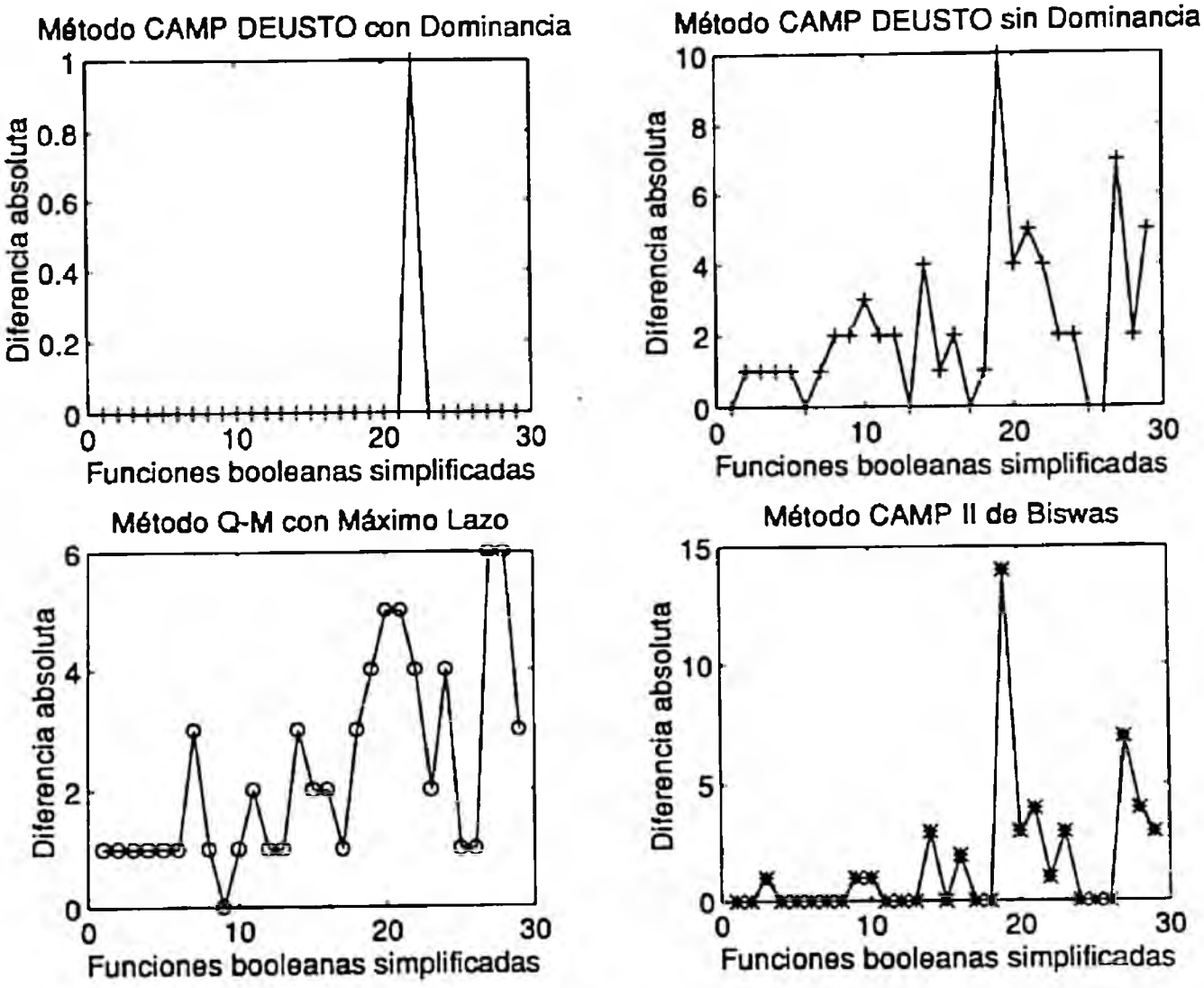


Figuras D.46 a D.48. Comparación CAMP DEUSTO vs Q-M según los tiempos medios y el número de variables de la función

APÉNDICE E: GRÁFICAS COMPARATIVAS DE LA OBTENCIÓN DE LA EXPRESIÓN MÍNIMA

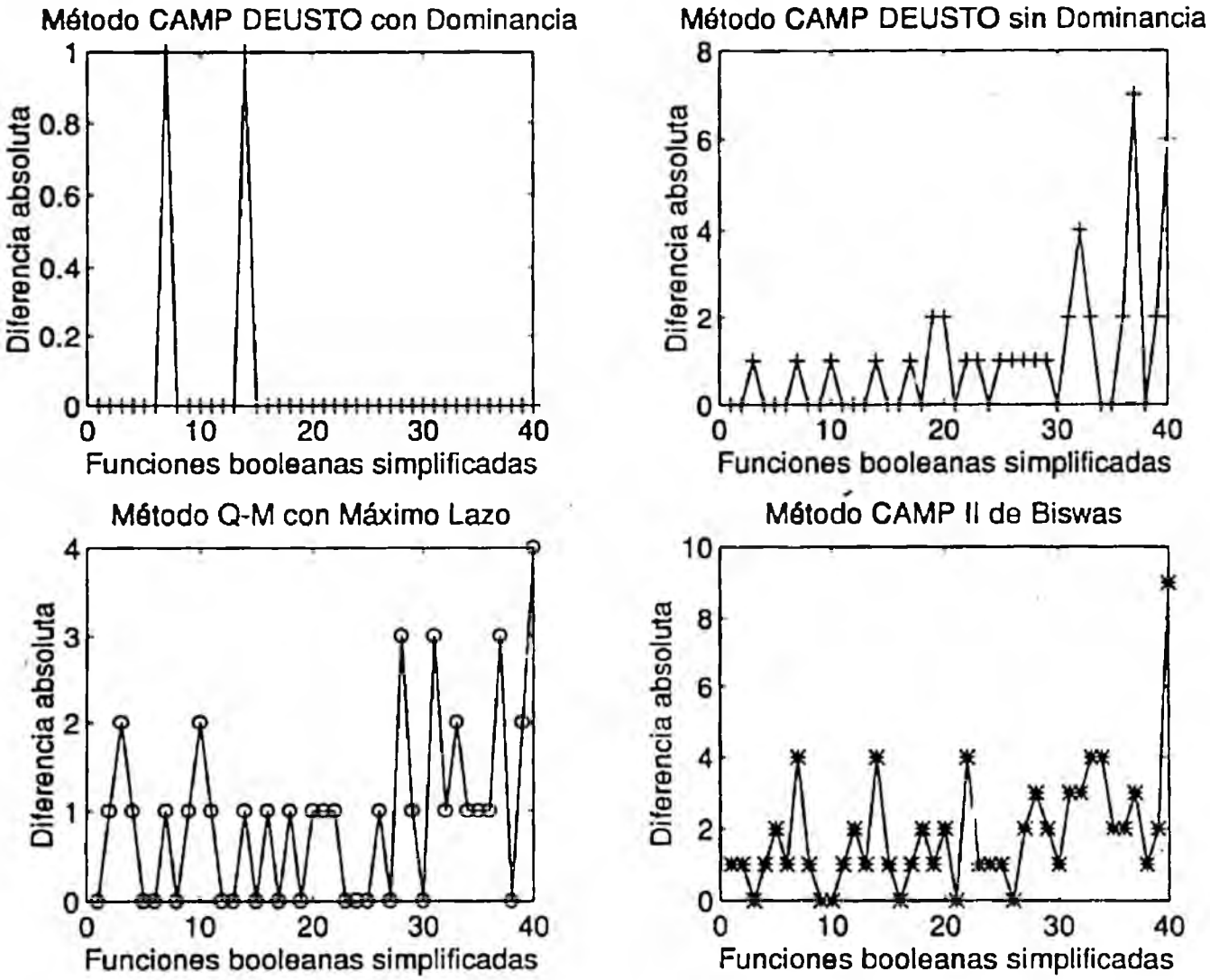
Las siguientes figuras permiten comparar la exactitud de las expresiones mínimas obtenidas por los distintos métodos: CAMP DEUSTO, CAMP DEUSTO con Dominancia, Quine-McCluskey con Máximo Lazo y CAMP II de Biswas.

