Re-implementing and Extending a Hybrid SAT–IP Approach to Maximum Satisfiability

Paul Saikko

HIIT & Dept. of Computer Science
University of Helsinki
Finland

1st August, 2017

HELSINKI INSTITUTE FOR INFORMATION TECHNOLOGY

Problems

Goal: Find exact solutions to computationally difficult problems

Decision

Determine if a solution exists

Optimization

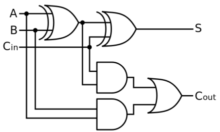
Find, with respect to a given objective function, the best solution

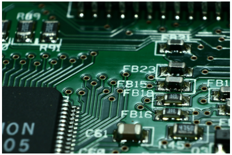
- smallest
- fastest
- cheapest
- most probable
- etc...

Problems

Decision

- Can a given propositional logic formula be satisfied? (SAT) [Cook, 1971]
- Hardware and software verification [Kropf, 2013, Silva et al., 2008]





Optimization

- Determining the locations of production and storage facilities and facility layout optimization [Azadivar and Wang, 2000]
- Scheduling: e.g. air traffic, course times in universities, shifts in workplaces [Lau, 1996]

Motivation

Many problems are NP–hard or harder

Why try to solve them exactly?

Motivation

Many problems are NP–hard or harder

Why try to solve them exactly?

- Exact solutions save time, money, resources
- Algorithms perform much better than worst–case on real–world problems
- Exactly solve simplified problems for better approximations

Declarative programming

Impractical to develop algoritms for every problem and every variation

Solution

1. **Model** problem using a constraint language
2. **Solve** using a generic algorithm (solver) for that constraint language

Benefits

- Easy to reformulate and refine problem definition
- Solver development benefits many different problem domains

Constraint languages

Many approaches to model and solve constrained optimization problems:

- Integer linear programming (IP / LP)
- Finite-domain constraint satisfaction/optimization (CP)
- Boolean satisfiability (SAT)
- Maximum satisfiability (MaxSAT)
- Prolog, Answer set programming (ASP), SMT, etc ...

Constraint languages

Many approaches to model and solve constrained optimization problems:

- **Integer linear programming (IP / LP)**
- Finite-domain constraint satisfaction/optimization (CP)
- **Boolean satisfiability (SAT)**
- **Maximum satisfiability (MaxSAT)**
- Prolog, Answer set programming (ASP), SMT, etc ...

Integer linear programming

Maximize or minimize a linear objective function f:

$$f(x_1, \ldots, x_n) = w_1 x_1 + \cdots + w_n x_n$$

Subject to linear constraints of type:

$$a_1 x_1 + \cdots + a_n x_n \leq k \quad \text{or} \quad a_1 x_1 + \cdots + a_n x_n \geq k$$

NP–hard if we restrict x_i to integer values

Example: Hitting sets

Given a collection of elements U and a set S of sets $s_0, \ldots, s_n \subset U$

A hitting set H of S contains at least one element from each s_i

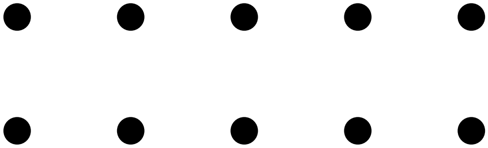
NP–hard problem: Find smallest or minimum-cost H [Karp, 1972]

Example: Hitting sets

Given a collection of elements U and a set S of sets $s_0, \ldots, s_n \subset U$

A hitting set H of S contains at least one element from each s_i

NP–hard problem: Find smallest or minimum-cost H [Karp, 1972]

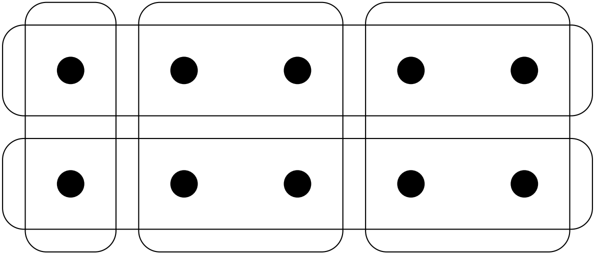


Example: Hitting sets

Given a collection of elements U and a set S of sets $s_0, \ldots, s_n \subset U$

A hitting set H of S contains at least one element from each s_i

NP–hard problem: Find smallest or minimum-cost H [Karp, 1972]

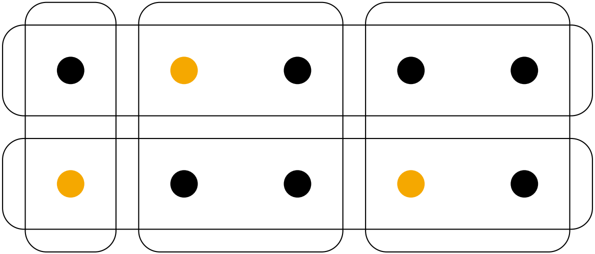


Example: Hitting sets

Given a collection of elements U and a set S of sets $s_0, \ldots, s_n \subset U$

