

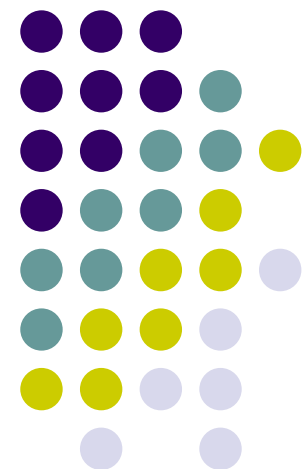
# Optimization and Relaxation in SAT Search

---

Sharad Malik  
Princeton University

Symposium on Satisfiability Solvers and Program  
Verification (SSPV)

Seattle  
August 11, 2006



Joint work with Zhaohui Fu

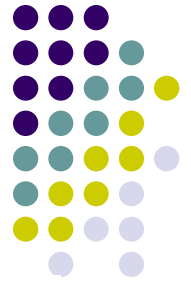
# What excites us...

## A new breed of fast SAT solvers

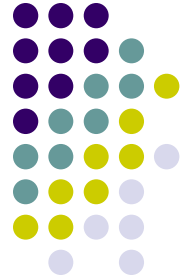


and others...

# This talk: Some other models in the SAT mall...



# MinCostSAT



 <b>ANCIRA KIA</b>			
6125 Bandera Rd. San Antonio, Tx 78238 (210) 681-2300 (877) 669-2300			
			
<b>SEDONA</b> STARTING AT \$21,620* EVERYTHING YOU CAN EXPECT IN A MINIVAN	<b>SORENTO</b> STARTING AT \$19,665* A MID-SIZE SUV WITH ALL THE QUALITIES YOU WANT	<b>SPORTAGE</b> STARTING AT \$16,490* COMPLETELY REDESIGNED AND READY TO IMPRESS	<b>AMANTI</b> STARTING AT \$26,140* A BIG SEDAN YOU'LL BE PROUD TO BE SEEN IN
			
<b>SPECTRA • SPECTRA 5</b> STARTING AT \$14,940* NEW AND READY FOR ADVENTURE, CARS YOU CAN AFFORD TO LOVE!		<b>OPTIMA</b> STARTING AT \$16,990* STEP UP TO A LUXURY MID-SIZE SEDAN	<b>RIO • RIO 5</b> STARTING AT \$11,110* STILL THE MOST AFFORDABLE 4-DOOR SEDAN IN AMERICA
CLICK ON ANY OF THE ABOVE VEHICLES FOR FURTHER INFORMATION.			
<b>*ALL PRICES SHOWN ARE MSRP. PLEASE CALL TO RECIEVE ANCIRA'S LOW DISCOUNTED PRICE.</b>			



# Problem Definition

- MinCostSAT: Given a Boolean formula  $\varphi$  with
  - $n$  variables  $x_1, x_2, \dots, x_n$   $x_i \in \{0, 1\}$   
each costs  $c_i \geq 0$   $1 \leq i \leq n$
- Find a variable assignment  $X \in \{0, 1\}^n$ 
  - $X$  satisfies  $\varphi$
  - $X$  minimizes

$$C = \sum_{i=1}^n c_i x_i$$



# Related Problems

- Min-One/Max-One Problem: positive/negative unit variable cost
- Binate Covering: synonym of MinCostSAT
- Unate Covering: single phase variables
- Problems easily translatable to MinCostSAT:
  - Partial MAX-SAT: some of the clauses are relaxable
  - MAX-SAT: all clauses are relaxable.



# Motivation

- MinCostSAT has various applications, e.g. Automatic Test Pattern Generation (ATPG), FPGA Routing, AI Planning, etc.
- (Applications in program verification:
  - Biased counterexamples
  - Shortest counterexamples)
- Best branch-and-bound solver *bsolo* is based on old SAT solver GRASP.
  - SAT techniques have advanced dramatically since then:
    - Two literal watching based fast BCP.
    - Better decision heuristics, e.g. VSIDS, Berkmin, Seige, MiniSAT

# Previous Work

Mathematical  
Optimization  
e.g. cplex



Classic Covering Algorithms  
using Branch and Bound Search

SAT Based Algorithms

MIS based  
lower bounds  
e.g. scherzo

Non-MIS based  
lower bounds,  
e.g. lp relaxation

Branch and Bound  
Search, e.g. bsolo  
(MIS based lower  
bounds)