Las figuras de E.1 a E.4 -de arriba abajo y de izquierda a derecha- representan la optimidad de los distintos métodos: CAMP DEUSTO con Dominancia, CAMP DEUSTO sin Dominancia, Quine-McCluskey con Máximo Lazo y CAMP II de Biswas, respectivamente. La optimidad se mide como la ***diferencia absoluta*** entre el número de términos obtenidos por cada método y el número exacto obtenido por Quine-McCluskey.



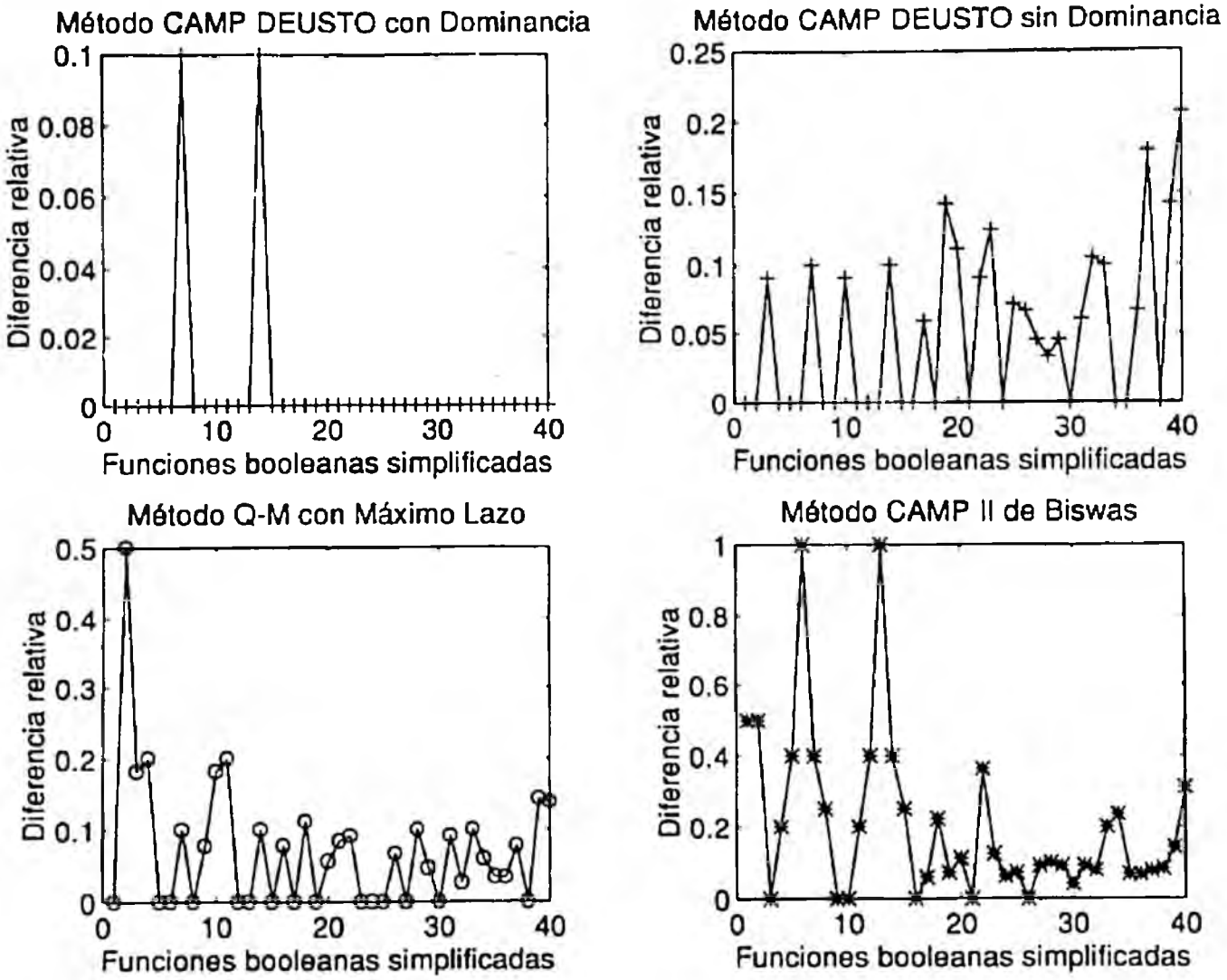
Figuras E.1 a E.4. Estudio de la optimidad de la expresión obtenida según los distintos métodos de simplificación para funciones sin condiciones libres

Las figuras de E.9 a E.12 -de arriba abajo y de izquierda a derecha- representan la optimidad de los distintos métodos: CAMP DEUSTO con Dominancia, CAMP DEUSTO sin Dominancia, Quine-McCluskey con Máximo Lazo y CAMP II de Biswas, respectivamente. La optimidad se mide como la ***diferencia absoluta*** entre el número de términos obtenidos por cada método y el número exacto obtenido por Quine-McCluskey.