A hitting set H of S contains at least one element from each s_i

NP–hard problem: Find smallest or minimum-cost H [Karp, 1972]



Example: Hitting sets

Minimum hitting set has a simple IP formulation:

For each element e in U, create a binary variable x_e

Meaning: $x_e = 1$ if $e \in H$ otherwise $x_e = 0$

$$\text{minimize } \sum_{e \in U} x_e,$$

Single linear constraint for each s:

$$\text{subject to } \sum_{e \in s} x_e \geq 1 \qquad \forall s \in S$$

Boolean Satisfiability

- First NP–complete problem [Cook, 1971]
- Given a propositional logic formula, does a truth assignment exist that satisfies the formula?
- Polynomial transformation to equivalent conjunctive normal form (CNF) formula [Tseitin, 1983]

Syntax of Boolean logic

- *Variables:* $x_1, x_2, x_3, \ldots$
- *Literals:* variable x_i or its negation $\neg x_i$
- *Clauses:* disjunction (logical OR) of literals
  e.g. $x_1 \vee \neg x_2 \vee x_3$
- *CNF Formula:* conjunction (logical AND) of clauses
  e.g. $(x_1 \vee x_2) \wedge (\neg x_3) \wedge (x_2) \wedge (x_1 \vee \neg x_2 \vee \neg x_3)$

Semantics of Boolean logic

- *Truth assignment:* $\tau : X \to \{0, 1\}$ gives each variable x_i a value of 0 or 1
- *Literals:* x_i is satisfied if $\tau(x_i) = 1$
  $\neg x_i$ is satisfied if $\tau(x_i) = 0$
- *Clauses:* satisfied if at least one of its literals is satisfied
- *CNF Formula:* satisfied if all of its clauses are satisfied

Example

$$F = \begin{array}{l} (x_1 \vee x_2 \vee x_3) \wedge \\ (\neg x_1 \vee x_2 \vee x_3) \wedge \\ (x_1 \vee \neg x_2 \vee x_3) \wedge \\ (x_1 \vee x_2 \vee \neg x_3) \wedge \\ (\neg x_1 \vee \neg x_2 \vee x_3) \wedge \\ (x_1 \vee \neg x_2 \vee \neg x_3) \wedge \\ (\neg x_1 \vee \neg x_2 \vee \neg x_3) \end{array}$$

Example

$$
F = \begin{array}{l}
(x_1 \lor x_2 \lor x_3) \land \\
(\neg x_1 \lor x_2 \lor x_3) \land \\
(x_1 \lor \neg x_2 \lor x_3) \land \\
(x_1 \lor x_2 \lor \neg x_3) \land \\
(\neg x_1 \lor \neg x_2 \lor x_3) \land \\
(x_1 \lor \neg x_2 \lor \neg x_3) \land \\
(\neg x_1 \lor \neg x_2 \lor \neg x_3)
\end{array}
$$

Satisfiable?

Example

$$F = \begin{aligned}&(x_1 \vee x_2 \vee x_3) \wedge \\ &(\neg x_1 \vee x_2 \vee x_3) \wedge \\ &(x_1 \vee \neg x_2 \vee x_3) \wedge \\ &(x_1 \vee x_2 \vee \neg x_3) \wedge \\ &(\neg x_1 \vee \neg x_2 \vee x_3) \wedge \\ &(x_1 \vee \neg x_2 \vee \neg x_3) \wedge \\ &(\neg x_1 \vee \neg x_2 \vee \neg x_3)\end{aligned}$$

Satisfiable?

$\tau : \{x_1 = 1, x_2 = 0, x_3 = 1\}$

Example

$$F = \begin{array}{l} (x_1 \vee x_2 \vee x_3) \wedge \\ (\neg x_1 \vee x_2 \vee x_3) \wedge \\ (x_1 \vee \neg x_2 \vee x_3) \wedge \\ (x_1 \vee x_2 \vee \neg x_3) \wedge \\ (\neg x_1 \vee \neg x_2 \vee x_3) \wedge \\ (x_1 \vee \neg x_2 \vee \neg x_3) \wedge \\ (\neg x_1 \vee \neg x_2 \vee \neg x_3) \end{array}$$

Satisfiable? YES

$\tau : \{x_1 = 1, x_2 = 0, x_3 = 1\}$

CNF Encodings

Clauses are very simple constraints, easy to reason about

More complex constraints must be encoded in CNF form to be used

CNF Encodings

Clauses are very simple constraints, easy to reason about

More complex constraints must be encoded in CNF form to be used

Example: "Exactly one of x_1, x_2, x_3 is true"

