Effective Bounding Techniques For Solving Unate and Binate Covering Problems

Xiao Yu Li
amazon.com.
Seattle, WA, USA

Matthias F. Stallmann
NC STATE UNIVERSITY
Raleigh, NC, USA

Franc Brglez
NC STATE UNIVERSITY
Raleigh, NC, USA

Related Work

- Unate (Set) Cover Problem (UCP)

  1987: *espresso*, Rudell et al

  1993: *espresso-signature*, McGeer at al

  ….

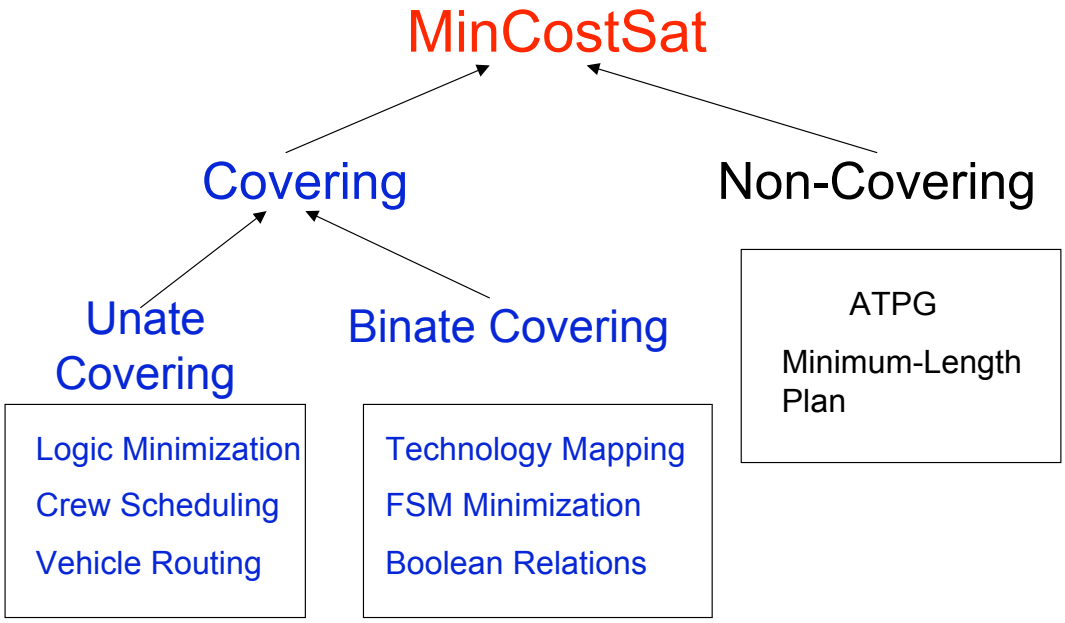
- Binate Cover Problem (BCP)

  1996: *scherzo*, Coudert

  2002: *bsolo*, Manquinho et al

  …

Native MinCostSat Problems



Outline

1. Background
2. Our UCP/BCP Solver
3. Experimental Results
4. Conclusions

Unate (Set) Covering

Three backup positions to fill.

Five players to choose from.

Want to minimize the number of players.





	Duncan	Francis	Malone	Stockton	Yao
Guard		1		1	
Forward	1		1		
Center	1		1		1

Unate Covering as CNF Sat

- Conjunctive Normal Form

Guard   F ∨ S

Forward   D ∨ M

Center    D ∨ M ∨ Y

- Goal: minimize F + S + D + M + Y

Min-cost solutions: F =1, D =1, S = M = Y = 0

M =1, S =1, D = F = Y = 0

Binate Covering Problem

Additional constraint 1:
Malone and Stockton
are together ($M \Leftrightarrow S$)

$\overline{M} \vee S$

$M \vee \overline{S}$

Additional constraint 2:
Duncan and Stockton
can't be signed together ($\overline{D \wedge S}$)

$\overline{D} \vee \overline{S}$

Binate Covering Problem (cont.)

