MaxSAT for Optimization Problems

Joao Marques-Silva[1,2]

[1]University College Dublin, Ireland

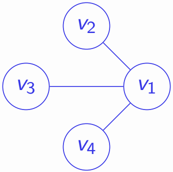
[2]IST/INESC-ID, Lisbon, Portugal

SAT/SMT Summer School 2011

MIT, Cambridge, MA, USA

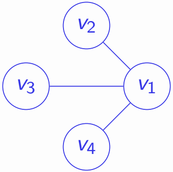
Example Problem: Minimum Vertex Cover

- The problem:
  - Graph $G = (V, E)$
  - Vertex cover $U \subseteq V$
    - For each $(v_i, v_j) \in E$, either $v_i \in U$ or $v_j \in U$
  - Minimum vertex cover: vertex cover U of minimum size



Example Problem: Minimum Vertex Cover

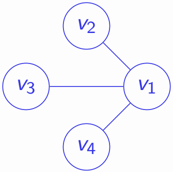
- The problem:
  - Graph $G = (V, E)$
  - Vertex cover $U \subseteq V$
    - For each $(v_i, v_j) \in E$, either $v_i \in U$ or $v_j \in U$
  - Minimum vertex cover: vertex cover U of minimum size



Vertex cover: $\{v_2, v_3, v_4\}$

Example Problem: Minimum Vertex Cover

- The problem:
  - Graph $G = (V, E)$
  - Vertex cover $U \subseteq V$
    - For each $(v_i, v_j) \in E$, either $v_i \in U$ or $v_j \in U$
  - Minimum vertex cover: vertex cover U of minimum size



Vertex cover: $\{v_2, v_3, v_4\}$

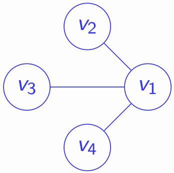
Min vertex cover: $\{v_1\}$

Example Problem: Minimum Vertex Cover

- Pseudo-Boolean Optimization (PBO) formulation:
  - Variables: x_i for each $v_i \in V$, with $x_i = 1$ iff $v_i \in U$
  - Clauses: $(x_i \vee x_j)$ for each $(v_i, v_j) \in E$
  - Objective function: minimize number of true x_i variables
    - I.e. minimize vertices included in U

Example Problem: Minimum Vertex Cover

- Pseudo-Boolean Optimization (PBO) formulation:
  - Variables: x_i for each $v_i \in V$, with $x_i = 1$ iff $v_i \in U$
  - Clauses: $(x_i \vee x_j)$ for each $(v_i, v_j) \in E$
  - Objective function: minimize number of true x_i variables
    - I.e. minimize vertices included in U



$$\begin{array}{ll} \text{minimize} & x_1 + x_2 + x_3 + x_4 \\ \text{subject to} & (x_1 \vee x_2) \wedge (x_1 \vee x_3) \wedge (x_1 \vee x_4) \end{array}$$

Boolean-Based Optimization

- Linear optimization over Boolean domains

Boolean-Based Optimization

- Linear optimization over Boolean domains
  - Can be mildly non-linear (e.g. basic Boolean operators)

Boolean-Based Optimization

- Linear optimization over Boolean domains
  - Can be mildly non-linear (e.g. basic Boolean operators)

- Concrete instantiations:
  - Maximum Satisfiability (MaxSAT)
  - Pseudo-Boolean Optimization (PBO, 0-1 ILP)
  - Weighted-Boolean Optimization (WBO)
  - Can map any problem to any other problem [e.g. HLO'08]

Boolean-Based Optimization

- Linear optimization over Boolean domains
  - Can be mildly non-linear (e.g. basic Boolean operators)
- Concrete instantiations:
  - Maximum Satisfiability (MaxSAT)
  - Pseudo-Boolean Optimization (PBO, 0-1 ILP)
  - Weighted-Boolean Optimization (WBO)
  - Can map any problem to any other problem [e.g. HLO'08]
- Related problems:
  - Optimization in SMT (MaxSMT)
  - Optimization in CSP (MaxCSP, etc.)
  - Integer Linear Programming (ILP)

This Talk

- Different ways of representing Boolean optimization problems are equivalent
  - Pseudo-Boolean Optimization (PBO) (or 0-1 ILP)
  - Maximum Satisfiability (MaxSAT)
  - Weighted Boolean Optimization (WBO)
  - etc.

- Optimization algorithms can (and **do!**) build on SAT solver technology
  - By using PBO
  - By using Core-guided MaxSAT

- Algorithms for MaxSAT can be readily extended to MaxSMT

Outline

Outline

What is Maximum Satisfiability?

- CNF Formula:

$x_6 \vee x_2$	$\neg x_6 \vee x_2$	$\neg x_2 \vee x_1$	$\neg x_1$
$\neg x_6 \vee x_8$	$x_6 \vee \neg x_8$	$x_2 \vee x_4$	$\neg x_4 \vee x_5$
$x_7 \vee x_5$	$\neg x_7 \vee x_5$	$\neg x_5 \vee x_3$	$\neg x_3$

What is Maximum Satisfiability?

- CNF Formula:

$x_6 \vee x_2$	$\neg x_6 \vee x_2$	$\neg x_2 \vee x_1$	$\neg x_1$
$\neg x_6 \vee x_8$	$x_6 \vee \neg x_8$	$x_2 \vee x_4$	$\neg x_4 \vee x_5$
$x_7 \vee x_5$	$\neg x_7 \vee x_5$	$\neg x_5 \vee x_3$	$\neg x_3$

- Formula is unsatisfiable
- MaxSAT:
  - Find assignment that maximizes number of satisfied clauses
    - For above formula, solution is 10
- There are a number of variants of MaxSAT

MaxSAT Problem(s)

- MaxSAT:
  - All clauses are soft
  - Maximize number of satisfied soft clauses
  - Minimize number of unsatisfied soft clauses

MaxSAT Problem(s)

- MaxSAT:
  - All clauses are soft
  - Maximize number of satisfied soft clauses
  - Minimize number of unsatisfied soft clauses
- Partial MaxSAT:
  - Hard clauses must be satisfied
  - Minimize number of unsatisfied soft clauses

MaxSAT Problem(s)

- MaxSAT:
  - All clauses are soft
  - Maximize number of satisfied soft clauses
  - Minimize number of unsatisfied soft clauses
- Partial MaxSAT:
  - Hard clauses must be satisfied
  - Minimize number of unsatisfied soft clauses
- Weighted MaxSAT
  - Weights associated with (soft) clauses
  - Minimize sum of weights of unsatisfied clauses

MaxSAT Problem(s)

- MaxSAT:
  - All clauses are soft
  - Maximize number of satisfied soft clauses
  - Minimize number of unsatisfied soft clauses
- Partial MaxSAT:
  - Hard clauses must be satisfied
  - Minimize number of unsatisfied soft clauses
- Weighted MaxSAT
  - Weights associated with (soft) clauses
  - Minimize sum of weights of unsatisfied clauses
- Weighted Partial MaxSAT
  - Weights associated with soft clauses
  - Hard clauses must be satisfied
  - Minimize sum of weights of unsatisfied soft clauses

MaxSAT Notation

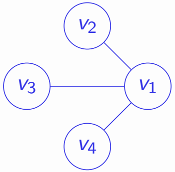
- (c_i, w_i): weighted clause
  - c_i is a set of literals (clause)
  - w_i is a non-negative integer or ∞ (or $\top$)
    - Cost of not satisfying c_i
- φ: set of weighted clauses
  - Soft clauses: (c_i, w_i), with $w_i < \infty$
    - Cost of not satisfying c_i is w_i
  - Hard clauses: (c_i, ∞)
    - Clause c_i must be satisfied

Modeling Example: Minimum Vertex Cover

- Partial MaxSAT formulation:
  - Variables: x_i for each $v_i \in V$, with $x_i = 1$ iff $v_i \in U$
  - Hard clauses: $(x_i \vee x_j)$ for each $(v_i, v_j) \in E$
  - Soft clauses: $(\neg x_i)$ for each $v_i \in V$
    - I.e. preferable not to include vertices in U

Modeling Example: Minimum Vertex Cover

- Partial MaxSAT formulation:
  - Variables: x_i for each $v_i \in V$, with $x_i = 1$ iff $v_i \in U$
  - Hard clauses: $(x_i \vee x_j)$ for each $(v_i, v_j) \in E$
  - Soft clauses: $(\neg x_i)$ for each $v_i \in V$
    - I.e. preferable not to include vertices in U



$$\varphi_H = \{(x_1 \vee x_2), (x_1 \vee x_3), (x_1 \vee x_4)\}$$
$$\varphi_S = \{(\neg x_1), (\neg x_2), (\neg x_3), (\neg x_4)\}$$