CNF Encodings

Clauses are very simple constraints, easy to reason about

More complex constraints must be encoded in CNF form to be used

Example: "Exactly one of x_1, x_2, x_3 is true"

1. $(x_1 \lor x_2 \lor x_3)$ "At least one of x_1, x_2, x_3 is true"

CNF Encodings

Clauses are very simple constraints, easy to reason about

More complex constraints must be encoded in CNF form to be used

Example: "Exactly one of x_1, x_2, x_3 is true"

1. $(x_1 \vee x_2 \vee x_3)$ "At least one of x_1, x_2, x_3 is true"

2. $(\neg x_1 \vee \neg x_2)$
   $(\neg x_2 \vee \neg x_3)$ "At least one of each pair of x_1, x_2, x_3 is false"
   $(\neg x_1 \vee \neg x_3)$

SAT solvers

- SAT solvers very efficient on real-world problems
- Often handle up to millions of variables and clauses
- Constraint driven clause learning (CDCL) algorithm *implicitly* exploits structure
- Solvers provide satisfying assignment or *proof of unsatisfiability*

MaxSAT

An optimization extension of SAT

Given an unsatisfiable formula F, find a truth assignment τ that maximizes the number of satisfied clauses

MaxSAT

An optimization extension of SAT

Given an unsatisfiable formula F, find a truth assignment τ that maximizes the number of satisfied clauses

Example:
$F = (x_1 \vee x_2) \wedge (\neg x_1 \vee x_2) \wedge (x_1 \vee \neg x_2) \wedge (\neg x_1 \vee \neg x_2) \wedge (\neg x_1) \wedge (x_2)$

MaxSAT

An optimization extension of SAT

Given an unsatisfiable formula F, find a truth assignment τ that maximizes the number of satisfied clauses

Example:
$F = (x_1 \vee x_2) \wedge (\neg x_1 \vee x_2) \wedge (x_1 \vee \neg x_2) \wedge (\neg x_1 \vee \neg x_2) \wedge (\neg x_1) \wedge (x_2)$

$\tau : \{x_1 = 0, x_2 = 1\}$

MaxSAT

An optimization extension of SAT

Given an unsatisfiable formula F, find a truth assignment τ that maximizes the number of satisfied clauses

Example:
$F = (x_1 \vee x_2) \wedge (\neg x_1 \vee x_2) \wedge (x_1 \vee \neg x_2) \wedge (\neg x_1 \vee \neg x_2) \wedge (\neg x_1) \wedge (x_2)$

$\tau : \{x_1 = 0, x_2 = 1\}$

MaxSAT

An optimization extension of SAT

Given an unsatisfiable formula F, find a truth assignment τ that maximizes the number of satisfied clauses

Example:
$F = (x_1 \vee x_2) \wedge (\neg x_1 \vee x_2) \wedge (x_1 \vee \neg x_2) \wedge (\neg x_1 \vee \neg x_2) \wedge (\neg x_1) \wedge (x_2)$

$\tau : \{x_1 = 0, x_2 = 1\}$

Variants of MaxSAT

Weighted MaxSAT

- Assign positive weights to clauses
- Maximize the total weight of satisfied clauses

Partial MaxSAT

- Mandatory (hard) and optional (soft) clauses
- Maximize the number of satisfied soft clauses such that **all** hard clauses are satisfied

Applications

Recently MaxSAT has been successfully utilized in many problem domains.

- design debugging [Chen et al., 2009]
- software dependencies [Argelich et al., 2010]
- data visualization [Bunte et al., 2014]
- causal discovery [Hyttinen et al., 2014]
- model–based diagnosis [Marques-Silva et al., 2015]
- abstract argumentation [Wallner et al., 2016]
- correlation clustering [Berg and Järvisalo, 2017]
- and more ...

Unsatisfiable cores

A subset of clauses κ of a formula F, which cannot be satisfied by the same truth assignment.

Found by SAT solver if formula unsatisfiable.

Unsatisfiable cores

A subset of clauses κ of a formula F, which cannot be satisfied by the same truth assignment.

Found by SAT solver if formula unsatisfiable.

Example:
$F = (\neg x_1 \vee x_2) \wedge (\neg x_1 \vee \neg x_2) \wedge (x_1 \vee \neg x_2) \wedge (x_1 \vee x_2) \wedge (x_1) \wedge (x_2)$

Unsatisfiable cores

A subset of clauses κ of a formula F, which cannot be satisfied by the same truth assignment.

Found by SAT solver if formula unsatisfiable.