Encoding based  
Search,  
e.g. opbdp,  
miniSAT+



# Previous Work

Mathematical  
Optimization  
e.g. cplex

**Classic Covering Algorithms  
using Branch and Bound Search**

**SAT Based Algorithms**

**MIS based  
lower bounds  
e.g. scherzo**

**Non-MIS based  
lower bounds,  
e.g. lpr relaxation**

**Branch and Bound  
Search, e.g. bsolo  
(MIS based lower  
bounds)**

**Encoding based  
Search,  
e.g. opbdp,  
miniSAT+**

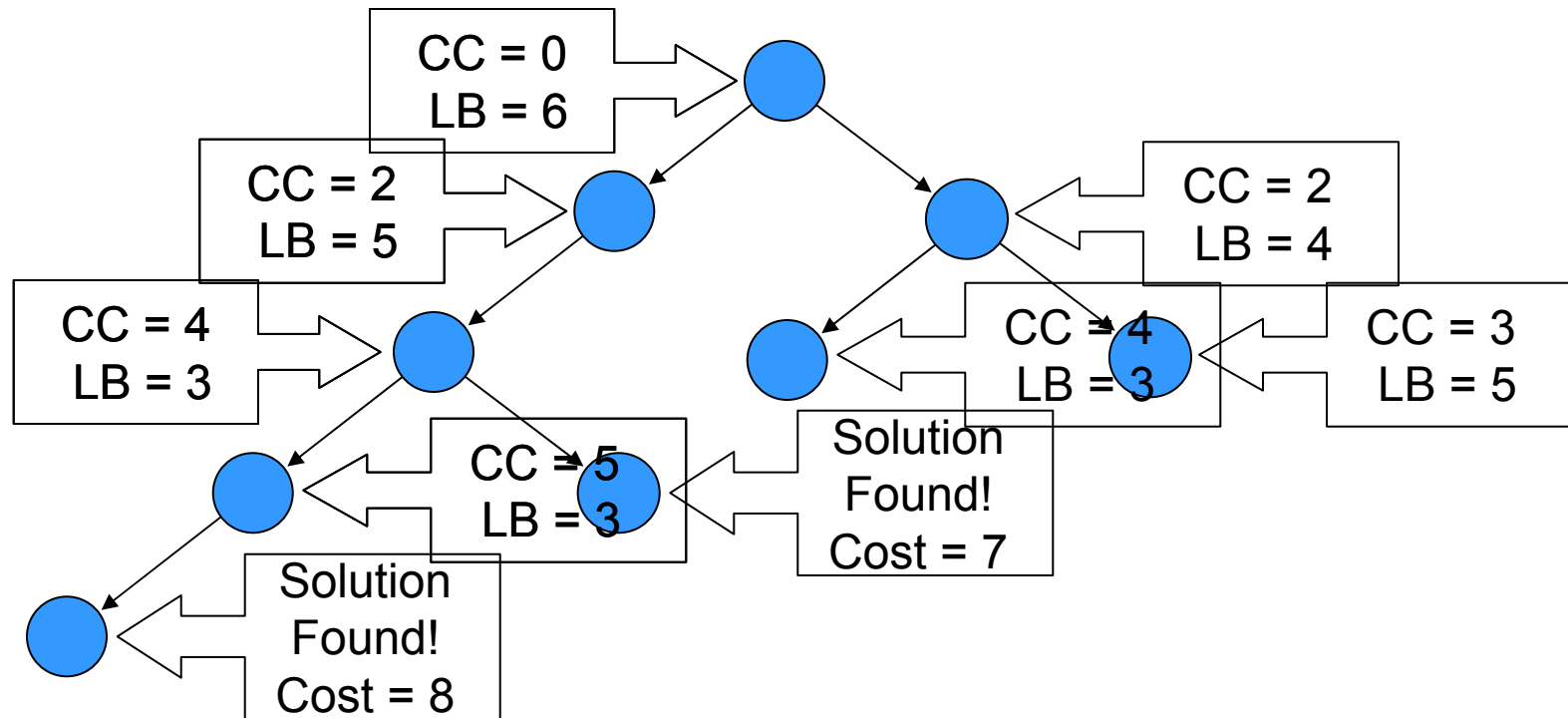


# Classic Covering Algorithms

- The earliest important forms of MinCostSAT are the Unate/Binate Covering Problems.

	<i>x</i>	<i>y</i>	<i>z</i>	<i>a</i>	<i>b</i>	<i>c</i>
<i>x+y</i>	1	1				
<i>x+z</i>	1		1			
<i>y+a</i>		1		1		
<i>z+c</i>			1			1
<i>a+b</i>				1	1	
<i>b+c</i>					1	1
<i>c</i>						1
<i>b</i>					1	

# Branch-and-Bound Search



**Solution = 7** ← **Optimal!**

CC: Current Cost  
LB: Lower Bound



# Previous Work

Mathematical  
Optimization  
e.g. cplex

Classic Covering Algorithms  
using Branch and Bound Search

SAT Based Algorithms

**MIS based  
lower bounds  
e.g. scherzo**

Non-MIS based  
lower bounds,  
e.g. lpr relaxation

Branch and Bound  
Search, e.g. bsolo  
(MIS based lower  
bounds)

Encoding based  
Search,  
e.g. opbdp,  
miniSAT+

# Maximum Independent Set (MIS) Based Lower Bounding Functions



	$x$	$y$	$z$	$a$	$b$	$c$
$x+y$	$1$	$1$				
$x+z$	$1$		$1$			
$y+a$		$1$		$1$		
$z+c$			$1$			$1$
$a+b$				$1$	$1$	
$b+c$					$1$	$1$
$c$						$1$
$b$					$1$	

Size of the MIS = 4

# Previous Work

Mathematical  
Optimization  
e.g. cplex



Classic Covering Algorithms  
using Branch and Bound Search

SAT Based Algorithms

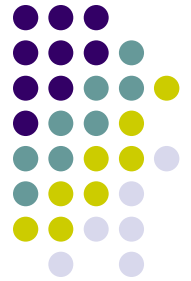
MIS based  
lower bounds  
e.g. scherzo

**Non-MIS based  
lower bounds,  
e.g. lpr relaxation**

Branch and Bound  
Search, e.g. bsolo  
(MIS based lower  
bounds)

Encoding based  
Search,  
e.g. opbdp,  
miniSAT+

# Non-MIS Based Lower Bounding Functions



- LP Relaxation:

Minimize:

$$C = \sum_{i=1}^n c_i x_i$$

positive phase literal

negative phase literal

Subject to:

$$x_i + (1-x_j) + \dots > 0$$

(for each clause)  
for all  $i$ ,  $0 \leq x_i \leq 1$

- Objective function minimization returns a lower bound for the original problem
- Better bounds than MIS, but more expensive computation



# Previous Work

Mathematical  
Optimization  
e.g. cplex

Classic Covering Algorithms  
using Branch and Bound Search

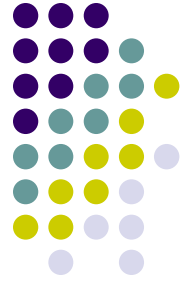
SAT Based Algorithms

MIS based  
lower bounds  
e.g. scherzo

Non-MIS based  
lower bounds,  
e.g. lpr relaxation

**Branch and Bound  
Search, e.g. bsolo  
(MIS based lower  
bounds)**

Encoding based  
Search,  
e.g. opbdp,  
miniSAT+



# SAT Based Algorithms

- SAT Based Branch-and-Bound Search
  - bsolo implemented on top of GRASP
- MIS Based Lower Bounding Functions
  - $\dots(x'_2+x_4)(x_1+x_2+x_3)(x_4+x_5)(x_5+x_6+x'_7)\dots$
  - $(x_1+x_2+x_3)(x_4+x_5)$  are independent.
  - At least 2 variables must be 1
  - MIS is computed dynamically, i.e. every time a decision is made.