  - Hard clauses have cost ∞
  - Soft clauses have cost 1

Pseudo-Boolean Constraints & Optimization

- Pseudo-Boolean (PB) Constraints:
  - Boolean variables: $x_1, \ldots, x_n$
  - Linear inequalities:

$$\sum_{j \in N} a_{ij} l_j \geq b_i, \quad l_j \in \{x_j, \bar{x}_j\}, x_j \in \{0,1\}, a_{ij}, b_i \in \mathbb{N}_0^+$$

- Pseudo-Boolean Optimization (PBO):

$$\begin{array}{ll} \text{minimize} & \sum_{j \in N} w_j \cdot x_j \\ \text{subject to} & \sum_{j \in N} a_{ij} l_j \geq b_i, \\ & l_j \in \{x_j, \bar{x}_j\}, x_j \in \{0,1\}, a_{ij}, b_i, w_j \in \mathbb{N}_0^+ \end{array}$$

Solving MaxSAT with PBO

- Create φ' from φ:
  - Replace each c_i with $c_i' = c_i \cup \{r_i\}$
    - Fresh relaxation variable r_i for each clause c_i
  - Note: Trivial to satisfy φ' by assigning $r_i = 1$, for all i
- Minimize cost function: $\sum r_i$

Solving MaxSAT with PBO

- Create φ' from φ:
  - Replace each c_i with $c_i' = c_i \cup \{r_i\}$
    - Fresh relaxation variable r_i for each clause c_i
  - Note: Trivial to satisfy φ' by assigning $r_i = 1$, for all i
- Minimize cost function: $\sum r_i$

- Example:
  - CNF formula φ:

$$\varphi = \{\{x_1, \neg x_2\}, \{x_1, x_2\}, \{\neg x_1\}\}$$

  - Modified formula φ':

$$\varphi' = \{\{x_1, \neg x_2, r_1\}, \{x_1, x_2, r_2\}, \{\neg x_1, r_3\}\}$$

  - Minimize cost function: $r_1 + r_2 + r_3$

Solving MaxSAT with PBO – General Case

- MaxSAT instance:
  - φ_H: hard clauses of the form (c_i, ∞)
  - φ_S: (weighted) soft clauses of the form (c_i, w_i)

Solving MaxSAT with PBO – General Case

- MaxSAT instance:
  - φ_H: hard clauses of the form (c_i, ∞)
  - φ_S: (weighted) soft clauses of the form (c_i, w_i)

- Create PBO instance:

$$\begin{array}{ll} \text{min} & \sum w_i\, r_i \\ \text{s.t.} & \varphi_T \end{array}$$

where,
  - $\varphi_T = \varphi'_H \cup \varphi'_S$
  - φ'_H:
    - Each hard clause $(c_i, \infty) \in \varphi_H$ is mapped into clause c_i in φ_T
  - φ'_S:
    - Each soft clause (c_i, w_i) is mapped into a clause $(c_i \vee r_i)$, and term $w_i\, r_i$ is added to cost function

Solving PBO with MaxSAT – General Case

- Binate covering instance:

$$\begin{array}{ll} \text{min} & \sum w_i\, x_i \\ \text{s.t.} & \varphi \end{array}$$

Solving PBO with MaxSAT – General Case

- Binate covering instance:

$$\begin{array}{ll} \min & \sum w_i \, x_i \\ \text{s.t.} & \varphi \end{array}$$

- Create MaxSAT instance:
  - $\varphi_H \triangleq \varphi$: hard clauses of the form (c_i, ∞)
  - φ_S: for each cost function term $w_i \, x_i$, create soft clause $(\neg x_i, w_i)$

- General PB constraints?
  - Encode PB constraints to CNF, or
  - Use Weighted Boolean Optimization

Outline

Boolean-Based Optimization

Example Applications

Fundamental Techniques

Practical Algorithms

Results, Conclusions & Research Directions

Software Package Upgrades with MaxSAT

[MBCV'06,TSJL'07,AL'08,ALMS'09,ALBL'10]

- Universe of software packages: $\{p_1, \ldots, p_n\}$
- Associate x_i with p_i: $x_i = 1$ iff p_i is installed
- Constraints associated with package p_i: (p_i, D_i, C_i)
  - D_i: dependencies (required packages) for installing p_i
  - C_i: conflicts (disallowed packages) for installing p_i
- Example problem: Maximum Installability
  - Maximum number of packages that can be installed
  - Package constraints represent hard clauses
  - Soft clauses: (x_i)

Package constraints:

$(p_1, \{p_2 \vee p_3\}, \{p_4\})$
$(p_2, \{p_3\}, \{p_4\})$
$(p_3, \{p_2\}, \emptyset)$
$(p_4, \{p_2, p_3\}, \emptyset)$

Software Package Upgrades with MaxSAT

[MBCV'06,TSJL'07,AL'08,ALMS'09,ALBL'10]

- Universe of software packages: $\{p_1, \ldots, p_n\}$
- Associate x_i with p_i: $x_i = 1$ iff p_i is installed
- Constraints associated with package p_i: (p_i, D_i, C_i)
  - D_i: dependencies (required packages) for installing p_i
  - C_i: conflicts (disallowed packages) for installing p_i
- Example problem: Maximum Installability
  - Maximum number of packages that can be installed
  - Package constraints represent hard clauses
  - Soft clauses: (x_i)

Package constraints:

$$(p_1, \{p_2 \vee p_3\}, \{p_4\})$$
$$(p_2, \{p_3\}, \{p_4\})$$
$$(p_3, \{p_2\}, \emptyset)$$
$$(p_4, \{p_2, p_3\}, \emptyset)$$

MaxSAT formulation:

$$\begin{aligned} \varphi_H &= \{(\neg x_1 \vee x_2 \vee x_3), (\neg x_1 \vee \neg x_4), \\ &\quad (\neg x_2 \vee x_3), (\neg x_2 \vee \neg x_4), (\neg x_3 \vee x_2), \\ &\quad (\neg x_4 \vee x_2), (\neg x_4 \vee x_3)\} \\ \varphi_S &= \{(x_1), (x_2), (x_3), (x_4)\} \end{aligned}$$

Other Applications

- Error localization in C code [JM'11]
- Debugging of hardware designs [e.g. SMVLS'07,CSMSV'10]

- Haplotyping with pedigrees [GLMSO'10]
- Course timetabling [AN'10]
- Combinatorial auctions [HLGS'08]

- Binate/unate covering
  - Haplotype inference [GMSLO'11]
  - Digital filter design [ACFM'08]
  - FSM synthesis [e.g. HS'96]
  - Logic minimization [e.g. HS'96]
  - ...
- ...

Outline

Main Techniques

- Unit propagation
  - For computing lower bounds in B&B MaxSAT
- Local Search
  - For computing upper bounds (e.g. B&B MaxSAT)
- Unsatisfiable subformulas (or cores)
  - Used in core-guided MaxSAT algorithms
- CNF encodings
  - Cardinality constraints
  - PB constraints

Outline

Cardinality Constraints

- How to handle cardinality constraints, $\sum_{j=1}^{n} x_j \leq k$?
  - How to handle AtMost1 constraints, $\sum_{j=1}^{n} x_j \leq 1$?
  - General form: $\sum_{j=1}^{n} x_j \bowtie k$, with $\bowtie \in \{<, \leq, =, \geq, >\}$

- Solution #1:
  - Use PB solver
  - Difficult to keep up with advances in SAT technology
  - For SAT/UNSAT, best solvers already encode to CNF
    - E.g. Minisat+, but also QMaxSat, MSUnCore, (W)PM2

Cardinality Constraints

- How to handle cardinality constraints, $\sum_{j=1}^{n} x_j \leq k$?
  - How to handle AtMost1 constraints, $\sum_{j=1}^{n} x_j \leq 1$?
  - General form: $\sum_{j=1}^{n} x_j \bowtie k$, with $\bowtie \in \{<, \leq, =, \geq, >\}$

- Solution #1:
  - Use PB solver
  - Difficult to keep up with advances in SAT technology
  - For SAT/UNSAT, best solvers already encode to CNF
    - E.g. Minisat+, but also QMaxSat, MSUnCore, (W)PM2

- Solution #2:
  - Encode cardinality constraints to CNF
  - Use SAT solver

Equals1, AtLeast1 & AtMost1 Constraints

- $\sum_{j=1}^{n} x_j = 1$: encode with $(\sum_{j=1}^{n} x_j \leq 1) \wedge (\sum_{j=1}^{n} x_j \geq 1)$
- $\sum_{j=1}^{n} x_j \geq 1$: encode with $(x_1 \vee x_2 \vee \ldots \vee x_n)$
- $\sum_{j=1}^{n} x_j \leq 1$ encode with:
  - Pairwise encoding
    - Clauses: $\mathcal{O}(n^2)$; No auxiliary variables
  - Sequential counter [S'05]
    - Clauses: $\mathcal{O}(n)$; Auxiliary variables: $\mathcal{O}(n)$
  - Bitwise encoding [P'07,FP'01]
    - Clauses: $\mathcal{O}(n \log n)$; Auxiliary variables: $\mathcal{O}(\log n)$
  - ...

