Incremental Maximum Satisfiability

Matti Järvisalo

University of Helsinki, Finland

joint work with Andreas Niskanen and Jeremias Berg

October 3 @ Shonan Art of SAT Meeting, Japan

Takeaways

- Need for incremental optimization techniques motivated by applications
- SAT-based MaxSAT approaches hold promise for enabling high levels of incrementality
- Promising avenue for further work!
  - Solvers & applications
  - Figuring out the "right" combinations of solving techniques & what make sense / can/cannot be made incremental
  - IPAMIR interface for solvers & applications

Incremental optimization *(a.k.a. reoptimization)*

- Various problem domains call for iterative solving procedures where **a sequence of related instances are solved**
  - types of incremental changes applied between instances:
    - adding, removing, or strengthening constraints
    - modifying objective function
- Solving each instance from scratch often too costly: aim to **reuse information obtained during previous calls**

In This Talk

Overview of recent developments in **Incremental MaxSAT**
Niskanen, Berg, and Järvisalo [2021, 2022], MaxSAT Eval 2022 + applications

- Unsatisfiability-based optimization:
  Particularly suited for incrementality (?)

- Forms of incrementality
- API for incremental MaxSAT solvers and their applications
- Application case studies
- Incremental IHS MaxSAT solving

Maximum satisfiability (MaxSAT)

Bacchus, Järvisalo, and Martins [2021]

- Optimization paradigm based on Boolean satisfiability (SAT)
  - *minimize:* linear objective function over 0-1 variables
  - *subject to:* constraints expressed in propositional logic
- Suitable **declarative modelling language** for various real-world optimization problems involving **logical constraints**
- Significant **progress in solving technology** over the past 10 years
  - state-of-the-art solvers build on the success of SAT solvers

Key to Success of MaxSAT

Ability of SAT solvers to efficiently **explain** unsatisfiability

Incremental MaxSAT

- Incremental SAT solving well-established Eén and Sörensson [2003]
  - extensively applied by MaxSAT solvers
- Application scenarios for incremental MaxSAT known, but...
- Currently **MaxSAT solvers offer limited support** for incrementality

Lifting Incrementality to MaxSAT

- Aim for solving a sequence of related MaxSAT instances efficiently, avoiding computation from scratch
- Different scenarios call for different forms of incremental changes
  - adding or removing hard constraints
  - modifying the objective function
  - solving under assumptions: partial assignments to variables

Adding Constraints

Consider an initial problem instance, and an iterative procedure:

- Compute an optimal solution to the current instance
- Check whether it satisfies a desired property:
  if not, exclude it (and other non-solutions) from consideration

Generic paradigm: **Counterexample-guided abstraction refinement** *with various instantiations employing MaxSAT*

Mangal, Zhang, Nori, and Naik [2015]; Niskanen and Järvisalo [2020]

$$\begin{aligned} \text{minimize:} \quad & x + 2y \\ \text{subject to:} \quad & x + y \geq 1 \\ & y + (1 - z) \geq 1 \end{aligned}$$

Adding Constraints

Consider an initial problem instance, and an iterative procedure:

- Compute an optimal solution to the current instance
- Check whether it satisfies a desired property:
  if not, exclude it (and other non-solutions) from consideration

Generic paradigm: **Counterexample-guided abstraction refinement** *with various instantiations employing MaxSAT*

Mangal, Zhang, Nori, and Naik [2015]; Niskanen and Järvisalo [2020]

$$\begin{aligned} \text{minimize:} \quad & x + 2y \\ \text{subject to:} \quad & x + y \geq 1 \\ & y + (1 - z) \geq 1 \end{aligned}$$

Adding Constraints

Consider an initial problem instance, and an iterative procedure:

- Compute an optimal solution to the current instance
- Check whether it satisfies a desired property:
  if not, exclude it (and other non-solutions) from consideration

Generic paradigm: **Counterexample-guided abstraction refinement** *with various instantiations employing MaxSAT*

Mangal, Zhang, Nori, and Naik [2015]; Niskanen and Järvisalo [2020]

$$\begin{aligned} \text{minimize:} \quad & x + 2y \\ \text{subject to:} \quad & x + y \geq 1 \\ & y + (1 - z) \geq 1 \end{aligned}$$

Optimal solution:
$x = 1,\ y = 0,\ z = 0$

Adding Constraints

Consider an initial problem instance, and an iterative procedure:

- Compute an optimal solution to the current instance
- Check whether it satisfies a desired property:
  if not, exclude it (and other non-solutions) from consideration

Generic paradigm: **Counterexample-guided abstraction refinement** *with various instantiations employing MaxSAT*

Mangal, Zhang, Nori, and Naik [2015]; Niskanen and Järvisalo [2020]

$$\begin{aligned} \text{minimize:} \quad & x + 2y \\ \text{subject to:} \quad & x + y \geq 1 \\ & y + (1 - z) \geq 1 \\ & (1 - x) + y + z \geq 1 \end{aligned}$$

~~Optimal solution~~:
$x = 1,\ y = 0,\ z = 0$

Adding Constraints

Consider an initial problem instance, and an iterative procedure:

- Compute an optimal solution to the current instance
- Check whether it satisfies a desired property: if not, exclude it (and other non-solutions) from consideration

Generic paradigm: **Counterexample-guided abstraction refinement** *with various instantiations employing MaxSAT*

Mangal, Zhang, Nori, and Naik [2015]; Niskanen and Järvisalo [2020]

$$\begin{aligned} \text{minimize:} \quad & x + 2y \\ \text{subject to:} \quad & x + y \geq 1 \\ & y + (1 - z) \geq 1 \\ & (1 - x) + y + z \geq 1 \end{aligned}$$

Optimal solution:
$x = 0,\ y = 1,\ z = 1$

Changes to Objective Function Coefficients

Consider an initial problem instance, and an iterative procedure:

- Compute an optimal solution to the current instance
- Give more priority to more diverse solutions and repeat

For example: Learning classifiers with the AdaBoost algorithm, MaxSAT employed for decision trees *Hu, Siala, Hebrard, and Huguet [2020]*

$$\begin{aligned} \text{minimize:} \quad & x + 2y \\ \text{subject to:} \quad & x + y \geq 1 \\ & y + (1 - z) \geq 1 \end{aligned}$$

Changes to Objective Function Coefficients

Consider an initial problem instance, and an iterative procedure:

- Compute an optimal solution to the current instance
- Give more priority to more diverse solutions and repeat

For example: Learning classifiers with the AdaBoost algorithm, MaxSAT employed for decision trees *Hu, Siala, Hebrard, and Huguet [2020]*

$$\begin{aligned} \text{minimize:} \quad & x + 2y \\ \text{subject to:} \quad & x + y \geq 1 \\ & y + (1 - z) \geq 1 \end{aligned}$$

Changes to Objective Function Coefficients

Consider an initial problem instance, and an iterative procedure:

- Compute an optimal solution to the current instance
- Give more priority to more diverse solutions and repeat

For example: Learning classifiers with the AdaBoost algorithm, MaxSAT employed for decision trees *Hu, Siala, Hebrard, and Huguet [2020]*

$$\begin{aligned} \text{minimize:} \quad & x + 2y \\ \text{subject to:} \quad & x + y \geq 1 \\ & y + (1 - z) \geq 1 \end{aligned}$$

Optimal solution:
$x = 1$, $y = 0$, $z = 0$

Changes to Objective Function Coefficients

Consider an initial problem instance, and an iterative procedure:

- Compute an optimal solution to the current instance
- Give more priority to more diverse solutions and repeat