# Previous Work

Mathematical  
Optimization  
e.g. cplex



Classic Covering Algorithms  
using Branch and Bound Search

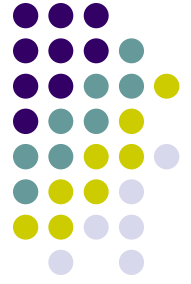
SAT Based Algorithms

MIS based  
lower bounds  
e.g. scherzo

Non-MIS based  
lower bounds,  
e.g. lpr relaxation

Branch and Bound  
Search, e.g. bsolo  
(MIS based lower  
bounds)

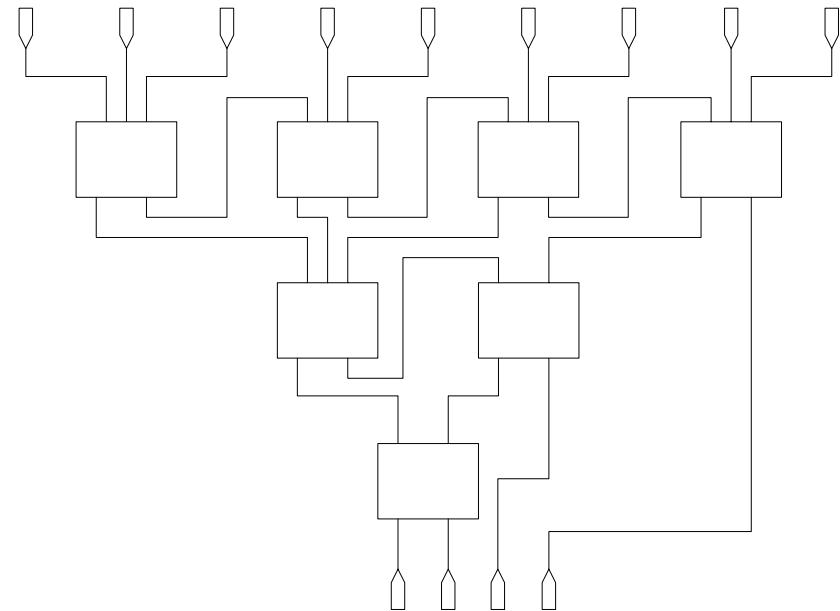
Encoding based  
Search,  
e.g. opbdp,  
miniSAT+

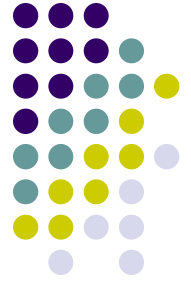


# Encoding Based Solvers

- Used in Pseudo-Boolean (PB) Solvers (0-1 ILP)
  - Linear objective function
  - Linear constraints
  - minCostSAT special case of PB solvers
- Linear Search with auxiliary adders: opbdp
  - Encode decision version of objective function using adders and comparators
- MiniSat+:
  - Converting pseudo-Boolean (PB) constraint to SAT clauses through BDDs.
  - PB constraint to a network of adders.
  - PB constraint to a network of sorters.
- Challenging with large  $n$ , coefficients

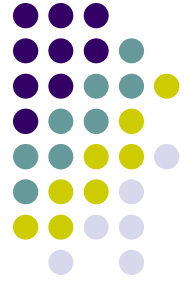
$$C = \sum_{i=1}^n c_i x_i$$





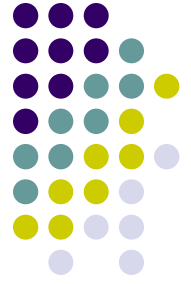
# MinCost-Chaff

- **MIS Based Lower Bounding Function**
  - Pre-computable static MIS
  - Dynamically maintained
- **Compact Blocking Clause**
  - Several effective techniques adopted from bsolo
- **SAT Optimization Techniques**
  - Branch variable selection
  - No expensive simplifications



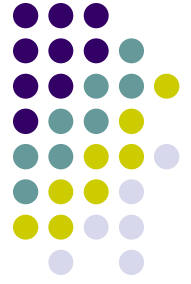
# MIS Based Lower Bound

- Why not LP?
  - MIS is simple
  - Large SAT instances more constrained than general LP ones
- Challenges
  - With two literal watching BCP no easy way to determine unresolved clauses
  - Need efficiency: LB is computed each time a decision is made.



# Pre-computed Static MIS

- An MIS of clauses is selected before the search.
  - Intuition
    - Larger learned clauses less likely to contribute to MIS
    - Smaller sized original clauses still likely to contribute a substantial MIS
- LB computation only considers the clauses in this MIS.
- The status of each clause in the MIS is dynamically checked against the current variable assignment.