Bitwise Encoding

- Encode $\sum_{j=1}^{n} x_j \leq 1$ with bitwise encoding:

- An example: $x_1 + x_2 + x_3 \leq 1$

Bitwise Encoding

- Encode $\sum_{j=1}^{n} x_j \leq 1$ with bitwise encoding:
  - Auxiliary variables $v_0, \ldots, v_{r-1}$; $r = \lceil \log n \rceil$ (with $n > 1$)
  - If $x_j = 1$, then $v_0 \ldots v_{j-1} = b_0 \ldots b_{j-1}$, the binary encoding $j - 1$

    $x_j \rightarrow (v_0 = b_0) \wedge \ldots \wedge (v_{j-1} = b_{j-1}) \Leftrightarrow (\neg x_j \vee (v_0 = b_0) \wedge \ldots \wedge (v_{j-1} = b_{j-1}))$

- An example: $x_1 + x_2 + x_3 \leq 1$

	$j-1$	$v_1 v_0$
x_1	0	00
x_2	1	01
x_3	2	10

Bitwise Encoding

- Encode $\sum_{j=1}^{n} x_j \leq 1$ with bitwise encoding:
  - Auxiliary variables $v_0, \ldots, v_{r-1}$; $r = \lceil \log n \rceil$ (with $n > 1$)
  - If $x_j = 1$, then $v_0 \ldots v_{j-1} = b_0 \ldots b_{j-1}$, the binary encoding $j - 1$

    $x_j \rightarrow (v_0 = b_0) \wedge \ldots \wedge (v_{j-1} = b_{j-1}) \Leftrightarrow (\neg x_j \vee (v_0 = b_0) \wedge \ldots \wedge (v_{j-1} = b_{j-1}))$
  - Clauses $(\neg x_j \vee (v_i \leftrightarrow b_i)) = (\neg x_j \vee l_i)$, $i = 0, \ldots, r - 1$, where
    - $l_i \equiv v_i$, if $b_i = 1$
    - $l_i \equiv \neg v_i$, otherwise

- An example: $x_1 + x_2 + x_3 \leq 1$

	$j - 1$	$v_1 v_0$
x_1	0	00
x_2	1	01
x_3	2	10

$$(\neg x_1 \vee \neg v_1) \wedge (\neg x_1 \vee \neg v_0)$$
$$(\neg x_2 \vee \neg v_1) \wedge (\neg x_2 \vee v_0)$$
$$(\neg x_3 \vee v_1) \wedge (\neg x_3 \vee \neg v_0)$$

Bitwise Encoding

- Encode $\sum_{j=1}^{n} x_j \leq 1$ with bitwise encoding:
  - Auxiliary variables $v_0, \ldots, v_{r-1}$; $r = \lceil \log n \rceil$ (with $n > 1$)
  - If $x_j = 1$, then $v_0 \ldots v_{j-1} = b_0 \ldots b_{j-1}$, the binary encoding $j - 1$
    $x_j \rightarrow (v_0 = b_0) \wedge \ldots \wedge (v_{j-1} = b_{j-1}) \Leftrightarrow (\neg x_j \vee (v_0 = b_0) \wedge \ldots \wedge (v_{j-1} = b_{j-1}))$
  - Clauses $(\neg x_j \vee (v_i \leftrightarrow b_i)) = (\neg x_j \vee l_i)$, $i = 0, \ldots, r-1$, where
    - $l_i \equiv v_i$, if $b_i = 1$
    - $l_i \equiv \neg v_i$, otherwise
  - If $x_j = 1$, assignment to v_i variables must encode $j - 1$
    - All other x variables must take value 0
  - If all $x_j = 0$, any assignment to v_i variables is consistent
  - $\mathcal{O}(n \log n)$ clauses ; $\mathcal{O}(\log n)$ auxiliary variables

- An example: $x_1 + x_2 + x_3 \leq 1$

	$j-1$	$v_1 v_0$
x_1	0	00
x_2	1	01
x_3	2	10

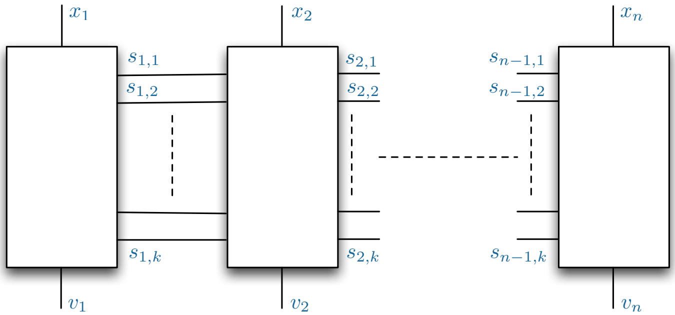
$(\neg x_1 \vee \neg v_1) \wedge (\neg x_1 \vee \neg v_0)$
$(\neg x_2 \vee \neg v_1) \wedge (\neg x_2 \vee v_0)$
$(\neg x_3 \vee v_1) \wedge (\neg x_3 \vee \neg v_0)$

General Cardinality Constraints

- General form: $\sum_{j=1}^{n} x_j \leq k$ (or $\sum_{j=1}^{n} x_j \geq k$)
  - Sequential counters [S'05]
    - Clauses/Variables: $\mathcal{O}(n\,k)$
  - BDDs [ES'06]
    - Clauses/Variables: $\mathcal{O}(n\,k)$
  - Sorting networks [ES'06]
    - Clauses/Variables: $\mathcal{O}(n \log^2 n)$
  - Cardinality Networks: [ANORC'09,ANORC'11]
    - Clauses/Variables: $\mathcal{O}(n \log^2 k)$
  - ...

Sequential Counter

- Encode $\sum_{j=1}^{n} x_j \leq k$ with sequential counter:



- Equations for each block $1 < i < n$, $1 < j < k$:

$$s_i = \sum_{j=1}^{i} x_j$$

s_i represented in unary

$$s_{i,1} = s_{i-1,1} \vee x_i$$
$$s_{i,j} = s_{i-1,j} \vee s_{i-1,j-1} \wedge x_i$$
$$v_i = s_{i-1,k} \wedge x_i = 0$$

Sequential Counter

- CNF formula for $\sum_{j=1}^{n} x_j \leq k$:
  - Assume: $k > 0 \land n > 1$
  - Indeces: $1 < i < n \ , \ 1 < j \leq k$

$$
\begin{array}{l}
(\neg x_1 \lor x_{1,1}) \\
(\neg s_{1,j}) \\
(\neg x_i \lor s_{i,1}) \\
(\neg s_{i-1,1} \lor s_{i,1}) \\
(\neg x_i \lor \neg s_{i-1,j-1} \lor s_{i,j}) \\
(\neg s_{i-1,j} \lor s_{i,j}) \\
(\neg x_i \lor \neg s_{i-1,k}) \\
(\neg x_n \lor \neg s_{n-1,k})
\end{array}
$$

- Recall: $\mathcal{O}(n\ k)$ clauses & variables

Outline

Pseudo-Boolean Constraints

- General form: $\sum_{j=1}^{n} a_j x_j \leq b$
  - Operational encoding [W'98]
    - Clauses/Variables: $\mathcal{O}(n)$
    - Does not guarantee arc-consistency
  - BDDs [ES'06]
    - Worst-case exponential number of clauses
  - Polynomial watchdog encoding [BBR'09]
    - Let $\nu(n) = \log(n) \log(a_{max})$
    - Clauses: $\mathcal{O}(n^3 \nu(n))$; Aux variables: $\mathcal{O}(n^2 \nu(n))$
  - ...

Pseudo-Boolean Constraints

- General form: $\sum_{j=1}^{n} a_j x_j \leq b$
  - Operational encoding [W'98]
    - Clauses/Variables: $\mathcal{O}(n)$
    - Does not guarantee arc-consistency
  - BDDs [ES'06]
    - Worst-case exponential number of clauses
  - Polynomial watchdog encoding [BBR'09]
    - Let $\nu(n) = \log(n) \log(a_{max})$
    - Clauses: $\mathcal{O}(n^3 \nu(n))$; Aux variables: $\mathcal{O}(n^2 \nu(n))$
  - ...
- How about $\sum_{j=1}^{n} a_j x_j = k$?

