Enabling Incrementality in the Implicit Hitting Set Approach to MaxSAT under Changing Weights

Andreas Niskanen Jeremias Berg Matti Järvisalo

HIIT, Department of Computer Science, University of Helsinki, Finland

CP 2021
Online

Motivation

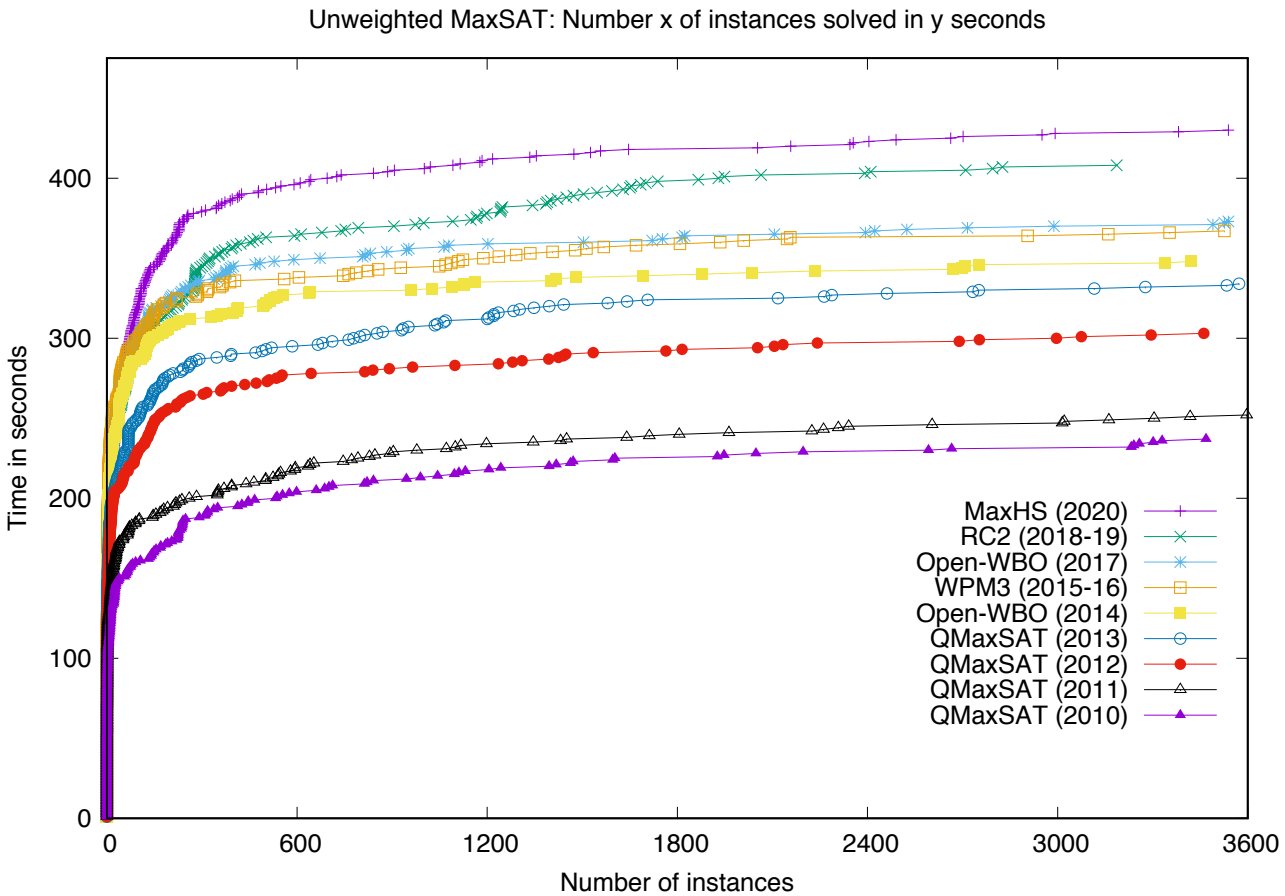
- MaxSAT: declarative optimisation based on propositional logic.
  - i.e. Boolean literals and clauses
- State-of-the-art solvers build on the success of CDCL SAT solvers.
- New application domains and solver improvements annually.

(Recent survey in [Bacchus, Järvisalo, and Martins, 2021])

Motivation

- MaxSAT: declarative optimisation based on propositional logic.
  - i.e. Boolean literals and clauses
- State-of-the-art solvers build on the success of CDCL SAT solvers.
- New application domains and solver improvements annually.