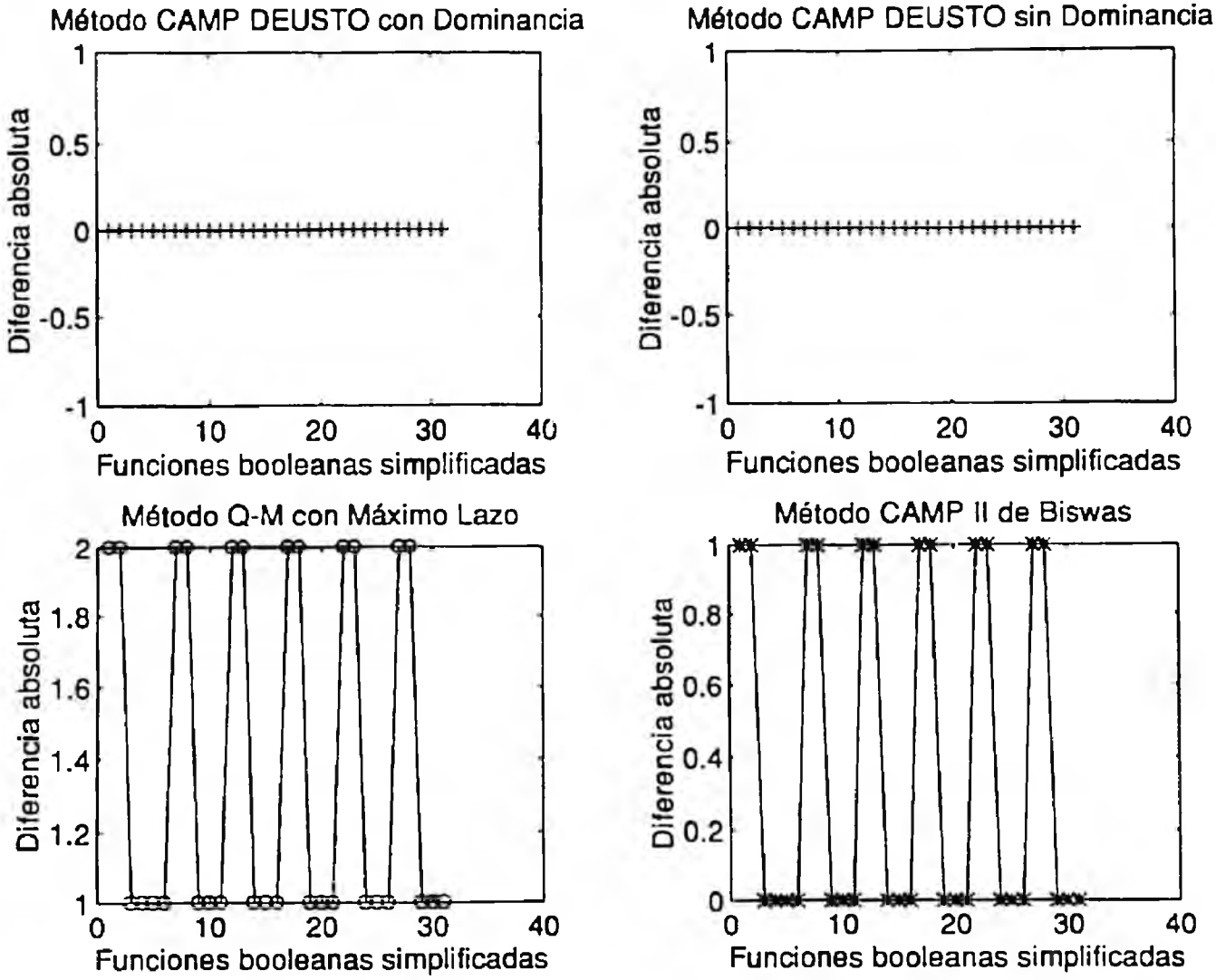
Figuras E.9 a E.12. Estudio de la optimidad de la expresión obtenida según los distintos métodos de simplificación para funciones con condiciones libres

Las figuras de E.13 a E.16 -de arriba abajo y de izquierda a derecha- representan la optimidad de los distintos métodos: CAMP DEUSTO con Dominancia, CAMP DEUSTO sin Dominancia, Quine-McCluskey con Máximo Lazo y CAMP II de Biswas, respectivamente. La optimidad se mide como la ***diferencia relativa*** entre el número de términos obtenidos por cada método y el número exacto obtenido por Quine-McCluskey.



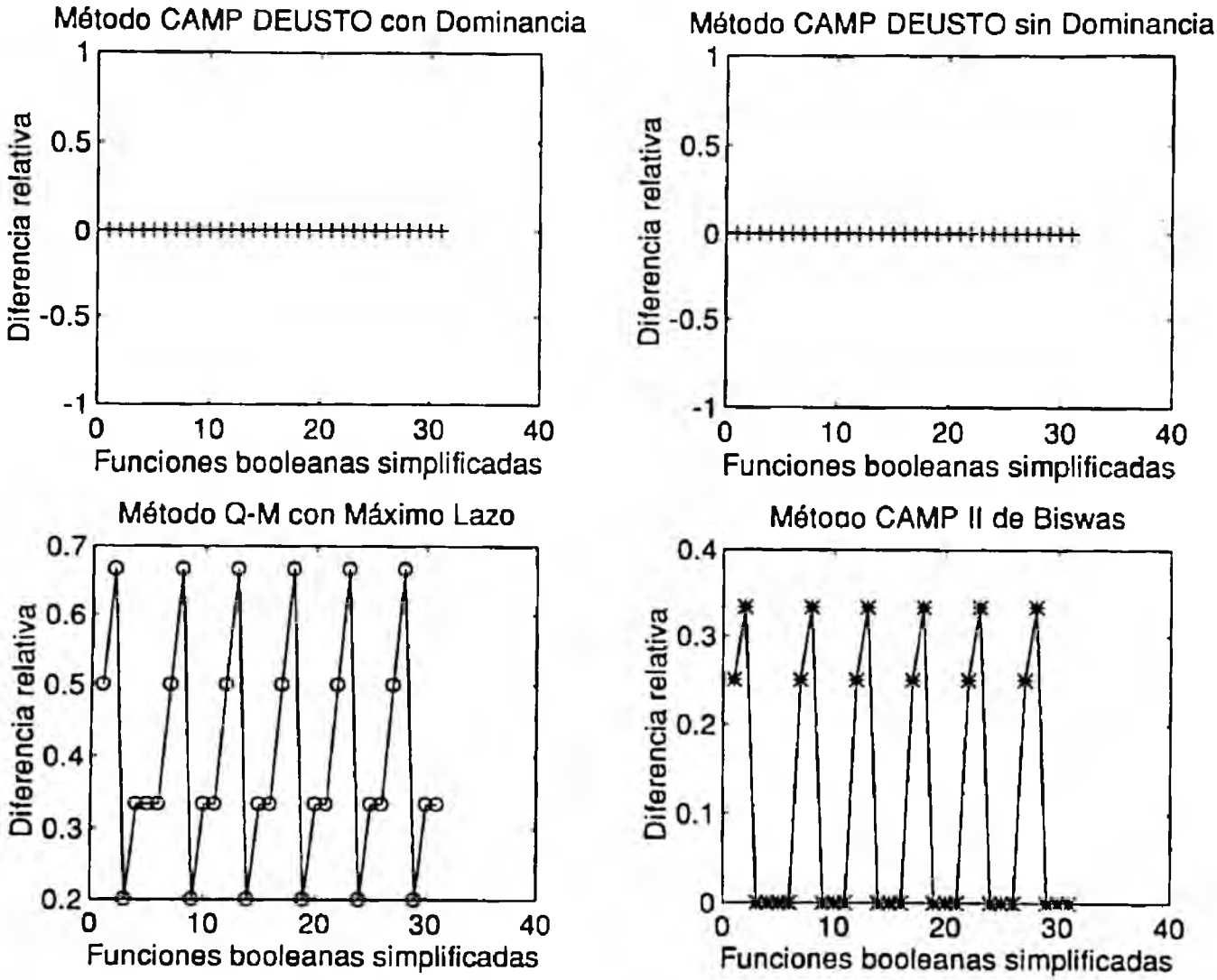
Figuras E.13 a E.16. Estudio de la optimidad de la expresión obtenida según los distintos métodos de simplificación para funciones con condiciones libres

Las figuras de E.17 a E.20 -de arriba abajo y de izquierda a derecha- representan la optimidad de los distintos métodos: CAMP DEUSTO con Dominancia, CAMP DEUSTO sin Dominancia, Quine-McCluskey con Máximo Lazo y CAMP II de Biswas, respectivamente. La optimidad se mide como la ***diferencia absoluta*** entre el número de términos obtenidos por cada método y el número exacto obtenido por Quine-McCluskey.