constraints

$F \vee S$

$D \vee M$

$D \vee M \vee Y$

$\overline{M} \vee S$

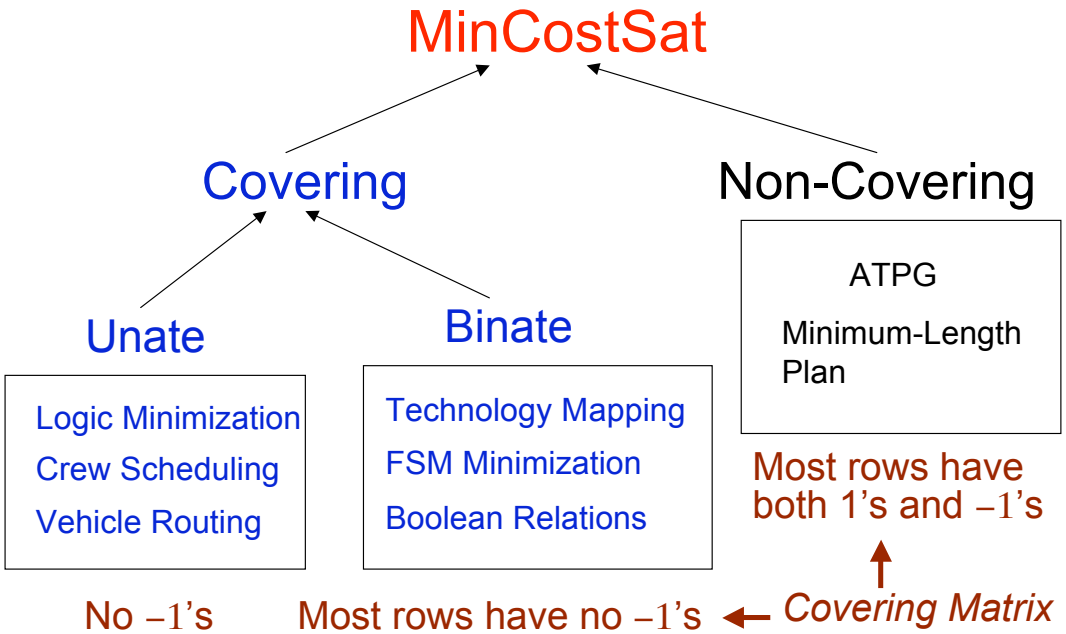
$M \vee \overline{S}$

$\overline{D} \vee \overline{S}$

covering matrix

D	F	M	S	Y
	1		1	
1		1		
1		1		1
		-1	1	
		1	-1	
-1			-1	

MinCostSat Problems Revisited



Solve MinCostSat as ILP

MinCostSat is a special case of 0-1 integer linear programming

For each constraint in the MinCostSat instance...

$$\overline{x_1} \vee x_2 \vee \overline{x_3} \vee x_4$$

...replace $\overline{x_i}$ with $(1 - x_i)$:

$$(1 - x_1) + x_2 + (1 - x_3) + x_4 \geq 1$$

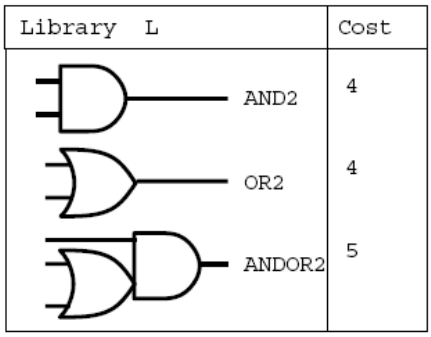
$$-x_1 + x_2 - x_3 + x_4 \geq -1$$

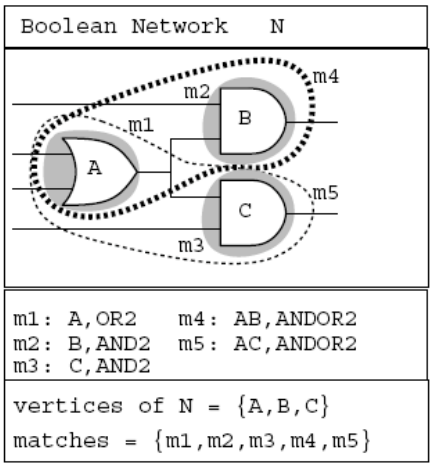
MinCostSat as ILP: an example

MinCostSat		ILP
$\overline{x1} \vee x2 \vee \overline{x3} \vee x4$		$-x1 + x2 - x3 + x4 \geq -1$
$\overline{x1} \vee \overline{x2} \vee \overline{x4}$	$\Rightarrow$	$-x1 - x2 + x4 \geq -2$
$x1 \vee x2 \vee x3$		$x1 + x2 + x3 \geq 1$

