PBS: A Pseudo-Boolean Solver and Optimizer

Fadi A. Aloul, Arathi Ramani,
Igor L. Markov, Karem A. Sakallah
University of Michigan

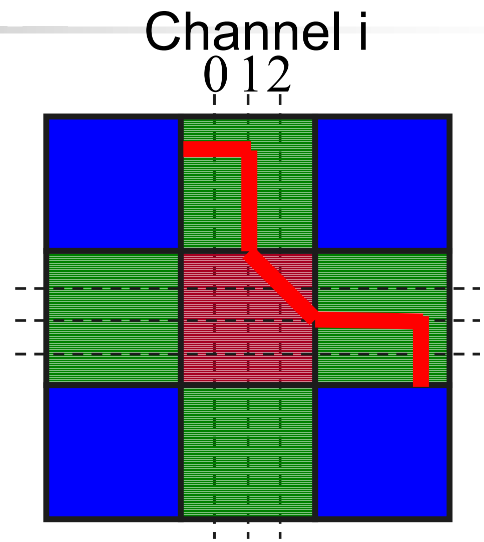
Motivation …

SAT Solvers
Apps: Verification, Routing,
ATPG, Timing Analysis
Problem Type: CSP
Problem Format: CNF
Example: Chaff, GRASP, SATO

Generic ILP Solvers
Apps: Routing, Planning,
Scheduling
Problem Type: CSP/Optimization
Problem Format: ILP
Example: CPLEX, LP_Solve

Specialized 0/1 ILP Solvers
Apps: Verification, Routing,
Binate Covering
Problem Type: CSP/Optimization
Problem Format: CNF/PB (0/1 ILP)
Example: Satire, BSOLO, OPBDP, WSAT

Motivation …

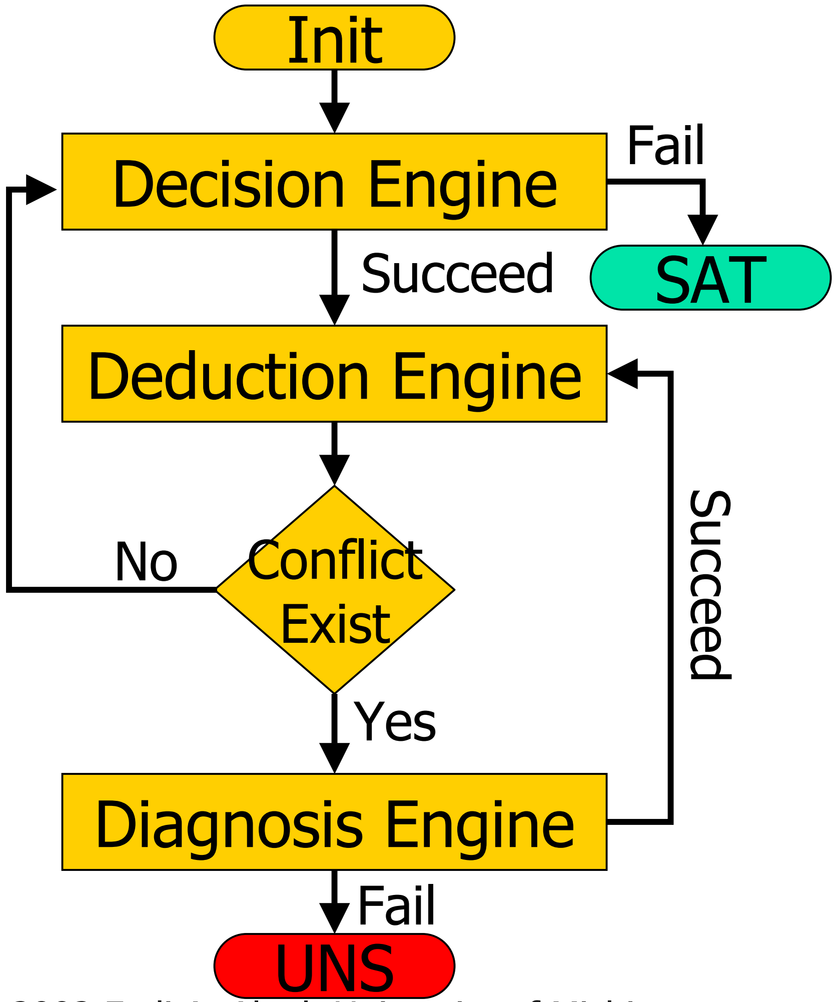


- Many applications require "Counting Constraints" that impose upper/lower bounds on number of objects
- Introduce a new specialized 0-1 ILP SAT solver
- Describe Pseudo-Boolean (PB) search algorithms
- Adapt SAT applications expressed in pure CNF to CNF/PB format
- Empirically demonstrate effectiveness in EDA applications