Example:
$F = (\neg x_1 \vee x_2) \wedge (\neg x_1 \vee \neg x_2) \wedge (x_1 \vee \neg x_2) \wedge (x_1 \vee x_2) \wedge (x_1) \wedge (x_2)$

Has (minimal) cores:

- $\{(\neg x_1 \vee x_2), (\neg x_1 \vee \neg x_2), (x_1 \vee \neg x_2), (x_1 \vee x_2)\}$
- $\{(\neg x_1 \vee x_2), (\neg x_1 \vee \neg x_2), (x_1)\}$
- $\{(\neg x_1 \vee \neg x_2), (x_1 \vee \neg x_2), (x_2)\}$

Solving (plain) MaxSAT with SAT solvers

Bounds-based algorithm (e.g. in [Martins et al., 2014])

1. Encode "k clauses in formula can be satisfied" as CNF
2. SAT solve original formula F with above constraints
   - Satisfiable? Increase k
   - Unsatisfiable? Decrease k
3. Repeat until largest satisfiable k found

Solving (plain) MaxSAT with SAT solvers

Bounds-based algorithm (e.g. in [Martins et al., 2014])

1. Encode "k clauses in formula can be satisfied" as CNF
2. SAT solve original formula F with above constraints
   - Satisfiable? Increase k
   - Unsatisfiable? Decrease k
3. Repeat until largest satisfiable k found

Core-based algorithm (e.g. [Fu and Malik, 2006])

1. SAT solve the formula F
   - Satisfiable? Optimum found
   - Unsatisfiable? Get a core κ
2. *Relax* F such that exactly one clause in κ can be left unsatified
3. Repeat until satisfiable

MaxSAT algorithms

SAT–based algorithms?

- Deal poorly with diverse clause weights
- SAT formula grows as constraints added or formula relaxed

MaxSAT algorithms

SAT–based algorithms?

- Deal poorly with diverse clause weights
- SAT formula grows as constraints added or formula relaxed

Rewrite formula as IP problem?

- Natively handles weighted objective functions
- IP solvers poorly suited to proving unsatisfiability

MaxSAT algorithms

SAT–based algorithms?

- Deal poorly with diverse clause weights
- SAT formula grows as constraints added or formula relaxed

Rewrite formula as IP problem?

- Natively handles weighted objective functions
- IP solvers poorly suited to proving unsatisfiability

Best of both worlds?

- Implicit hitting set algorithm [Moreno-Centeno and Karp, 2013] for MaxSAT [Davies, 2013]

Solutions, cores and hitting sets

- A MaxSAT solution cannot satisfy every clause in any core

Solutions, cores and hitting sets

- A MaxSAT solution cannot satisfy every clause in any core
- For every core, a solution leaves at least one clause unsatisfied

Solutions, cores and hitting sets

- A MaxSAT solution cannot satisfy every clause in any core
- For every core, a solution leaves at least one clause unsatisfied
- Unsatisfied clauses form a hitting set of the set of all cores K

Solutions, cores and hitting sets

- A MaxSAT solution cannot satisfy every clause in any core
- For every core, a solution leaves at least one clause unsatisfied
- Unsatisfied clauses form a hitting set of the set of all cores K
- If the solution is optimal, this is a **minimum hitting set**

Implicit hitting set algorithm

Do we need the set of all cores K?

- Enough to find large enough $K' \subset K$ that K' has same minimum hitting set H

Implicit hitting set algorithm

Do we need the set of all cores K?

- Enough to find large enough $K' \subset K$ that K' has same minimum hitting set H

How do we know if we have enough cores?

- Test satisfiability of $F \setminus H$
- If satisfiable, all cores are hit by H

Implicit hitting set algorithm

Do we need the set of all cores K?

- Enough to find large enough $K' \subset K$ that K' has same minimum hitting set H

How do we know if we have enough cores?

- Test satisfiability of $F \setminus H$
- If satisfiable, all cores are hit by H

IHS algorithm loop

Repeat:

1. SAT solve $F \setminus H$
   - Satisfiable? Optimal solution found
   - Unsatisfiable? Add core κ to K
2. $H \leftarrow \mathit{MinimumCostHittingSet}(K)$

Example

Input: $\mathrm{F} = (\neg x_1 \lor x_2, 7) \land (\neg x_1 \lor \neg x_2, 8) \land (x_1 \lor \neg x_2, 7) \land (x_1 \lor x_2, 3) \land (x_1, 3) \land (x_2, 3)$