# MIS Construction

Cost:

- $c_1 = 1$
- $c_2 = 2$
- $c_3 = 1$
- $c_4 = 6$
- $c_5 = 3$
- $c_6 = 4$
- $c_7 = 9$
- $c_8 = 7$
- $c_9 = 5$

Clauses

Expect Cost

MIS

$$(x_1 + x_2 + x_3) = 4/3$$

$$(x_1 + x'_2 + x_6 + x_8) = 12/4$$

$$(x_3 + x_9) = 6/2$$

$$(x'_3 + x_4 + x_5 + x_6) = 13/4$$

$$(x'_5 + x_7 + x_8) = 16/3$$

$$(x_6 + x'_7) = 4/2$$

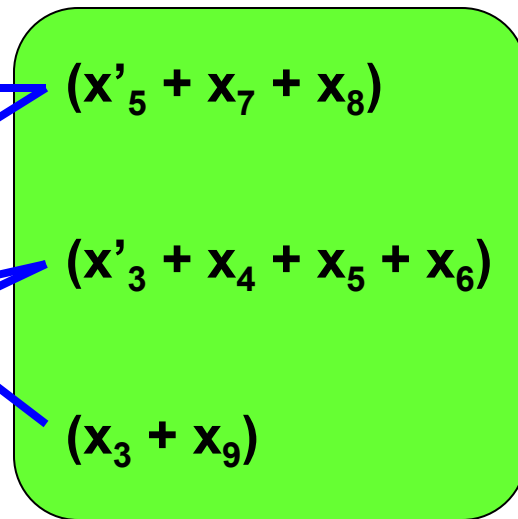
$$(x_7 + x'_9) = 9/2$$

$$(x'_8 + x_5) = 3/2$$

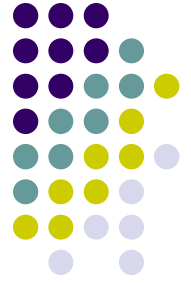
$$(x'_5 + x_7 + x_8)$$

$$(x'_3 + x_4 + x_5 + x_6)$$

$$(x_3 + x_9)$$



# Computing Lower Bound using Pre-computed MIS



Cost:

$$c_1 = 1$$

$$c_2 = 2$$

$$c_3 = 1$$

$$c_4 = 6$$

$$c_5 = 3$$

$$c_6 = 4$$

$$c_7 = 9$$

$$c_8 = 7$$

$$c_9 = 5$$

Assignment:

$$x_2 = 1$$

$$x_4 = 0$$

$$x_5 = 1$$

$$x_8 = 0$$

MIS

$$(x'_5 + x_7 + x_8)$$

$$(x'_3 + x_4 + x_5 + x_6)$$

$$(x_3 + x_9)$$

Total Cost:

9

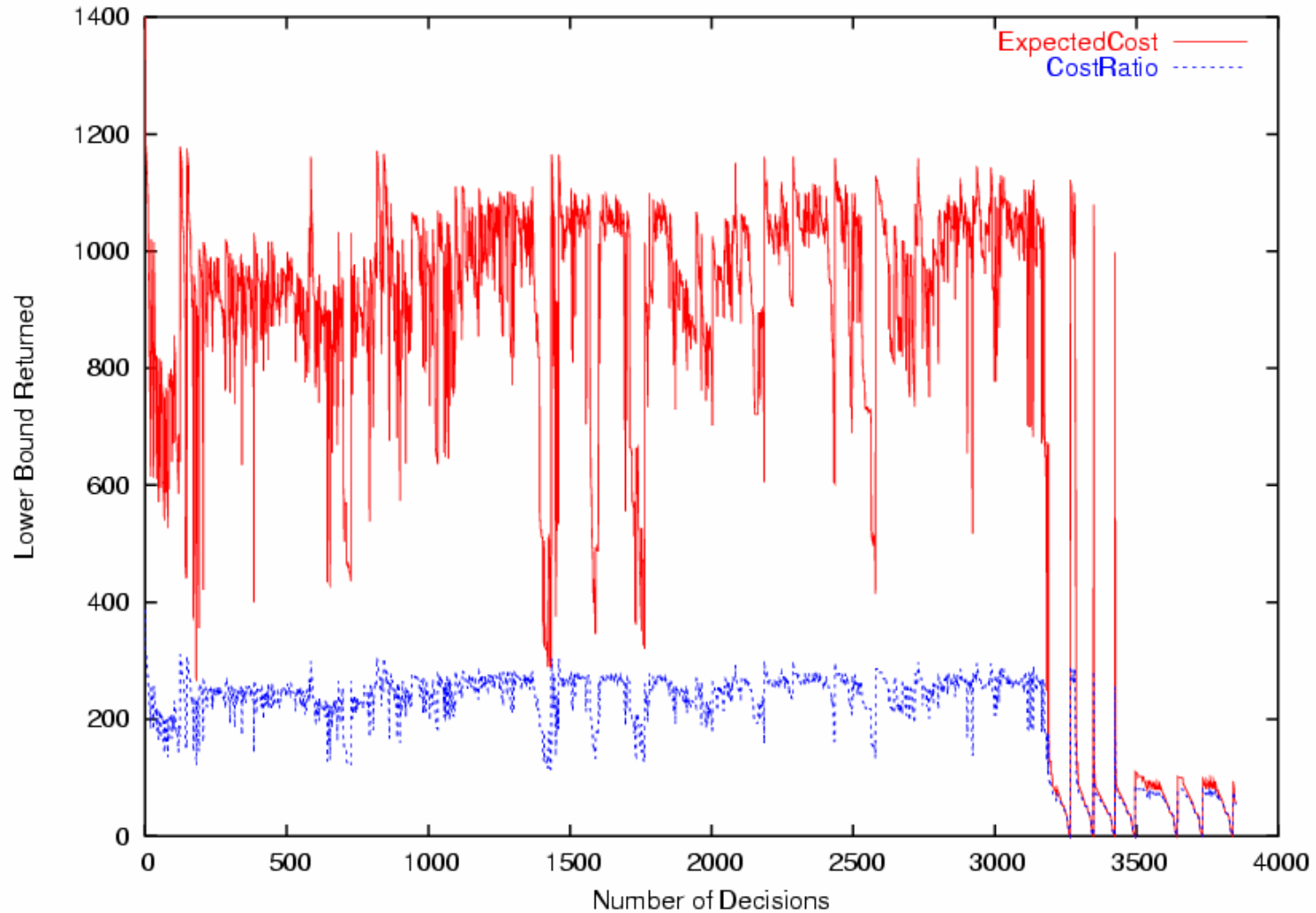
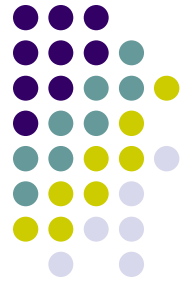
0