Assume unit cost, minimize: $x1 + x2 + x3 + x4$

Technology Mapping





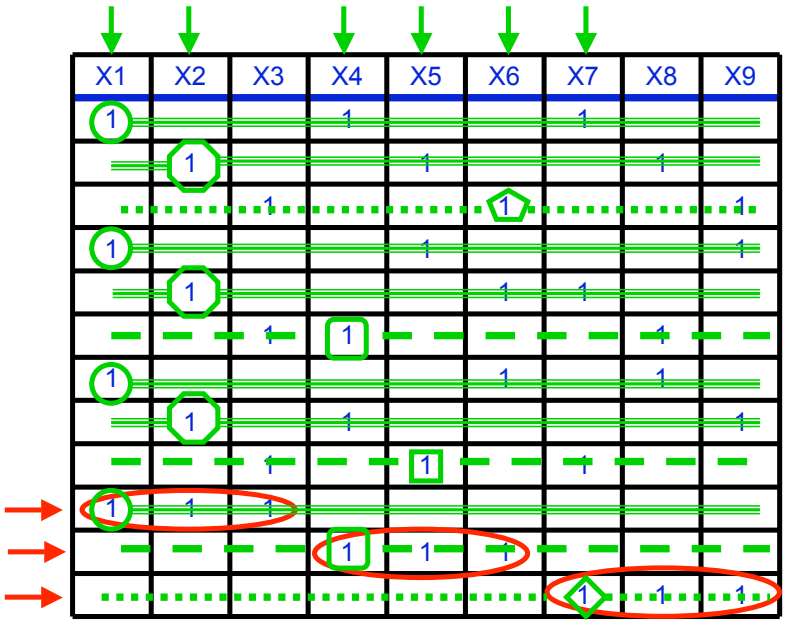
Cover A	$m1 \vee m4 \vee m5$
Cover B	$m2 \vee m4$
Cover C	$m3 \vee m5$
$m2 \rightarrow m1$	$\overline{m2} \vee m1$
$m3 \rightarrow m1$	$\overline{m3} \vee m1$

Our UCP/BCP Solver

Classical branch and bound

1. **Upper/lower bound** calculation (for a unate example)
2. **Branching** (variable assignment)
3. One iteration of the main algorithm
4. Effect of better (global) upper bounds and (local) lower bounds.

An Example (Steiner 9)



