Essential Prime Implicants

Cube \( p \) is an essential prime implicant of \( f \) if it contains a minterm not contained by any other prime implicant \( p' \) of \( f \).

All essential prime implicants are present in every optimal SOP.
Essential Prime Implicants

- Only $xz, x'y$ are **essential** (exclusively contains minterms $w'xy'z$ and $wxy'z$ of $f$).

Unate SOPs Revisited

An SOP in which each variable appears only as positive literals or only as negative literals is called **Unate**.

Unate:

$$x_1x_3 + x_1x_3x_4 + x_2x_3 + x_2x_3x_4 = x_1x_3 + x_2x_3$$

Non-Unate: $xz + x'y$ (Note possible consensus)

If $F$ is a unate SOP, $\text{ABS}(F)$ is the complete sum of the function represented by $F$, and all the primes are essential.
Picking a Subset of the Primes

\[ F = yz + x'y + y'z' + xyz + x'z' \]
\[ \text{MCF}(F) = x'y'z' + x'yz' + x'yz + xyz + xy'z' \]
\[ \text{CS}(F) = x'y + x'z' + y'z' + yz \]

\[
\begin{array}{cccc}
p_1 & p_2 & p_3 & p_4 \\
\text{primes} & x'y'z' & 0 & 1 & 1 & 0 \\
x'y'z & 1 & 0 & 0 & 1 \\
x'yz & 0 & 0 & 1 & 0 \\
xy'z' & 0 & 0 & 0 & 1 \\
xyz & 0 & 0 & 0 & 1 \\
\end{array}
\]

Picking a Subset of the Primes

The implied constraint is
\[ F_{\text{Min}} = p_3p_4(p_1 + p_2) = 1 \]

We regard the \( p_i \) as Boolean variables.
\[ p_3 = 1 \]

Means that \( p_3 \) is included in the selected subset.

So there are just 2 possible subsets: \( p_3p_4p_1 \) or \( p_3p_4p_1 \)
Picking a Subset of the Primes

\[ F = yz + x'y + y'z' + xy'z + x'z' \]

\[ \text{MCF}(F) = x'y'z' + x'yz + x'yz + xy'z' + xy'z + xyz \]

\[ \text{CS}(F) = x'y + x'z' + y'z' + yz + xy' + xz \]

Cyclic!

No essentials, all primes redundant

<table>
<thead>
<tr>
<th>( p_1 )</th>
<th>( p_2 )</th>
<th>( p_3 )</th>
<th>( p_4 )</th>
<th>( p_5 )</th>
<th>( p_6 )</th>
<th>( p_4 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( x'yz )</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>( x'yz' )</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>( x'y'z' )</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>( xy'z' )</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>( xy'z )</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>( xyz )</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

Generalization

- Logic Minimization is just one of many important problems that can be formulated as a matrix covering problem.
- Others include Technology Mapping, FSM State Minimization with DCs, and many others from various fields.
- So we study this problem intensively.
The Astronaut and the Cookies

Optimal Nutrition:

A good diet should contain adequate amounts of proteins (P), vitamins (V), fats (F), and cookies (C). An astronaut, who must travel light, can choose from five different preparations:

- Preparation 1 (logic variable \( p_1 \)) contains: V and P;
- Preparation 2 (logic variable \( p_2 \)) contains: V and F;
- Preparation 3 (logic variable \( p_3 \)) contains: P and F;
- Preparation 4 (logic variable \( p_4 \)) contains: V;
- Preparation 5 (logic variable \( p_5 \)) contains: C.

Can the astronaut have a balanced diet with only two preparations?

Proteins: \(( p_1 + p_3 )\), Vitamins: \(( p_1 + p_2 + p_4 )\),...