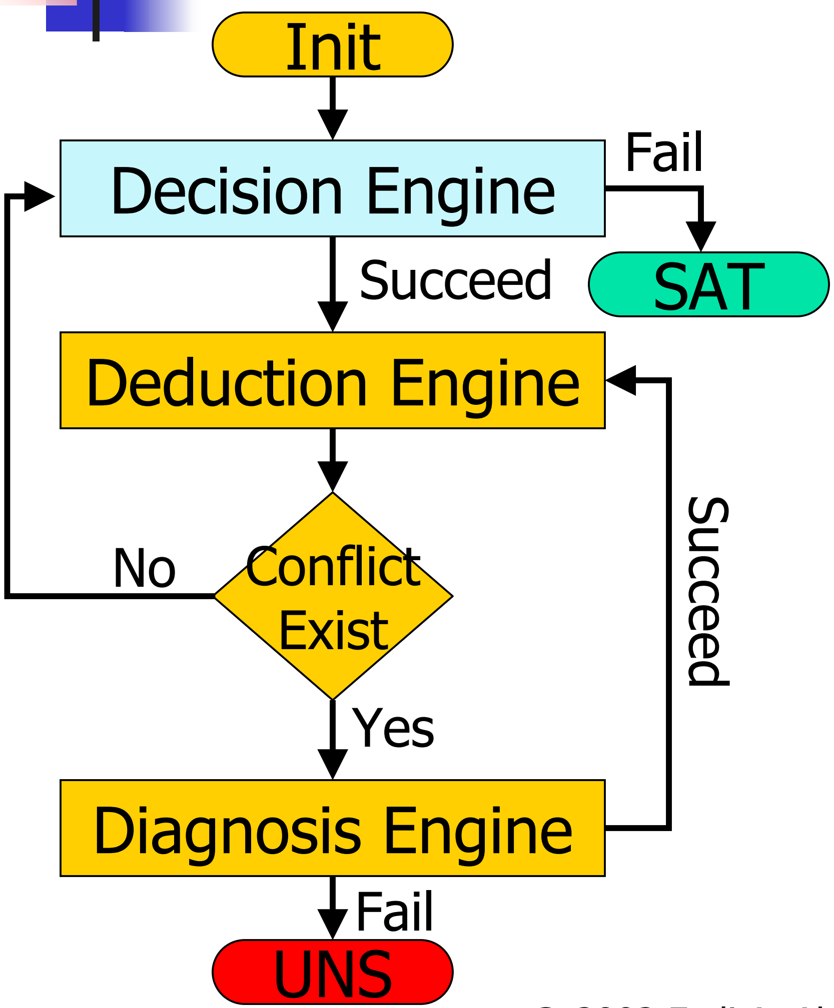
Outline

- Boolean Satisfiability advances
- Processing Pseudo-Boolean constraints
- Applications
  - CSP
  - Optimization
- Experimental evaluation
- Conclusions

Backtrack Search (DPLL)