Pseudo-Boolean Constraints

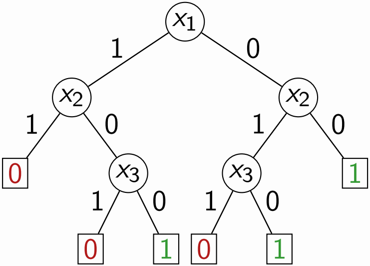
- General form: $\sum_{j=1}^{n} a_j x_j \leq b$
  - Operational encoding [W'98]
    - Clauses/Variables: $\mathcal{O}(n)$
    - Does not guarantee arc-consistency
  - BDDs [ES'06]
    - Worst-case exponential number of clauses
  - Polynomial watchdog encoding [BBR'09]
    - Let $\nu(n) = \log(n)\log(a_{max})$
    - Clauses: $\mathcal{O}(n^3\nu(n))$; Aux variables: $\mathcal{O}(n^2\nu(n))$
  - ...
- How about $\sum_{j=1}^{n} a_j x_j = k$?
  - Can use $(\sum_{j=1}^{n} a_j x_j \geq k) \wedge (\sum_{j=1}^{n} a_j x_j \leq k)$, but...

Pseudo-Boolean Constraints

- General form: $\sum_{j=1}^{n} a_j x_j \leq b$
  - Operational encoding [W'98]
    - Clauses/Variables: $\mathcal{O}(n)$
    - Does not guarantee arc-consistency
  - BDDs [ES'06]
    - Worst-case exponential number of clauses
  - Polynomial watchdog encoding [BBR'09]
    - Let $\nu(n) = \log(n)\log(a_{max})$
    - Clauses: $\mathcal{O}(n^3\nu(n))$; Aux variables: $\mathcal{O}(n^2\nu(n))$
  - ...
- How about $\sum_{j=1}^{n} a_j x_j = k$?
  - Can use $(\sum_{j=1}^{n} a_j x_j \geq k) \wedge (\sum_{j=1}^{n} a_j x_j \leq k)$, but...
    - $\sum_{j=1}^{n} a_j x_j = k$ is a subset-sum constraint (special case of a knapsack constraint)
    - Cannot find all consequences in polynomial time [S'03,FS'02,T'03]

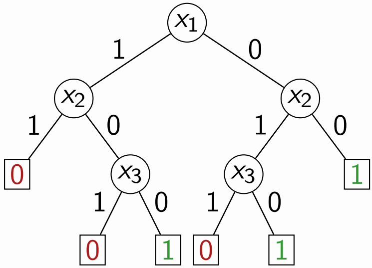
Encoding PB Constraints with BDDs I

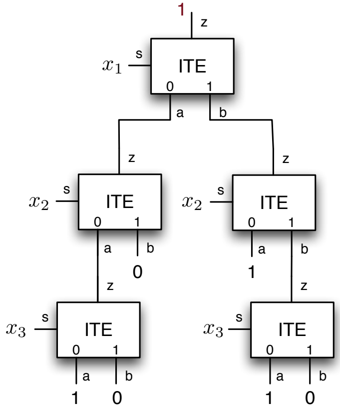
- Encode $3x_1 + 3x_2 + x_3 \leq 3$
- Construct BDD
  - E.g. analyze variables by decreasing coefficients
- Extract ITE-based circuit from BDD



Encoding PB Constraints with BDDs I

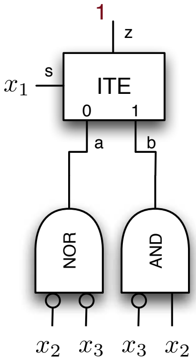
- Encode $3x_1 + 3x_2 + x_3 \leq 3$
- Construct BDD
  - E.g. analyze variables by decreasing coefficients
- Extract ITE-based circuit from BDD





Encoding PB Constraints with BDDs II

- Encode $3x_1 + 3x_2 + x_3 \leq 3$
- Extract ITE-based circuit from BDD
- Simplify and create final circuit:



Outline

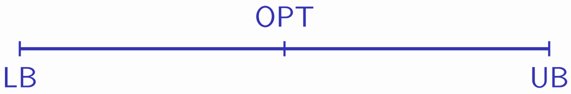
Outline

Definitions

- Cost of assignment:
  - Sum of weights of unsatisfied clauses
- Optimum solution (OPT):
  - Assignment with minimum cost
- Upper Bound (UB):
  - Assignment with cost not less than OPT
  - E.g. $\sum_{c_i \in \varphi} w_i + 1$; hard clauses may be inconsistent
- Lower Bound (LB):
  - Assignment with cost less (or no larger) than OPT
  - E.g. -1; it may be possible to satisfy all soft clauses

Definitions

- Cost of assignment:
  - Sum of weights of unsatisfied clauses
- Optimum solution (OPT):
  - Assignment with minimum cost
- Upper Bound (UB):
  - Assignment with cost not less than OPT
  - E.g. $\sum_{c_i \in \varphi} w_i + 1$; hard clauses may be inconsistent
- Lower Bound (LB):
  - Assignment with cost less (or no larger) than OPT
  - E.g. -1; it may be possible to satisfy all soft clauses



Outline

Branch-and-Bound Search for MaxSAT

- Unit propagation is unsound for MaxSAT [e.g. BLM'07]

$$\{\{x_1\}, \{\neg x_1, \neg x_2\}, \{\neg x_1, \neg x_3\}, \{x_2\}, \{x_3\}\}$$

- Standard B&B search [LMP'07,HLO'08,LHG'08]
  - No unit propagation
- Refine UBs on number of empty clauses
- Estimate LBs
  - Unit propagation provides LBs
  - Bound search when LB $\geq$ UB
- Inference rules to prune search [HL'06,LMP'07]
- Optionally: use local search to identify UBs [HLO'08]

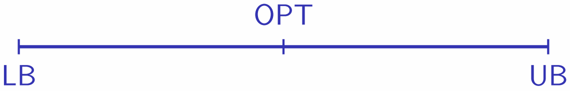
Branch-and-Bound Search for PBO

$$\begin{array}{ll} \text{minimize} & \sum_{j \in N} w_j \cdot x_j \\ \text{subject to} & \sum_{j \in N} a_{ij} l_j \geq b_i, \\ & l_j \in \{x_j, \bar{x}_j\}, x_j \in \{0,1\}, a_{ij}, b_i, w_j \in \mathbb{N}_0^+ \end{array}$$

- Standard B&B search [MMS'02,MMS'04,MMS'06,SS'06,NO'06]
- Refine UBs on value of cost function
  - Any model for the constraints refines UB
- Estimate LBs
  - Standard techniques: LP relaxations; MIS; etc.
  - Bound search when LB $\geq$ UB
- Native handling of PB constraints
- Integrate SAT techniques
  - Unit propagation; Clause learning; Restarts; VSIDS; etc.

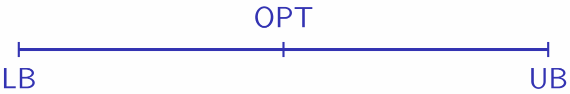
Outline

Iterative SAT Solving



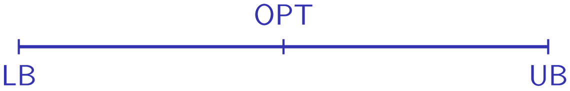
- Iteratively refine upper bound (UB) until UB = OPT
  - Linear search SAT-UNSAT (or strengthening)

Iterative SAT Solving



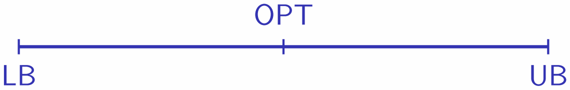
- Iteratively refine upper bound (UB) until UB = OPT
  - Linear search SAT-UNSAT (or strengthening)
- Iteratively refine lower bound (LB) until LB = OPT
  - Linear search UNSAT-SAT

Iterative SAT Solving



- Iteratively refine upper bound (UB) until UB = OPT
  - Linear search SAT-UNSAT (or strengthening)
- Iteratively refine lower bound (LB) until LB = OPT
  - Linear search UNSAT-SAT
- Iteratively refine lower & upper bounds until $LB_k = UB_k - 1$
  - Linear search by refining LB&UB
  - Binary search on cost of unsatisfied clauses

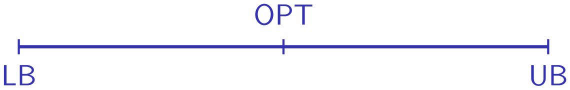
Iterative SAT Solving



- Iteratively refine upper bound (UB) until UB = OPT
  - Linear search SAT-UNSAT (or strengthening)
- Iteratively refine lower bound (LB) until LB = OPT
  - Linear search UNSAT-SAT
- Iteratively refine lower & upper bounds until $\text{LB}_k = \text{UB}_k - 1$
  - Linear search by refining LB&UB
  - Binary search on cost of unsatisfied clauses

- By default:
  - All soft clauses relaxed: replace c_i with $c_i \cup \{r_i\}$
  - Cardinality/PB constraints to represent LBs & UBs