For example: Learning classifiers with the AdaBoost algorithm, MaxSAT employed for decision trees Hu, Siala, Hebrard, and Huguet [2020]

minimize: $3x + y$

subject to: $x + y \geq 1$

$y + (1 - z) \geq 1$

~~Optimal~~ solution:

$x = 1,\ y = 0,\ z = 0$

Changes to Objective Function Coefficients

Consider an initial problem instance, and an iterative procedure:

- Compute an optimal solution to the current instance
- Give more priority to more diverse solutions and repeat

For example: Learning classifiers with the AdaBoost algorithm, MaxSAT employed for decision trees *Hu, Siala, Hebrard, and Huguet [2020]*

$$\begin{aligned} \text{minimize:} \quad & 3x + y \\ \text{subject to:} \quad & x + y \geq 1 \\ & y + (1 - z) \geq 1 \end{aligned}$$

Optimal solution:
$x = 0,\ y = 1,\ z = 1$

Optimizing under Different Assumptions

Consider an initial problem instance, and an iterative procedure:

- Extract information about the current state of the world
- Incorporate it to the instance and compute an optimal solution

Example: Timetabling under disruptions, time or room slots may become unavailable

Lemos, Monteiro, and Lynce [2020]

$$\begin{aligned} \text{minimize:} \quad & x + 2y \\ \text{subject to:} \quad & x + y \geq 1 \\ & y + (1 - z) \geq 1 \end{aligned}$$

- Unlike hard constraints, assumptions are revertable
  - removal of hard constraints can be simulated with assumptions

Optimizing under Different Assumptions

Consider an initial problem instance, and an iterative procedure:

- Extract information about the current state of the world
- Incorporate it to the instance and compute an optimal solution

Example: Timetabling under disruptions, time or room slots may become unavailable

Lemos, Monteiro, and Lynce [2020]

$$\begin{aligned} \text{minimize:} \quad & x + 2y \\ \text{subject to:} \quad & x + y \geq 1 \\ & y + (1 - z) \geq 1 \end{aligned}$$

- Unlike hard constraints, assumptions are revertable
  - removal of hard constraints can be simulated with assumptions

Optimizing under Different Assumptions

Consider an initial problem instance, and an iterative procedure:

- Extract information about the current state of the world
- Incorporate it to the instance and compute an optimal solution

Example: Timetabling under disruptions, time or room slots may become unavailable

Lemos, Monteiro, and Lynce [2020]

$$\begin{aligned} \text{minimize:} \quad & x + 2y \\ \text{subject to:} \quad & x + y \geq 1 \\ & y + (1 - z) \geq 1 \end{aligned}$$

Optimal solution:
$x = 1,\ y = 0,\ z = 0$

- Unlike hard constraints, assumptions are revertable
  - removal of hard constraints can be simulated with assumptions

Optimizing under Different Assumptions

Consider an initial problem instance, and an iterative procedure:

- Extract information about the current state of the world
- Incorporate it to the instance and compute an optimal solution

Example: Timetabling under disruptions, time or room slots may become unavailable

Lemos, Monteiro, and Lynce [2020]

minimize: $x + 2y$

subject to: $x + y \geq 1$

$y + (1 - z) \geq 1$

assuming: $z = 1$

~~Optimal solution~~:
$x = 1,\ y = 0,\ z = 0$

- Unlike hard constraints, assumptions are revertable
  - removal of hard constraints can be simulated with assumptions

Optimizing under Different Assumptions

Consider an initial problem instance, and an iterative procedure:

- Extract information about the current state of the world
- Incorporate it to the instance and compute an optimal solution

Example: Timetabling under disruptions, time or room slots may become unavailable

Lemos, Monteiro, and Lynce [2020]

$$\begin{aligned} \text{minimize:} \quad & x + 2y \\ \text{subject to:} \quad & x + y \geq 1 \\ & y + (1 - z) \geq 1 \\ \text{assuming:} \quad & z = 1 \end{aligned}$$

Optimal solution:
$x = 0,\ y = 1,\ z = 1$

- Unlike hard constraints, assumptions are revertable
  - removal of hard constraints can be simulated with assumptions

Optimizing under Different Assumptions

Consider an initial problem instance, and an iterative procedure:

- Extract information about the current state of the world
- Incorporate it to the instance and compute an optimal solution

Example: Timetabling under disruptions, time or room slots may become unavailable

Lemos, Monteiro, and Lynce [2020]

$$\begin{aligned} \text{minimize:} \quad & x + 2y \\ \text{subject to:} \quad & x + y \geq 1 \\ & y + (1 - z) \geq 1 \\ \text{assuming:} \quad & z = 1 \end{aligned}$$

Optimal solution:
$x = 0,\ y = 1,\ z = 1$

- Unlike hard constraints, assumptions are revertable
  - removal of hard constraints can be simulated with assumptions

IPAMIR: Incremental API for MaxSAT

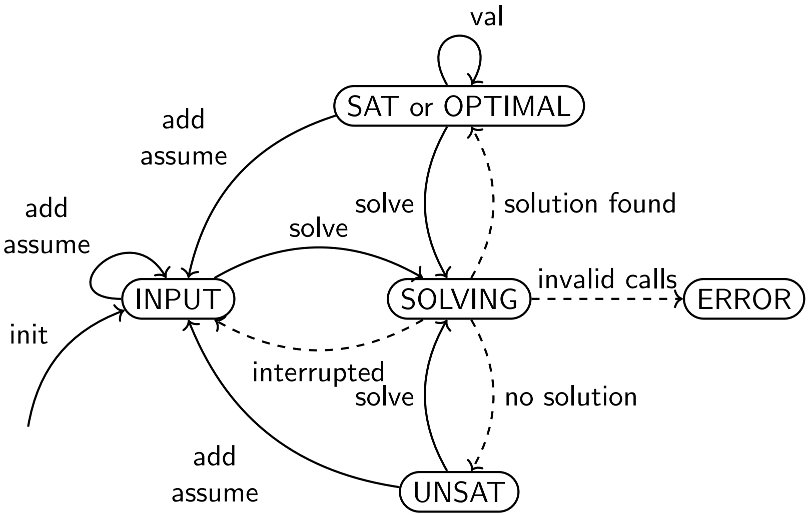
- Aim for a **generic interface** for incremental MaxSAT
  - for MaxSAT solvers providing support for incrementality
  - for applications making use of incrementality
- Specifies **incremental changes** to a MaxSAT instance
  - adding hard constraints
  - adding terms to or changing coefficients of the objective function
  - assumptions on variables
- + other essential declarations
  - constructing and releasing a solver
  - solving, variable assignments, objective values

IPAMIR: Incremental API for MaxSAT

```
// Construct a MaxSAT solver and return a pointer to it.
void * ipamir_init ();
// Deallocate all resources of the MaxSAT solver.
void ipamir_release (void * solver);
// Add a literal to a hard clause or finalize the clause with zero.
void IPAMIR_ADD_HARD (void * solver, int32_t lit_or_zero);
// Add a weighted soft literal.
void IPAMIR_ADD_SOFT_LIT (void * solver, int32_t lit, uint64_t weight);
// Assume a literal for the next solver call.
void IPAMIR_ASSUME (void * solver, int32_t lit);
// Solve the MaxSAT instance under the current assumptions.
int ipamir_solve (void * solver);
// Compute the cost of the solution.
uint64_t ipamir_val_obj (void * solver);
// Extract the truth value of a literal in the solution.
int32_t ipamir_val_lit (void * solver, int32_t lit);
// Set a callback function for terminating the solving procedure.
void ipamir_set_terminate (void * solver, void * state,
                           int (*terminate)(void * state));
```

Interface and example applications openly available:
`https://bitbucket.org/coreo-group/ipamir`