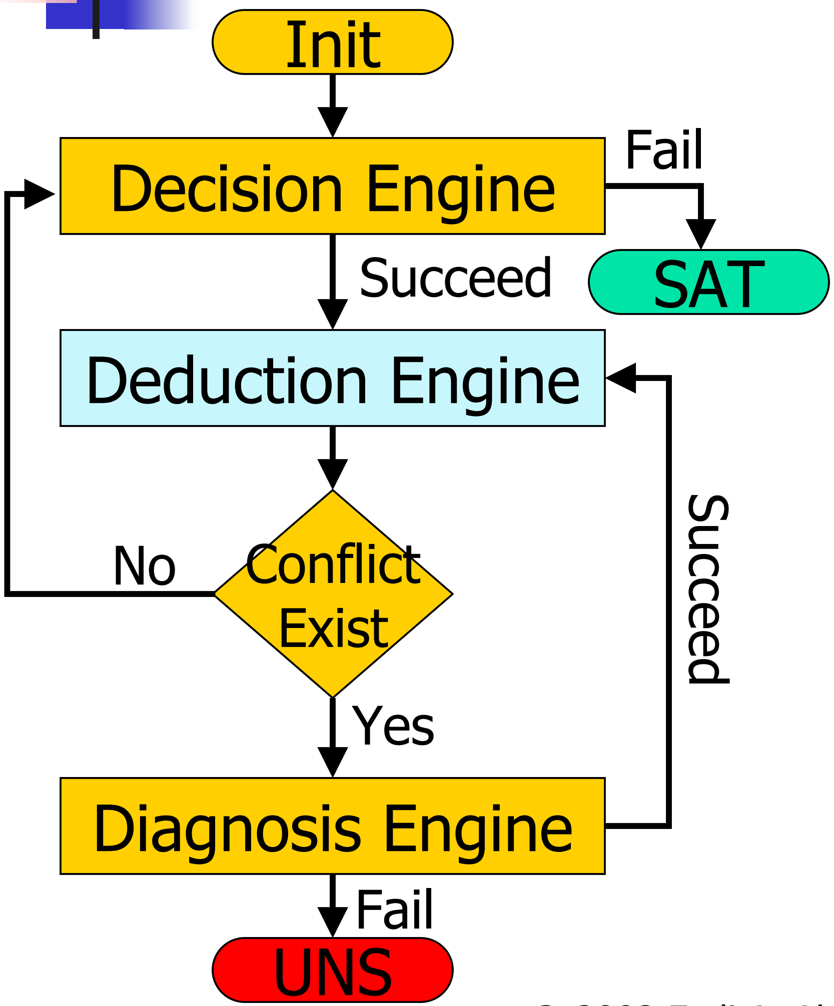


Decision Strategy



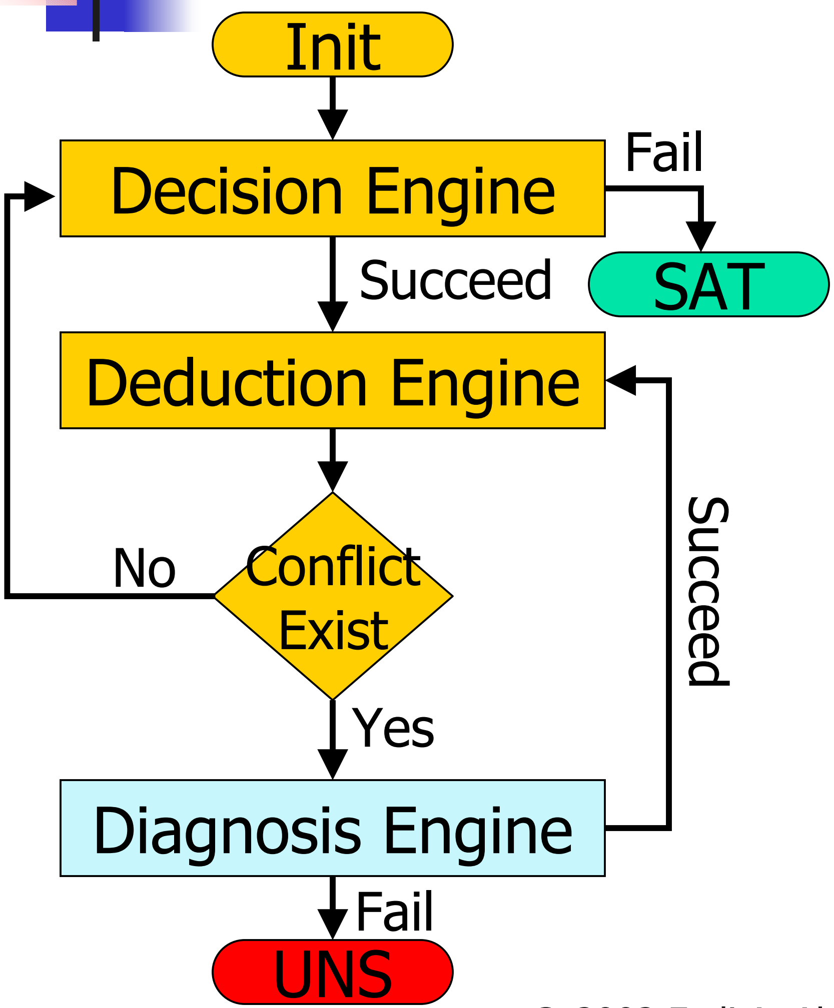
- Significantly improves the search performance
- Classified as:
  - Static
  - Dynamic
- Chaff introduced dynamic VSIDS:
  - Shown to be effective on most benchmarks
  - Selects most common literal and emphasizes variables in recent conflicts