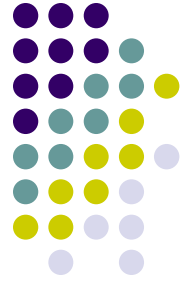
1

$\Sigma$

10

# Performance of the Static MIS During Search





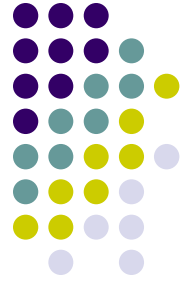
# Additional Optimizations

- Improve the lower bounding with multiple static MISs
- No lower bounding function at all?
- Compute lower bound adaptively
- Compact Blocking Clause
- Branch Variable Selection
- No Expensive Simplifications



# Experiments

Category	Benchmark	Num. Var.	Num. Cls.	Min. Cost	Cnt. BCP	2 Literal Watching Based BCP				Other Solvers			
					Dynamic	Dyna.	Static	No LB	Adapt.	MiniSat+	scherzo	bsolo	cplex
Minimum Size Test Pattern Generation	c1908_F469	2826	12735	11	2.59	3.68	0.35	0.31	0.32	0.53	-	32.10	3182.86
	c1908_F953	3540	17796	4	8.01	11.82	0.90	0.65	0.70	0.63	-	32.36	208.58
	c3540_F20	7600	41312	6	48.40	58.19	3.75	4.50	4.79	2.49	-	413.26	6*
	c3540_F45	6000	29076	9	43.35	34.27	5.93	5.75	5.75	2.62	-	3428.20	9*
	c432_F37	966	5054	9	7.32	8.25	1.09	1.04	1.02	0.21	-	29.55	47.95
	c5315_F54	3778	16000	5	21.08	48.29	1.33	0.44	0.61	0.61	-	27.84	4.55
	c6288_F35	6791	30396	4	5.31	4.53	0.27	0.26	0.27	0.97	2751.05	69.96	5*
	c6288_F69	7539	36257	6	6.13	12.97	1.01	0.98	0.98	2.00	-	606.60	10*
	9symm1_F6	735	4048	9	0.12	0.14	0.06	0.06	0.06	0.14	-	1.35	74.95
	alu4_Fj	2881	16732	6	0.88	1.29	0.21	0.20	0.21	0.71	95.21*	10.10	2066.50
	alu4_Fl	2908	16980	6	0.90	1.76	0.33	0.31	0.32	0.68	180.37*	8.18	1068.59
	duke2_Fv5	1836	10162	5	0.71	1.71	0.11	0.09	0.09	0.39	-	6.67	157.87
	misex3_Fa0	2051	10088	9	0.61	1.12	0.23	0.22	0.23	0.36	-	7.62	664.52
	misex3_Fb1	2348	12574	8	1.30	2.03	0.46	0.45	0.45	0.48	476.47	13.87	1611.92
spla_Fv14	2168	10687	8	0.48	0.74	0.10	0.10	0.10	0.36	171.81	7.77	907.18	
BMC	2bitcomp-5	125	310	39	25.43	3.36	1.91	19.81	1.92	0.54	22.54	26.53	1.70
	bmc-ibm-2	2810	11683	940	1136.86	155.80	97.22	940*	97.71	210.50	-	20.85	7.51
	bmc-ibm-3	14930	72106	6356	211.45	147.30	1.96	6365*	1.98	6373*	-	76.53	-
Coloring	3col120_5	240	1026	110	24.31	22.02	8.13	8.07	8.13	2.53	-	110*	241.97
	3col140_5	280	1196	124	155.77	53.41	19.97	22.03	19.96	4.46	-	124*	579.73
	3col160_5	320	1366	139	2476.38	241.53	131.90	128.03	132.05	53.98	-	144*	139*
	3col180_5	360	1536	153	157*	153*	2025.98	1997.27	2027.62	153*	-	163*	171*
Planning	logistics.b	843	7301	138	7.26	6.80	0.02	3027.81	0.02	8.85	29.67	138*	87.11
	logistics.c	1141	10719	162	15.73	17.89	0.04	2705.87	0.04	17.86	941.35	162*	162*
	rocket_ext.b	351	2398	69	0.36	0.56	0.02	0.14	0.01	0.36	338.40	5.19	737.26
	bw_large.c	3016	50457	265	14.46	15.67	0.37	0.37	0.38	8.45	-	24.99	165.40
	bw_large.d	6325	131973	431	245.02	141.69	2.92	2.86	2.94	92.10	-	683.77	251.49
Covering	9sym.b	310	976	5	56.16	200.22	583.08	633.16	3205.81	0.68	0.19	62.42	0.11
	alu4.b	808	1838	50	55*	56*	55*	56*	56*	52*	-	50*	189.10
	apex4.a	4317	11912	776	801*	800*	800*	800*	800*	848*	4.07	2358.90	0.33
	ex5.pi	2460	873	65	101*	99*	95*	95*	97*	85*	-	508.90	35.93
	rot.b	1452	2984	115	140*	142*	141*	141*	141*	126*	-	117*	5.81



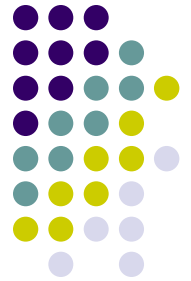
# MinCostSAT Conclusions

- MinCostChaff applies advanced techniques in SAT to MinCostSAT using Branch-and-Bound search.
- Pre-computed, dynamically maintained MIS to work with two literal watching BCP.
- The adaptive lower bounding is effective for constrained problems.
- However, MinCostChaff does not work well on classic covering benchmarks (large satisfying space), covering or mathematical optimization works much better
- The performance of MinCostChaff (search with lower bounding) is comparable to MiniSat+ (encoding based) on benchmarks with low total cost, does much better when total cost is high.