IPAMIR


val
SAT or OPTIMAL
add
assume
solve
solution found
add
assume
solve
INPUT
SOLVING
invalid calls
ERROR
init
interrupted
solve
no solution
add
assume
UNSAT

MSE 2022 Incremental track: Submissions

5 benchmark submissions:

- **Bi-objective Boolean optimization**: adding hard clauses
- **MLIC-SeeSaw**: adding hard clauses + assumptions
- **Extension enforcement in abstract argumentation**: adding hard clauses
- **Learning boosted decision trees via AdaBoost**: changing weights of soft literals
- **Proof obligations in bit-level PDR**: assumptions

3 solver submissions:

- **EvalMaxSAT**: core-guided
- **iMaxHS**: implicit hitting set based
- **UWrMaxSat** (2 versions): core-guided (+ ILP)

Incremental track: Results

Solver	rank (number of solved instances)				
	BiOptSat	SeeSaw	ExtEnf	AdaBoost	PDR
EvalMaxSAT	4 (28)	**1** (19)	2 (40)	4 (16)	**1** (44)
iMaxHS	3 (45)	2 (18)	**1** (48)	**1** (23)	3 (36)
UWrMaxSat	**1** (50)	3 (6)	3 (38)	2 (17)	2 (38)
UWrMaxSat+SCIP	2 (50)	N/A	4 (37)	3 (17)	4 (31)

- Solver performance application-dependent
  - EvalMaxSAT, iMaxHS, and UWrMaxSat ranked first on some benchmark, all solvers ranked second on some benchmark

Incremental track: Results

Solver	rank (number of solved instances)				
	BiOptSat	SeeSaw	ExtEnf	AdaBoost	PDR
EvalMaxSAT	4 (28)	**1** (19)	2 (40)	4 (16)	**1** (44)
iMaxHS	3 (45)	2 (18)	**1** (48)	**1** (23)	3 (36)
UWrMaxSat	**1** (50)	3 (6)	3 (38)	2 (17)	2 (38)
UWrMaxSat+SCIP	2 (50)	N/A	4 (37)	3 (17)	4 (31)

- Solver performance application-dependent
  - EvalMaxSAT, iMaxHS, and UWrMaxSat ranked first on some benchmark, all solvers ranked second on some benchmark

Incremental track: Observations

- Adaptive benchmarks: the sequence of IPAMIR calls depends on the results of previous solve calls
  - How to clearly rank solvers in this case?
- Unit tests and fuzzers for IPAMIR?
- Likely much room for solver performance improvements!

MSE'23 incremental track failed due to lack of participants

2024?

Incremental MaxSAT Solving (IHS style)

Implicit Hitting Set (IHS) based MaxSAT solving

Davies and Bacchus [2011, 2013]

An iterative approach: identify *sources of inconsistency* and *repair the inconsistencies* optimally

- *core*: partial assignment over objective function variables *which cannot be extended to satisfy the constraints*
  - SAT solver as *core extractor*
- *hs*: a *hitting set* over a set of cores
  - *cost* of a hitting set determined by coefficients of the objective
  - IP solver for computing *minimum-cost* hitting sets

Reasoning and *optimization* effectively decoupled

- *upper bounds* from assignments given by the SAT solver
- *lower bounds* from costs of optimal hitting sets

Implicit Hitting Set (IHS) based MaxSAT solving

Davies and Bacchus [2011, 2013]

An iterative approach: identify *sources of inconsistency* and *repair the inconsistencies* optimally

- *core*: partial assignment over objective function variables *which cannot be extended to satisfy the constraints*
  - SAT solver as *core extractor*
- *hs*: a *hitting set* over a set of cores
  - *cost* of a hitting set determined by coefficients of the objective
  - IP solver for computing *minimum-cost* hitting sets

Reasoning and *optimization* effectively decoupled

- *upper bounds* from assignments given by the SAT solver
- *lower bounds* from costs of optimal hitting sets

Implicit Hitting Set (IHS) based MaxSAT solving

Davies and Bacchus [2011, 2013]

An iterative approach: identify *sources of inconsistency*
and *repair the inconsistencies* optimally

- *core*: partial assignment over objective function variables *which cannot be extended to satisfy the constraints*
  - SAT solver as *core extractor*
- *hs*: a *hitting set* over a set of cores
  - *cost* of a hitting set determined by coefficients of the objective
  - IP solver for computing *minimum-cost* hitting sets

Reasoning and *optimization* effectively decoupled

- *upper bounds* from assignments given by the SAT solver
- *lower bounds* from costs of optimal hitting sets

Implicit Hitting Set (IHS) based MaxSAT solving

Davies and Bacchus [2011, 2013]

An iterative approach: identify *sources of inconsistency* and *repair the inconsistencies* optimally

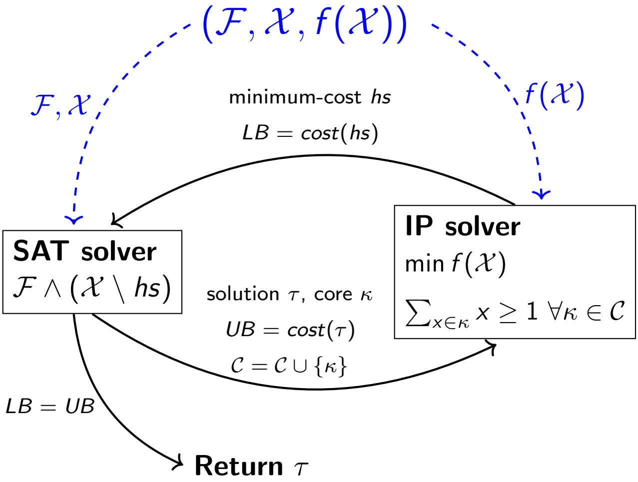
- *core*: partial assignment over objective function variables *which cannot be extended to satisfy the constraints*
  - SAT solver as *core extractor*
- *hs*: a *hitting set* over a set of cores
  - *cost* of a hitting set determined by coefficients of the objective
  - IP solver for computing *minimum-cost* hitting sets

Reasoning and *optimization* effectively decoupled

- *upper bounds* from assignments given by the SAT solver
- *lower bounds* from costs of optimal hitting sets

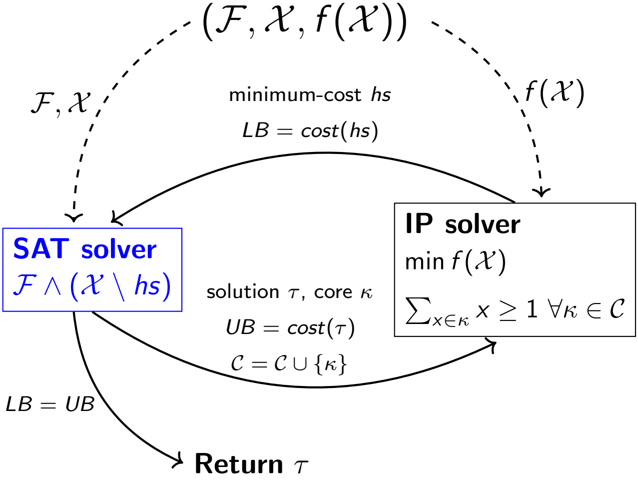
Implicit Hitting Set (IHS) based MaxSAT solving

Davies and Bacchus [2011, 2013]