Improved BCP



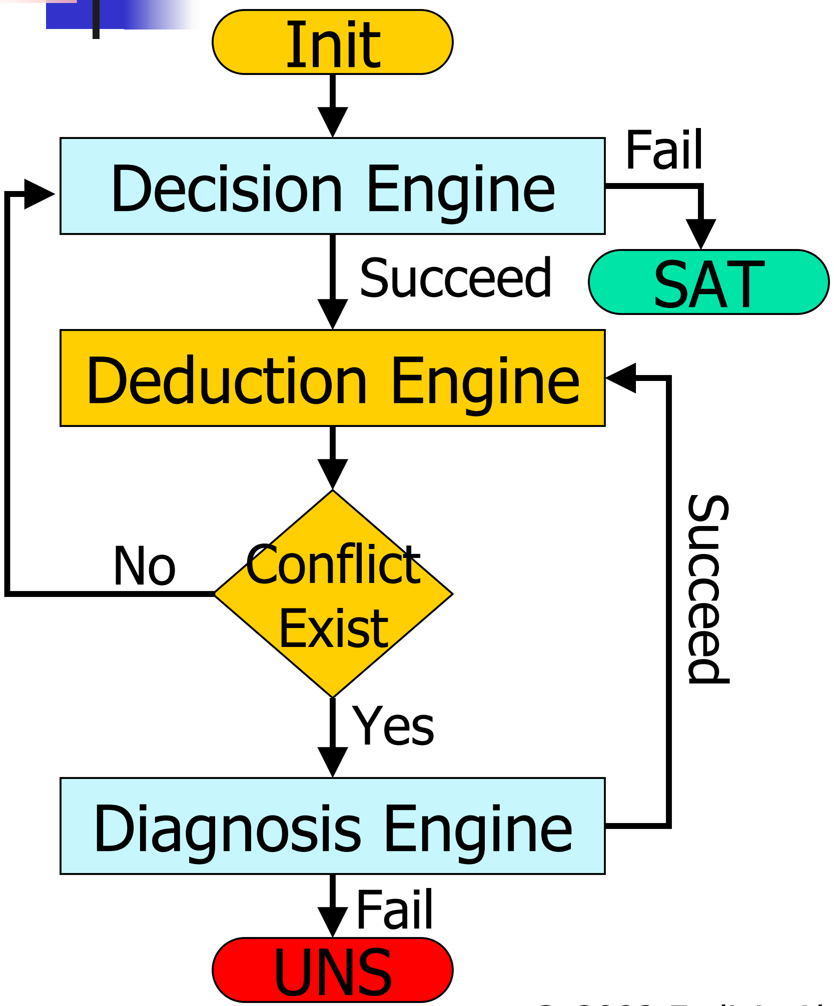
- Keeps track of any two unresolved literals in each clause instead of keeping track of all literals
- Leads to significant improvements over conventional BCP [Moskewicz et al., Zhang et al.]

Conflict Diagnosis and Clause Deletion



- Add conflict-induced clauses to avoid regenerating similar conflicts in future parts of the search process
- Very effective in expediting the search process
- Allows non-chronological backtracking
- 1UIP learning scheme shown to perform best among other learning schemes [Zhang et al.]

Random Restarts and Backtracking



- Solver often gets stuck in local non-useful search space
- Random restarts periodically unassigns all decisions and randomly selects a new decision sequence
- Restarts ensures that different sub-trees are searched at every restart
- Randomization can be combined with backtracking

Outline

- Boolean Satisfiability advances
- Processing Pseudo-Boolean constraints
- Applications
  - CSP
  - Optimization
- Experimental evaluation
- Conclusions

Pseudo-Boolean Constraints

$$c_1 x_1 + \cdots + c_n x_n \sim g$$

$$c_i, g \in Z$$

$$\sim \in \{=, \leq, \geq\}$$

$$x_i \in Literals$$

- Clauses can be generalized as a PB constraint: $(x + y) \rightarrow (x + y \geq 1)$
- None of the presented algorithms rely on the integrality of c_i and can be implemented for floating-point c_i