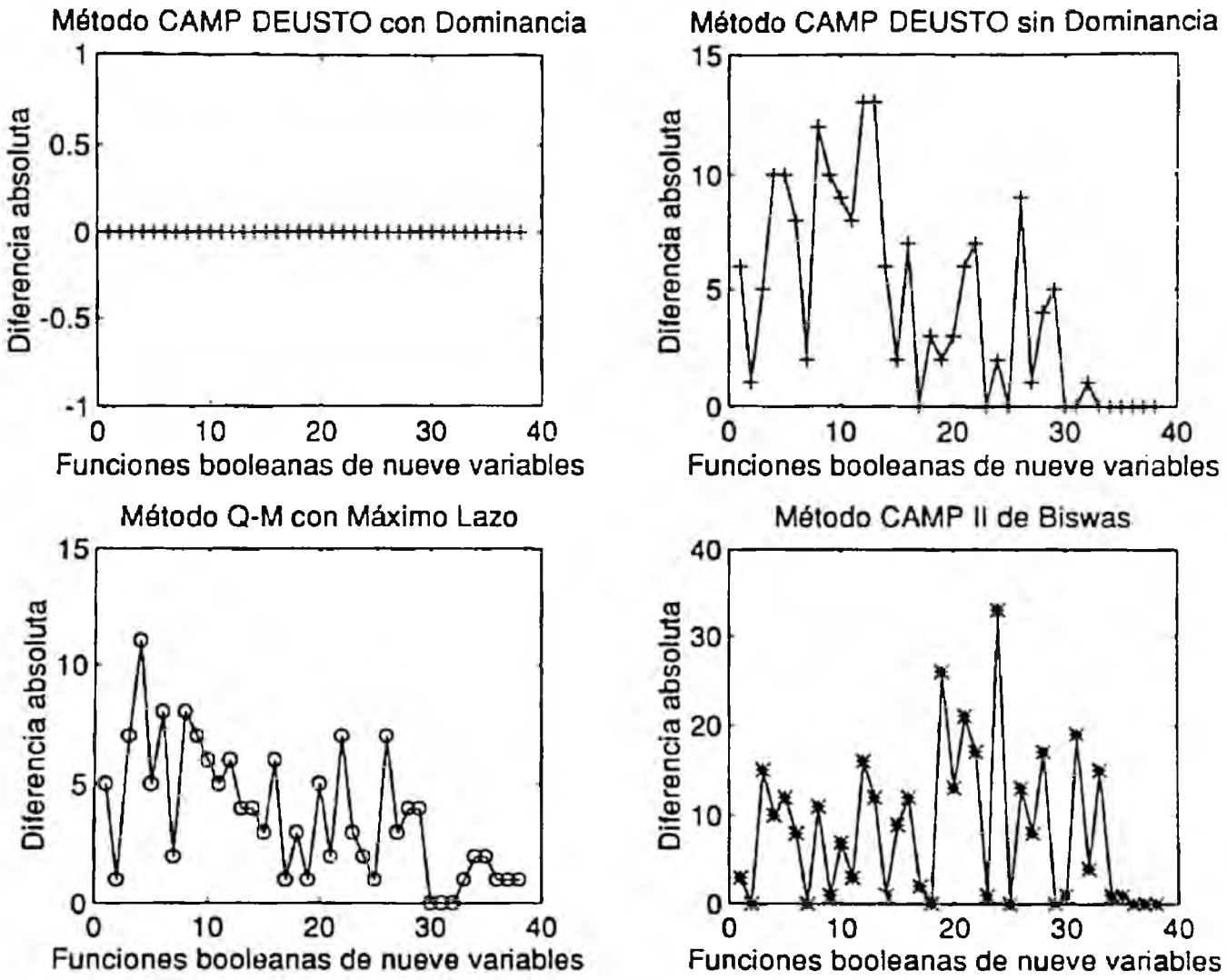
Figuras E.17 a E.20. Estudio de la optimidad de la expresión obtenida según los distintos métodos de simplificación para funciones cíclicas totales

Las figuras de E.21 a E.24 -de arriba abajo y de izquierda a derecha- representan la optimidad de los distintos métodos: CAMP DEUSTO con Dominancia, CAMP DEUSTO sin Dominancia, Quine-McCluskey con Máximo Lazo y CAMP II de Biswas, respectivamente. La optimidad se mide como la ***diferencia relativa*** entre el número de términos obtenidos por cada método y el número exacto obtenido por Quine-McCluskey.



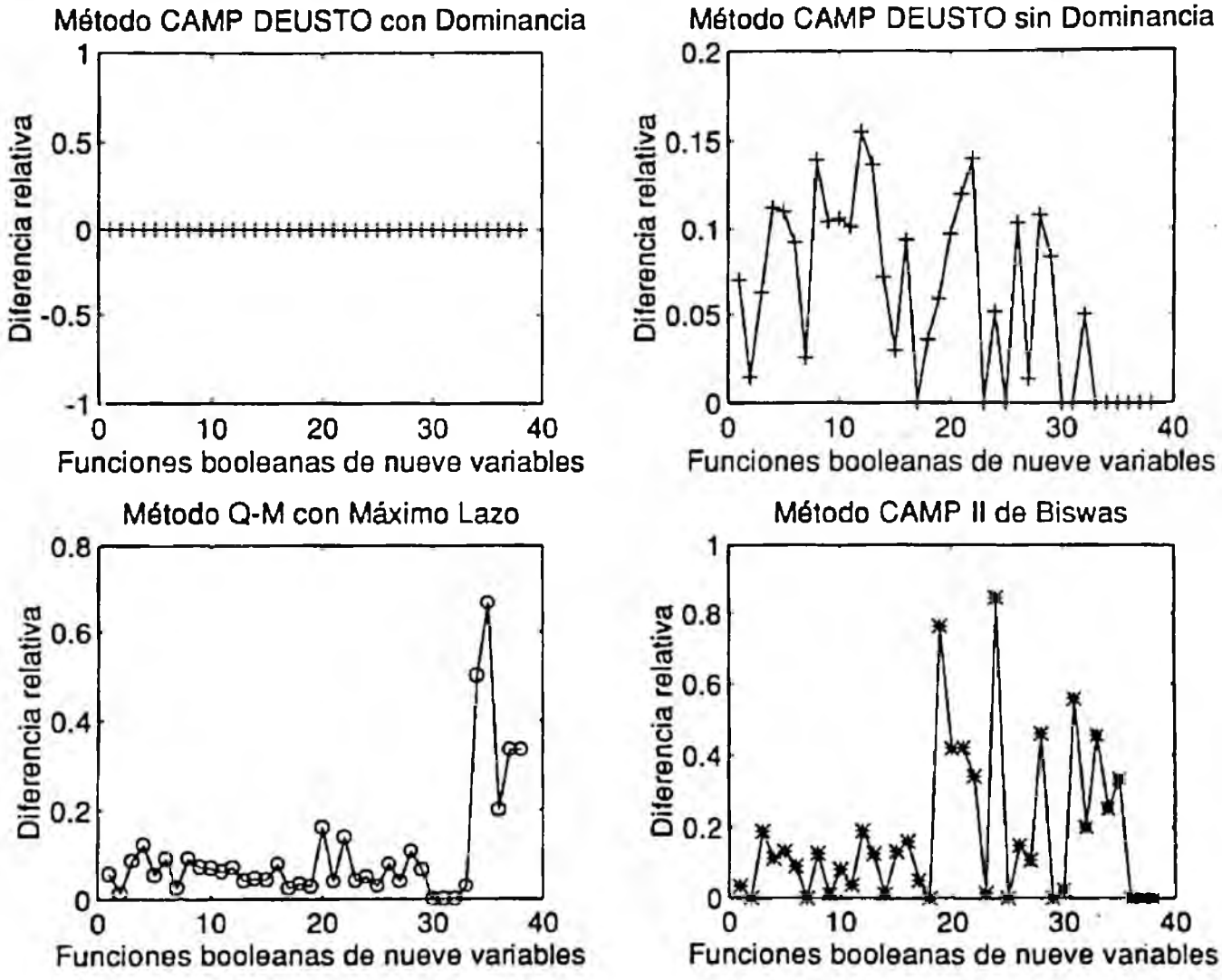
Figuras E.21 a E.24. Estudio de la optimidad de la expresión obtenida según los distintos métodos de simplificación para funciones cíclicas totales