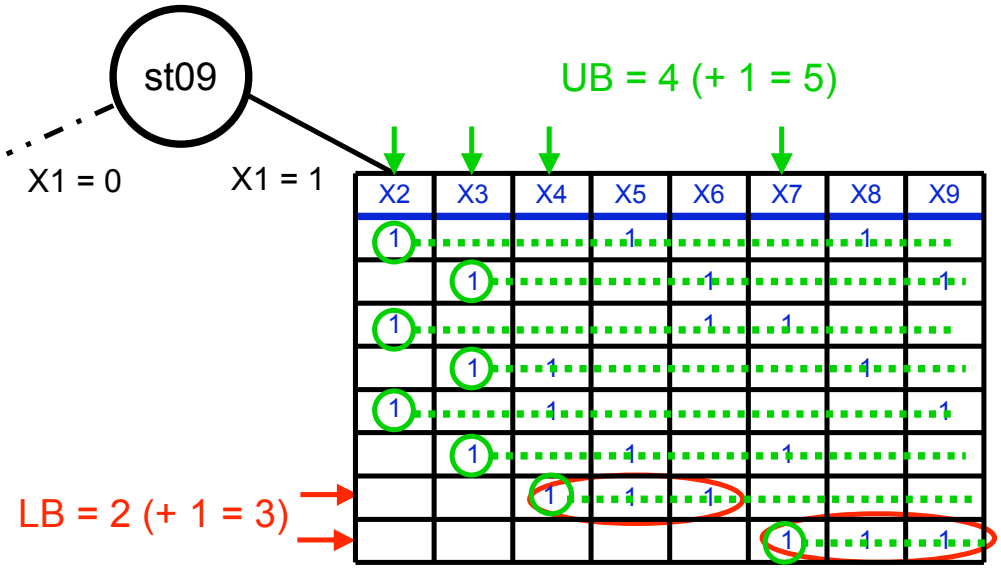
UB = 6

find "cover" to get UB

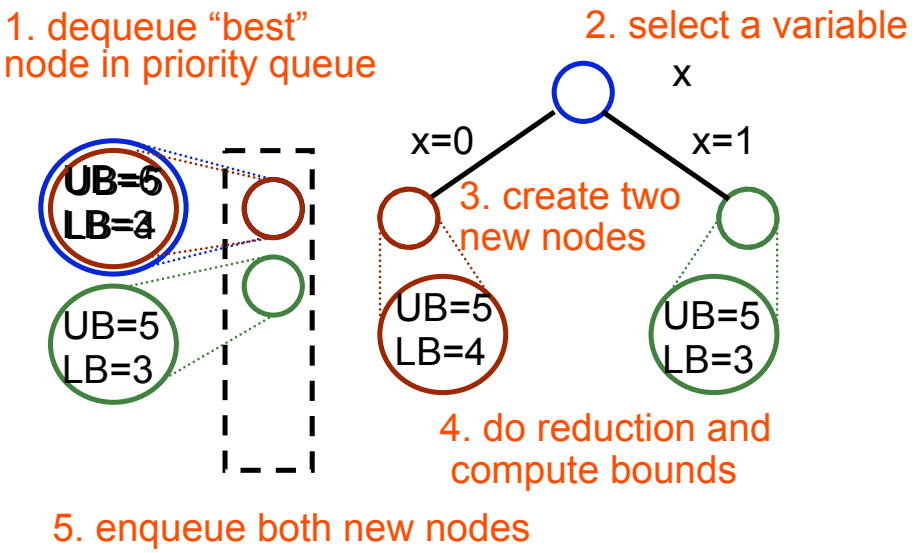
LB = 3

based on max. ind. set (MIS)

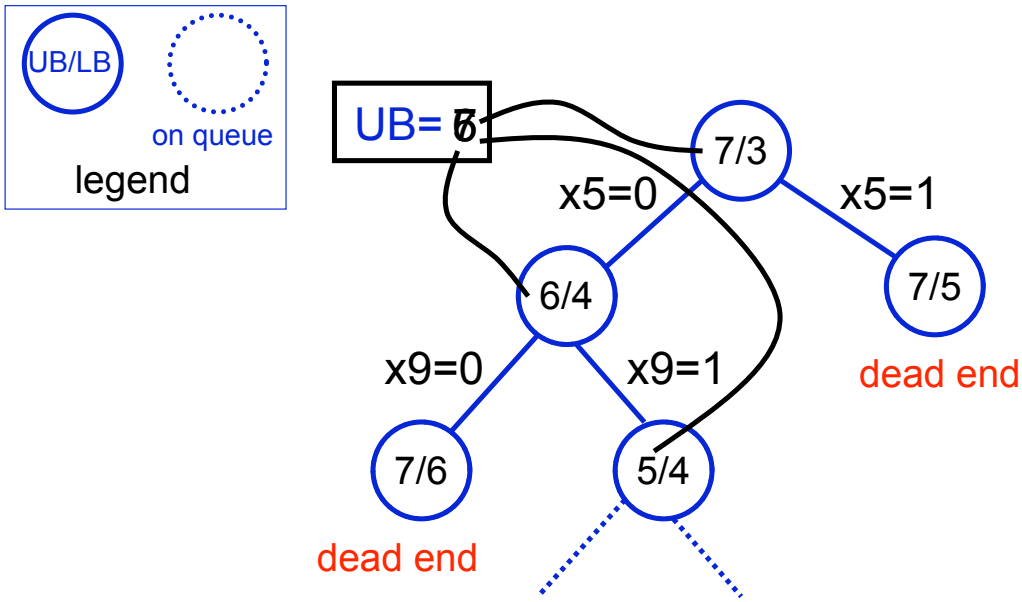
After assignment of a variable



The algorithm: one iteration



Lower and Upper Bounds



Branch-and-bound solver: eclipse

The algorithm

```
Initialize root; Q.enqueue(root); globalUB=UB(root)
while ( Q is not empty ) {
   node = Q.dequeue();
   if ( LB(node) < globalUB ) {
      select branching variable x_i
      for ( b = 0 to 1 ) {
         n_b = (node for x_i = b)
         do reduction and bounds for n_b
         if ( UB(n_b) < globalUB ) { globalUB = UB(n_b) }
         if ( LB(n_b) < globalUB ) { Q.enqueue( n_b) }
 }}}
```