Motivating Example

- Objective:
  - limit the true assignments to k vars out of the n vars
- Solution:
  - CNF: $\binom{n}{k+1}$ clauses
    Each of size $(k+1)$
  - PB: single PB constraint

- "at most 2 out of v_1, v_2, v_3, v_4, v_5, can be true"
  - Pure CNF:

$$(\bar{v}_1 + \bar{v}_2 + \bar{v}_3) \cdot (\bar{v}_1 + \bar{v}_2 + \bar{v}_4) \cdot$$
$$(\bar{v}_1 + \bar{v}_2 + \bar{v}_5) \cdot (\bar{v}_1 + \bar{v}_3 + \bar{v}_4) \cdot$$
$$(\bar{v}_1 + \bar{v}_3 + \bar{v}_5) \cdot (\bar{v}_1 + \bar{v}_4 + \bar{v}_5) \cdot$$
$$(\bar{v}_2 + \bar{v}_3 + \bar{v}_4) \cdot (\bar{v}_2 + \bar{v}_3 + \bar{v}_5) \cdot$$
$$(\bar{v}_2 + \bar{v}_4 + \bar{v}_5) \cdot (\bar{v}_3 + \bar{v}_4 + \bar{v}_5)$$

  - PB form:

$$(1v_1 + 1v_2 + 1v_3 + 1v_4 + 1v_5 \le 2)$$

PB Constraint Data Structure

- Struct PBConstraint {
  - Goal n; constraint type $\sim$; list of c_i and x_i's;
  - initLHS;   // sum of all c_i's
  - LHS;   // value of LHS based on current variable assignment
  - maxLHS; // maximal possible value of LHS given the current variable assignment }

- For efficiency:
  - Sort the list of $c_i x_i$ in order of increasing c_i
  - Convert all negative c_i to positive:
    i.e. $c_1 x_1 - c_2 x_2 \leq n$

    $c_1 x_1 - c_2 (1 - \bar{x}_2) \leq n$

    $c_1 x_1 + c_2 \bar{x}_2 \leq n + c_2$

Algorithms for PB Search

- Assigning v_i to 1:
  For each literal x_i of v_i
  - If positive x_i, LHS += c_i
  - If negative x_i, maxLHS -= c_i
- Unassigning v_i from 1:
  For each literal x_i of v_i
  - If positive x_i, LHS -= c_i
  - If negative x_i, maxLHS += c_i
- PB constraint state:
  - $\geq$ type
    - SAT: LHS $\geq$ goal
    - UNS: maxLHS < goal
  - $\leq$ type
    - SAT: maxLHS $\leq$ goal
    - UNS: LHS > goal

$5x_1+6x_2+3x_3 \leq 12$

LHS = 0
maxLHS = 14

$5x_1+6x_2+3x_3 \leq 12$

LHS = 5
maxLHS = 14

$5x_1+6x_2+3x_3 \leq 12$

LHS = 5
maxLHS = 8
SATISFIABLE

Algorithms for PB Search

- Identifying implications
  - $\leq$ type
    - if c_i > goal – LHS, x_i = 0
    - Implied by literals in PB assigned to 1
  - $\geq$ type
    - if c_i > maxLHS – goal, x_i =1
    - Implied by literals in PB assigned to 0

$5x_1+6x_2+3x_3 \leq 12$

LHS = 0
maxLHS = 14
goal - LHS = 12

$5x_1+6x_2+3x_3 \leq 12$

LHS = 8
maxLHS = 14
goal - LHS = 4
Imply x_2=0

Algorithms for PB Search

- Identifying implications
  - $\leq$ type
    - if c_i > goal – LHS, $x_i = 0$
    - Implied by literals in PB assigned to 1
  - $\geq$ type
    - if c_i > maxLHS – goal, $x_i = 1$
    - Implied by literals in PB assigned to 0

$5x_1+6x_2+3x_3 \geq 10$

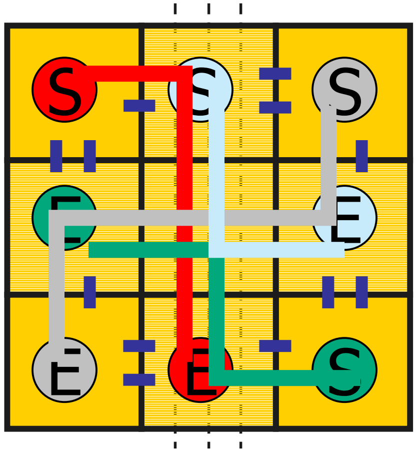
LHS = 0
maxLHS = 14
maxLHS - goal = 4
Imply $x_1=x_2=1$

