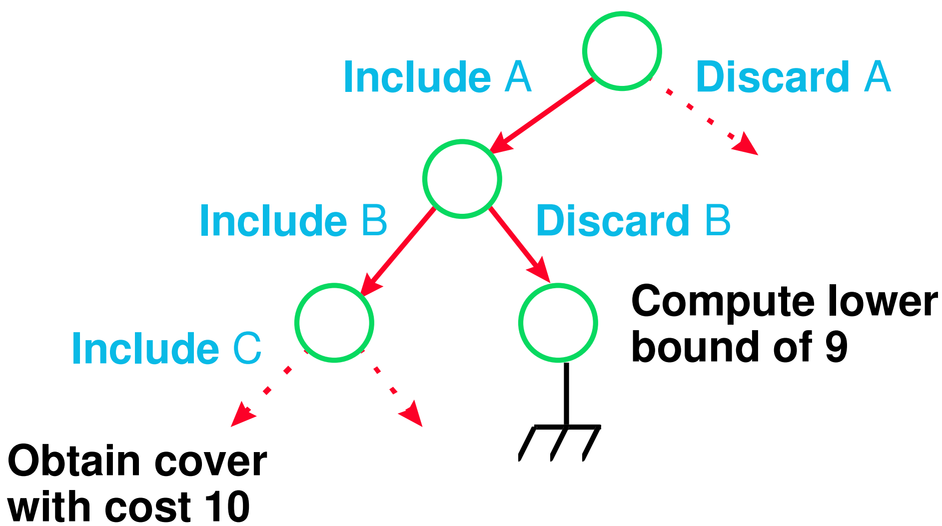
Espresso-Exact (1987)

Efficient lower bounding at Step 4(b) to terminate unprofitable searches high in the recursion



Size of selected set + Lower bound ≤ best solution already known, quit level of recursion

Lower Bounding

	A	B	C	D	E	F
0	X				X	
1	X		X			X
4					X	
6		X	X			
8		X				
10				X		X
12				X		X

Lower bound: Maximal independent set of rows all of which are pairwise disjoint

Maximal independent set = {1, 4, 8} or {0, 6, 10}

Need to select at least one PI/column to cover each row.

NOTE: Finding <u>maximum</u> independent set is itself worst-case exponential

Complexity of Q-M based Methods

(1) There exist functions for which the number of prime implicants is $O(3^n)$ (n is number of inputs)

(2) Given a PI table, recursive branching could require $O(2^m)$ time (m is the number of PIs)

Current logic minimizers able to find exact solutions for functions with 20-25 input variables

$\Rightarrow$ Need heuristic methods for larger functions

Heuristic Logic Minimization

Presently, there appears to be a limit of ~20-25 input variables in problems that can be handled by exact minimizers

HISTORY

50's	Karnaugh Map	$\leq$ 5 variables
60's	Q-M method	< 10 variables
70's	Starner, Dietmeyer	< 15 variables
1974	MINI	heuristic
1980-84	ESPRESSO	approaches
1986	McBoole	< 25 variables
1987	ESPRESSO-EXACT	< 25 variables

Multiple Output Functions

Truth table is AND-OR representation

AND			OR	
a	b	c	f	g
0	1	–	1	0
0	1	1	0	1
1	0	1	0	1

What does vector 0 1 1 **produce?**

ON-SET of f = {0 1 –, 0 1 1} = {0 1 –}

ON-SET of g = {0 1 1, 1 0 1}

Multiple-Output Function Primes

Same definition as in single-output case

- Cube with most minterms that will intersect OFF-SET if you add any more minterms to them

	f g
0 0 0 0	1 0
0 0 0 1	1 0
1 0 0 1	1 0
0 0 0 0	0 1
0 0 1 0	0 1
1 0 0 1	0 1

CUBE		TYPE
0 0 0 0	1 0	
0 0 0 –	1 0	
1 0 0 1	1 0	
1 0 0 1	1 1	
0 0 0 –	1 1	

MINI

S.J. Hong, R.G. Cain, D.L. Ostapko - 1974

Final solution is obtained from initial solution by iterative improvement rather than by generating and covering prime implicants

Three basic modifications are performed

- Reduction of implicants while maintaining coverage
- Reshaping implicants in pairs
- Expansion of implicants (and removal of covered implicants)