---

The Astronaut and the Cookies

Brute Force Enumeration Approach:

- Encode candidates with binary variables \( p_i \) (like primes)
- Express constraints in POS form (like minterm coverage)
- Convert POS to SOP (enumerate all possibilities)
- Pick “smallest” cube (fewest literals)

\[
1 = (\text{Proteins})(\text{Vitamins})(\text{Fats})(\text{Cookies})
\]
\[
= (p_1 + p_3)(p_1 + p_2 + p_4)(p_2 + p_3)(p_5)
\]
\[
= (p_1 + p_3(p_2 + p_4))(p_2 + p_3)(p_5)
\]
\[
= (p_1)(p_2 + p_3)(p_5) + p_3(p_2 + p_4)(p_5)
\]
\[
= p_1p_2p_5 + p_1p_3p_5 + p_2p_3p_5 + p_3p_4p_5
\]

All possible solutions!
The Trouble with Enumeration

• POS to SOP conversion, like many enumeration algorithms, is in class NP (up to $2^n$ minterms)

• Thus either it has no efficient (sub-exponential) implementation, or a host of other exhaustively studied problems, like the traveling salesman problem, also have efficient solutions

• So if you find a linear or quadratic algorithm for any of these problems, you’ll be rich and famous

“Effective” Enumeration: Branch and Bound

• Accept the inevitability of exponential worst case performance--”Implicitly Enumerate” the search space

• **Reduce** the problem into simplest equivalent problem--at least one optimum solution remains

• Prune the search space by computing **lower bounds**
Reductions: Essential Columns

- Variables with singleton constraints, like \( p_5 \) in the cookies covering problem, are essential and are present in every optimum solution.

- If Row \( i \) of \( M \) contains a single nonzero in Col \( j \), then add \( j \) to the partial solution and delete all rows of \( M \) with a nonzero in Col \( j \).

- Col \( j \), plus the optimum solution to the reduced matrix, is an optimum solution of the original problem.

**Essential Columns: Example**

\[
1 = (p_1 + p_3)(p_1 + p_2 + p_4)(p_2 + p_3)(p_5)
\]

\[
\begin{array}{c|ccccc}
 & p_1 & p_2 & p_3 & p_4 & p_5 \\
\hline
R_1 & 1 & 0 & 1 & 0 & 0 \\
R_2 & 1 & 1 & 0 & 1 & 0 \\
R_3 & 0 & 1 & 1 & 0 & 0 \\
R_4 & 0 & 0 & 0 & 0 & 1 \\
\end{array}
\]

\[
\begin{array}{c|ccccc}
 & p_1 & p_2 & p_3 & p_4 \\
\hline
R_1 & 1 & 0 & 1 & 0 \\
R_2 & 1 & 1 & 0 & 1 \\
R_3 & 0 & 1 & 1 & 0 \\
\end{array}
\]

\[
S = \{ p_5 \} \cup S' \]

\( p_5 \), plus an optimum solution (say \( p_1 p_2 \)) to the reduced matrix, is an optimum solution of the original problem.
Reductions: Row Dominance

- Row $i$ of $M$ dominates Row $k$ if every nonzero of Row $k$ is matched by a nonzero of Row $i$ in the same column, that is, $M_{kj} = 1 \implies M_{ij} = 1$, $\forall j$
- Any set of columns that covers Row $k$ will also cover Row $i$
- Hence dominating rows may be deleted without affecting the size of the optimum solution

- Such deletion may lead to column dominance (later today)

Row Dominance: Example

\[
M = \begin{bmatrix}
1 & 0 & 0 & 1 & 0 & 1 \\
1 & 1 & 0 & 1 & 0 & 1 \\
0 & 1 & 1 & 0 & 0 & 1 \\
1 & 0 & 1 & 0 & 1 & 0
\end{bmatrix}, \quad M' = \begin{bmatrix}
1 & 0 & 0 & 1 & 0 & 1 \\
1 & 1 & 0 & 1 & 0 & 1 \\
0 & 1 & 1 & 0 & 0 & 1 \\
0 & 1 & 1 & 0 & 0 & 1
\end{bmatrix}
\]