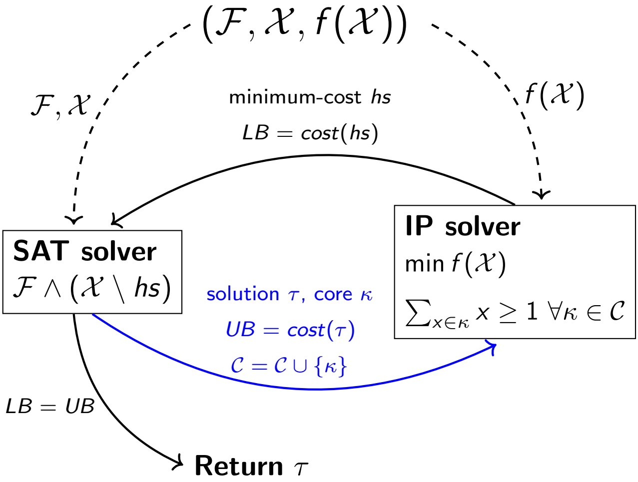
Implicit Hitting Set (IHS) based MaxSAT solving

Davies and Bacchus [2011, 2013]



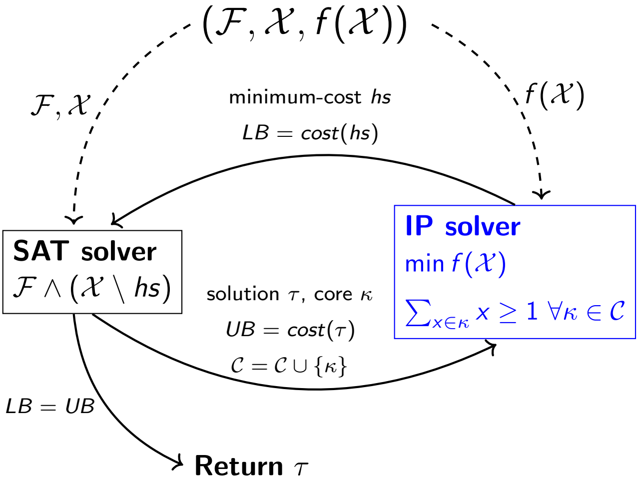
Implicit Hitting Set (IHS) based MaxSAT solving

Davies and Bacchus [2011, 2013]



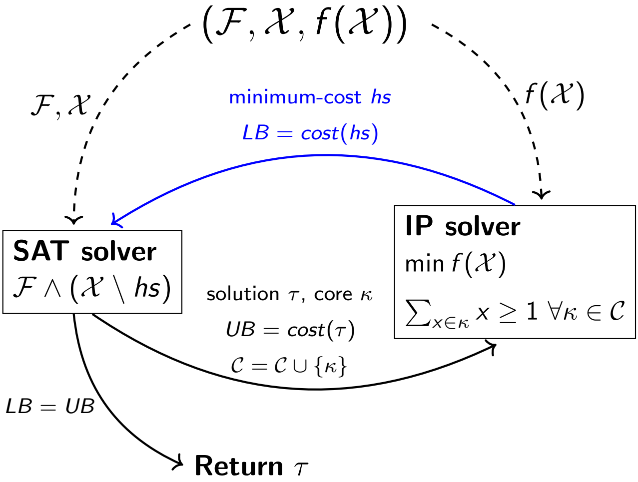
Implicit Hitting Set (IHS) based MaxSAT solving

Davies and Bacchus [2011, 2013]



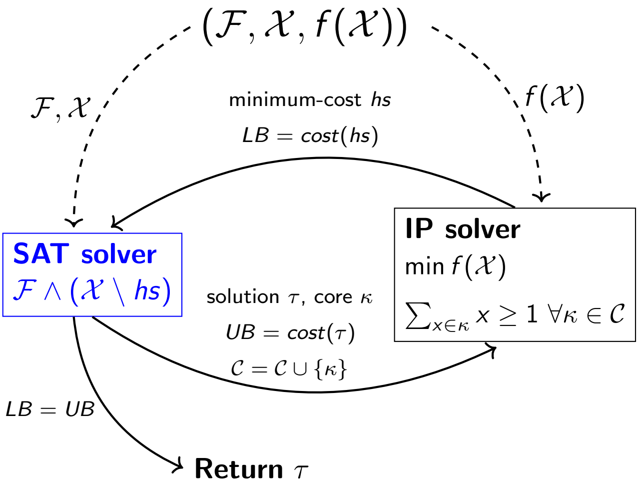
Implicit Hitting Set (IHS) based MaxSAT solving

Davies and Bacchus [2011, 2013]



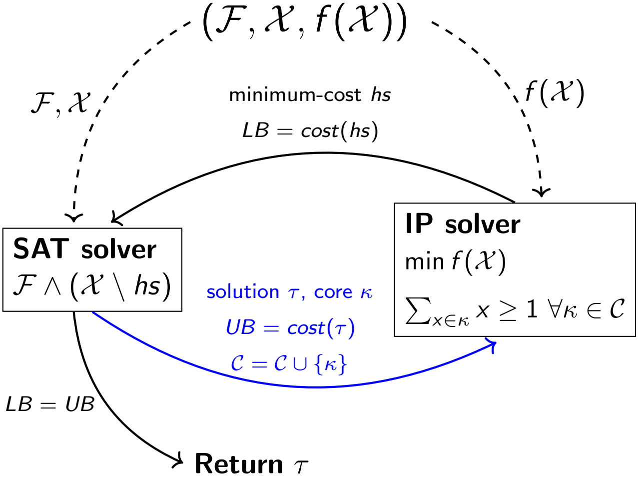
Implicit Hitting Set (IHS) based MaxSAT solving

Davies and Bacchus [2011, 2013]



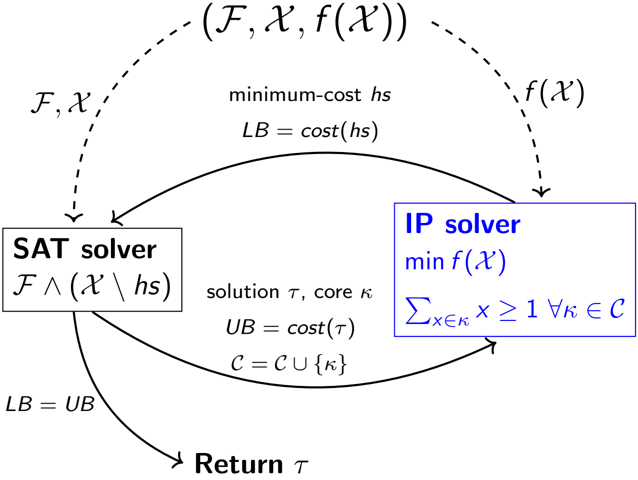
Implicit Hitting Set (IHS) based MaxSAT solving

Davies and Bacchus [2011, 2013]



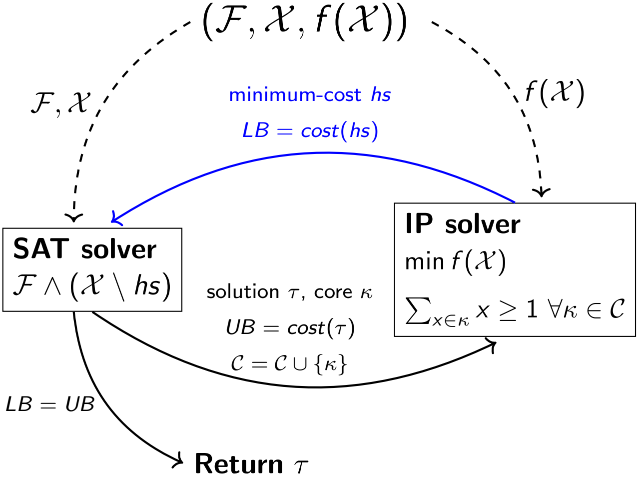
Implicit Hitting Set (IHS) based MaxSAT solving