MINI Algorithm

MINI (F, DC) {

F is ON-SET
DC is Don't Care Set
U is universe cube

1. $\overline{F} = U - F$
2. F = Expand F against $\overline{F}$
   p = Compute solution size
3. F = Reduce each cube of F
   against other cubes of $F \vee DC$
4. Reshape F
5. F = Expand F against $\overline{F}$
   n = compute solution size
6. If n < p go to 3, else, exit

}

Expanding a Cube

Goal is to expand a cube such that it covers a maximum number of other cubes

Cubes can be "grown" in many ways

		1 0 1 0	11	Prime
Implicant	1 0 1 0 01	1 0 1 –	01	Prime
		– – 1 0	01	Prime

OFF-SET 0 0 1 0 for 1st output
– – 0 – for 2nd output

Expand against OFF-SET

Expansion Example

Step 2 in MINI:

Expand F against $\overline{F}$

F		$F_{expanded}$		$\overline{F}$	
1 0 0 1	0 1	1 0 0 1	0 1	1 1 1 0	1 0
0 1 1 0	1 0	0 – 1 0	1 0	0 1 0 1	1 0
1 1 0 1	1 0	1 1 0 1	1 0	0 0 1 0	0 1
1 0 0 0	1 0	1 0 – 0	1 0	0 0 0 1	0 1
1 0 1 0	0 1	1 0 1 0	0 1	0 0 0 0	1 0
– 1 0 0	1 0	– 1 0 0	1 0	– 1 0 1	0 1
– 1 1 1	0 1	– – 1 1	1 1	– 0 0 0	0 1
1 0 1 1	0 1				
– 1 1 1	1 0				
– 0 1 0	1 0				
– 1 – 0	0 1	– 1 – 0	0 1		
– 0 – 1	1 0	– 0 – 1	1 0		

Order small cubes first

Reduction

Reduce the size of cubes in F without affecting coverage

The smaller the size of the cube, the more likely it will be covered by an expanded cube

Reduction Examples

Reducing covers:

F

1	–	–	1
–	1	–	1
–	–	1	1

$F_{reduced}$

1	0	0	1
–	1	–	1
–	–	1	1

1	0	0	1	0	1
0	–	1	0	1	0
1	1	0	1	1	0
1	0	–	0	1	0
1	0	1	0	0	1
–	1	0	0	1	0
–	–	1	1	1	1
–	1	–	0	0	1
–	0	–	1	1	0

$\rightarrow$

–	–	1	1	1	1
–	0	–	1	1	0
–	1	–	0	0	1
1	0	–	0	1	0
–	1	0	0	1	0
0	–	1	0	1	0
1	1	0	1	1	0
1	0	1	0	0	1
1	0	0	1	0	1

Larger cubes first

(– 0 1 1   1 0)

–	–	1	1	1	1
–	0	0	1	1	0
–	1	–	0	0	1
1	0	–	0	1	0
–	1	0	0	1	0
0	–	1	0	1	0
1	1	0	1	1	0
1	0	1	0	0	1
1	0	0	1	0	1

Reshaping

Attempt to change the shape of the cubes without changing coverage or number

Reshaping transforms a pair of cubes into another pair such that coverage is unaffected (perturbs solution so next expand does things differently)

Reshaping Example

F

–	–	1	1	1	1
–	0	0	1	1	0
–	1	–	0	0	1
1	0	–	0	1	0
–	1	0	0	1	0
0	–	1	0	1	0
1	1	0	1	1	0
1	0	1	0	0	1
1	0	0	1	0	1

$F_{ordered}$

–	–	1	1	1	1
–	1	–	0	0	1
1	0	–	0	1	0
–	0	0	1	1	0
–	1	0	0	1	0
0	–	1	0	1	0
1	1	0	1	1	0
1	0	1	0	0	1
1	0	0	1	0	1

1	–	–	1	1	1	1
2	–	1	–	0	0	1
3	1	0	–	0	1	0
4	–	0	0	1	1	0
5	–	1	0	0	1	0
6	0	–	1	0	1	0
7	1	1	0	1	1	0
8	1	0	1	0	0	1
9	1	0	0	1	0	1

$F_{reshaped}$

1	–	–	1	1	1	1
(2,5)	–	1	1	0	0	1
	–	1	0	0	1	1
(3,8)	1	0	0	0	1	0
	1	0	1	0	1	1
(4,9)	0	0	0	1	1	0
	1	0	0	1	1	1
6	0	–	1	0	1	0
7	1	1	0	1	1	0