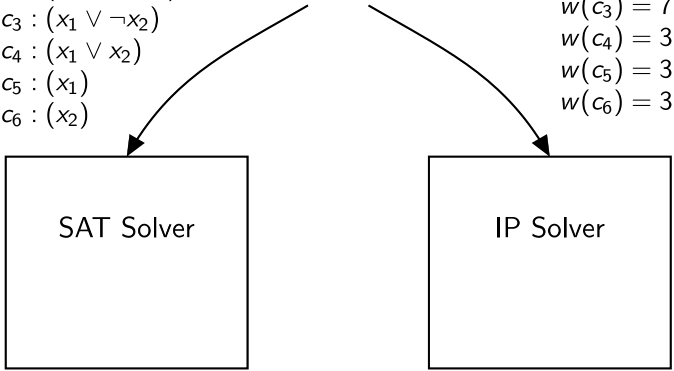
SAT Solver

IP Solver

Example

$c_1 : (\neg x_1 \vee x_2)$
$c_2 : (\neg x_1 \vee \neg x_2)$
$c_3 : (x_1 \vee \neg x_2)$
$c_4 : (x_1 \vee x_2)$
$c_5 : (x_1)$
$c_6 : (x_2)$

$w(c_1) = 7$
$w(c_2) = 8$
$w(c_3) = 7$
$w(c_4) = 3$
$w(c_5) = 3$
$w(c_6) = 3$



Example

$$w(c_1) = 7$$
$$w(c_2) = 8$$
$$w(c_3) = 7$$
$$w(c_4) = 3$$
$$w(c_5) = 3$$
$$w(c_6) = 3$$

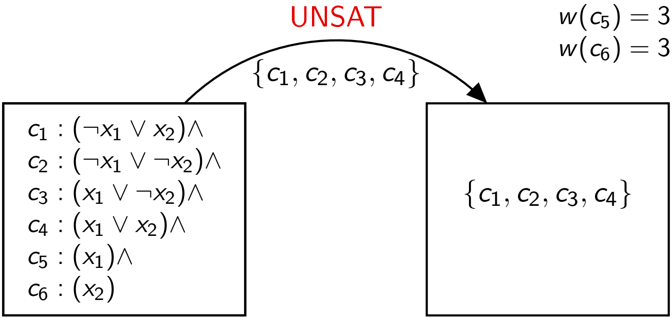
SAT?

$c_1 : (\neg x_1 \lor x_2) \land$
$c_2 : (\neg x_1 \lor \neg x_2) \land$
$c_3 : (x_1 \lor \neg x_2) \land$
$c_4 : (x_1 \lor x_2) \land$
$c_5 : (x_1) \land$
$c_6 : (x_2)$

IP Solver

Example

$w(c_1) = 7$
$w(c_2) = 8$
$w(c_3) = 7$
$w(c_4) = 3$
$w(c_5) = 3$
$w(c_6) = 3$



Example

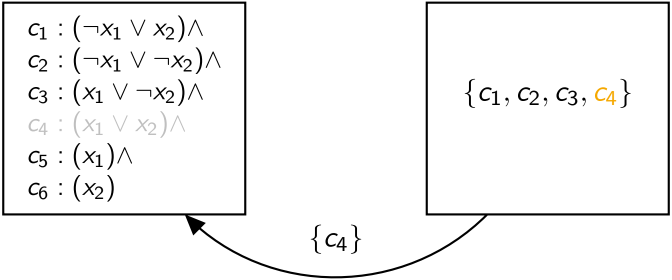
$$w(c_1) = 7$$
$$w(c_2) = 8$$
$$w(c_3) = 7$$
$$w(c_4) = 3$$
$$w(c_5) = 3$$
$$w(c_6) = 3$$

$c_1 : (\neg x_1 \vee x_2) \wedge$
$c_2 : (\neg x_1 \vee \neg x_2) \wedge$
$c_3 : (x_1 \vee \neg x_2) \wedge$
$c_4 : (x_1 \vee x_2) \wedge$
$c_5 : (x_1) \wedge$
$c_6 : (x_2)$

$\{c_1, c_2, c_3, c_4\}$ OPT?

Example

$w(c_1) = 7$
$w(c_2) = 8$
$w(c_3) = 7$
$w(c_4) = 3$
$w(c_5) = 3$
$w(c_6) = 3$



Example

$w(c_1) = 7$
$w(c_2) = 8$
$w(c_3) = 7$
$w(c_4) = 3$
$w(c_5) = 3$
$w(c_6) = 3$

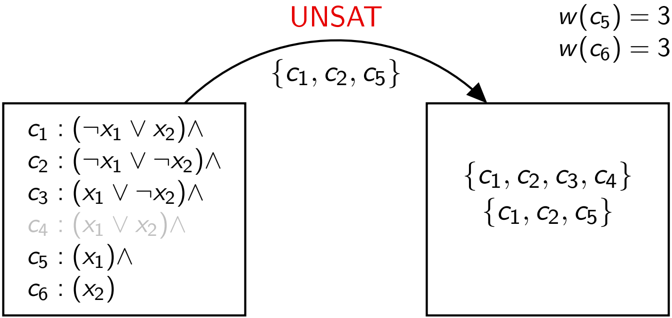
SAT?