Davies and Bacchus [2011, 2013]



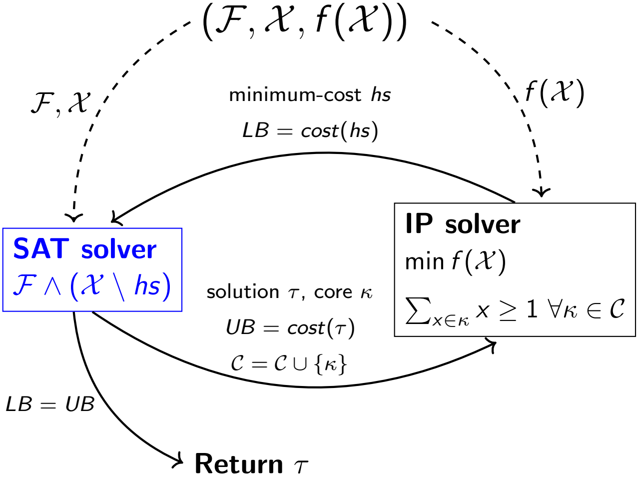
Implicit Hitting Set (IHS) based MaxSAT solving

Davies and Bacchus [2011, 2013]



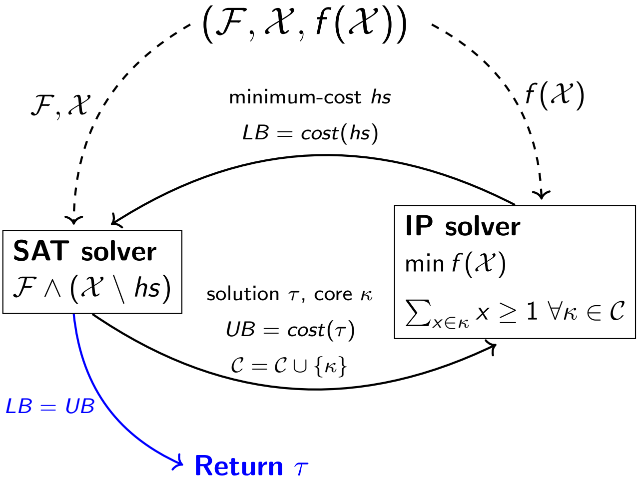
Implicit Hitting Set (IHS) based MaxSAT solving

Davies and Bacchus [2011, 2013]



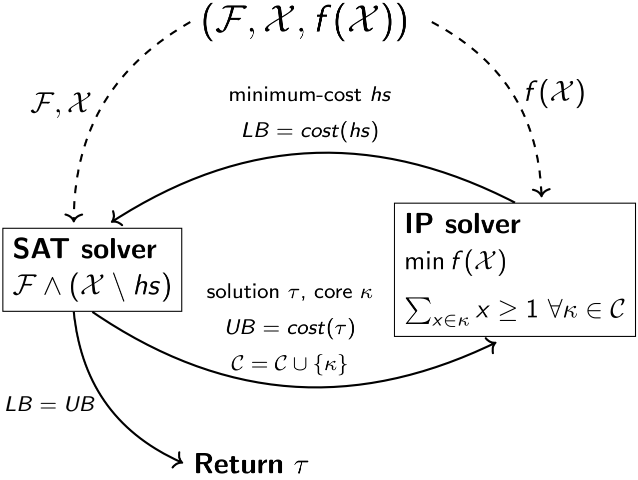
Implicit Hitting Set (IHS) based MaxSAT solving

Davies and Bacchus [2011, 2013]



Implicit Hitting Set (IHS) based MaxSAT solving

Davies and Bacchus [2011, 2013]



Incremental IHS

Niskanen, Berg, and Järvisalo [2021, 2022]

Observations:

- Add a constraint or a term to the objective function?
  Or change objective variable coefficients?
  Extracted cores still valid
  - **Cores can be preserved** between solver invocations
  - only objective needs to be altered in the IP solver (for hitting sets)
- The SAT solver knows nothing about the objective
  - add constraints directly to the SAT solver
  - no need to reinitialize

Assumptions require more care: *conditional cores*

- **no need to reset the SAT solver**
- IP solver reinitialized with *restrictions* of all conditional cores valid under current assumptions

Incremental IHS

Niskanen, Berg, and Järvisalo [2021, 2022]

Observations:

- Add a constraint or a term to the objective function?
  Or change objective variable coefficients?
  Extracted cores still valid
  - **Cores can be preserved** between solver invocations
  - only objective needs to be altered in the IP solver (for hitting sets)
- The SAT solver knows nothing about the objective
  - add constraints directly to the SAT solver
  - no need to reinitialize

Assumptions require more care: *conditional cores*

- **no need to reset the SAT solver**
- IP solver reinitialized with *restrictions* of all conditional cores valid under current assumptions

Incremental IHS

In practice

Realizing incrementality requires a non-trivial amount of engineering . . . due to e.g.

- **Simplifications before solution**
  - variable mappings between internal and external representations
  - fixed variables need to be handled correctly
  - ...
- **Maintaining conditional cores:** use another SAT solver as a database for storing conditional cores
  - removes redundant cores and simplifies them
- **Other techniques** to account for . . . e.g.
  - reduced cost fixing — Bacchus, Hyttinen, Järvisalo, and Saikko [2017]
  - abstract cores — Berg, Bacchus, and Poole [2020]

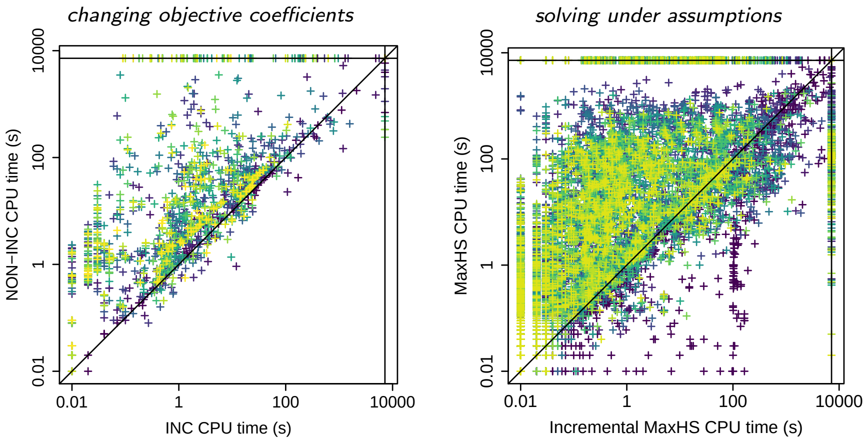
Incremental IHS

In practice

Realizing incrementality requires a non-trivial amount of engineering . . . due to e.g.

- **Simplifications before solution**
  - variable mappings between internal and external representations
  - fixed variables need to be handled correctly
  - ...
- **Maintaining conditional cores:** use another SAT solver as a database for storing conditional cores
  - removes redundant cores and simplifies them
- **Other techniques** to account for . . . e.g.
  - reduced cost fixing — Bacchus, Hyttinen, Järvisalo, and Saikko [2017]
  - abstract cores — Berg, Bacchus, and Poole [2020]

Practical Benefits of Incrementality



- blue points → earlier iterations
- yellow points → later iterations

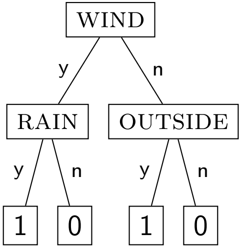
Learning (boosted) decision trees

Narodytska, Ignatiev, Pereira, and Marques-Silva [2018]; Hu, Siala, Hebrard, and Huguet [2020]

Decision Trees:

- Input:
  - Examples and their classes
  - Max depth U
- Learn tree $\mathcal{T}$ s.t:
  - $\textsc{Depth}(\mathcal{T}) \leq U$
  - $|\{e_i \mid c_i = \mathcal{T}(e_i)\}|$ is maximized.