Eclipse Performance Factors

- Seven Performance Factors
  1. Lower Bounding (ILP techniques)
  2. Upper Bounding (stochastic search)
  3. Search-Tree Exploration Strategies
  4. Branching Variable Selection
  5. Search Pruning
  6. Reductions
  7. Data Structures

Factor 1: Lower Bounding

Three lower-bounding strategies:

- Maximum Independent Set (MIS)
- Linear Programming Relaxation (LPR)
- Cutting Planes (CP)

Lower Bounding: MIS

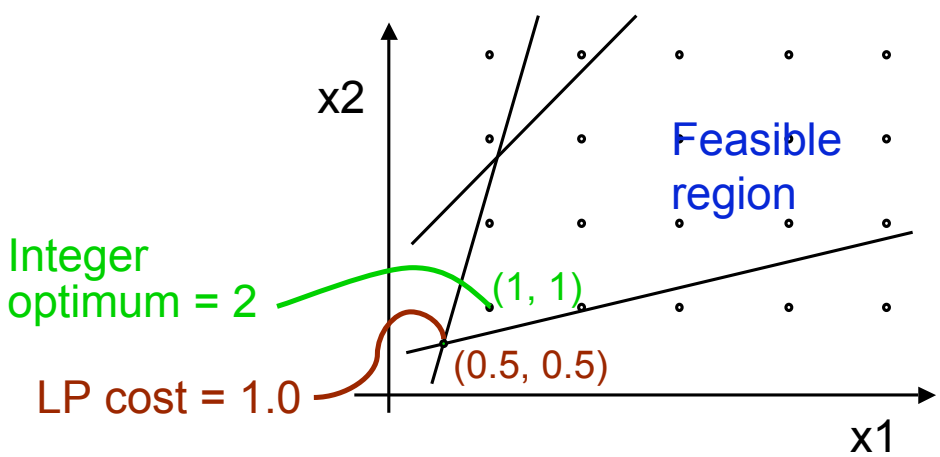
- MIS - maximum independent set of rows

	x1	x2	x3	x4	x5	x6
row1	1	1				
row2			1	1	1	
row3			1	1		
row4					1	1
row5			-1		-1	1

- Row 1, 2 form an independent set $\Rightarrow$ LB = 2
- Row 1, 3, 4 form an independent set $\Rightarrow$ LB = 3