$c_1 : (\neg x_1 \vee x_2) \wedge$
$c_2 : (\neg x_1 \vee \neg x_2) \wedge$
$c_3 : (x_1 \vee \neg x_2) \wedge$
$c_4 : (x_1 \vee x_2) \wedge$
$c_5 : (x_1) \wedge$
$c_6 : (x_2)$

$\{c_1, c_2, c_3, c_4\}$

Example

$w(c_1) = 7$
$w(c_2) = 8$
$w(c_3) = 7$
$w(c_4) = 3$
$w(c_5) = 3$
$w(c_6) = 3$



Example

$$w(c_1) = 7$$
$$w(c_2) = 8$$
$$w(c_3) = 7$$
$$w(c_4) = 3$$
$$w(c_5) = 3$$
$$w(c_6) = 3$$

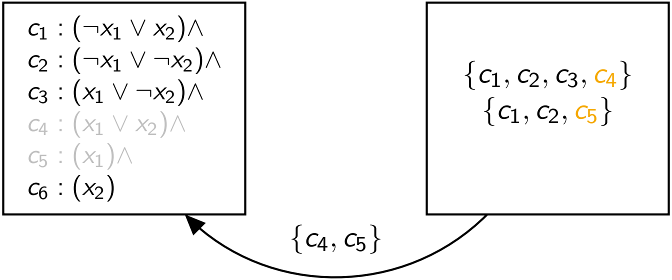
$c_1 : (\neg x_1 \vee x_2) \wedge$
$c_2 : (\neg x_1 \vee \neg x_2) \wedge$
$c_3 : (x_1 \vee \neg x_2) \wedge$
$c_4 : (x_1 \vee x_2) \wedge$
$c_5 : (x_1) \wedge$
$c_6 : (x_2)$

$\{c_1, c_2, c_3, c_4\}$
$\{c_1, c_2, c_5\}$

OPT?

Example

$$w(c_1) = 7$$
$$w(c_2) = 8$$
$$w(c_3) = 7$$
$$w(c_4) = 3$$
$$w(c_5) = 3$$
$$w(c_6) = 3$$



Example

$$w(c_1) = 7$$
$$w(c_2) = 8$$
$$w(c_3) = 7$$
$$w(c_4) = 3$$
$$w(c_5) = 3$$
$$w(c_6) = 3$$

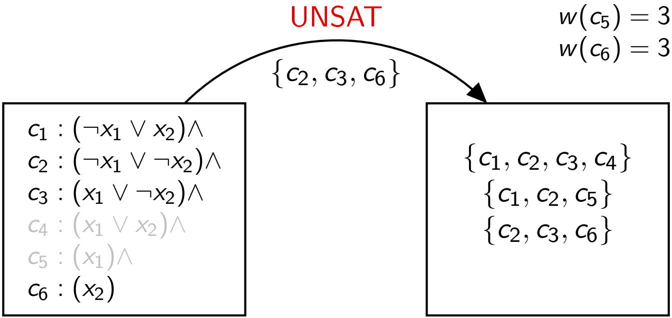
SAT?

$c_1 : (\neg x_1 \vee x_2) \wedge$
$c_2 : (\neg x_1 \vee \neg x_2) \wedge$
$c_3 : (x_1 \vee \neg x_2) \wedge$
$c_4 : (x_1 \vee x_2) \wedge$
$c_5 : (x_1) \wedge$
$c_6 : (x_2)$

$\{c_1, c_2, c_3, c_4\}$
$\{c_1, c_2, c_5\}$

Example

$w(c_1) = 7$
$w(c_2) = 8$
$w(c_3) = 7$
$w(c_4) = 3$
$w(c_5) = 3$
$w(c_6) = 3$



Example

$$w(c_1) = 7$$
$$w(c_2) = 8$$
$$w(c_3) = 7$$
$$w(c_4) = 3$$
$$w(c_5) = 3$$
$$w(c_6) = 3$$

$c_1 : (\neg x_1 \vee x_2) \wedge$
$c_2 : (\neg x_1 \vee \neg x_2) \wedge$
$c_3 : (x_1 \vee \neg x_2) \wedge$
$c_4 : (x_1 \vee x_2) \wedge$
$c_5 : (x_1) \wedge$
$c_6 : (x_2)$

$\{c_1, c_2, c_3, c_4\}$
$\{c_1, c_2, c_5\}$
$\{c_2, c_3, c_6\}$

OPT?