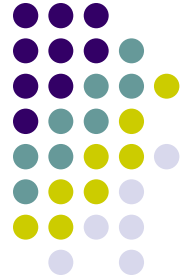
# Converting UNSAT instances...



to SAT instances



# Partial MAX-SAT (PM-SAT) Problem Definition



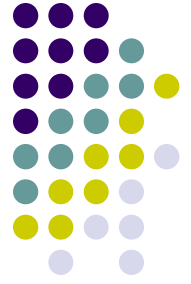
- Two sets of clauses
  - Non-relaxable or hard
  - Relaxable or soft

$$(x'_1 + x_2)(x'_1 + x'_2)[x_1 + x_3][x_1]$$

- Objective – A truth assignment that
  - Satisfies all non-relaxable clauses
  - Satisfies maximum number of relaxable clauses

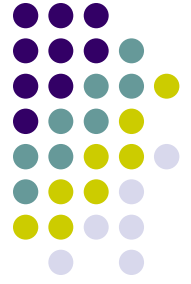
$$x_1 = 0, x_2 = 0, x_3 = 1$$

- Along the spectrum of SAT and MAX-SAT
  - All clauses are non-relaxable → Classical SAT
  - All clauses are relaxable → MAX-SAT



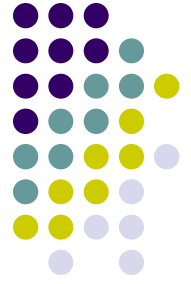
# Why Bother?

- Simply determining that an instance is UNSAT may not be enough.
- We want the optimal way to make the instance satisfiable by allowing for *some* clauses to be unsatisfied.
- AI
  - University course scheduling
  - ...
- EDA
  - Over constrained system analysis
  - FPGA routing
  - ...



# Previous Work

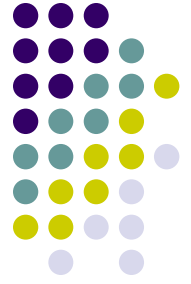
- PM-SAT problem defined by Miyazaki *et al.* in 1996
- Heuristic (incomplete) based methods
  - First heuristic (local search) by Kautz *et al.* in 1996
  - Another local search heuristic by Cha *et al.* in 1997 for course scheduling
- Complete (exact) methods
  - In 2005, Li used two MinCostSat solvers, *eclipse-stoc* [12] and *wpack* to solve the transformed PM-SAT problem
  - In 2005, Argelich and Manyà uses a branch and bound approach for the over constrained MAX-SAT problem



# Two Approaches

- Diagnosis Based Approach
  - Iterative UNSAT Core Elimination
- Encoding Based Approach
  - Constructing an auxiliary counter

# Diagnosis Based Approach: Iterative UNSAT Core Elimination

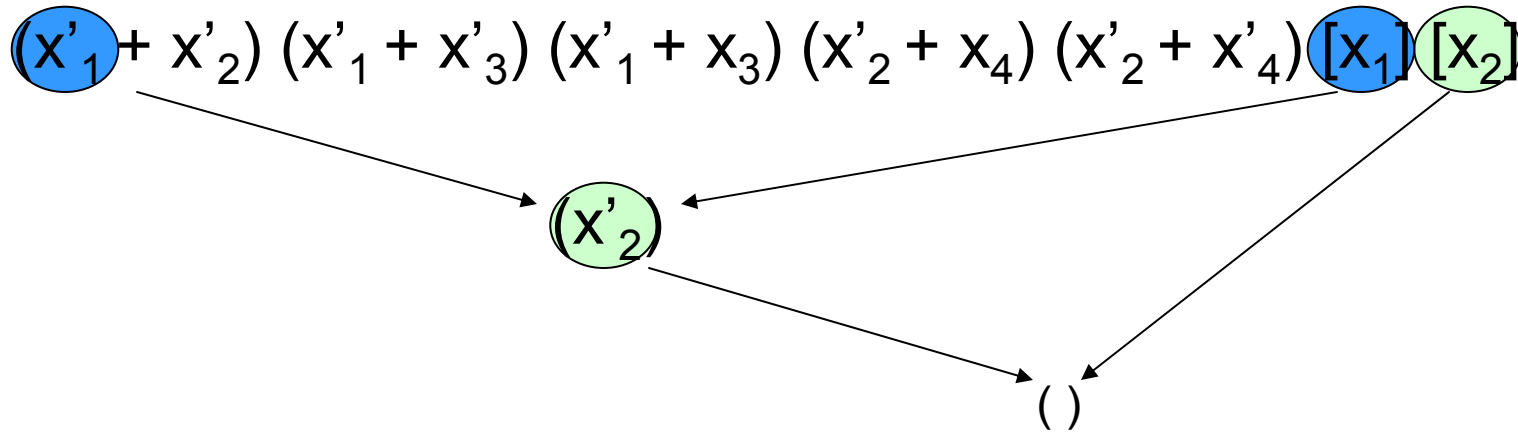
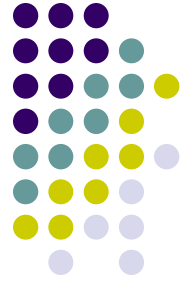