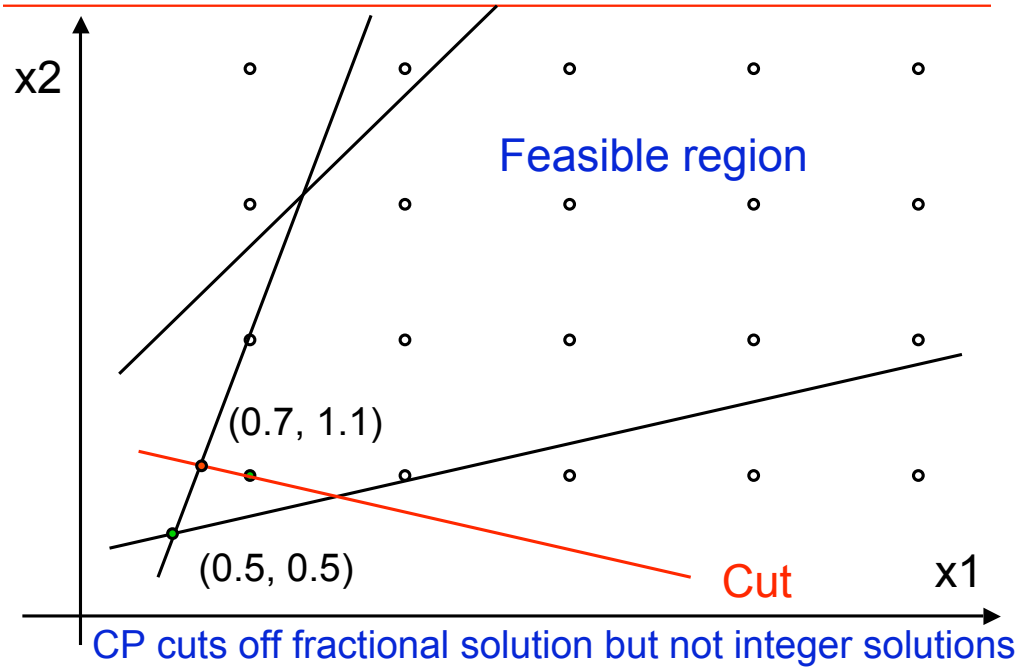
Lower Bounding: LP Relaxation

Relax integer constraints (LPR):



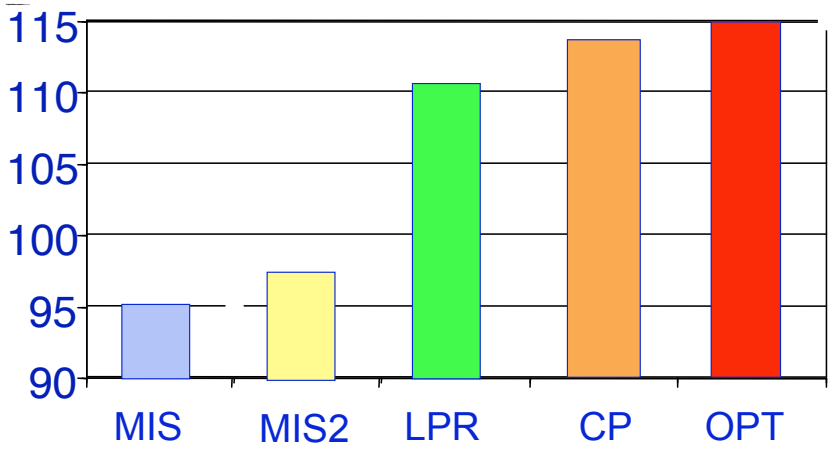
LP optimum is a lower bound on integer optimum

Lower Bounding: Cutting Planes (CP)



CP cuts off fractional solution but not integer solutions

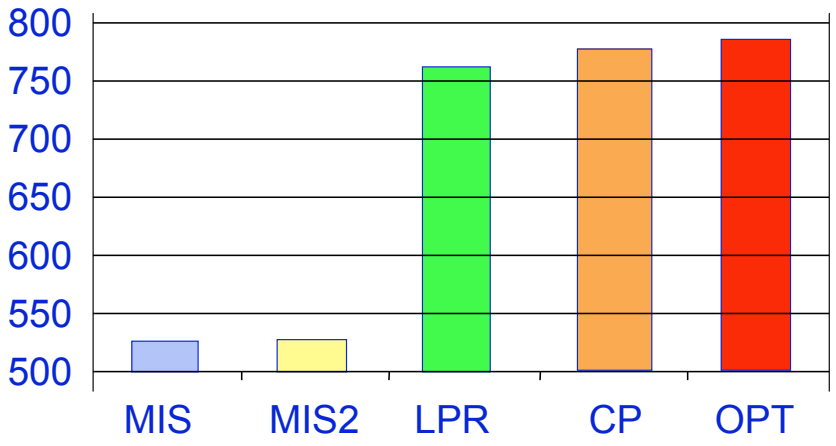
Lower Bound Methods (rot.b)



(MIS2 is MIS with extra tricks)

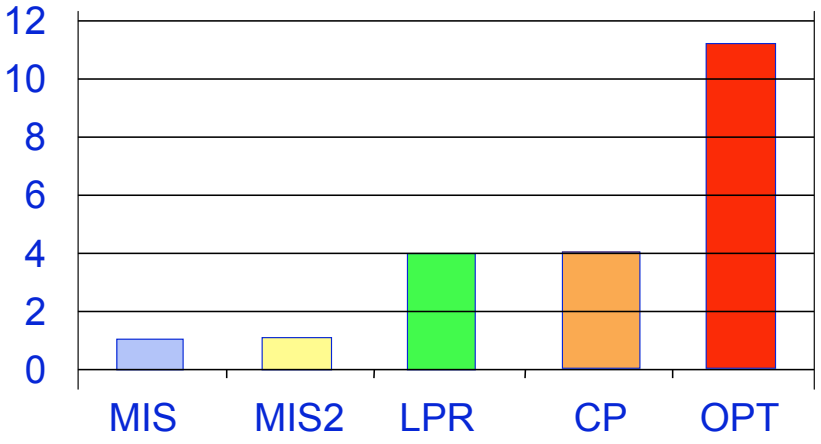
Most effective by far: CP

Lower Bound Methods (apex4.a)