Outline

- Boolean Satisfiability advances
- Processing Pseudo-Boolean constraints
- Applications
  - CSP
  - Optimization
- Experimental evaluation
- Conclusions

Applications - CSP

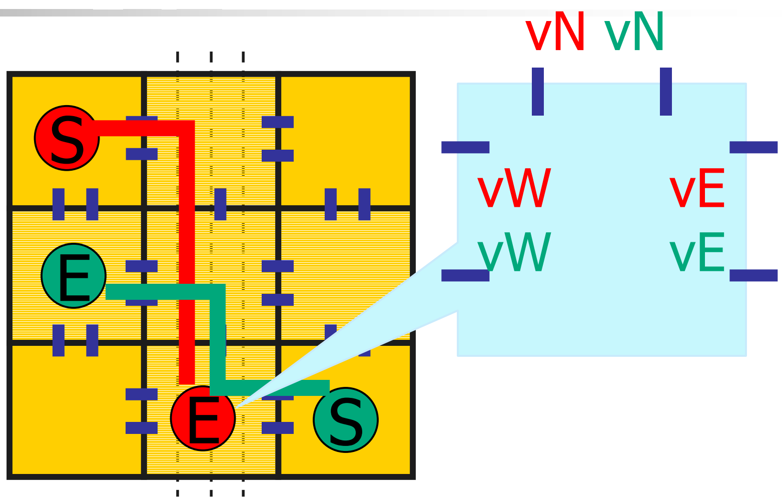
- Global Routing



- 2-D grid of cells arranged in rows/columns
- Cell boundaries are edges
- Capacity C is associated with each edge (no more than C routes can pass)
- Goal: route number of 2-pin connections in the grid with edge capacities
- Generate satisfiable instances using randomized flooding

Global Routing Formulation

- Connectivity constraints (for each net)
  - Exactly one edge selected at start/end point
  - If cell is a mid-point, either two or no edges are selected
- Capacity constraints
  - A net can use a single track across an edge
  - No two nets can use the same track across an edge

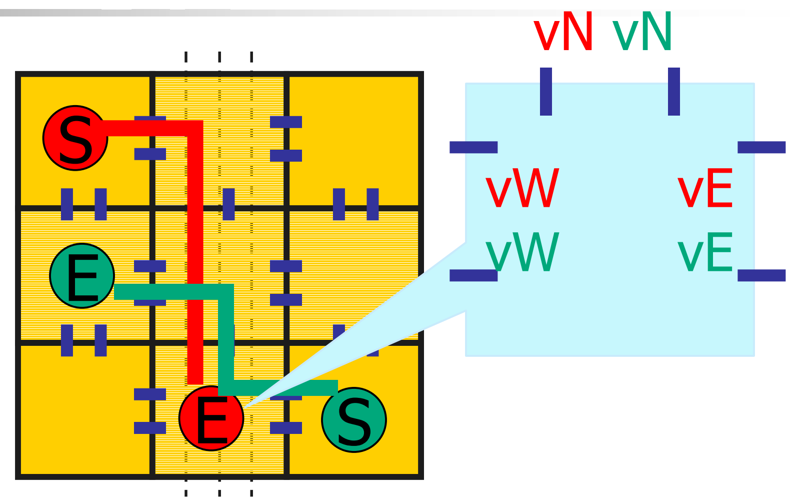


- Create a variable for each edge/net
  2 x 12 = 24 variables

$$(\overline{vN}+\overline{vW})(\overline{vN}+\overline{vE})(\overline{vW}+\overline{vE})$$
$$(vW+vN+vE)$$

Global Routing Formulation

- Connectivity constraints (for each net)
  - Exactly one edge selected at start/end point
  - If cell is a mid-point, either two or no edges are selected
- Capacity constraints
  - A net can use a single track across an edge
  - No two nets can use the same track across an edge