Ex.	RAIN	WIND	OUTSIDE	c_i	$\mathcal{T}(e_i)$
e_1	yes	no	yes	0	1
e_2	yes	yes	yes	1	1
e_3	no	yes	yes	0	0
e_4	yes	no	no	1	0

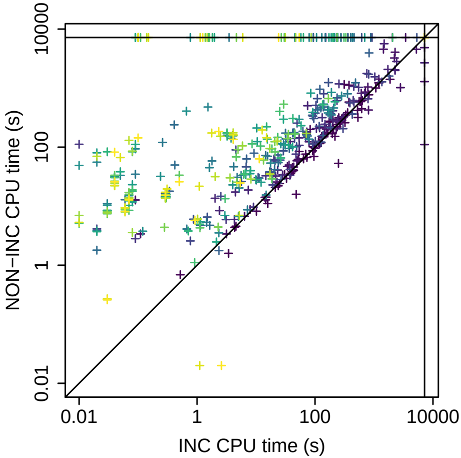


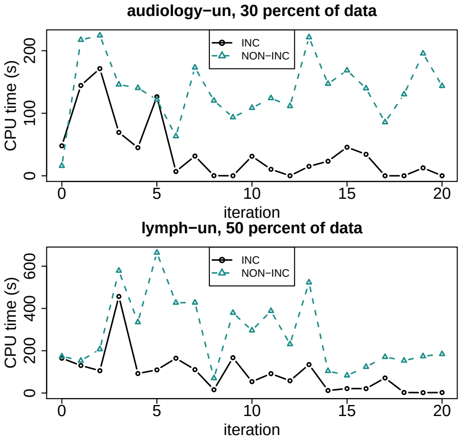
Incrementality: AdaBoost

Weights = priorities of examples.
Learn diverse trees that prioritize different examples.

Results

Decision Trees





Learning interpretable decision rules

Malioutov and Meel [2018]

Decision Rules:

- Input:
  - examples and their classes
  - integer λ
- Learn rule $\mathcal{R}$ minimizing:

$$\sum_{C \in \mathcal{R}} |C| + \lambda \cdot |\{e_i \mid c_i \neq \mathcal{R}(e_i)\}|$$

Ex.	RAIN	WIND	OUTSIDE	c_i	$\mathcal{T}(e_i)$
e_1	yes	no	yes	0	1
e_2	yes	yes	yes	1	1
e_3	no	yes	yes	0	0
e_4	yes	no	no	1	0

$$\mathcal{R} = (x_{\text{RAIN}}) \wedge (x_{\text{WIND}} \vee x_{\text{OUTSIDE}})$$

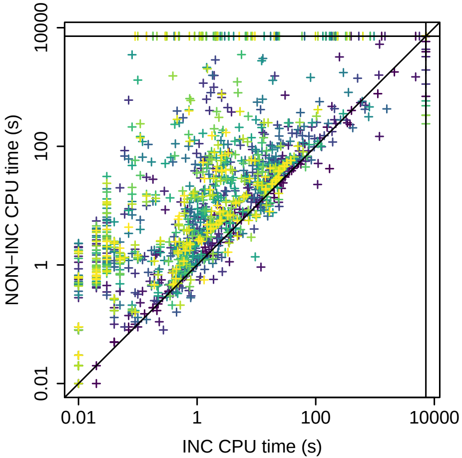
Incrementality

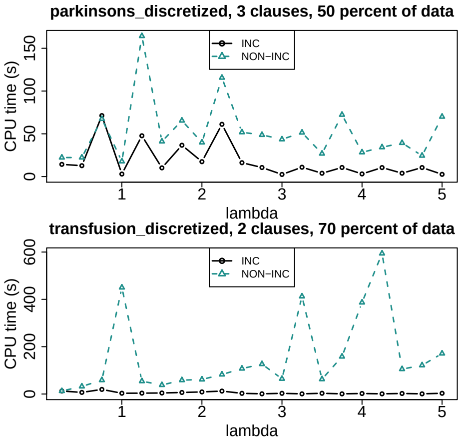
λ = trade-off between interpretability and accuracy.

Learn rules for different λ.

Results

Decision Rules





MaxSAT-based Column Generation

Computing the η Inconsistency Measure Niskanen, Kuhlmann, Thimm, and Järvisalo [2023]

$\mathcal{I}_\eta(\mathcal{K})$ is the **optimal value** of the following **linear program** (LP), where $\Omega(\mathsf{At})$ is the set of **all truth assignments** τ over At.

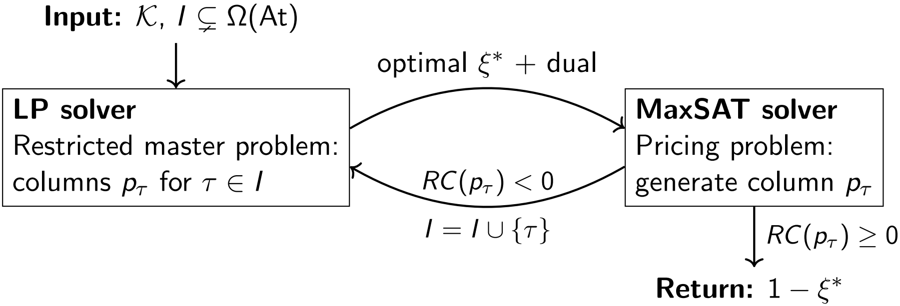
$$
\begin{aligned}
\text{minimize} \quad & 1 - \xi \\
\text{subject to} \quad & \sum_{\tau \in \Omega(\mathsf{At})} p_\tau = 1, \\
& \sum_{\tau \models \phi} p_\tau \geq \xi & \forall \phi \in \mathcal{K}, \\
& p_\tau \geq 0 & \forall \tau \in \Omega(\mathsf{At}).
\end{aligned}
$$

Problem: **exponential number of columns** (variables) p_τ!

MaxSAT-based Column Generation

Computing the η Inconsistency Measure

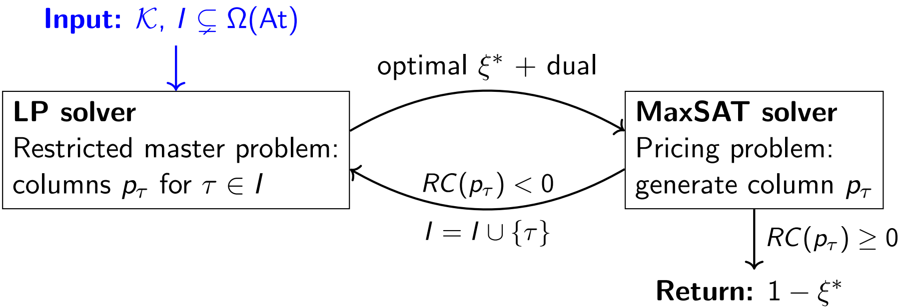
Niskanen, Kuhlmann, Thimm, and Järvisalo [2023]



Iterative approach to computing $\mathcal{I}_\eta(\mathcal{K})$: employ an **LP solver** on a **restricted linear program**, and a **MaxSAT solver** for **generating a new column**.