Most effective: CP

Lower Bound Methods (c1908_F)



no bounding method appears effective here …

(not a case of covering problem instance)

Next Performance Factor:

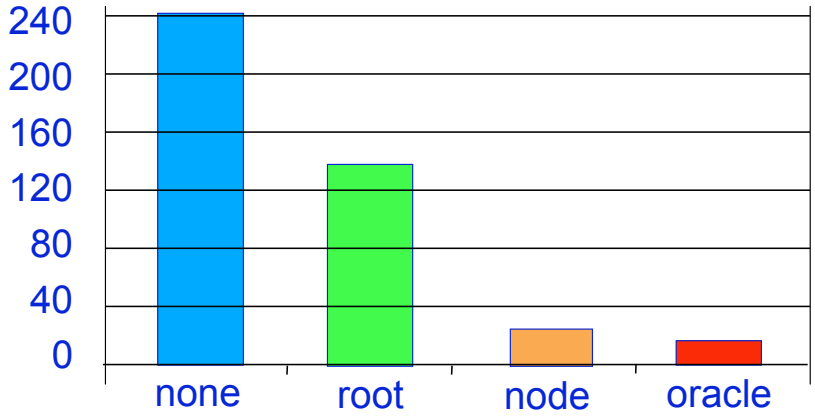
1. Lower Bounding (ILP techniques)
2. **Upper Bounding (stochastic search)**
3. Search-Tree Exploration Strategies
4. Branching Variable Selection
5. Search Pruning
6. Reductions
7. Data Structures

Factor 2: Upper Bounding

- Stochastic local search for optimal solution:
  1. Apply at the root only or **at each node**
  2. Initialization – solution based on linear programming for lower bound
- For comparison purposes we initialize the global UB to the cost of the optimal solution (oracle)

Upper Bound Methods (max1024)

we report average cpu-seconds to solve the instance with different UB strategies using the same LB (the best possible, i.e. CP)



Local Search Initialization

… random initialization of local search is not effective, see the table of runtimes (in cpu-seconds):

benchmark	random	rounded LP solution
ex5	16.3	15.1
exam.pi	20.8	5.4
max1024	143.9	27.5
bench1.pi	300**	4.7

**timeout value

Factor 3: Tree-Exploration Strategy

… search using priority queue performs better than depth-first search (recursion - the usual method):

(runtimes in cpu-seconds)

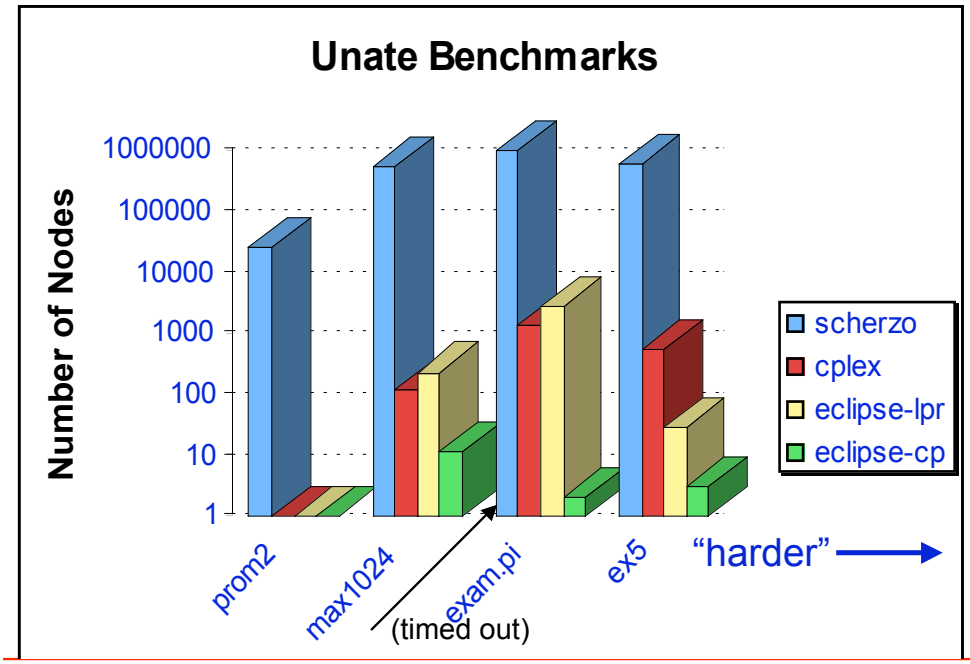
benchmark	depth-first	priority*
exam.pi	7.0	5.4
bench1.pi	7.9	4.7
ex5	16.2	15.1
max1024	74.1	27.5