Iterative SAT Solving

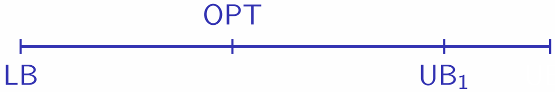


- Iteratively refine upper bound (UB) until UB = OPT
  - Linear search SAT-UNSAT (or strengthening)
- Iteratively refine lower bound (LB) until LB = OPT
  - Linear search UNSAT-SAT
- Iteratively refine lower & upper bounds until $\text{LB}_k = \text{UB}_k - 1$
  - Linear search by refining LB&UB
  - Binary search on cost of unsatisfied clauses
- By default: Not for core-guided approaches !
  - All soft clauses relaxed: replace c_i with $c_i \cup \{r_i\}$
  - Cardinality/PB constraints to represent LBs & UBs

Iterative SAT Solving – Refine UB

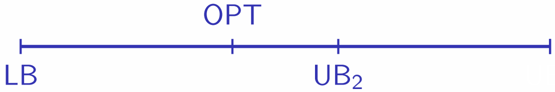
- Require $\sum w_i\, r_i \leq UB_0 - 1$

Iterative SAT Solving – Refine UB



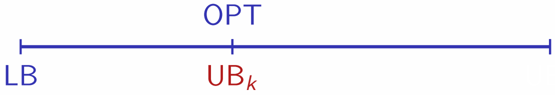
- Require $\sum w_i\, r_i \leq UB_0 - 1$
- While SAT, refine UB
  - New UB given by cost of unsatisfied clauses, i.e. $\sum w_i\, r_i$

Iterative SAT Solving – Refine UB



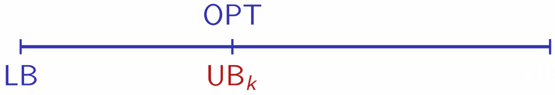
- Require $\sum w_i \, r_i \leq UB_0 - 1$
- While SAT, refine UB
  - New UB given by cost of unsatisfied clauses, i.e. $\sum w_i \, r_i$

Iterative SAT Solving – Refine UB



- Require $\sum w_i \, r_i \leq UB_0 - 1$
- While SAT, refine UB
  - New UB given by cost of unsatisfied clauses, i.e. $\sum w_i \, r_i$
- Repeat until constraint $\sum w_i \, r_i \leq UB_k - 1$ becomes UNSAT
  - UB_k denotes the optimum value

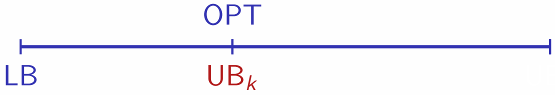
Iterative SAT Solving – Refine UB



- Require $\sum w_i \, r_i \leq UB_0 - 1$
- While SAT, refine UB
  - New UB given by cost of unsatisfied clauses, i.e. $\sum w_i \, r_i$
- Repeat until constraint $\sum w_i \, r_i \leq UB_k - 1$ becomes UNSAT
  - UB_k denotes the optimum value

- Worst-case # of iterations **exponential** on instance size

Iterative SAT Solving – Refine UB

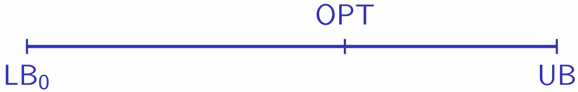


- Require $\sum w_i \, r_i \leq UB_0 - 1$
- While SAT, refine UB
  - New UB given by cost of unsatisfied clauses, i.e. $\sum w_i \, r_i$
- Repeat until constraint $\sum w_i \, r_i \leq UB_k - 1$ becomes UNSAT
  - UB_k denotes the optimum value

- Worst-case # of iterations **exponential** on instance size

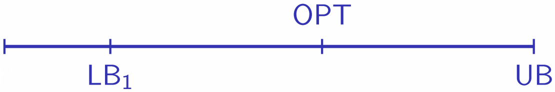
- Example tools:
  - Minisat+: CNF encoding of constraints [ES'06]
  - SAT4J: native handling of constraints [LBP'10]
  - QMaxSat: CNF encoding of constraints
  - ...

Iterative SAT Solving – Refine LB



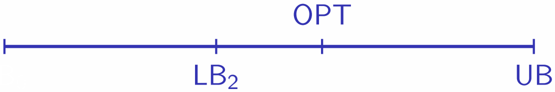
- Require $\sum w_i \, r_i \leq LB_0 + 1$

Iterative SAT Solving – Refine LB



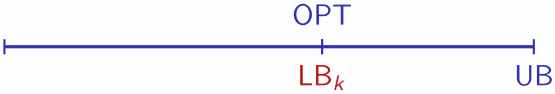
- Require $\sum w_i \, r_i \leq LB_0 + 1$
- While UNSAT, refine LB, i.e. $LB_k \leftarrow LB_{k-1} + 1$

Iterative SAT Solving – Refine LB



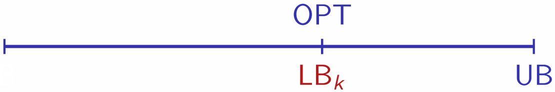
- Require $\sum w_i\, r_i \leq LB_0 + 1$
- While UNSAT, refine LB, i.e. $LB_k \leftarrow LB_{k-1} + 1$

Iterative SAT Solving – Refine LB



- Require $\sum w_i\, r_i \leq LB_0 + 1$
- While UNSAT, refine LB, i.e. $LB_k \leftarrow LB_{k-1} + 1$
- Repeat until constraint $\sum w_i\, r_i \leq LB_k$ becomes SAT
  - LB_k denotes the optimum value

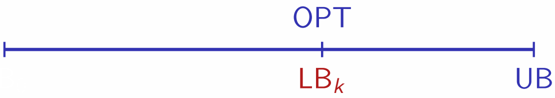
Iterative SAT Solving – Refine LB



- Require $\sum w_i \, r_i \leq LB_0 + 1$
- While UNSAT, refine LB, i.e. $LB_k \leftarrow LB_{k-1} + 1$
- Repeat until constraint $\sum w_i \, r_i \leq LB_k$ becomes SAT
  - LB_k denotes the optimum value

- Worst-case # of iterations **exponential** on instance size

Iterative SAT Solving – Refine LB

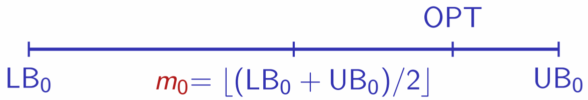


- Require $\sum w_i \, r_i \leq LB_0 + 1$
- While UNSAT, refine LB, i.e. $LB_k \leftarrow LB_{k-1} + 1$
- Repeat until constraint $\sum w_i \, r_i \leq LB_k$ becomes SAT
  - LB_k denotes the optimum value

- Worst-case # of iterations **exponential** on instance size

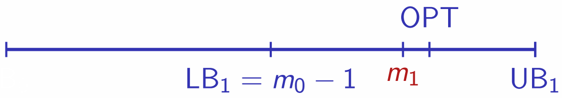
- Example tools:
  - No known implementations
  - Some core-guided MaxSAT solvers [e.g. FM'06,MSP'08,MMSP'09,ABL'09]
    - **But** policies for updating LB are different
    - **Unclear** whether worst-case # of iterations remains exponential

Iterative SAT Solving – Binary Search



- Invariant: $LB_k \leq UB_k - 1$
- Require $\sum w_i \, r_i \leq m_0$

Iterative SAT Solving – Binary Search



- Invariant: $LB_k \leq UB_k - 1$
- Require $\sum w_i \, r_i \leq m_0$
- Repeat
  - If UNSAT, refine $LB_1 = m_0, \ldots$
  - Compute new mid value $m_1, \ldots$

Iterative SAT Solving – Binary Search

- Invariant: $\mathrm{LB}_k \leq \mathrm{UB}_k - 1$
- Require $\sum w_i\, r_i \leq m_0$
- Repeat
  - If UNSAT, refine $\mathrm{LB}_1 = m_0, \ldots$
  - Compute new mid value $m_1, \ldots$
  - If SAT, refine $\mathrm{UB}_3 = m_2, \ldots$

Iterative SAT Solving – Binary Search

- Invariant: $\mathrm{LB}_k \leq \mathrm{UB}_k - 1$
- Require $\sum w_i\, r_i \leq m_0$
- Repeat
  - If UNSAT, refine $\mathrm{LB}_1 = m_0, \ldots$
  - Compute new mid value $m_1, \ldots$
  - If SAT, refine $\mathrm{UB}_3 = m_2, \ldots$

Iterative SAT Solving – Binary Search

- Invariant: $\text{LB}_k \leq \text{UB}_k - 1$
- Require $\sum w_i\, r_i \leq m_0$
- Repeat
  - If UNSAT, refine $\text{LB}_1 = m_0, \ldots$
  - Compute new mid value $m_1, \ldots$
  - If SAT, refine $\text{UB}_3 = m_2, \ldots$
- Until $\text{LB}_k = \text{UB}_k - 1$

Iterative SAT Solving – Binary Search

- Invariant: $\mathrm{LB}_k \leq \mathrm{UB}_k - 1$
- Require $\sum w_i \, r_i \leq m_0$
- Repeat
  - If UNSAT, refine $\mathrm{LB}_1 = m_0, \ldots$
  - Compute new mid value $m_1, \ldots$
  - If SAT, refine $\mathrm{UB}_3 = m_2, \ldots$
- Until $\mathrm{LB}_k = \mathrm{UB}_k - 1$
- Worst-case # of iterations **linear** on instance size

Iterative SAT Solving – Binary Search

- Invariant: $LB_k \leq UB_k - 1$
- Require $\sum w_i \, r_i \leq m_0$
- Repeat
  - If UNSAT, refine $LB_1 = m_0, \ldots$
  - Compute new mid value $m_1, \ldots$
  - If SAT, refine $UB_3 = m_2, \ldots$
- Until $LB_k = UB_k - 1$
- Worst-case # of iterations **linear** on instance size
- Example tools:
  - Counter-based MaxSAT solver [FM'06]
  - MathSAT [CFGSS'10]
  - MSUnCore [HMMS'11]

Outline

What are Core-Guided MaxSAT Algorithms?