MaxSAT-based Column Generation

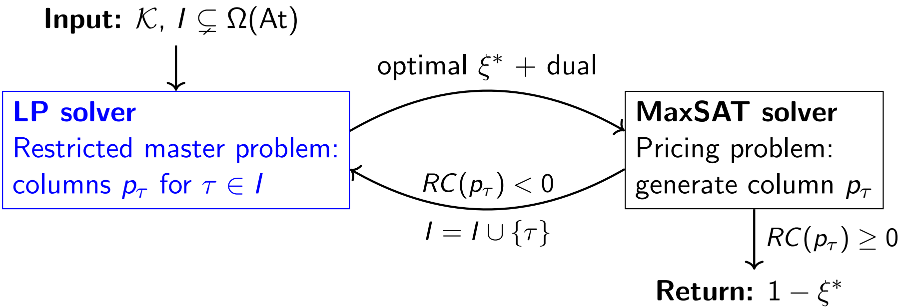
Computing the η Inconsistency Measure Niskanen, Kuhlmann, Thimm, and Järvisalo [2023]



An **initial set of truth assignments** $I \subseteq \Omega(\mathrm{At})$ obtained by iteratively calling a SAT solver for a satisfiable truth assignment for each $\phi \in \mathcal{K}$.

MaxSAT-based Column Generation

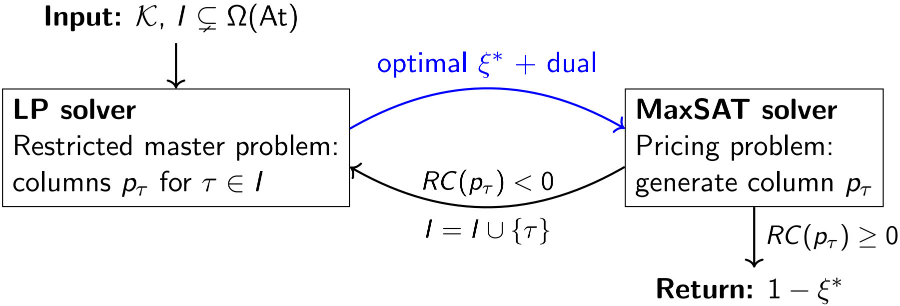
Computing the η Inconsistency Measure Niskanen, Kuhlmann, Thimm, and Järvisalo [2023]



Restricted master problem includes columns corresponding to variables p_τ for assignments $\tau \in I$, instead of all $\tau \in \Omega(\mathrm{At})$.

MaxSAT-based Column Generation

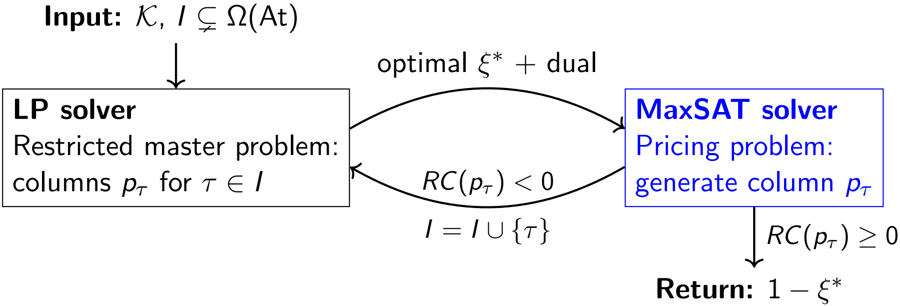
Computing the η Inconsistency Measure   Niskanen, Kuhlmann, Thimm, and Järvisalo [2023]



Solving the restricted LP yields an optimal solution ξ^* and the corre-
sponding
optimal dual solution.

MaxSAT-based Column Generation

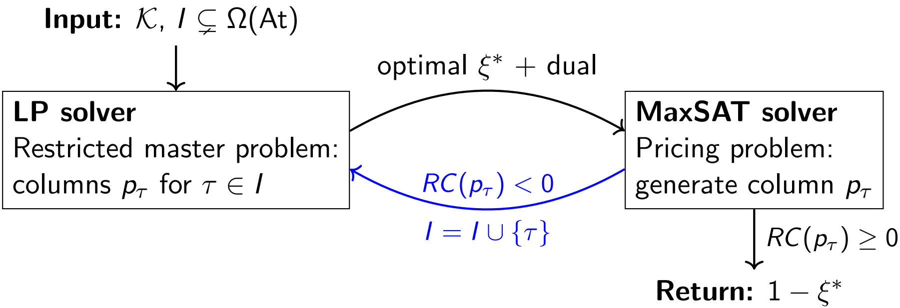
Computing the η Inconsistency Measure Niskanen, Kuhlmann, Thimm, and Järvisalo [2023]



Pricing problem generates a column p_τ with $\tau \notin I$ which may improve $1 - \xi^*$ via minimizing **reduced cost** $RC(p_\tau)$, defined using the optimal dual solution.
This is equivalent to an **incremental MaxSAT problem**.

MaxSAT-based Column Generation

Computing the η Inconsistency Measure Niskanen, Kuhlmann, Thimm, and Järvisalo [2023]

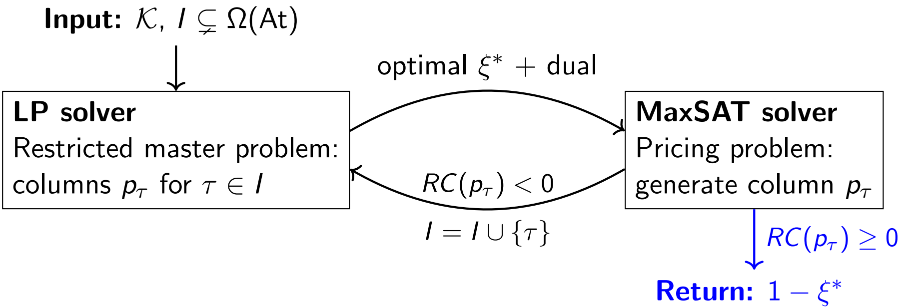


If minimum RC(p_τ) is negative, add assignment τ to I and **continue**.

MaxSAT-based Column Generation

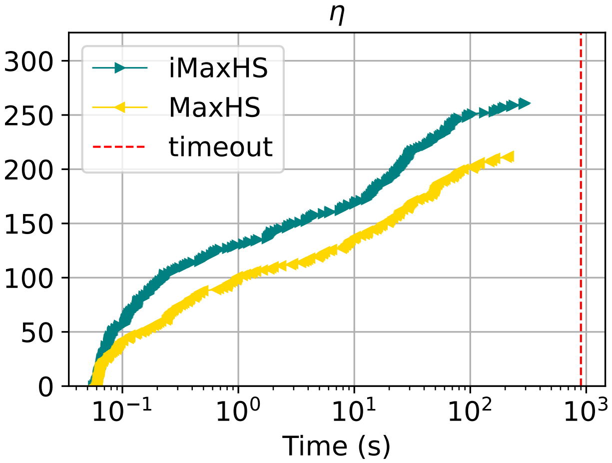
Computing the η Inconsistency Measure

Niskanen, Kuhlmann, Thimm, and Järvisalo [2023]



If minimum $\mathrm{RC}(p_\tau)$ is non-negative, the current solution $1 - \xi^*$ is **optimal**.

Inconsistency measurement: Incremental vs non-incremental MaxSAT



Summary

- **IPAMIR:** incremental API for MaxSAT
  - provides a standard interface to facilitate the development of solvers and applications
- **Incremental MaxHS:** incremental MaxSAT solver
  - supports all IPAMIR functionality
  - preserves cores and does not reset SAT solver between invocations
- **Applications:** clear benefit from incrementality

Going Further

- More applications — understanding limits and realizing potential
- Realizing again the MSE incremental track?
- Making hitting set computations more incremental
- Extensions beyond MaxSAT, e.g. incremental PBO-IHS

Smirnov, Berg, and Järvisalo [2021, 2022]

Thank you for your attention!

matti.jarvisalo@helsinki.fi

Bibliography I

Fahiem Bacchus, Antti Hyttinen, Matti Järvisalo, and Paul Saikko. Reduced cost fixing in MaxSAT. In J. Christopher Beck, editor, *Principles and Practice of Constraint Programming - 23rd International Conference, CP 2017, Melbourne, VIC, Australia, August 28 - September 1, 2017, Proceedings*, volume 10416 of *Lecture Notes in Computer Science*, pages 641–651. Springer, 2017. doi: 10.1007/978-3-319-66158-2_41.