* smallest lower bound first (to try to reduce UB quickly)

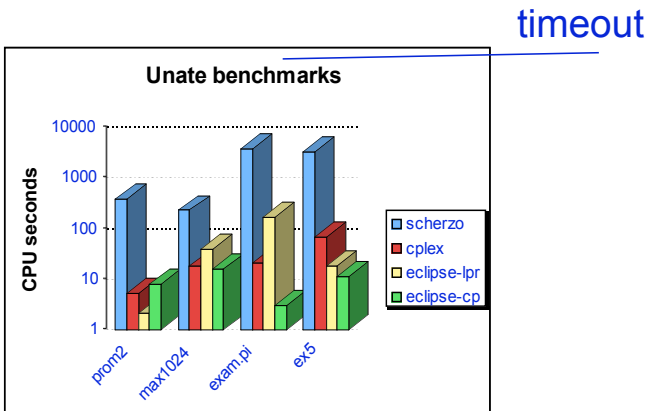
Experimental Results

- The solvers
  1. Scherzo [Coudert, DAC1996]
  2. Cplex (State of the art IP/ILP Solver)
  3. Eclipse-lpr (LP Relaxation)
  4. Eclipse-cp (Cutting Planes)
- The benchmarks
  - Logic minimization (unate covering)
  - FSM minimization (binate covering)

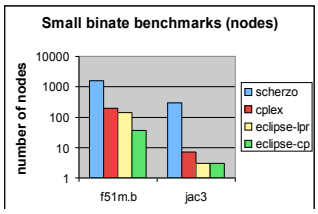
Unate Results (nodes visited)



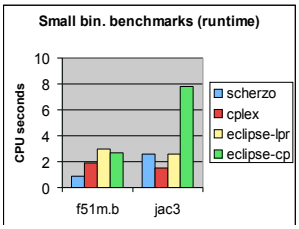
Typical Results: Unate (runtime)



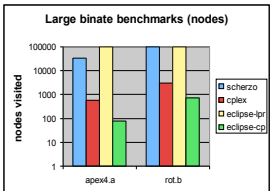
Typical Results: small binate



eclipse-cp spends a lot of time with cuts at each node (cplex visits more nodes and does cuts less frequently)



Large binate benchmarks

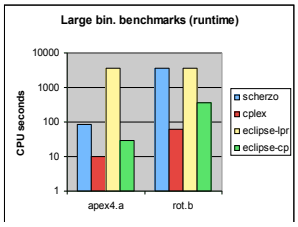


CPLEX only does cuts at some nodes.

We visit fewer nodes...

... but spend more time

We do cuts everywhere
-- could be more judicious



latte = espresso – mincov + eclipse-cp

	espresso (hours)		latte (secs)	
benchmark	**cover**	**total**	**cover**	**total**
prom2	---	12**	12.9	14.7
ex5	---	12**	129.5	139.2
max1024	---	12**	329.1	329.5

**timeout

benchmark	prod-orig	prod-heur	prod-latte
prom2	287	287	287
ex5	256	74	65
max1024	1024	274	259

The Last Words …

Acknowledgments:

- for the benchmarks
- for the solvers (*aura, bsolo, espresso, scherzo*)
- for constructive reviews

Under construction:

- code tune-ups before the release of *eclipse*
- additional experiments
  - -- for the journal submission
  - -- for comprehensive web-posting of results