Some Problems are Hard
Using Exact Algorithms vs. Heuristics

- Quine-McCluskey
  - Calculated all prime implicants to derive the optimal solution(s)
  - Petrick’s Method derives all covers to determine minimum cover set(s)
  - Number of prime implicants grow quickly -- solution space is huge!
  - Finding the minimum cover set in a class of NP complete problems
    - Determining optimal solution is difficult

- Move to heuristics
  - Look at generating a quality solution quickly (not necessarily optimal)

Local Search

- Don’t generating all prime implicants and minterms
- Instead, ESPRESSO successively modify a given initial cover
  - This technique is called a local search algorithm

- Idea behind local search
  - Search space or solution space - set of all possible values and cost associated with solution
  - Start with an initial value
  - Search all points in neighborhood for a feasible point whose cost is less than current
  - Different problems have different neighborhood definitions
  - If one is found, start process over

- Drawback of local searches is local optimality
  - Solution is locally optimal if its neighborhood does not contain any solutions with a lower cost
  - Locally optimal solution may not be the optimal solution
  - Modify local search so we don’t get stuck at the local minimum
Espresso

- Espresso utilizes local search (keeping in mind local minimum problem)
  - Probably most popular minimization algorithm
  - Extremely efficient Boolean manipulation
- Composed of three main operations
  - EXPAND, REDUCE, IRREDUNDANT
- Other operations include
  - COMPLEMENT, ESSENTIAL PRIMES, LASTGASP, MAKESPARSE

- Espresso Heuristic (in a nutshell)
  - Apply Expand and Irredundant operators to optimize the current function specification
  - Uses the reduce operator to get out of local minimum
  - Iterated until the solution converges

Espresso – Expand Operator Overview

- Goal is to expand a non-prime implicants to prime with the least number of literals

Espresso – Reduce Operator Overview

- REDUCE
  - Adding one or more literals
  - Check for validity

Expand abc by removing c (results in ab)
Is it valid? Yes.

Expand abc by removing a (results in bc)
Is it valid? No.

Reduce a' by adding b' (results in a'b')
Is it valid? Yes.

Reduce a' by adding c (results in ac)
Is it valid? Yes.
Espresso – Reduce Operator Overview

- Goal is to decrease the size of implicants such that expansion may lead to a better solution
- Avoiding a local minimum

Expand $x'y'$ to $yz'$
Is it valid? Yes.

Reduce $x'y$ to $x'y'z'$
No implicant can be expanded
Is it valid? Yes.

Reduce $x'z$ to $x'yz'$
Is it valid? Yes.

Expand $x'y'$ to $yz'$
Is it valid? Yes.

F = $x'z + yz' + xy'$
Reduction helped find a better solution!

Espresso – Irredundant Operator Overview

- **IRREDUNDANT**
  - Implicant in a cover is redundant if all the minterms covered by it are contained in other implicants in the cover

yz' is redundant
$x'y$ and $xz'$ cover all minterms contained in $yz'$

Espresso – Additional Concerns

- **Additional concerns**
  - Validity check operations
  - Which direction should the move make?

Which way should we expand?
Which implicant should we reduce?
Which literal should we add?
Espresso

espresso(F,D) {
  R = complement(F U D);
  F = expand(F,R);
  // initial expansion
  F = irredundant(F,D);
  // initial irredundant cover
  E = essentials(F,D);
  // detect essential prime implicants
  F = F – E;
  // remove essential prime implicants from F
  D = D U E;
  // add essential prime implicants to D
  repeat {
    φ₁ = |F |;
    F = reduce(F,D);
    F = expand(F,R);
    F = irredundant(F,D);
  } until (|F | ≥ φ₁);
  F = F U E;
  D = D – E;
  RETURN F;
}

F is the on-set, D is the don't care set

repeated application of REDUCE, EXPAND, IRREDUNDANT operations while cost keeps decreasing

ESPRESSO, to be continued...

- We’ve seen the high-level idea behind ESPRESSO
  - ESPRESSO performs extremely efficient Boolean manipulation
- How are these operations actually performed?
- How is data represented?