Note: Row 4 dominates Row 1

$OptSol(M) = OptSol(M')$
Reductions: Column Dominance

- Col \( j \) of \( M \) dominates Col \( k \) if every nonzero of Col \( k \) is matched by a nonzero of Col \( j \) in the same row, that is \( M_{ik} = 1 \implies M_{ij} = 1, \forall i \)
- Any set of columns that contains Col \( j \) will also cover all rows \( i \) covered by Col \( k \).
- Hence \textit{dominated} columns may be deleted without affecting the size of the optimum solution

- Such deletion may lead to row dominance or even reveal new essential columns

\begin{align*}
\text{Column Dominance: Example} \\
&\begin{array}{c|ccccc}
& p_1 & p_2 & p_3 & p_4 & p_5 \\
\hline
1 & 1 & 1 & 0 & 1 & 0 \\
2 & 1 & 1 & 0 & 1 & 0 \\
3 & 0 & 1 & 1 & 0 & 0 \\
4 & 1 & 0 & 1 & 0 & 1 \\
\end{array}
\quad
\begin{array}{c|cc}
& p_1 & p_2 & p_3 \\
\hline
1 & 1 & 0 & 1 \\
2 & 1 & 1 & 0 \\
3 & 0 & 1 & 1 \\
\end{array}
\end{align*}

\[ M = 2, \quad M' = 3 \]

Cost\( (OptSol(M)) = Cost(\text{OptSol}(M')) \)

- Col 1 \textit{dominates} Cols 4 and 5
- Row 4 \textit{co-dominates} Row 1
- Note \textit{cyclic core}
- Note Solutions including \( p_4 \) or \( p_5 \) are ignored
Matrix Reduction

Procedure REDUCE($M$)
1 $EC = COLS_OF_SINGLETON_ROWS(M)$
2 "delete cols in $EC$ and rows with cols in $EC$"
3 "Add cols in $EC$ to optimum solution"
4 "delete rows which dominate other rows"
5 "delete cols which are dominated by other cols"
6 if($M \neq \emptyset$, and has changed) repeat 1-5

- Essential Column reduction
- Row Dominance reduction
- Column Dominance reduction
- Iterate: Result is unique “Cyclic Core”

Solved Problem 4-14: Reduction

<table>
<thead>
<tr>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\times$</td>
<td>$\times$</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

| 1 | 1 | 1 | 0 | 0 | 1 |
| 2 | 0 | 1 | 1 | 1 | 0 |
| 3 | 0 | 0 | 1 | 1 | 1 |
| 4 | 1 | 1 | 0 | 1 | 0 |

$\rightarrow$

<table>
<thead>
<tr>
<th>2</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
</table>

| 1 | 1 | 0 | 1 |
| 2 | 1 | 1 | 0 |
| 3 | 0 | 1 | 1 |
| 4 | $\times$ | 1 | 1 | 0 |

Cols 2,3 Dominate Row 4 (Co-) Cyclic Core
1,4 Respectively dominates Row 2

Note: No essential columns
\[
\begin{array}{c|ccccc}
    & p_1 & p_2 & p_3 & p_4 & p_5 \\
\hline
1   & 1   & 0   & 1   & 0   & 1 \\
2   & 1   & 1   & 0   & 1   & 0 \\
3   & 0   & 1   & 1   & 0   & 0 \\
\end{array}
\]

\[ M = 2 \]

\[
\begin{array}{c|cc}
    & p_1 & p_2 & p_3 \\
\hline
1   & 1   & 0   & 1 \\
2   & 1   & 1   & 0 \\
3   & 0   & 1   & 1 \\
\end{array}
\]

\[ M' = \]

\[
\begin{array}{c|cc}
    & p_1 & p_2 & p_3 \\
\hline
1   & 1   & 0   & 1 \\
2   & 1   & 1   & 0 \\
3   & 0   & 1   & 1 \\
\end{array}
\]