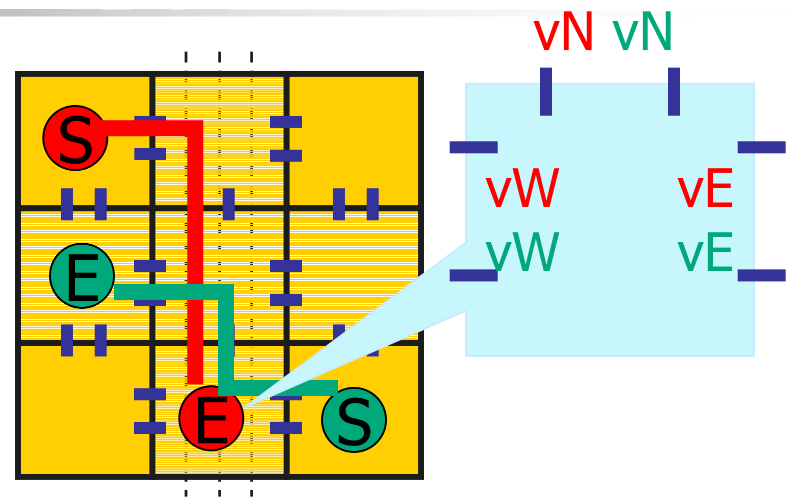


- Create a variable for each edge/net
  2 x 12 = 24 variables

$$(\overline{vN} + vE + vW)(vN + \overline{vE} + vW)$$
$$(vN + vE + \overline{vW})(\overline{vN} + \overline{vE} + \overline{vW})$$

Global Routing Formulation

- Connectivity constraints (for each net)
  - Exactly one edge selected at start/end point
  - If cell is a mid-point, either two or no edges are selected
- Capacity c
  - A net car track acr
  - No two n same track



Create a variable for each edge/net

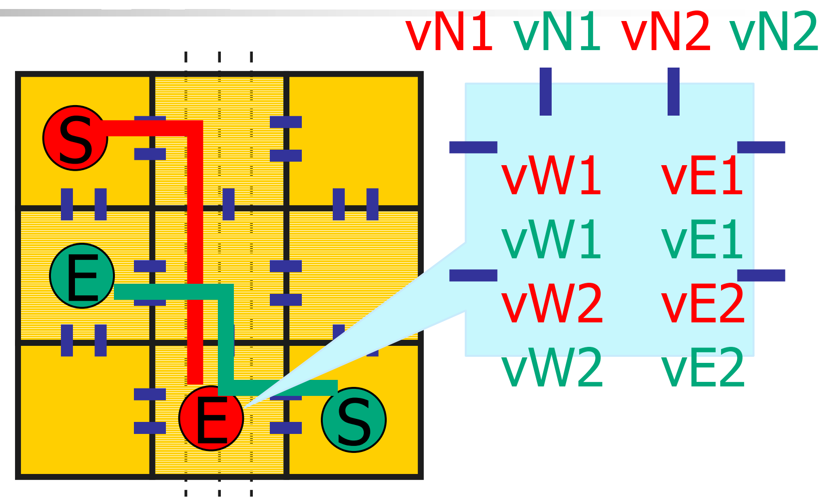
2 x 12 = 24 variables

30 Nets
10 Cap
#Cl = 55M
vs.
1 PB

pureCNF	CNF/PB
$\#Cl = \binom{\#Nets}{Cap+1}$	$\sum_{all\_nets} vi \leq Cap$

Global Routing Formulation

- Connectivity constraints (for each net)
  - Exactly one edge selected at start/end point
  - If cell is a mid-point, either two or no edges are selected
- Capacit
  - A net
    track a
  - No two n
    same track across an edge



Additional Variables & Clauses

- Create *Cap* variables per edge/net
  2 x 2 x 12 = 48 variables

pureCNF

$$\#Cl = (\#Nets)\binom{Cap}{2} + (Cap)\binom{\#Nets}{2}$$

Applications - optimization

- Max-ONEs
  - Seeks an assignment that
    - Satisfies all constraints
    - Maximizes the number of variables assigned to true
  - Useful to represent "Max-Clique" problems
  - "Vertex Cover" can be reduced to Min-ONEs
  - Use a single PB constraint of type "≥" that includes each variable with coefficient "1"
  - Iteratively increase the lower bound until the problem becomes unsatisfiable
  - Extendable to "Weighted Max-ONEs"

Applications - optimization

- Max-SAT
  - Finds an assignment that
    - Satisfies maximum possible number of clauses
  - Generalization of SAT
    - Provides more info for unsatisfiable instances
  - Used to represent “Max-CUT” problems
  - Expressed using a single PB constraint
  - Solved using PBS
  - Addressed indirectly using WalkSAT