Fahiem Bacchus, Matti Järvisalo, and Ruben Martins. Maximum satisfiability. In Armin Biere, Marijn Heule, Hans van Maaren, and Toby Walsh, editors, *Handbook of Satisfiability, Second Edition*, volume 336 of *Frontiers in Artificial Intelligence and Applications*, chapter 24, pages 929–991. IOS Press, 2021. doi: 10.3233/FAIA201008.

Jeremias Berg, Fahiem Bacchus, and Alex Poole. Abstract cores in implicit hitting set MaxSat solving. In Luca Pulina and Martina Seidl, editors, *Theory and Applications of Satisfiability Testing - SAT 2020 - 23rd International Conference, Alghero, Italy, July 3-10, 2020, Proceedings*, volume 12178 of *Lecture Notes in Computer Science*, pages 277–294. Springer, 2020. doi: 10.1007/978-3-030-51825-7_20.

Jessica Davies and Fahiem Bacchus. Solving MAXSAT by solving a sequence of simpler SAT instances. In Jimmy Ho-Man Lee, editor, *Principles and Practice of Constraint Programming - CP 2011 - 17th International Conference, CP 2011, Perugia, Italy, September 12-16, 2011, Proceedings*, volume 6876 of *Lecture Notes in Computer Science*, pages 225–239. Springer, 2011. doi: 10.1007/978-3-642-23786-7_19.

Jessica Davies and Fahiem Bacchus. Postponing optimization to speed up MAXSAT solving. In Christian Schulte, editor, *Principles and Practice of Constraint Programming - 19th International Conference, CP 2013, Uppsala, Sweden, September 16-20, 2013, Proceedings*, volume 8124 of *Lecture Notes in Computer Science*, pages 247–262. Springer, 2013. doi: 10.1007/978-3-642-40627-0_21.

Niklas Eén and Niklas Sörensson. Temporal induction by incremental SAT solving. *Electronic Notes in Theoretical Computer Science*, 89(4):543–560, 2003. doi: 10.1016/S1571-0661(05)82542-3.

Hao Hu, Mohamed Siala, Emmanuel Hebrard, and Marie-José Huguet. Learning optimal decision trees with MaxSAT and its integration in AdaBoost. In Christian Bessiere, editor, *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, IJCAI 2020*, pages 1170–1176. ijcai.org, 2020. doi: 10.24963/ijcai.2020/163.

Bibliography II

Alexandre Lemos, Pedro T. Monteiro, and Inês Lynce. Minimal perturbation in university timetabling with maximum satisfiability. In Emmanuel Hebrard and Nysret Musliu, editors, *Integration of Constraint Programming, Artificial Intelligence, and Operations Research - 17th International Conference, CPAIOR 2020, Vienna, Austria, September 21-24, 2020, Proceedings*, volume 12296 of *Lecture Notes in Computer Science*, pages 317–333. Springer, 2020. doi: 10.1007/978-3-030-58942-4_21.

Dmitry Malioutov and Kuldeep S. Meel. MLIC: A MaxSAT-based framework for learning interpretable classification rules. In John N. Hooker, editor, *Principles and Practice of Constraint Programming - 24th International Conference, CP 2018, Lille, France, August 27-31, 2018, Proceedings*, volume 11008 of *Lecture Notes in Computer Science*, pages 312–327. Springer, 2018. doi: 10.1007/978-3-319-98334-9_21.

Ravi Mangal, Xin Zhang, Aditya V. Nori, and Mayur Naik. Volt: A lazy grounding framework for solving very large maxsat instances. In Marijn Heule and Sean A. Weaver, editors, *Theory and Applications of Satisfiability Testing - SAT 2015 - 18th International Conference, Austin, TX, USA, September 24-27, 2015, Proceedings*, volume 9340 of *Lecture Notes in Computer Science*, pages 299–306. Springer, 2015. doi: 10.1007/978-3-319-24318-4_22.

Nina Narodytska, Alexey Ignatiev, Filipe Pereira, and João Marques-Silva. Learning optimal decision trees with SAT. In Jérôme Lang, editor, *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI 2018, July 13-19, 2018, Stockholm, Sweden*, pages 1362–1368. ijcai.org, 2018. doi: 10.24963/ijcai.2018/189.

Andreas Niskanen and Matti Järvisalo. Strong refinements for hard problems in argumentation dynamics. In Giuseppe De Giacomo, Alejandro Catalá, Bistra Dilkina, Michela Milano, Senén Barro, Alberto Bugarín, and Jérôme Lang, editors, *ECAI 2020 - 24th European Conference on Artificial Intelligence, 29 August-8 September 2020, Santiago de Compostela, Spain, August 29 - September 8, 2020 - Including 10th Conference on Prestigious Applications of Artificial Intelligence (PAIS 2020)*, volume 325 of *Frontiers in Artificial Intelligence and Applications*, pages 841–848. IOS Press, 2020. doi: 10.3233/FAIA200174.

Andreas Niskanen, Jeremias Berg, and Matti Järvisalo. Enabling incrementality in the implicit hitting set approach to maxsat under changing weights. In Laurent D. Michel, editor, *27th International Conference on Principles and Practice of Constraint Programming, CP 2021, Montpellier, France (Virtual Conference), October 25-29, 2021*, volume 210 of *LIPIcs*, pages 44:1–44:19. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2021. doi: 10.4230/LIPIcs.CP.2021.44.

Bibliography III

Andreas Niskanen, Jeremias Berg, and Matti Järvisalo. Incremental maximum satisfiability. In Kuldeep S. Meel and Ofer Strichman, editors, *25th International Conference on Theory and Applications of Satisfiability Testing, SAT 2022, Haifa, Israel, August 2-5, 2022*, volume 236 of *LIPIcs*, pages 14:1–14:19. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2022. To appear.

Andreas Niskanen, Isabelle Kuhlmann, Matthias Thimm, and Matti Järvisalo. MaxSAT-Based inconsistency measurement. In *Proc. ECAI 2023*. IOS Press, 2023.

Pavel Smirnov, Jeremias Berg, and Matti Järvisalo. Pseudo-boolean optimization by implicit hitting sets. In Laurent D. Michel, editor, *27th International Conference on Principles and Practice of Constraint Programming, CP 2021, Montpellier, France (Virtual Conference), October 25-29, 2021*, volume 210 of *LIPIcs*, pages 51:1–51:20. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2021. doi: 10.4230/LIPIcs.CP.2021.51. URL https://doi.org/10.4230/LIPIcs.CP.2021.51.

Pavel Smirnov, Jeremias Berg, and Matti Järvisalo. Improvements to the implicit hitting set approach to pseudo-boolean optimization. In Kuldeep S. Meel and Ofer Strichman, editors, *25th International Conference on Theory and Applications of Satisfiability Testing, SAT 2022, August 2-5, 2022, Haifa, Israel*, volume 236 of *LIPIcs*, pages 13:1–13:18. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2022. doi: 10.4230/LIPIcs.SAT.2022.13. URL https://doi.org/10.4230/LIPIcs.SAT.2022.13.