Note: Row dominance kills constraints

\[
M: (p_1 + p_3 + p_5)(p_1 + p_2 + p_4)(p_2 + p_3)(p_1 + p_3 + p_4 + p_5) \\
= (p_1 + (p_3 + p_5)(p_2 + p_4))(p_2 + p_3) \\
= p_1(p_2 + p_3) + (p_3 + p_5)(p_2 + p_3) \\
= p_1p_2 + p_1p_3 + p_2p_3 + p_3p_4 + p_2p_5 \\
\]

\[
M': (p_1 + p_3)(p_1 + p_2)(p_2 + p_3) \\
= (p_1 + p_2p_3)(p_2 + p_3) = p_1p_2 + p_1p_3 + p_2p_3 \\
\]

Note: Col dominance kills alternative solutions

---

\[
\begin{array}{c|cccccc}
    & p_1 & p_2 & p_3 & p_4 & p_5 & p_6 \\
\hline
1   & 1   & 0   & 1   & 0   & 0   & 0 \\
2   & 0   & 1   & 0   & 1   & 0   & 0 \\
3   & 0   & 0   & 0   & 0   & 1   & 1 \\
4   & 1   & 0   & 1   & 0   & 1   & 1 \\
\end{array}
\]

Lower Bounds

- Note rows 1,2,3 are “column disjoint”, that is, the nonzero column sets \{1,3\},\{2,4\},\{5,6\} are pair-wise disjoint
- Thus at least 3 columns are required to cover the rows of \(M\)
- Thus if we somehow find a solution of size 3, we can take it and quit --- it is an optimum solution.
Lower Bounds

- Row $i$ of $M$ is disjoint from Row $k$ if no nonzero of Row $i$ is matched by a nonzero of Row $k$ in the same column, and conversely. That is, $M_{ij} = 1 \Rightarrow M_{kj} = 0$, $M_{kj} = 1 \Rightarrow M_{ij} = 0$, $\forall j$
- $m$ distinct columns are required to cover $m$ pair-wise disjoint rows
- Hence the cost of covering a matrix that contains $m$ pair-wise disjoint rows is at least $m$
- Lower bounds can prune vast regions of the search

Quick Lower Bound Algorithm

Procedure MIS QUICK($M$)
1 $MIS = \emptyset$
2 do {
3 
4 $i = $CHOOSE SHORTEST ROW($M$)
5 $MIS = MIS \cup \{i\}$
6 $M = $DELETE INTERSECTING ROWS($M$, $i$)
7 } while(||$M|| > 0) continue
8 return row set $MIS$

This is a cheap heuristic: Finding “best” lower bound can be harder than solving the original covering problem
**Procedure** MIS_QUICK($M$)
1 $MIS = \emptyset$
2 do {
3 \hspace{1em} $i = \text{CHOOSE\_SHORTEST\_ROW}(M)$
4 \hspace{1em} $MIS = MIS \cup \{i\}$
5 \hspace{1em} $M = \text{DELETE\_INTERSECTING\_ROWS}(M,i)$
6 } while($|M| > 0$) continue
7 return row set $MIS$

---

**Example**

1 $\times$ 1 1 0 0 0 0 $\leftarrow MIS = \{1\}$
2 $\times$ 0 1 1 0 0 0
3 0 0 1 1 0 0 $\rightarrow$ 3 $\times$ 0 0 1 1 0 0 $\leftarrow MIS = \{1,3\}$
4 0 0 0 1 1 0 4 $\times$ 0 0 0 1 1 0
5 0 0 0 0 1 1 5 0 0 0 0 1 1
6 $\times$ 1 0 0 0 0 1 5 $\times$ 0 0 0 0 1 1 $\leftarrow MIS = \{1,3,5\}$

---

**Example Showing Lower Bounding**

Initial upper bound: $U = 6 + 1 = 7$

<table>
<thead>
<tr>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

- Cyclic Core: no essential columns, Row dominance, or Column dominance
- Row 1 “Intersects” rows 2-4