Las figuras E.25 a E.28 miden la optimidad como *diferencia absoluta* para nueve variables. En este caso el proceso Q-M conlleva un gran número de recursividades, que consumiendo la memoria no obtienen la expresión mínima. Es decir, no existe el referente exacto de Q-M para medir la optimidad, siendo sustituido por el mejor método para cada función simplificada.



Figuras E.25 a E.28. Estudio de la optimidad de la expresión obtenida según los distintos métodos de simplificación para funciones de nueve variables

Las figuras E.29 a E.32 miden la optimidad como ***diferencia relativa*** para nueve variables. En este caso el proceso Q-M conlleva un gran número de recursividades, que consumiendo la memoria no obtienen la expresión mínima. Es decir, no existe el referente exacto de Q-M para medir la optimidad, siendo sustituido por el mejor método para cada función simplificada.



Figuras E.29 a E.32. Estudio de la optimidad de la expresión obtenida según los distintos métodos de simplificación para funciones de nueve variables

BIBLIOGRAFÍA

La bibliografía principal utilizada en la tesis se presenta clasificada de dos formas: alfabéticamente y por temas.

RELACIÓN ALFABÉTICA

Althoen, S. C. y Bumcrot, R. J. 1988. *Introduction to the discret mathematics.* PWS-Kent, Boston, pp: 99-108.

Arevalo, Z.; Bredeson, J. G. 1978. «A method to simplify a boolean function into a near minimal sum-of-products for programmable logic arrays». *IEEE Trans. on Comp.* Vol. C-27, Nº 11, pp: 1028-1039.

Bartholomeus, M; Man, H. 1985. PRESTOL-II: «Yet another logic minimizer for programmed logic arrays». *Proc. 1985 Int. Symp. Circuits and Systems*, Kyoto.

Besslich, P. W. 1986. «A heuristic minimization of MVL functions: a direct cover approach». *IEEE Trans. on Comp.* Vol. C-35, Nº 2, Feb. 1986, pp: 134-144.

Besslich, P. W.; Pichlbauer, P. «Fast transform procedure for the generation of near minimal covers of boolean functions». *IEEE Proc.* Vol. 128, Pt. E Nº 6, pp: 250-254.

Biswas, N. N. 1971. «Minimization of boolean functions». *IEEE Trans. on Comp.*, Vol. C-20, Nº 8, Aug. 1971, pp: 925-929.

Biswas, N. N. 1984. «Computer aided minimization procedure for boolean functions». *Proc. 21st Design Automation Conference*, Alburquerque, N.Mex., June 1984, pp: 699-702.

Biswas, N. N. 1986. «Computer aided minimization procedure for boolean functions». *IEEE Trans Computer-Aided Design Integrated Circuits Systems*, Vol. CAD-5, N° 2, April 1986, pp: 303-304.

Biswas, N. N. 1990. «On covering distant minterms by the CAMP algorithm». *IEEE Trans. Computer-Aided design of integrated circuits systems*, Vol. CAD-9, N° 7, July 1990, pp: 786-789.

Biswas, N. N. 1993. *Logic design theory*. Prentice Hall, pp: 63-91.

Brayton, R. K.; Cohen, J. D.; Hachtel; Trager, B. M.; Yun, D. Y. Y. 1982. «Fast recursive boolean function manipulation» *Proc. 1982 Int. Symp. on Circ. and Syst.*, pp: 58-62.

Brayton, R. K.; Hachtel, G. D.;McMullen, C. T.; Sangiovanni-Vincentelli; A.L. 1984. *Logic minimization algorithms for VLSI synthesis*. Kluwer academic publishers.

Brayton, R.; Hachtel, G. D., Hemachandra, A. R.; Newton, A. R.; Sangiovanni-Vincentelli, A. L. 1982. «A comparision of logic minimization strategies using ESPRESSO. An APL program package for partitioned logic minimization». *Proc. Int. Symp. on Circ. and Syst.* pp: 43-49, Roma.

Brayton, R.; McMullen, C. 1982. «The descomposition and factorization of boolean expressions». *Proc. Int. Symp. on Circ. and Syst.* pp: 49-54, Roma.

Breeding, K. J. 1992. *Digital design fundamentals*. Prentice Hall, segunda edición, pp: 58-88.

Bryant, R. E. 1986. «Graph-Based algorithms for boolean function manipulation». *IEEE Trans. on Comp.* Vol. C-35, N° 8, Aug. 1986, pp: 677-691.

Cahill, S. J. 1993. *Digital and microprocessor engineering*. Ellis Horwood, segunda edición, pp: 81-115.

Chakravarty, S. 1993. «A characterization of binary decision diagrams». *IEEE Trans. on Comp.* Vol. 42, N° 2, Feb. 1993, pp: 129-137.

Chan, A. H. 1987. «Using decision trees to derive the complement of a binary function with multiple-valued inputs». *IEEE Trans. on Comp.* Vol. C-36, Nº 2, Feb. 1987, pp: 212-214.

Dagenais, M. R.; Agarwal, V. K.; Rumin, N. C. 1985. «The McBoole logic minimizer». *Proc. 22nd Design Automation Conf.*, Las Vegas, página: 667.

Dagenais, M. R.; Agarwal, V. K.; Rumin, N. C. 1986. «McBoole: A new procedure for exact logic minimization». *IEEE Trans. CAD of Int. Circ. Syst.*, Vol CAD-5, Nº 1, pp: 229-238.