- Drawbacks of iterative SAT solving
  - All soft clauses are relaxed
    - Number of soft clauses can be large
  - PB/cardinality constraints with large number of variables and (possibly) large rhs
    - Can result in large CNF encodings

- Core-guided MaxSAT algorithms use unsatisfiable cores for:
  - Relax soft clauses on demand, i.e. relax clauses only when needed, or
  - Relax all soft clauses, but use unsatisfiable cores for creating simpler PB/cardinality constraints

Many Core-Guided MaxSAT Algorithms

- Algorithms:
  - (W)MSU1.X/WPM1 [FM'06,MSM'08,MMSP'09,ABL'09]
  - (W)MSU3 [MSP'07]
  - (W)MSU4 [MSP'08]
  - (W)PM2 [ABL'09,ABL'10]
  - Core-guided binary search (w/ disjoint cores) [HMMS'11]
    - Bin-Core, Bin-Core-Dis

Many Core-Guided MaxSAT Algorithms

- Algorithms:
  - (W)MSU1.X/WPM1 [FM'06,MSM'08,MMSP'09,ABL'09]
  - (W)MSU3 [MSP'07]
  - (W)MSU4 [MSP'08]
  - (W)PM2 [ABL'09,ABL'10]
  - Core-guided binary search (w/ disjoint cores) [HMMS'11]
    - Bin-Core, Bin-Core-Dis

- Other properties:

Algorithm	Type	Relaxation	
		Vars p/ Clause	On Demand
(W)MSU1.X/WPM1	UNSAT-SAT	Multiple	Y
(W)MSU3	UNSAT-SAT	Single	Y
(W)MSU4	Refine LB&UB	Single	Y
(W)PM2	UNSAT-SAT	Single	N/Y
Bin-Core	Bin Search	Single	Y
Bin-Core-Dis	Bin Search	Single	Y

An Example: (W)MSU1.X

$x_6 \vee x_2$	$\neg x_6 \vee x_2$	$\neg x_2 \vee x_1$	$\neg x_1$
$\neg x_6 \vee x_8$	$x_6 \vee \neg x_8$	$x_2 \vee x_4$	$\neg x_4 \vee x_5$
$x_7 \vee x_5$	$\neg x_7 \vee x_5$	$\neg x_5 \vee x_3$	$\neg x_3$

Example CNF formula

An Example: (W)MSU1.X

$x_6 \vee x_2$	$\neg x_6 \vee x_2$	$\neg x_2 \vee x_1$	$\neg x_1$
$\neg x_6 \vee x_8$	$x_6 \vee \neg x_8$	$x_2 \vee x_4$	$\neg x_4 \vee x_5$
$x_7 \vee x_5$	$\neg x_7 \vee x_5$	$\neg x_5 \vee x_3$	$\neg x_3$

Formula is UNSAT; OPT $\leq |\varphi| - 1$; Get unsat core

An Example: (W)MSU1.X

$x_6 \vee x_2$	$\neg x_6 \vee x_2$	$\neg x_2 \vee x_1 \vee r_1$	$\neg x_1 \vee r_2$
$\neg x_6 \vee x_8$	$x_6 \vee \neg x_8$	$x_2 \vee x_4 \vee r_3$	$\neg x_4 \vee x_5 \vee r_4$
$x_7 \vee x_5$	$\neg x_7 \vee x_5$	$\neg x_5 \vee x_3 \vee r_5$	$\neg x_3 \vee r_6$

$$\sum_{i=1}^{6} r_i \leq 1$$

Add relaxation variables and AtMost1 constraint

An Example: (W)MSU1.X

$x_6 \vee x_2$	$\neg x_6 \vee x_2$	$\neg x_2 \vee x_1 \vee r_1$	$\neg x_1 \vee r_2$
$\neg x_6 \vee x_8$	$x_6 \vee \neg x_8$	$x_2 \vee x_4 \vee r_3$	$\neg x_4 \vee x_5 \vee r_4$
$x_7 \vee x_5$	$\neg x_7 \vee x_5$	$\neg x_5 \vee x_3 \vee r_5$	$\neg x_3 \vee r_6$
$\sum_{i=1}^{6} r_i \leq 1$			

Formula is (again) UNSAT; $\text{OPT} \leq |\varphi| - 2$; Get unsat core

An Example: (W)MSU1.X

$x_6 \vee x_2 \vee r_7$ $\quad$ $\neg x_6 \vee x_2 \vee r_8$ $\quad$ $\neg x_2 \vee x_1 \vee r_1 \vee r_9$ $\quad$ $\neg x_1 \vee r_2 \vee r_{10}$

$\neg x_6 \vee x_8$ $\quad$ $x_6 \vee \neg x_8$ $\quad$ $x_2 \vee x_4 \vee r_3$ $\quad$ $\neg x_4 \vee x_5 \vee r_4$

$x_7 \vee x_5 \vee r_{11}$ $\quad$ $\neg x_7 \vee x_5 \vee r_{12}$ $\quad$ $\neg x_5 \vee x_3 \vee r_5 \vee r_{13}$ $\quad$ $\neg x_3 \vee r_6 \vee r_{14}$

$\sum_{i=1}^{6} r_i \leq 1$ $\quad$ $\sum_{i=7}^{14} r_i \leq 1$

Add new relaxation variables and AtMost1 constraint

An Example: (W)MSU1.X

$x_6 \vee x_2 \vee r_7$	$\neg x_6 \vee x_2 \vee r_8$	$\neg x_2 \vee x_1 \vee r_1 \vee r_9$	$\neg x_1 \vee r_2 \vee r_{10}$
$\neg x_6 \vee x_8$	$x_6 \vee \neg x_8$	$x_2 \vee x_4 \vee r_3$	$\neg x_4 \vee x_5 \vee r_4$
$x_7 \vee x_5 \vee r_{11}$	$\neg x_7 \vee x_5 \vee r_{12}$	$\neg x_5 \vee x_3 \vee r_5 \vee r_{13}$	$\neg x_3 \vee r_6 \vee r_{14}$
$\sum_{i=1}^{6} r_i \leq 1$	$\sum_{i=7}^{14} r_i \leq 1$		

Instance is now SAT

An Example: (W)MSU1.X

$x_6 \vee x_2 \vee r_7$	$\neg x_6 \vee x_2 \vee r_8$	$\neg x_2 \vee x_1 \vee r_1 \vee r_9$	$\neg x_1 \vee r_2 \vee r_{10}$
$\neg x_6 \vee x_8$	$x_6 \vee \neg x_8$	$x_2 \vee x_4 \vee r_3$	$\neg x_4 \vee x_5 \vee r_4$
$x_7 \vee x_5 \vee r_{11}$	$\neg x_7 \vee x_5 \vee r_{12}$	$\neg x_5 \vee x_3 \vee r_5 \vee r_{13}$	$\neg x_3 \vee r_6 \vee r_{14}$
$\sum_{i=1}^{6} r_i \leq 1$	$\sum_{i=7}^{14} r_i \leq 1$		

MaxSAT solution is $|\varphi| - \mathcal{I} = 12 - 2 = 10$

Organization of MSU1.X

- Clauses characterized as:
  - Soft: initial set of soft clauses
  - Hard: initially hard, or added during execution of algorithm
    - E.g. clauses from AtMost1 constraints
- While exist unsatisfiable cores
  - Add fresh set B of relaxation variables to soft clauses in core
  - Add new AtMost1 constraint

$$\sum_{b_i \in B} b_i \leq 1$$

    - At most 1 relaxation variable from set B can take value 1
- (Partial) MaxSAT solution is $|\varphi| - \mathcal{I}$
  - $\mathcal{I}$: number of iterations ($\equiv$ number of computed unsat cores)