MIS = \{1\}, $L = 0 + 1 = 1$

Split on Column 1
Example Showing Lower Bounding

Recursive Call for  $p_1 = 1$

\[
\begin{array}{cccccc}
1 & 2 & 3 & 4 & 5 & 6 \\
\times \\
1 & 1 & 0 & 1 & 0 & 1 & 0 & 2 & 3 & 4 & 5 & 6 \\
2 & 1 & 0 & 0 & 1 & 0 & 1 & p_1 = 1 & \times \times \times \\
3 & 0 & 1 & 1 & 0 & 0 & 1 & \Rightarrow & 3 & 1 & 1 & 0 & 0 & 1 \\
4 & 0 & 1 & 0 & 1 & 1 & 0 & 4 & 1 & 0 & 1 & 1 & 0 \\
\end{array}
\]

currentSol = \{p_1\}, \ EC = \{p_2\}, \ U = 1+1 = 2

\{p_1, p_2\} is a best solution which includes $p_1$.

Example Showing Lower Bounding

Recursive Call for  $p_1 = 0$

\[
\begin{array}{cccccc}
1 & 2 & 3 & 4 & 5 & 6 \\
\times \\
1 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \ \\
2 & 1 & 0 & 0 & 1 & 0 & 1 & p_1 = 0 & 2 & 0 & 0 & 1 & 0 & 1 \\
3 & 0 & 1 & 1 & 0 & 0 & 1 & \Rightarrow & 3 & 1 & 1 & 0 & 0 & 1 \\
4 & 0 & 1 & 0 & 1 & 1 & 0 & 4 & 1 & 0 & 1 & 1 & 0 \\
\end{array}
\]

Cyclic core

EC = \{ \}, \ MIS = \{1,2\}, \ L = 0 + 2 = 2 \geq U
Procedure BCP($M, U, currentSol$) {
  1. $(M, currentSol) = \text{REDUCE}(M, currentSol)$
  2. if (terminalCase($M$)) { \hspace{1cm} $\|M\| = 0$
    3. \hspace{1cm} if (COST($currentSol$) < $U$) {
        \hspace{2cm} $U = \text{COST}(currentSol)$
        \hspace{2cm} \text{return} (currentSol)
    4. \hspace{1cm} else return("no (better) solution (in this subspace)")
  5. }
  6. \hspace{1cm} $L = \text{LOWER\_BOUND}(M, currentSol)$
   \hspace{1cm} if ($L \geq U$) return("no (better) solution (in this subspace)")
  7. \hspace{1cm} $x_i = \text{CHOOSE\_VAR}(M)$ \hspace{1cm} \hspace{1cm} \hspace{1cm} \hspace{1cm} longest column
  8. \hspace{1cm} $S^1 = \text{BCP}(M_{x_i}, U, currentSol \cup \{x_i\})$
  9. \hspace{1cm} if (COST($S^1$) = $L$) return($S^1$)
    \hspace{1cm} $S^0 = \text{BCP}(M_{x_i}, U, currentSol)$
  10. \hspace{1cm} \text{return \ BEST\_SOLUTION \ ($S^1, S^0$)}
}