Example

$$w(c_1) = 7$$
$$w(c_2) = 8$$
$$w(c_3) = 7$$
$$w(c_4) = 3$$
$$w(c_5) = 3$$
$$w(c_6) = 3$$

$c_1 : (\neg x_1 \vee x_2) \wedge$
$c_2 : (\neg x_1 \vee \neg x_2) \wedge$
$c_3 : (x_1 \vee \neg x_2) \wedge$
$c_4 : (x_1 \vee x_2) \wedge$
$c_5 : (x_1) \wedge$
$c_6 : (x_2)$

$\{c_1, c_2, c_3, c_4\}$
$\{c_1, c_2, c_5\}$
$\{c_2, c_3, c_6\}$

$\{c_2\}$

Example

$$w(c_1) = 7$$
$$w(c_2) = 8$$
$$w(c_3) = 7$$
$$w(c_4) = 3$$
$$w(c_5) = 3$$
$$w(c_6) = 3$$

SAT?

$c_1 : (\neg x_1 \vee x_2) \wedge$
$c_2 : (\neg x_1 \vee \neg x_2) \wedge$
$c_3 : (x_1 \vee \neg x_2) \wedge$
$c_4 : (x_1 \vee x_2) \wedge$
$c_5 : (x_1) \wedge$
$c_6 : (x_2)$

$\{c_1, c_2, c_3, c_4\}$
$\{c_1, c_2, c_5\}$
$\{c_2, c_3, c_6\}$

Example

$w(c_1) = 7$
$w(c_2) = 8$
$w(c_3) = 7$
$w(c_4) = 3$
$w(c_5) = 3$
$w(c_6) = 3$

$c_1 : (\neg x_1 \lor x_2) \land$
$c_2 : (\neg x_1 \lor \neg x_2) \land$
$c_3 : (x_1 \lor \neg x_2) \land$
$c_4 : (x_1 \lor x_2) \land$
$c_5 : (x_1) \land$
$c_6 : (x_2)$

$\{c_1, c_2, c_3, c_4\}$
$\{c_1, c_2, c_5\}$
$\{c_2, c_3, c_6\}$

SAT

$\{x_1 = 1, x_2 = 1\}$
$cost = 8$

Output

M.Sc. Thesis work

LMHS Solver [Saikko et al., 2016a]

- Implement implicit hitting set algorithm for MaxSAT from scratch.
- MiniSat as SAT solver
- IBM CPLEX as IP solver

M.Sc. Thesis work

LMHS Solver [Saikko et al., 2016a]

- Implement implicit hitting set algorithm for MaxSAT from scratch.
- MiniSat as SAT solver
- IBM CPLEX as IP solver

MaxSAT Evaluations

Entered in 2015, 2016, 2017 international evaluations of state-of-the-art MaxSAT solvers

- 2015: 1st (of 29) in both categories of weighted partial MaxSAT
- 2016: 2nd and 3rd

Going further...

LMHS solver development has led to:

- In thesis: LMHS incremental API used to solve sub–problems in Bayesian network structure solver
- IJCAI'15: Integrated MaxSAT preprocessing [Berg et al., 2015]
- KR'16: Implicit hitting–set approach extended to abductive reasoning [Saikko et al., 2016b]
- CP'17: Use IP technique of reduced–cost fixing in the algorithm to simplify the problem during search [Bacchus et al., 2017]
- IJCAI'17: Domain–specific application for learning optimal causal graphs [Hyttinen et al., 2017]

Summary

1. Constrained optimization problems

Summary

1. Constrained optimization problems
2. Boolean logic and satisfiability

Summary

1. Constrained optimization problems
2. Boolean logic and satisfiability
3. MaxSAT

Summary

1. Constrained optimization problems
2. Boolean logic and satisfiability
3. MaxSAT
4. Implicit hitting set algorithms

Summary

1. Constrained optimization problems
2. Boolean logic and satisfiability
3. MaxSAT
4. Implicit hitting set algorithms
5. The LMHS solver and recent work

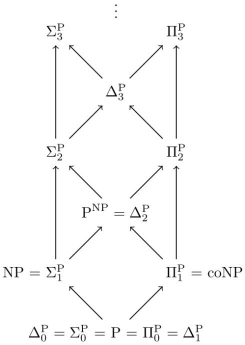
Thanks

Questions?

Slides with complete references at
http://cs.helsinki.fi/u/psaikko/msc-slides.pdf

Extension to abductive reasoning

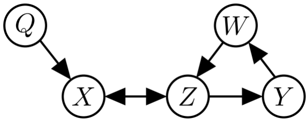
- Logical reasoning problem:
- Given a theory T, set of possible hypothesis H, observations M:
  Find a subset of H that is consistent with T and entails M.
- Σ_2^P–complete, harder than NP
- Extend IHS algorithm with two–phase core extraction
- KR paper [Saikko et al., 2016b]