A Complete Example

cd \ ab	f: 00	f: 01	f: 11	f: 10	g: 00	g: 01	g: 11	g: 10	
00		9	9	10		5	5		00
01	4		3	4				2	01
11	1	1	1	1	1	1	1	1	11
10	8	7		8		5	5	6	10

initial F

	a b c d	f g			
1	– – 1 1	1 1		– – 1 1	1 1
2	1 0 0 1	0 1		1 0 0 1	0 1
3	1 1 0 1	1 0	expand	1 1 0 1	1 0
4	– 0 0 1	1 0	→	– 0 – 1	1 0
5	– 1 – 0	0 1		– 1 – 0	0 1
6	1 0 1 0	0 1		1 0 1 0	0 1
7	0 1 1 0	1 0	→	0 – 1 0	1 0 ←
8	– 0 1 0	1 0			
9	– 1 0 0	1 0		– 1 0 0	1 0
10	1 0 0 0	1 0	→	1 0 – 0	1 0 ←

Example - 2

cd \ ab	00	01	11	10	00	01	11	10	
00		9	9	10		5	5		00
01	4		3	4				2	01
11	1,4	1	1	1,4	1	1	1	1	11
10	7	7		10		5	5	6	10

expanded F

	a	b	c	d	f	g
1	–	–	1	1	1	1
2	1	0	0	1	0	1
3	1	1	0	1	1	0
4	–	0	–	1	1	0
5	–	1	–	0	0	1
6	1	0	1	0	0	1
7	0	–	1	0	1	0
9	–	1	0	0	1	0
10	1	0	–	0	1	0

reduce →

–	–	1	1	1	1
1	0	0	1	0	1
1	1	0	1	1	0
–	0	0	1	1	0
–	1	–	0	0	1
1	0	1	0	0	1
0	–	1	0	0	1
–	1	0	0	1	0
1	0	–	0	1	0

Example - 3

cd \ ab	00	01	11	10	00	01	11	10	
00		9	9	10		5	5		00
01	4		3	4				2	01
11	1	1	1	1	1	1	1	1	11
10	7	7		10		5	5	6	10

reduced F

	a	b	c	d	f	g
1	–	–	1	1	1	1
2	1	0	0	1	0	1
3	1	1	0	1	1	0
4	–	0	0	1	1	0
5	–	1	–	0	0	1
6	1	0	1	0	0	1
7	0	–	1	0	1	0
9	–	1	0	0	1	0
10	1	0	–	0	1	0

reshape

–	–	1	1	1	1	
1	0	0	1	1	1	2,4
1	1	0	1	1	0	
0	0	0	1	1	0	2,4
–	1	1	0	0	1	5,9
1	0	1	0	1	1	6,10
0	–	1	0	1	0	
–	1	0	0	1	1	5,9
1	0	0	0	1	0	6,10

Example - 4

cd \ ab	00	01	11	10	00	01	11	10	
00		9	9	10		9	9		00
01	4		3	2				2	01
11	1	1	1	1	1	1	1	1	11
10	7	7		6		5	5	6	10

reshaped F

	a b c d	f g	expand		
1	– – 1 1	1 1		– – 1 1	1 1
2	1 0 0 1	0 1	→	1 0 – 1	1 1
3	1 1 0 1	1 0	→	1 – 0 –	1 0 ←
4	0 0 0 1	1 0	→	– 0 – 1	1 0
5	– 1 1 0	0 1	→	– 1 – 0	0 1
6	1 0 1 0	1 1	→	1 0 1 –	1 1
7	0 – 1 0	1 0		0 – 1 0	1 0
9	– 1 0 0	1 1		– 1 0 0	1 1
10	1 0 0 0	1 0			

Example - 5

cd \ ab	00	01	11	10	00	01	11	10	
00		9	3,9	3		5,9	5,9		00
01	4		3	2,3,4				2	01
11	1,4,7	1,7	1	1,2,4,6	1	1	1	1,2,6	11
10	7	7		6		5	5	6	10

final expanded F

final F

	a	b	c	d	f	g
1	–	–	1	1	1	1
2	1	0	–	1	1	1
3	1	–	0	–	1	0
4	–	0	–	1	1	0
5	–	1	–	0	0	1
6	1	0	1	–	1	1
7	0	–	1	0	1	0
9	–	1	0	0	1	1

Summary of MINI

Iterative local optimization (as opposed to global) appears to work very well

– Properties of logic functions?

For large problems, where the exact solution is known, MINI produces an exact or near-exact solution much more quickly

Can minimize problems, exact minimizers can’t handle