Procedure BCP($M, U, currentSol$) {
  1. $(M, currentSol) = \text{REDUCE}(M, currentSol)$
  2. if (terminalCase($M$)) { \hspace{1cm} $\|M\| = 0$
    3. \hspace{1cm} if (COST($currentSol$) < $U$) {
        \hspace{2cm} $U = \text{COST}(currentSol)$
        \hspace{2cm} \text{return} (currentSol)
    4. \hspace{1cm} else return("no (better) solution (in this subspace)")
  5. }
  6. \hspace{1cm} $L = \text{LOWER\_BOUND}(M, currentSol)$
   \hspace{1cm} if ($L \geq U$) return("no (better) solution (in this subspace)")
  7. \hspace{1cm} $p_1 = 1$ \hspace{1cm} $p_2 = 1$ \hspace{1cm} $p_3 = 0$ \hspace{1cm} $p_4 = 1$ \hspace{1cm} $p_5 = 1$
    \hspace{1cm} $p_1 = 1$ \hspace{1cm} $p_2 = 1$ \hspace{1cm} $p_3 = 0$
    \hspace{1cm} $M = \begin{bmatrix}
    1 & 1 & 0 & 1 & 0 & 1
    \end{bmatrix}
    \begin{bmatrix}
    2 & 1 & 1 & 0 & 1 & 0
    \end{bmatrix}$
    \hspace{1cm} $M' = \begin{bmatrix}
    2 & 1 & 1 & 0
    \end{bmatrix}$
    \hspace{1cm} $3 \times 1 \hspace{1cm} 0 \hspace{1cm} 1 \hspace{1cm} 1 \hspace{1cm} 1$
  8. \hspace{1cm} $3 \times 1 \hspace{1cm} 0 \hspace{1cm} 1 \hspace{1cm} 1 \hspace{1cm} 1$
  9. \hspace{1cm} $U \times 1 \hspace{1cm} 0 \hspace{1cm} 1 \hspace{1cm} 1 \hspace{1cm} 1$
  10. \hspace{1cm} $U = 3$
  11. \hspace{1cm} $U = 5 + 1 = 6$
  12. \hspace{1cm} $L = \#\text{ESS} + |MIS| = 0 + 1 = 1$ \hspace{1cm} (or 2)
}

For cyclic problems
Recursion Tree

Solved Problem 4-14: “Splitting”

Row counts

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>1</th>
<th>0</th>
<th>1</th>
<th>2</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td></td>
<td></td>
<td>A</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
<td></td>
<td>B</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
<td></td>
<td>C</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

MIS = {1}, U = 6

L = 0 + |MIS| = 0 + 1 = 1

¬(L ≥ U) (So recur)

Column counts

<table>
<thead>
<tr>
<th></th>
<th>2</th>
<th>2</th>
<th>2</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Splitting Variable Heuristic: Pick “long” columns which cover many “short” rows
**Picking the “Splitting Variable”**

1. \((M, currentSol) = \text{REDUCE}(M, currentSol)\)
   
   if (terminalCase(M))
   
   \{ \)
   
   if (COST(currentSol) < U) \{ \)
   
   \(
   U = \text{COST}(\text{currentSol})
   \)
   
   return (currentSol) \} \}
   
   \}}

else return ("no (better) solution") \} \}

2. \(L = \text{LOWER\_BOUND}(M, currentSol)\)

3. if (L ≥ U) return ("no (better) solution")

4. \(x_i = \text{CHOOSE\_VAR}(M)\)

5. \(S^l = \text{BCP}(M_{x_i}, U, currentSol \cup \{x_i\})\)

6. if (COST(S^l) = L) return (S^l)

7. \(S^0 = \text{BCP}(M_{x_i}, U, currentSol)\)

8. return BEST\_SOLUTION \((S^l, S^0)\)

---

**Solved Problem 14: Binary Recursion**

\[
\begin{array}{ccc}
2 & 4 & 5 \\
\times & & \\
1 & 1 & 0 & 1 \\
\Rightarrow & 3 & 1 & 1 \\
2 & 0 & 1 & 1
\end{array}
\]

\[
\begin{array}{ccc}
2 & 4 & 5 \\
\times & & \\
1 & 1 & 0 & 1 & \Rightarrow & 1 & 0 & 1 \\
2 & 1 & 1 & 0 & \Rightarrow & 2 & 1 & 0 \\
3 & 0 & 1 & 1 & \Rightarrow & 3 & 1 & 1
\end{array}
\]

\[
\begin{array}{ccc}
\text{currentSol} = \{2\}, \\
\text{EC} = \{4\}, \\
U = 1 + 1 = 2
\end{array}
\]

\[
\begin{array}{ccc}
\text{currentSol} = \{\}, \\
\text{EC} = \{4,5\}, \\
U = 0 + 2 = 2
\end{array}
\]
13x11 Example