Core-Guided Approach to Learning Optimal Causal Graphs

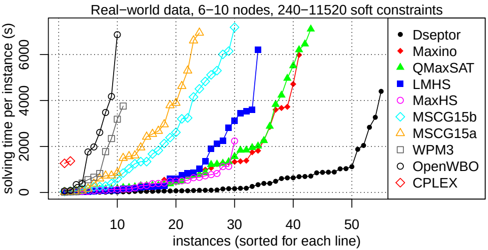
Dseptor Solver

- LMHS with domain–specific features
- Improves on state-of-the-art performance
- IJCAI paper [Hyttinen et al., 2017]



Domain–specific improvements

- Precomputed cores
- Tighter bounds from underlying graph
- Core extraction heuristics



References I

Argelich, J., Berre, D. L., Lynce, I., Silva, J. P. M., and Rapicault, P. (2010).
Solving linux upgradeability problems using boolean optimization.
In *Proc. LoCoCo*, pages 11–22.

Azadivar, F. and Wang, J. (2000).
Facility layout optimization using simulation and genetic algorithms.
International Journal of Production Research, 38(17):4369–4383.

Bacchus, F., Hyttinen, A., Järvisalo, M., and Saikko, P. (2017).
Reduced cost fixing in maxsat.
In *Proc. CP*.
To appear.

Berg, J. and Järvisalo, M. (2017).
Cost-optimal constrained correlation clustering via weighted partial maximum satisfiability.
Artif. Intell., 244:110–142.

Berg, J., Saikko, P., and Järvisalo, M. (2015).
Improving the effectiveness of SAT-based preprocessing for MaxSAT.
In *Proc. IJCAI*, pages 239–245. AAAI Press.

Bunte, K., Järvisalo, M., Berg, J., Myllymäki, P., Peltonen, J., and Kaski, S. (2014).
Optimal neighborhood preserving visualization by maximum satisfiability.
In *Proc. AAAI*, pages 1694–1700.

Chen, Y., Safarpour, S., Veneris, A. G., and Silva, J. P. M. (2009).
Spatial and temporal design debug using partial maxsat.
In *Proc. GLSVLSI*, pages 345–350.

References II

Cook, S. A. (1971).
The complexity of theorem-proving procedures.
In *Proc. STOC*, pages 151–158. ACM.

Davies, J. (2013).
Solving MAXSAT by Decoupling Optimization and Satisfaction.
PhD thesis, University of Toronto.

Fu, Z. and Malik, S. (2006).
On solving the partial MAX-SAT problem.
In *Proc. SAT*, volume 4121 of *LNCS*, pages 252–265. Springer.

Hyttinen, A., Eberhardt, F., and Järvisalo, M. (2014).
Constraint-based causal discovery: Conflict resolution with answer set programming.
In *Proc. UAI*, pages 340–349.

Hyttinen, A., Saikko, P., and Järvisalo, M. (2017).
A core-guided approach to learning optimal causal graphs.
In *Proc. IJCAI*, pages 645–651. AAAI Press.

Karp, R. M. (1972).
Reducibility among combinatorial problems.
In *Proc. Complexity of Computer Computations*, The IBM Research Symposia Series, pages 85–103. Plenum Press.

Kropf, T. (2013).
Introduction to Formal Hardware Verification.
Springer Science & Business Media.

References III

Lau, H. C. (1996).
On the complexity of manpower shift scheduling.
Computers & Operations Research, 23(1):93–102.

Marques-Silva, J., Janota, M., Ignatiev, A., and Morgado, A. (2015).
Efficient model based diagnosis with maximum satisfiability.
In *Proc. IJCAI*, pages 1966–1972.

Martins, R., Manquinho, V. M., and Lynce, I. (2014).
Open-WBO: A modular MaxSAT solver,.
In *Proc. SAT*, volume 8561 of *LNCS*, pages 438–445. Springer.

Moreno-Centeno, E. and Karp, R. M. (2013).
The implicit hitting set approach to solve combinatorial optimization problems with an application to multigenome alignment.
Operations Research, 61(2):453–468.

Saikko, P., Berg, J., and Järvisalo, M. (2016a).
LMHS: A SAT-IP hybrid maxsat solver.
In *Proc. SAT*, pages 539–546.

Saikko, P., Wallner, J. P., and Järvisalo, M. (2016b).
Implicit hitting set algorithms for reasoning beyond NP.
In *Proc. KR*, pages 104–113.

Silva, V. D., Kroening, D., and Weissenbacher, G. (2008).
A survey of automated techniques for formal software verification.
IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 27(7):1165–1178.

References IV

Tseitin, G. S. (1983).
On the complexity of derivation in propositional calculus.
In *Automation of Reasoning*, pages 466–483. Springer.

Wallner, J. P., Niskanen, A., and Järvisalo, M. (2016).
Complexity results and algorithms for extension enforcement in abstract argumentation.
In *Proc. AAAI*, pages 1088–1094.