Dietmeyer, D. L. 1978. *Logic design of digital systems*. Allyn and Bacon Inc, segunda edición, Boston, pp: 609-665.

Dormido, S.; Mira, J.; Delgado, A. E.; Canto, M. A. 1994. *Electrónica digital*. Sanz y Torres, pp: 21-25.

Downs, T. 1988. *Logic design with Pascal*. Van Nostrand Reinhold.

Floyd, T. L. 1994. *Digital fundamentals*. Prentice Hall, quinta edición.

García, J. E.; Gil, D.; Martínez, M. 1992. *Circuitos y sistemas digitales*. Tebar Flores, pp: 83-107.

García Zubía, J.; Kahoraho, K. 1995. «Simplificación de funciones booleanas bajo Matlab: Un nuevo método computacional». I congreso de usuarios de Matlab, Madrid, 22 y 23 de Mayo de 1995, pp: 389-396.

Gergov, J. y Meinel, Ch. 1994. «Efficient boolean manipulation with OBDD's can be extended to FBDD's». *IEEE Trans. on Comp.* Vol. 43, Nº 10, Oct. 1994, pp: 1197-1209.

Grass, W. 1987. «A depth-first branch-bound algorithm for optimal PLA folding». *Proc. 19th Design automation conference*, pp: 133-140, Las Vegas.

Hayes, J. P. 1993. *Introduction to digital logic design*. Addison Wesley, pp: 316-347.

Hill, F. J.; Peterson, G. R. 1993. *Computer aided logical design with emphasis on VLSI*. John Willey & Sons, cuarta edición, pp: 119-163.

Hong, S. J.; Cain R. G.; Ostapko, D. L. 1974. «MINI: A heuristic approach to logic minimizations». *IBM Journal of research and development*, Vol. 18, Sept. 1974 pp: 443-458.

Hong, S. J.; Ostapko, D. L. 1972. «On complementation of boolean functions». *IEEE Trans. on Comp.* Vol. C-21, página: 1022.

Karnaugh, M. 1953. «The map method for synteshis of combinational logic circuits». *Transactions AIEEE*, 72, Nov. pp: 593-599.

Kohavi, Z. 1978. *Switching and finite automata theory*. McGraw-Hill, segunda edición, pp: 74-106.

Kuo, Y. S. 1987. «Generating essential primes for a boolean function with multiple-valued inputs». *IEEE Trans. on Comp.* Vol. C-36, Nº 3, March 1987, pp: 356-358.

Liaw, H. T. y Lin, Ch. S. 1992. «On the OBDD-Representation of general boolean functions». *IEEE Trans. on Comp.* Vol. 41, Nº 6, June 1992, pp: 661-664.

Mandado, E. 1984. *Sistemas electrónicos digitales*. Marcombo.

Mano, M. M. 1991. *Digital design*. Prentice Hall, segunda edición, pp: 72-110.

McCluskey, E. J. 1956. «Minimization of boolean functions». *Bell systems technical journal*, 35, Nov. 1956, páginas: 1417-1444.

McCluskey, E. J. 1986. *Logic design principles: with emphasis on testable semicuston*. Prentice Hall series in computer engineering, pp: 191-238.

McMullen, C. y Shearer, J. 1986. «Prime implicants, minimum covers, and the complexity of logic simplification». *IEEE Trans. on Comp.* Vol. C-35, Nº 8, Aug. 1986, pp: 761-762.

Mott, T. H. 1960. «Determination of the irredundant normal forms of a truth function by iterated consensus of the prime implicants». *IRE Trans. Electron. Computers*, Vol. EC-9, Nº 2, 1960, pp: 245-252.

Muroga, S. 1979. *Logic design and switching theory*. Wiley-Interscience Publication, pp: 68-188.

Muroga, S. 1982. *VLSI system design*. John Wiley and Sons, New York.

Muzio, J. C.y Rosenberg I. C. 1986. «Introduction: Multiple-Valued logic». *IEEE Trans. on Comp.* Vol. C-35, Nº 2, Feb. 1986, pp: 97-98.

Newton, A. R.; Sangiovanni-Vincentelli, A.; 1987. «CAD tools for ASIC design». *Proc. IEEE* , Vol. 75, Nº 6, June 1987, pp: 765-776.

Ostapko, D. L.; Hong, S. J. 1974. «Generating tests examples for heuristic boolean minimization». *IBM Journal of Res. and Dev.* Vol. 18, pp: 459-464.

Petrick, S. R. 1956. «A direct determination of the irredundant forms of a boolean function from the set of prime implicants». *Air force Cambridge research center, Cambridge, Massachusetts Tech report AFCRC-TR-56-110.*

Petrick, S. R. 1959. «On the minimization of boolean functions». *Proceedings of symposium on switching theory,* ICIP, Paris.

Petrick, S. R. 1960. «On the minimizations of boolean functions». *Proceedings of the international conference on information processing.* Paris: Unesco, pp: 422-423.

Pomper, G.; Armstrong, R. J. 1979. «An efficient multivalued minimization algorithm». *Proceedings ISMVL-79.*

Quine, W. V. 1952. «The problem of simplifying truth functions». *American math monthly,* 59, Oct. 1952, pp: 521-531.

Quine, W. V. 1955. «A way to simplify truth functions». *American mathematics monthly,* Vol. 62, páginas: 627-631.

Rensch, B. 1975. «Generation of prime implicants from subfunctions and a unifying approach to the covering problem». *IEEE Trans. on Comp.,* Vol. C-24, Nº 9, pp: 924-930.

Rhyne, V. T.; Noe, P. S., McKinny, M. N.; Pooch, U. W. 1977. «A new technique for the fast minimization of switching function». *IEEE Trans. on Comp.,* Vol. C-26, Nº. 8, pp: 757-764.

Roth, J. P. 1958. «Algebraic topological methods for the synthesis of switching function». *Trans. Amer. Math. Soc.* Vol. 88, pp: 301-326.

Roth, J. P. 1978. «Programmed logic array optimization». *IEEE Trans. on Comp.*, pp: 174-176.

Roth, J. P. 1980. *Computer logic, testing and verification*. Computer Science Press.

Roth, J. P. 1986. «Minimization by the D algorithm». *IEEE Trans. on Comp. Vol.* C-35, Nº 5, May 1986, pp: 476-478.

Rudell, R. L.; Sangiovanni-Vincentelli, A. 1985. «ESPRESSO-MV: algorithms for multiple-valued logic minimization». *IEEE Proc. Cust. Int. Circ. Conf*, May 1985, pp: 230-234.

Rudell, R. L.; Sangiovanni-Vincentelli, A. 1987. «Multiple-Valued minimization for PLA optimization». *IEEE Trans. CAD*, Vol. CAD-6, Nº 5, pp: 727-750.

Sasao, T. 1989. «On the optimal design of multiple-valued PLA's». *IEEE Trans. on Comp.* Vol. 38., Nº 4, April 1989, pp: 582-592.