- An unsatisfiable core is a subset of the original CNF clauses that are unsatisfiable by themselves

$$\dots\dots(x'_1 + x_2)(x'_1 + x'_2) (x_1) \dots\dots$$

- Modern SAT solvers provide the UNSAT core as a byproduct of the proof of unsatisfiability, e.g. zcore from zchaff
  - L. Zhang and S. Malik. Validating SAT solvers using an independent resolution-based checker: Practical implementations and other applications. In *DATE'03*.
- All cores must be eliminated to make the instance SAT
  - Effectively need to find the minimal set of clauses that will cover all the UNSAT cores

# Iterative UNSAT Core Elimination

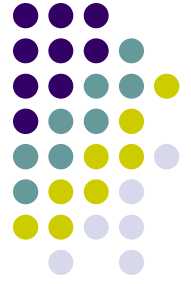


Unsatisfiable core:  $(x'_1 + x'_2)$   $[x_1]$   $[x_2]$

Relaxation with one-hot constraint using  $r_1$  and  $r_2$

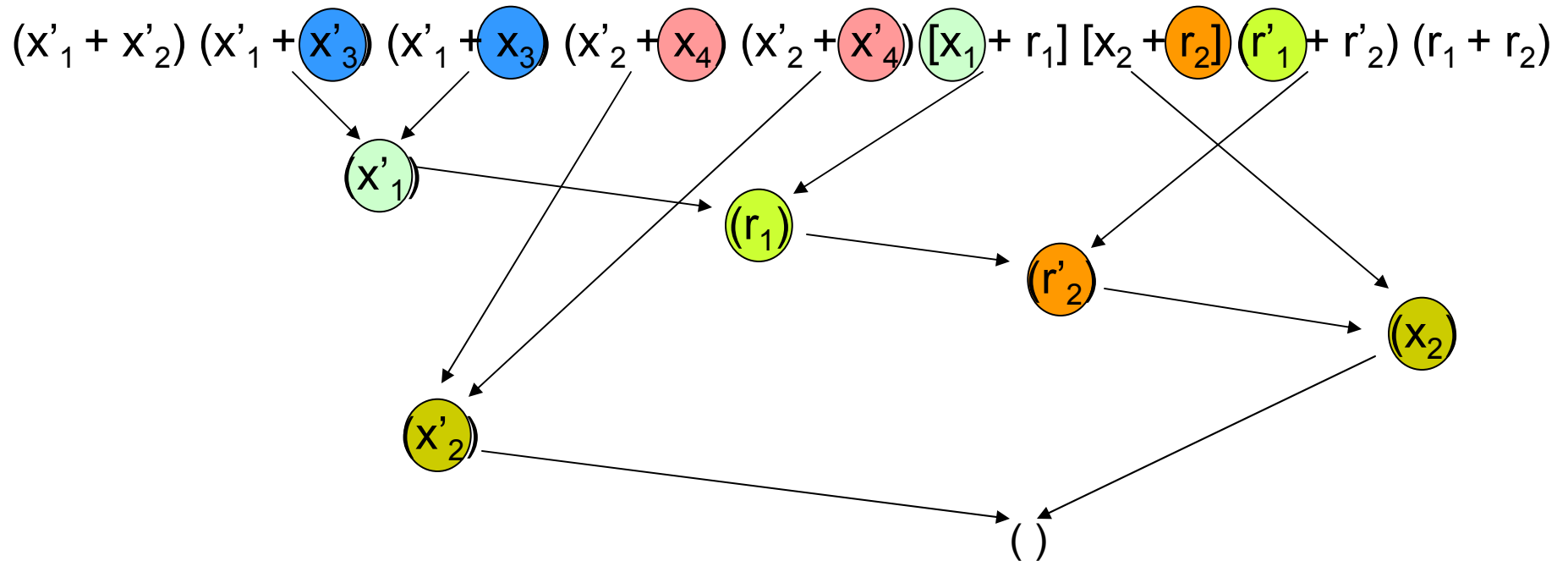
$$(x'_1 + x'_2) [x_1 + r_1] [x_2 + r_2] (r'_1 + r'_2) (r_1 + r_2)$$

# Iterative UNSAT Core Elimination



$$(x'_1 + x'_2) (x'_1 + x'_3) (x'_1 + x_3) (x'_2 + x_4) (x'_2 + x'_4) [x_1][x_2]$$

Relaxation with one-hot constraint using  $r_1$  and  $r_2$



Unsatisfiable core:  $(x'_1 + x'_3) (x'_1 + x_3) (x'_2 + x_4) (x'_2 + x'_4) [x_1 + r_1] [x_2 + r_2] (r'_1 + r'_2)$

## Slide 37

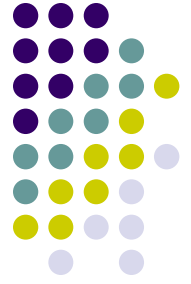
---

**SM8**

The animation does not show how the relaxation variables are added as a result of the UNSAT core. It will be helpful to show that.

Sharad Malik, 6/12/2006

# Iterative UNSAT Core Elimination



$$(x'_1 + x'_2) (x'_1 + x'_3) (x'_1 + x_3) (x'_2 + x_4) (x'_2 + x'_4) [x_1 + r_1] [x_2 + r_2] (r'_1 + r'_2) (r_1 + r_2)$$

Relaxation with one-hot constraint using  $r_3$  and  $r_4$

$$(x'_1 + x'_2) (x'_1 + x'_3) (x'_1 + x_3) (x'_2 + x_4) (x'_2 + x'_4) [x_1 + r_1 + r_3] [x_2 + r_2 + r_4] (r'_1 + r'_2) (r_1 + r_2) (r'_3 + r'_4) (r_3 + r_4)$$

Satisfiable!