Binary Search For MaxSAT (Bin)

[e.g. FM'06]

$(R, \varphi_W) \leftarrow \text{Relax}(\emptyset, \varphi, \text{Soft}(\varphi))$
$(\lambda, \mu, \mathcal{A}_M) \leftarrow (-1, \sum_{i=1}^{m} w_i + 1, \emptyset)$
while $\lambda < \mu - 1$ **do**
    $\nu \leftarrow \lfloor(\lambda + \mu)/2\rfloor$
    $\varphi_E \leftarrow \text{CNF}(\sum_{r_i \in R} w_i\, r_i \leq \nu)$
    $(\text{st}, \mathcal{A}) \leftarrow \text{SAT}(\varphi_W \cup \varphi_E)$
    if $\text{st} = \mathit{true}$ **then**
        $(\mathcal{A}_M, \mu) \leftarrow (\mathcal{A}, \sum_{i=1}^{m} w_i\, \mathcal{A}\langle r_i \rangle)$
    else
        $\lambda \leftarrow \nu$
return $\text{Init}(\mathcal{A}_M)$

Core-Guided Binary Search (Bin-Core)

[HMMS'11]

$(R, \varphi_W, \varphi_S) \leftarrow (\emptyset, \varphi, \text{Soft}(\varphi))$
$(\lambda, \mu, \mathcal{A}_M) \leftarrow (-1, \sum_{i=1}^{m} w_i + 1, \emptyset)$
while $\lambda < \mu - 1$ **do**
    $\nu \leftarrow \lfloor (\lambda + \mu)/2 \rfloor$
    $\varphi_E \leftarrow \text{CNF}(\sum_{r_i \in R} w_i \, r_i \leq \nu)$
    $(\text{st}, \varphi_C, \mathcal{A}) \leftarrow \text{SAT}(\varphi_W \cup \varphi_E)$
    if $\text{st} = \textit{true}$ **then**
        $(\mathcal{A}_M, \mu) \leftarrow (\mathcal{A}, \sum_{i=1}^{m} w_i \, \mathcal{A}\langle r_i \rangle)$
    else
        if $\varphi_C \cap \varphi_S = \emptyset$ **then**
            $\lambda \leftarrow \nu$
        else
            $(R, \varphi_W) \leftarrow \text{Relax}(R, \varphi_W, \varphi_C \cap \varphi_S)$
return $\text{Init}(\mathcal{A}_M)$

Outline

Boolean-Based Optimization

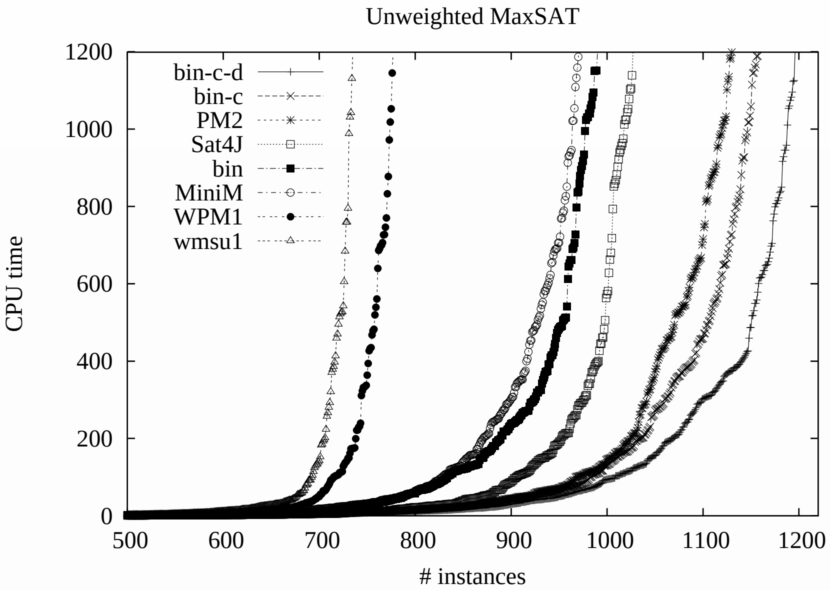
Example Applications

Fundamental Techniques

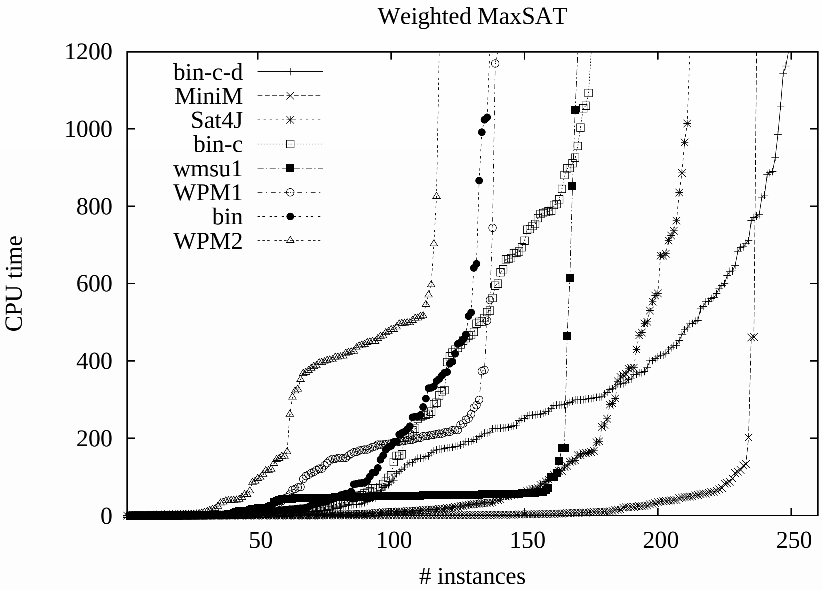
Practical Algorithms

Results, Conclusions & Research Directions

Results for Industrial & Crafted Instances



Results for Industrial & Crafted Instances



Conclusions

- Equivalence between Boolean optimization representations
  - Pseudo-Boolean Optimization (PBO) (or 0-1 ILP)
  - Maximum Satisfiability (MaxSAT)
  - etc.

- Overview of SAT-based Boolean optimization algorithms
  - B&B PBO
  - B&B MaxSAT
  - Iterative SAT solving
  - Core-guided MaxSAT

- Core-guided algorithms exhibit (moderate) performance edge
  - Disclaimer: Industrial & crafted instances from MaxSAT evaluations

Research Directions

- Core-guided MaxSAT algorithms
  - More algorithms?
  - Can we do better than core-guided binary search?
  - Theoretical analysis?
    - Worst-case # of iterations? [HMMS'11]
- MaxSAT vs. MaxSMT
  - Can use the same algorithms
- MaxSAT vs. MaxCSP
  - Effective alternative?
- MaxSAT vs. ILP
  - Complementary approaches?
- More practical applications
  - Recent promising applications
    - Error localization in C code [JM'11]
  - Practical applications drive development of efficient algorithms

Thank You

References – CNF Encodings I

B'68	K. Batcher: Sorting Networks and Their Applications. AFIPS Spring Joint Computing Conference 1968: 307-314
W'98	J. Warners: A Linear-Time Transformation of Linear Inequalities into Conjunctive Normal Form. Inf. Process. Lett. 68(2): 63-69 (1998)
FP'01	A. Frisch, T. Peugniez: Solving Non-Boolean Satisfiability Problems with Stochastic Local Search. IJCAI 2001: 282-290
FS'02	T. Fahle, M. Sellmann: Cost Based Filtering for the Constrained Knapsack Problem. Annals OR 115(1-4): 73-93 (2002)
T'03	M. Trick: A Dynamic Programming Approach for Consistency and Propagation for Knapsack Constraints. Annals OR 118(1-4): 73-84 (2003)
S'03	M. Sellmann: Approximated Consistency for Knapsack Constraints. CP 2003: 679-693
S'05	C. Sinz: Towards an Optimal CNF Encoding of Boolean Cardinality Constraints. CP 2005: 827-831
ES'06	N. Een, N. Sorensson: Translating Pseudo-Boolean Constraints into SAT. JSAT 2(1-4): 1-26 (2006)

References – CNF Encodings II

P'07	S. Prestwich: Variable Dependency in Local Search: Prevention Is Better Than Cure. SAT 2007: 107-120
ANORC'09	R. Asin, R. Nieuwenhuis, A. Oliveras, E. Rodríguez-Carbonell: Cardinality Networks and Their Applications. SAT 2009: 167-180
BBR'09	O. Bailleux, Y. Boufkhad, O. Roussel: New Encodings of Pseudo-Boolean Constraints into CNF. SAT 2009: 181-194
P'09	S. Prestwich: CNF Encodings. Handbook of Satisfiability 2009: 75-97
CZI'10	M. Codish, M. Zazon-Ivry: Pairwise Cardinality Networks. LPAR (Dakar) 2010: 154-172
ANORC'11	R. Asin, R. Nieuwenhuis, A. Oliveras, E. Rodríguez-Carbonell: Cardinality Networks: a theoretical and empirical study. Constraints 16(2): 195-221 (2011)
ANORC'11	I. Abio, R. Nieuwenhuis, A. Oliveras, E. Rodriguez-Carbonell: BDDs for Pseudo-Boolean Constraints - Revisited. SAT 2011