Tirumalai, P. P. y Butler, J. T. 1991. «Minimization algorithms for multiple-valued programmable logic arrays» *IEEE Trans. on Comp.* Vol. 40, Nº 2, Feb. 1991, pp: 167-177.

Tison, P. 1967. «Generalization of consensus theory and application to the minimization of boolean functions». *IEEE Transactions on electronic computers*, Vol. EC-16, Aug. 1967, pp: 446-456.

Tocci, R. J. 1995. *Digital systems. Principles and applications*. Prentice Hall, sexta edición, pp: 93-115.

Veitch, W. V. 1952. «A chart method for simplifying truth functions». *Proc. ACM*, Pittsburg, May 1952, pp: 127-133.

Wakerly, J. F. 1990. *Digital design principles and practices*. Prentice Hall series in computer engineering, pp: 173-206.

Wegener, I. 1994. «The size of reduced OBDD's and optimal read-once branching programs for almost all boolean functions». *IEEE Trans. on Comp.* Vol. 43, Nº 11, Nov. 1994, pp: 1262-1268.

CLASIFICACIÓN POR TEMAS

General: Simplificación

Althoen, S. C. y Bumcrot, R. J. 1988 Introduction to the discret mathematics. PWS-Kent, Boston, pp: 99-108.

Biswas, N. N. 1993. *Logic design theory*. Prentice Hall, primera edición, pp: 63-91.

Brayton, R. K.; Hachtel, G. D.;McMullen, C. T.; Sangiovanni-Vincentelli; A.L. 1990. *Logic minimization algorithms for VLSI synthesis*. Kluwer academic publishers.

Breeding, K. J. 1992. *Digital design fundamentals*. Prentice Hall, segunda edición, pp: 58-88.

Cahill, S. J. 1993. *Digital and microprocessor engineering*. Ellis Horwood, segunda edición, pp: 81-115.

Dietmeyer, D. L. 1978. *Logic design of digital systems*. Allyn and Bacon Inc, segunda edición, Boston, pp: 609-665.

Dormido, S.; Mira, J.; Delgado, A. E.; Canto, M. A. 1994. *Electrónica digital*. Sanz y Torres, pp: 21-25.

Downs, T. 1988. *Logic design with Pascal*. Van Nostrand Reinhold.

Floyd, T. L. 1994. *Digital fundamentals*. Prentice Hall, quinta edición.

García, J. E.; Gil, D.; Martínez, M. 1992. *Circuitos y sistemas digitales*. Tebar Flores, pp: 83-107.

Hayes, J. P. 1993. *Introduction to digital logic design*. Addison Wesley, pp: 316-347.

Hill, F. J.; Peterson, G. R. 1993. *Computer aided logical design with emphasis on VLSI*. John Willey & Sons, cuarta edición, pp: 119-163.

Kohavi, Z. 1978. *Switching and finite automata theory*. McGraw-Hill, segunda edición, pp: 74-106.

Mandado, E. 1984. *Sistemas electrónicos digitales.* Marcombo.

Mano, M. M. 1991. *Digital design.* Prentice Hall, segunda edición, pp: 72-110.

McCluskey, E. J. 1986. *Logic design principles: with emphasis on testable semicuston.* Prentice Hall series in computer engineering, pp: 191-238.

Muroga, S. 1979. *Logic design and switching theory.* Wiley-Interscience Publication, pp: 68-188.

Muroga, S. 1982. *VLSI system design.* John Wiley and Sons, New York.

Roth, J. P. 1980. *Computer logic, testing and verification.* Computer Science Press.

Tocci, R. J. 1995. *Digital systems. Principles and applications.* Prentice Hall, sexta edición, pp: 93-115.

Wakerly, J. F. 1990. *Digital design principles and practices.* Prentice Hall series in computer engineering, pp: 173-206.

Manipulación booleana

Biswas, N. N. 1971. «Minimization of boolean functions». *IEEE Trans. on Comp.*, Vol. C-20, Nº 8, Aug. 1971, pp: 925-929.

Brayton, R. K.; Cohen, J. D.; Hachtel; Trager, B. M.; Yun, D. Y. Y. 1982. «Fast recursive boolean function manipulation» *Proc. 1982 Int. Symp. on Circ. and Syst.*, pp: 58-62.

Brayton, R.; McMullen, C. 1982. «The descomposition and factorization of boolean expressions». *Proc. Int. Symp. on Circ. and Syst.* pp: 49-54, Roma.

Hong, S. J.; Ostapko, D. L. 1972. «On complementation of boolean functions». *IEEE Trans. on Comp.* Vol. C-21, página: 1022.

Newton, A. R.; Sangiovanni-Vincentelli, A.; 1987. «CAD tools for ASIC design». *Proc. IEEE* , Vol. 75, Nº 6, June 1987, pp: 765-776.

Ostapko, D. L.; Hong, S. J. 1974. «Generating tests examples for heuristic boolean minimization». *IBM Journal of Res. and Dev.* Vol. 18, pp: 459-464.

Roth, J. P. 1958. «Algebraic topological methods for the synthesis of switching function». *Trans. Amer. Math. Soc.* Vol. 88, pp: 301-326.

Roth, J. P. 1978. «Programmed logic array optimization». *IEEE Trans. on Comp.*, pp: 174-176.

Implicados primos

Kuo, Y. S. 1987. «Generating essential primes for a boolean function with multiple-valued inputs». *IEEE Trans. on Comp.* Vol. C-36, Nº 3, March 1987, pp: 356-358.

McMullen, C. y Shearer, J. 1986. «Prime implicants, minimum covers, and the complexity of logic simplification». *IEEE Trans. on Comp.* Vol. C-35, Nº 8, Aug. 1986, pp: 761-762.

Rensch, B. 1975. «Generation of prime implicants from subfunctions and a unifying approach to the covering problem». *IEEE Trans. on Comp.*, Vol. C-24, Nº 9, pp: 924-930.

Método Quine-McCluskey

McCluskey, E. J. 1956. «Minimization of boolean functions». *Bell systems technical journal*, 35, Nov. 1956, páginas: 1417-1444.

Quine, W. V. 1952. «The problem of simplifying truth functions». *American math monthly*, 59, Oct. 1952, pp: 521-531.

Quine, W. V. 1955. «A way to simplify truth functions». *American mathematics monthly*, Vol. 62, páginas: 627-631.

Método Petrick

Petrick, S. R. 1956. «A direct determination of the irredundant forms of a boolean function from the set of prime implicants». *Air force Cambridge research center, Cambridge, Massachusetts Tech report AFCRC-TR-56-110.*

Petrick, S. R. 1959. «On the minimization of boolean functions». *Proceedings of symposium on switching theory,* ICIP, Paris.

Petrick, S. R. 1960. «On the minimizations of boolean functions». *Proceedings of the international conference on information processing.* Paris: Unesco, pp: 422-423.

Consenso iterativo

Mott, T. H. 1960. «Determination of the irredundant normal forms of a truth function by iterated consensus of the prime implicants». *IRE Trans. Electron. Computers,* Vol. EC-9, N" 2, 1960, pp: 245-252.

Tison, P. 1967. «Generalization of consensus theory and application to the minimization of boolean functions». *IEEE Transactions on electronic computers,* Vol. EC-16, Aug. 1967, pp: 446-456.