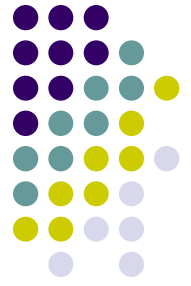
- $x_1 = 0, x_2 = 0, x_3 = 1, x_4 = 1$
- $r_1 = 1, r_2 = 0, r_3 = 0, r_4 = 1$
- Need to relax  $[x_1][x_2]$  in the original instance
- Iterative elimination provides a provably minimal relaxation.



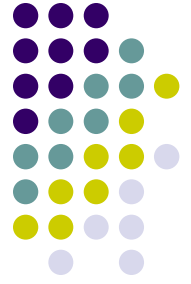
# Proof (Sketch) of Optimality

- Iterative Unsat Core Elimination (IUCE) finds  $K$  clauses to be relaxed, i.e. removing cores in  $U = (u_1, u_2, \dots, u_K)$
- Some algorithm finds  $M$  clauses,  $|M| < |U| = K$
- One of  $M$  clauses ( $c$ ) removes more than one core in  $U$ ,  $u_i$  and  $u_j$
- Only one of  $u_i, u_j$  can be discovered by IUCE since relaxing ( $c$ ) removes both of them
- Contradiction, so no such algorithm exists!

# Relaxation without Unsat Core: A Naïve Approach

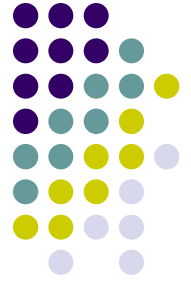


- Add a relaxation variable to **each** relaxable clause of the instance
- Require this batch of relaxation variables to be one-hot for every iteration until SAT
- Also guarantees optimality
- Practically inefficient due to the quadratic number of additional clauses (one-hot constraint)!
  - 100 relaxable clauses  $\rightarrow$  4951 additional clauses (naïve)
  - Only 3 out of 100 appear in the core  $\rightarrow$  4 additional clauses (Unsat Core diagnosis)



# Recap

- One-hot constraints are used to enforce that one and only one clause from each iterative unsat core is relaxed
- Instance eventually satisfiable by relaxing some subset of relaxable clauses
- Iterative UNSAT core elimination does not require the minimal core from the SAT solver
- Core diagnosis dramatically reduces the number of relaxation variables and one-hot constraint clauses (compared to the naïve approach)
- Incremental SAT is used across iterations



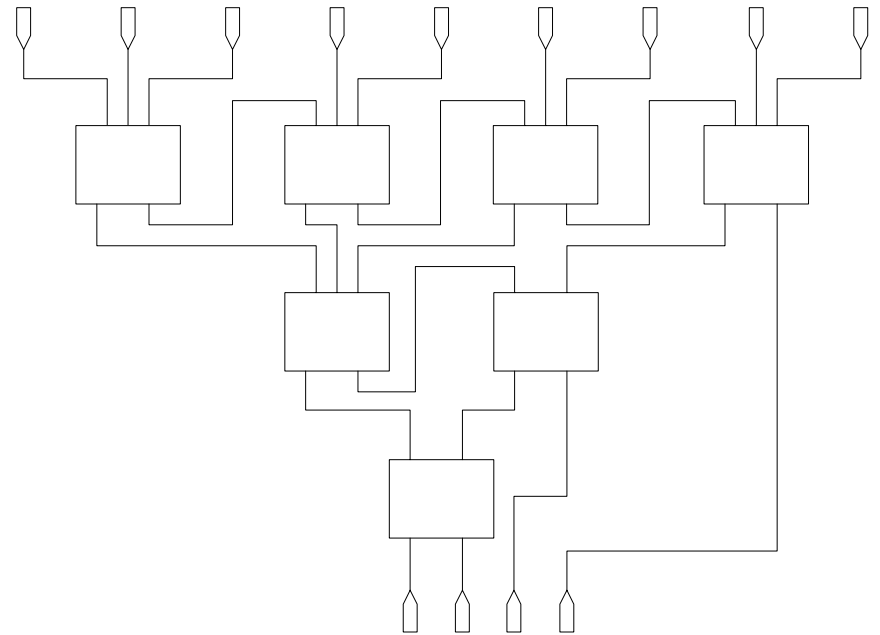
# Two Approaches

- Diagnosis Based Approach
  - Iterative UNSAT Core Elimination
- Encoding Based Approach
  - Constructing an auxiliary counter

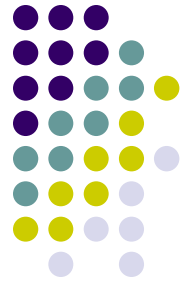
# Encoding Based Approach: Constructing An Auxiliary Counter



- A relaxation variable is added to each relaxable clause
- Minimum clauses relaxed  $\equiv$  Minimum relaxation variables assigned to 1 (true)
- Use an adder and a comparator to count the number of true relaxation variables



# Summary of Experimental Results



$n$  = number of relaxable clauses

$k$  = minimum number of clauses that need to be relaxed

- Diagnosis based wins when:
  - Small cores, large  $n$ , small  $k$
- Encoding based wins when:
  - Small  $n$
- Mathematical optimization wins:
  - Never
- Nothing wins:
  - Large cores, large  $n$



# With rising gas prices...



... need hybrid techniques

Y. Yu and S. Malik, "Lemma Learning in SMT on Linear Constraints,"  
SAT 2006, Sunday, August 13, 10:30am

# Acknowledgements



- We would like to thank Richard Rudell, Olivier Coudert and Vasco Manquinho for providing us various solvers and benchmarks