Outline

- Boolean Satisfiability advances
- Processing Pseudo-Boolean constraints
- Applications
  - CSP
  - Optimization
- Experimental evaluation
- Conclusions

Experimental Setup

- Platform: Pentium-II 300 MHz with 512MB RAM running Linux
- Runtime limit: 5000 sec
- PBS Implemented in C++
- PBS settings:
  - VSIDS decision heuristic
  - Optimized BCP
  - Random Restarts
  - 1st UIP conflict analysis learning scheme
  - Clause deletion/random backtracking disabled

Global Routing Experiment

Instance	CNF + pseudo-Boolean						pure CNF			PBS Speedup		
	V	C	#PB	PBS	SATIRE	OPBDP	V	C	Chaff	Satire	OPBDP	Chaff
grout3.3-1	216	572	12	1.72	0.41	4.51	864	7592	40.43	0	3	24
grout3.3-2	264	700	12	0.33	0.96	4.65	1056	10864	11.3	3	14	34
grout3.3-3	240	636	12	0.09	1.1	6.65	960	9156	37.21	12	74	413
grout3.3-4	228	604	12	1.29	0.2	4.73	912	8356	103.13	0	4	80
grout3.3-5	240	634	12	0.84	0.35	6.88	960	9154	71.21	0	8	85
grout4.3-1	672	2004	24	3.46	109.7	5000	2688	33924	1361.6	32	1445	394
grout4.3-2	648	1928	24	1.92	32.13	5000	2592	31736	5000	17	2604	2604
grout4.3-3	648	1930	24	5.52	319.47	5000	2592	31738	5000	58	906	906
grout4.3-4	696	2072	24	16.3	3772	5000	2784	36176	2523	231	307	155
grout4.3-5	720	2144	24	2.06	567.12	5000	2880	38504	3915	275	2427	1900
grout4.3-6	624	1860	24	134	5000	5000	2496	29628	5000	37	37	37
grout4.3-7	672	2006	24	55	5000	5000	2688	33926	772.6	91	91	14
grout4.3-8	432	1280	24	2.9	177.8	5000	1728	15320	125	61	1724	43
grout4.3-9	840	2502	24	376	5000	5000	3360	51222	3203	13	13	9
grout4.3-10	840	2504	24	7.4	5000	5000	3360	51224	3465	676	676	468

MaxONE Experiment

Bench-mark	Satisfiable Instance	V	C	Max-ONEs	PBS	SATIRE	OPBDP	PBS Speedup	
								SATIRE	OBPDP
DIMACS	aim-50-1_6-yes1-1	50	80	29	0.01	0.01	0.02	1	2
	aim-100-1_6-yes1-1	100	160	43	0.01	0.02	7.19	2	719
	aim-200-2_0-yes1_1	200	400	96	0.01	0.06	5000	6	500000
	ii8b1	336	2068	275	4.69	3180	56.2	678	12
	jnh1	100	850	55	0.32	2.2	0.12	7	0.38
	jnh204	100	800	58	0.28	1.63	0.14	6	0.50
	par8-1	350	1149	79	0.01	0.06	0.05	6	5
	par8-2-c	68	270	20	0.01	0.02	0.01	2	1
Beijing	3blocks	283	9690	63	4.83	49.53	4494	10	930
QG	qg7-09	729	22060	81	0.1	5.41	9.8	54	98
	qg6-09	729	21844	81	0.21	5.56	45	26	214
Planning	bw_a	459	4675	73	0.03	0.43	0.21	14	7
	bw_b	1087	13772	136	0.58	6.39	17.86	11	31
	bw_c	3016	50457	272	24.37	315.5	5000	13	205

Conclusions

- Adapting SAT apps to use CNF/PB constraints
  - leads to memory savings and runtime reductions
- Proposed new specialized 0-1 ILP solver, PBS
- Confirmed effectiveness on real world examples:
  - Global routing consistency instances
  - Max-ONEs optimization problems (extendable to Max-SAT, Min-ONEs)

Future Works

- Compare state-of-the-art Generic ILP solvers, such as CPLEX, to specialized 0-1 ILP solvers
- Apply PBS to Max-SAT and Min-ONEs problems
- Study applications to Max-Clique, Max Independent Set, and Min Vertex Cover