Método Veitch-Karnaugh

Karnaugh, M. 1953. «The map method for synteshis of combinational logic circuits». *Transactions AIEEE,* 72, Nov. pp: 593-599.

Veitch, W. V. 1952. «A chart method for simplifying truth functions». *Proc. ACM,* Pittsburg, May 1952, pp: 127-133.

Método CAMP II

Biswas, N. N. 1984. «Computer aided minimization procedure for boolean functions». *Proc. 21st Design Automation Conference,* Alburquerque, N.Mex., June 1984, pp: 699-702.

Biswas, N. N. 1986. «Computer aided minimization procedure for boolean functions». *IEEE Trans Computer-Aided Design Integrated Circuits Systems*, Vol. CAD-5, Nº 2, April 1986, pp: 303-304.

Biswas, N. N. 1990. «On covering distant minterms by the CAMP algorithm». *IEEE Trans. Computer-Aided design of integrated circuits systems*, Vol. CAD-9, Nº 7, July 1990, pp: 786-789.

Biswas, N. N. 1993. *Logic design theory*. Prentice Hall, pp: 63-91.

Método McBoole

Dagenais, M. R.; Agarwal, V. K.; Rumin, N. C. 1985. «The McBoole logic minimizer». *Proc. 22nd Design Automation Conf.*, Las Vegas, página: 667.

Dagenais, M. R.; Agarwal, V. K.; Rumin, N. C. 1986. «McBoole: A new procedure for exact logic minimization». *IEEE Trans. CAD of Int. Circ. Syst.*, Vol CAD-5, Nº 1, pp: 229-238.

Método Espresso

Brayton, R. K.; McMullen, C.; Hachtel, G. D.; Sangiovanni-Vincentelli, A. 1984. *Logic minimization algorithms for VLSI synthesis*. Kluwer academic publisher.

Brayton, R.; Hachtel, G. D., Hemachandra, A. R.; Newton, A. R.; Sangiovanni-Vincentelli, A. L. 1982. «A comparision of logic minimization strategies using ESPRESSO. An APL program package for partitioned logic minimization». *Proc. Int. Symp. on Circ. and Syst.* pp: 43-49, Roma.

Rudell, R. L.; Sangiovanni-Vincentelli, A. 1985. «ESPRESSO-MV: algorithms for multiple-valued logic minimization». *IEEE Proc. Cust. Int. Circ. Conf*, May 1985, pp: 230-234.

Otros métodos

Arevalo, Z.; Bredeson, J. G. 1978. «A method to simplify a boolean function into a near minimal sum-of-products for programmable logic arrays». *IEEE Trans. on Comp.* Vol. C-27, Nº 11, pp: 1028-1039.

Bartholomeus, M; Man, H. 1985. PRESTOL-II: «Yet another logic minimizer for programmed logic arrays». *Proc. 1985 Int. Symp. Circuits and Systems*, Kyoto.

Besslich, P. W.; Pichlbauer, P. «Fast transform procedure for the generation of near minimal covers of boolean functions». *IEEE Proc.* Vol. 128, Pt. E Nº 6, pp: 250-254.

García Zubía, J.; Kahoraho, K. 1995. «Simplificación de funciones booleanas bajo Matlab: Un nuevo método computacional». *I congreso de usuarios de Matlab*, Madrid, 22 y 23 de Mayo de 1995, pp: 389-396.

Grass, W. 1982. «A depth-first branch-bound algorithm for optimal PLA folding». *Proc. 19th Design automation conference*, pp: 133-140, Las Vegas.

Hong, S. J.; Cain R. G.; Ostapko, D. L. 1974. «MINI: A heuristic approach to logic minimizations». *IBM Journal of research and development*, Vol. 18, Sept. 1974 pp: 443-458.

Pomper, G.; Armstrong, R. J. 1979. «An efficient multivalued minimization algorithm». *Proceedings ISMVL-79*.

Rhyne, V. T.; Noe, P. S., McKinny, M. N.; Pooch, U. W. 1977. «A new technique for the fast minimization of switching function». *IEEE Trans. on Comp.*, Vol. C-26, Nº. 8, pp: 757-764.

Roth, J. P. 1986. «Minimization by the D algorithm». *IEEE Trans. on Comp. Vol.* C-35, Nº 5, May 1986, pp: 476-478.

Funciones multivalentes

Besslich, P. W. 1986. «A heuristic minimization of MVL functions: a direct cover approach». *IEEE Trans. on Comp.* Vol. C-35, Nº 2, Feb. 1986, pp: 134-144.

Chan, A. H. 1987. «Using decision trees to derive the complement of a binary function with multiple-valued inputs». *IEEE Trans. on Comp.* Vol. C-36, Nº 2, Feb. 1987, pp: 212-214.

Kuo, Y. S. 1987. «Generating essential primes for a boolean function with multiple-valued inputs». *IEEE Trans. on Comp.* Vol. C-36, Nº 3, March 1987, pp: 356-358.

Muzio, J. C.y Rosenberg I. C. 1986. «Introduction: Multiple-Valued logic». *IEEE Trans. on Comp.* Vol. C-35, Nº 2, Feb. 1986, pp: 97-98.

Rudell, R. L.; Sangiovanni-Vincentelli, A. 1987. «Multiple-Valued minimization for PLA optimization». *IEEE Trans. CAD*, Vol. CAD-6, Nº 5, pp: 727-750.

Sasao, T. 1989. «On the optimal design of multiple-valued PLA's». *IEEE Trans. on Comp.* Vol. 38., Nº 4, April 1989, pp: 582-592.

Tirumalai, P. P. y Butler, J. T. 1991. «Minimization algorithms for multiple-valued programmable logic arrays» *IEEE Trans. on Comp.* Vol. 40, Nº 2, Feb. 1991, pp: 167-177.

Árboles de decisión binarios

Bryant, R. E. 1986. «Graph-Based algorithms for boolean function manipulation». *IEEE Trans. on Comp.* Vol. C-35, Nº 8, Aug. 1986, pp: 677-691.

Chakravarty, S. 1993. «A characterization of binary decision diagrams». *IEEE Trans. on Comp.* Vol. 42, Nº 2, Feb. 1993, pp: 129-137.

Chan, A. H. 1987. «Using decision trees to derive the complement of a binary function with multiple-valued inputs». *IEEE Trans. on Comp.* Vol. C-36, Nº 2, Feb. 1987, pp: 212-214.

Gergov, J. y Meinel, Ch. 1994. «Efficient boolean manipulation with OBDD's can be extended to FBDD's». *IEEE Trans. on Comp.* Vol. 43, Nº 10, Oct. 1994, pp: 1197-1209.

Liaw, H. T. y Lin, Ch. S. 1992. «On the OBDD-Representation of general boolean functions». *IEEE Trans. on Comp.* Vol. 41, Nº 6, June 1992, pp: 661-664.

Wegener, I. 1994. «The size of reduced OBDD's and optimal read-once branching programs for almost all boolean functions». *IEEE Trans. on Comp.* Vol. 43, Nº 11, Nov. 1994, pp: 1262-1268.