MIS={1,3,5,7}, \ L=0+4=4

忽略了选择最长列的启发式：选择列1

M^1_0 \ \ \text{Drop Col 1}

M^1_1 \ \ \text{Drop Col 1, and}
\ \ \text{Rows 1,4,12}

13x11 Example

Cols 2,4 are dominated, so
Col 3 becomes essential

MIS={7,9}, \ L=2+2 = 4

Pick longest column, Col 5

M^2_0 \ \ \text{Drop Col 5}

M^2_{00} \ \ \text{Drop Col 5, and}
\ \ \text{Rows 5,9-13}
Cyclic (takes 2 Cols),
Cost = 3+2=5=U
\[ p_1, p_3, p_5 \quad p_6, p_7 \]

\( M_{11}^2 \)
\[
\begin{array}{cccc}
1 & 1 & 0 & 1 \\
0 & 1 & 1 & 0 \\
1 & 0 & 1 & 1 \\
\end{array}
\]

\( M_{10}^2 \)
\[
\begin{array}{cccc}
1 & 0 & 0 & 0 \\
1 & 1 & 0 & 0 \\
0 & 1 & 1 & 0 \\
0 & 0 & 1 & 1 \\
0 & 0 & 1 & 1 \\
0 & 1 & 0 & 0 \\
1 & 0 & 1 & 0 \\
1 & 0 & 0 & 1 \\
1 & 0 & 0 & 1 \\
\end{array}
\]

\( S^1 = \text{BCP}(M_{11}^2, U, \text{currentSol} \cup \{x_i\}) \)
\( S^0 = \text{BCP}(M_{10}^2, U, \text{currentSol}) \)

\( S^1 = \text{BCP}(M_{11}^2, U, \text{currentSol}) \)

\( S^0 = \text{BCP}(M_{10}^2, U, \text{currentSol}) \)

\( \text{return} \quad \text{BEST\_SOLUTION} \quad (S^1, S^0) \)

MIS=\{5, 10, 11\}, \quad L=2+3 = 5

\( (M, \text{currentSol}) = \text{REDUCE}(M, \text{currentSol}) \)
\( \text{if} \quad \text{(terminalCase}(M)) \quad \{ \)
\( \quad \text{if} \quad \text{(COST}(\text{currentSol}) < U) \quad \{ \)
\( \quad \quad \text{U} = \text{COST}(\text{currentSol}) \)
\( \quad \quad \text{return} \quad \text{currentSol} \quad \} \)
\( \quad \text{else return} \quad \text{"no (better) solution"} \quad \} \)
\( \quad \text{L} = \text{LOWER\_BOUND}(M, \text{currentSol}) \)
\( \quad \text{if} \quad (L \geq U) \quad \text{return} \quad \text{"no (better) solution"} \)
\( x_i = \text{CHOOSE\_VAR}(M) \)
\( S^1 = \text{BCP}(M_{x_i}, U, \text{currentSol} \cup \{x_i\}) \)
\( S^0 = \text{BCP}(M_{x_i}, U, \text{currentSol}) \)
\( \text{return} \quad \text{BEST\_SOLUTION} \quad (S^1, S^0) \)
Example Summary

5 Recursive Calls
5 Reductions
5 Lower Bounds
3 Splitting Choices
Minimizing Weighted Cost

• To make cost sensitive to transistor count, set the “weight” of a prime implicant to its literal count

\[
\text{COST}(x'y) = 2, \ \text{COST}(uvw'yz') = 6
\]

• Thus a dominated column is only remove if the dominating column has lower (or the same) cost

• The rest of BCP is unaffected
• Theorem 8.2.3, Page 339