References – B&B MaxSAT

HJ'90 P. Hansen, B. Jaumard: Algorithms for the Maximum Satisfiability Problem. Computing 44: 279-303 (1990)

LMP'05 C.-M. Li, F. Manya, J. Planes: Exploiting Unit Propagation to Compute Lower Bounds in Branch and Bound Max-SAT Solvers. CP 2005: 403-414

HL'06 F. Heras, J. Larrosa: New Inference Rules for Efficient Max-SAT Solving. AAAI 2006

HLO'07 F. Heras, J. Larrosa, A. Oliveras: MiniMaxSat: A New Weighted Max-SAT Solver. SAT 2007: 41-55

LMP'07 C.-M. Li, F. Manya, J. Planes: New Inference Rules for Max-SAT. J. Artif. Intell. Res. (JAIR) 30: 321-359 (2007)

BLM'07 M. Bonet, J. Levy, F. Manya: Resolution for Max-SAT. Artif. Intell. 171(8-9): 606-618 (2007)

HLO'08 F. Heras, J. Larrosa, A. Oliveras: MiniMaxSAT: An Efficient Weighted Max-SAT solver. J. Artif. Intell. Res. (JAIR) 31: 1-32 (2008)

LHG'08 J. Larrosa, F. Heras, S. de Givry: A logical approach to efficient Max-SAT solving. Artif. Intell. 172(2-3): 204-233 (2008)

LM'09 C.-M. Li, F. Manya: MaxSAT, Hard and Soft Constraints. Handbook of Satisfiability 2009: 613-631

References – PBO I

ACLB'96 L. Amgoud, C. Cayrol, D. Le Berre: Comparing Arguments Using Preference Ordering for Argument-Based Reasoning. ICTAI 1996: 400-403

MMS'00 V. Manquinho, J. Marques-Silva: Search Pruning Conditions for Boolean Optimization. ECAI 2000: 103-107

ARMS'02 F. Aloul, A. Ramani, I. Markov, K. Sakallah: Generic ILP versus specialized 0-1 ILP: an update. ICCAD 2002: 450-457

MMS'02 V. Manquinho, J. Marques-Silva: Search pruning techniques in SAT-based branch-and-bound algorithms for the binate covering problem. IEEE Trans. on CAD of Integrated Circuits and Systems 21(5): 505-516 (2002)

CK'03 D. Chai, A. Kuehlmann: A fast pseudo-boolean constraint solver. DAC 2003: 830-835

MMS'04 V. Manquinho, J. Marques-Silva: Satisfiability-Based Algorithms for Boolean Optimization. Ann. Math. Artif. Intell. 40(3-4): 353-372 (2004)

CK'06 D. Chai, A. Kuehlmann: A fast pseudo-Boolean constraint solver. IEEE Trans. on CAD of Integrated Circuits and Systems 24(3): 305-317 (2005)

References – PBO II

MMS'06 V. Manquinho, J. Marques-Silva: On Using Cutting Planes in Pseudo-Boolean Optimization. JSAT 2(1-4): 209-219 (2006)

SS'06 H. Sheini, Karem A. Sakallah: Pueblo: A Hybrid Pseudo-Boolean SAT Solver. JSAT 2(1-4): 165-189 (2006)

ARSM'07 F. Aloul, A. Ramani, K. Sakallah, I. Markov: Solution and Optimization of Systems of Pseudo-Boolean Constraints. IEEE Trans. Computers 56(10): 1415-1424 (2007)

RM'09 O. Roussel, V. Manquinho: Pseudo-Boolean and Cardinality Constraints. Handbook of Satisfiability 2009: 695-733

LBP'10 D. Le Berre, A. Parrain: The Sat4j library, release 2.2. JSAT 7(2-3): 59-6 (2010)

References – MaxSMT

NO'06 R. Nieuwenhuis, A. Oliveras: On SAT Modulo Theories and Optimization Problems. SAT 2006: 156-169

MMS'10 A. Morgado, J. Marques-Silva: Combinatorial Optimization Solutions for the Maximum Quartet Consistency Problem. Fundam. Inform. 102(3-4): 363-389 (2010)

CFGSS'10 A. Cimatti, A. Franzén, A. Griggio, R. Sebastiani, C. Stenico: Satisfiability Modulo the Theory of Costs: Foundations and Applications. TACAS 2010: 99-113

References – Core-Guided Algorithms

FM'06 Z. Fu, S. Malik: On Solving the Partial MAX-SAT Problem. SAT 2006: 252-265

MSP'07 J. Marques-Silva, J. Planes: On Using Unsatisfiability for Solving Maximum Satisfiability CoRR abs/0712.1097: (2007)

MSP'08 J. Marques-Silva, Jordi Planes: Algorithms for Maximum Satisfiability using Unsatisfiable Cores. DATE 2008: 408-413

MSM'08 J. Marques-Silva, V. Manquinho: Towards More Effective Unsatisfiability-Based Maximum Satisfiability Algorithms. SAT 2008: 225-230

MMSP'09 V. Manquinho, J. Marques Silva, J. Planes: Algorithms for Weighted Boolean Optimization. SAT 2009: 495-508

ABL'09 C. Ansotegui, M. Bonet, J. Levy: Solving (Weighted) Partial MaxSAT through Satisfiability Testing. SAT 2009: 427-440

ABL'10 C. Ansotegui, M. Bonet, J. Levy: A New Algorithm for Weighted Partial MaxSAT. AAAI 2010

HMMS'11 F. Heras, A. Morgado, J. Marques-Silva: Core-Guided Binary Search Algorithms for Maximum Satisfiability. AAAI 2011.

References – Applications I

HS'96 G. Hachtel, F. Somenzi: Logic synthesis and verification algorithms. Kluwer 1996

MBCV'06 F. Mancinelli, J. Boender, R. Di Cosmo, J. Vouillon, B. Durak, X. Leroy, R. Treinen: Managing the Complexity of Large Free and Open Source Package-Based Software Distributions. ASE 2006: 199-208

SMVLS'07 S. Safarpour, H. Mangassarian, A. Veneris, M. Liffiton, K. Sakallah: Improved Design Debugging Using Maximum Satisfiability. FMCAD 2007: 13-19

TSJL'07 C. Tucker, D. Shuffelton, R. Jhala, S. Lerner: OPIUM: Optimal Package Install/Uninstall Manager. ICSE 2007: 178-188

HLGS'08 F. Heras, J. Larrosa, S. de Givry, T. Schiex: 2006 and 2007 Max-SAT Evaluations: Contributed Instances. JSAT 4(2-4): 239-250 (2008)

ACFM'08 L. Aksoy, E. Costa, P. Flores, J. Monteiro: Exact and Approximate Algorithms for the Optimization of Area and Delay in Multiple Constant Multiplications. IEEE Trans. on CAD of Integrated Circuits and Systems 27(6): 1013-1026 (2008)

AL'08 J. Argelich, I. Lynce: CNF Instances from the Software Package Installation Problem. RCRA 2008

References – Applications II

ALMS'09 J. Argelich, I. Lynce, J. Marques-Silva: On Solving Boolean Multilevel Optimization Problems. IJCAI 2009: 393-398

CSMSV'10 Y. Chen, S. Safarpour, J. Marques-Silva, A. Veneris: Automated Design Debugging With Maximum Satisfiability. IEEE Trans. on CAD of Integrated Circuits and Systems 29(11): 1804-1817 (2010)

AN'10 R. Asin, R Nieuwenhuis: Curriculum-based Course Timetabling with SAT and MaxSAT. PATAT 2010

ALBL'10 J. Argelich, D. Le Berre, I. Lynce, J. Marques-Silva, P. Rapicault: Solving Linux Upgradeability Problems Using Boolean Optimization. LoCoCo 2010: 11-22

GLMSO'10 A. Graca, I. Lynce, J. Marques-Silva, and A. Oliveira: Efficient and Accurate Haplotype Inference by Combining Parsimony and Pedigree Information. ANB 2010

GMSLO'11 A. Graca, J. Marques-Silva, I. Lynce, A. Oliveira: Haplotype inference with pseudo-Boolean optimization. Annals OR 184(1): 137-162 (2011)

JM'11 M. Jose, R. Majumdar: Cause clue clauses: error localization using maximum satisfiability. PLDI 2011: 437-446