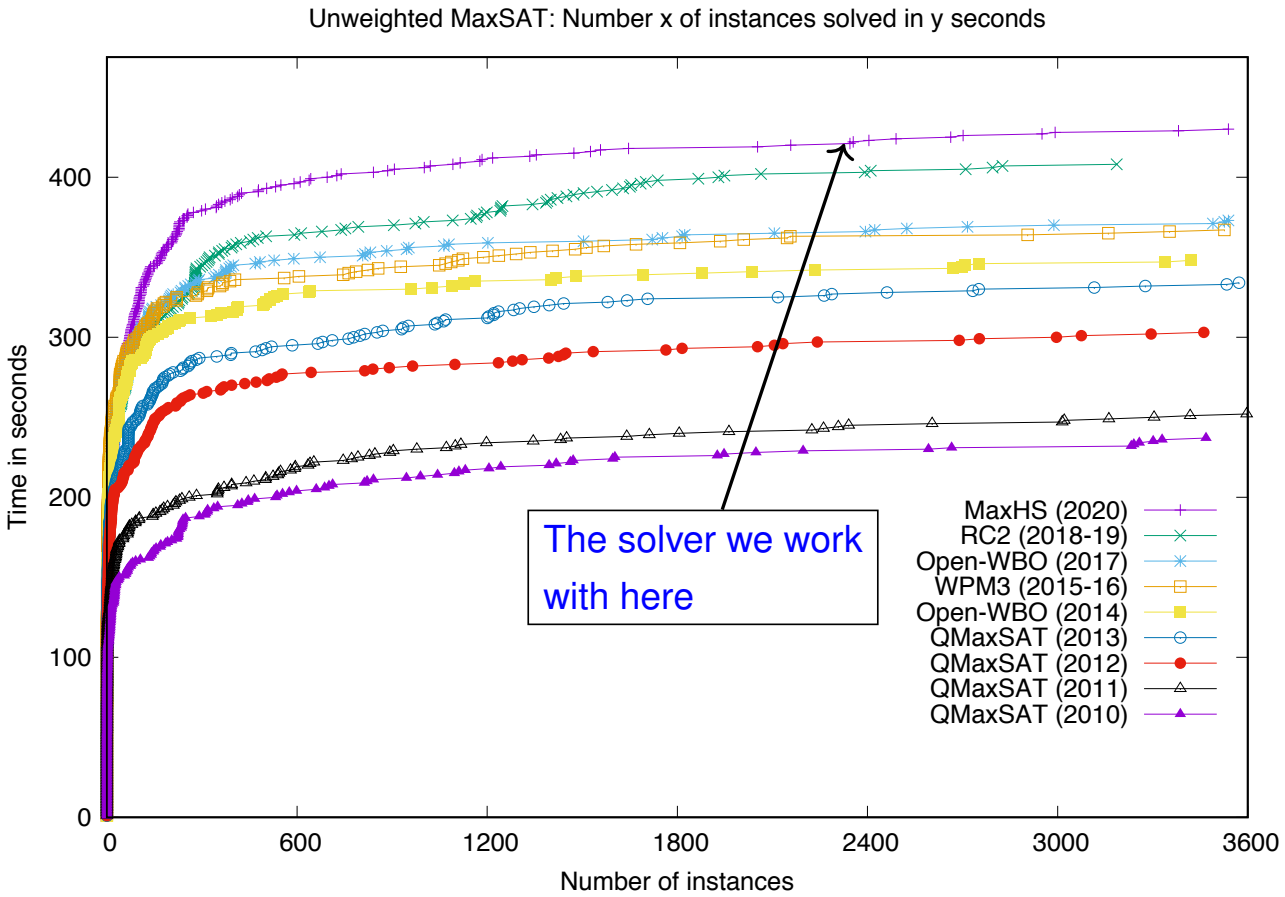
(Recent survey in [Bacchus, Järvisalo, and Martins, 2021])



Motivation

- MaxSAT: declarative optimisation based on propositional logic.
  - i.e. Boolean literals and clauses
- State-of-the-art solvers build on the success of CDCL SAT solvers.
- New application domains and solver improvements annually.

(Recent survey in [Bacchus, Järvisalo, and Martins, 2021])



Incremental Optimisation

- *Incremental solving* $\rightarrow$ solving a sequence of similar instances.
- Incremental SAT solving:
  - well established
  - extensively applied (e.g. by MaxSAT solvers).
- Applications for incremental MaxSAT known
  - XAI
  - predict & optimize
  - markov logic networks
  - ...
- So far, very little support from solvers
  - adding clauses.
  - restarting core-guided search when search stagnates.

Mangal et al. [2016]
Cimatti et al. [2013]
Zhang et al. [2016]
Richardson and Domingos [2006]
Malioutov and Meel [2018]
Ghosh and Meel [2019]
Hu et al. [2020]
Yu et al. [2020]
Mulamba et al. [2021]
Saikko et al. [2016]
Ignatiev et al. [2019]
Si et al. [2016]
⋮

Incremental Optimisation

- *Incremental solving* $\rightarrow$ solving a sequence of similar instances.
- Incremental SAT solving:
  - well established
  - extensively applied (e.g. by MaxSAT solvers).
- Applications for incremental MaxSAT known
  - XAI
  - predict & optimize
  - markov logic networks
  - ...
- So far, very little support from solvers
  - adding clauses.
  - restarting core-guided search when search stagnates.

Mangal et al. [2016]
Cimatti et al. [2013]
Zhang et al. [2016]
Richardson and Domingos [2006]
Malioutov and Meel [2018]
Ghosh and Meel [2019]
Hu et al. [2020]
Yu et al. [2020]
Mulamba et al. [2021]
Saikko et al. [2016]
Ignatiev et al. [2019]
Si et al. [2016]
⋮

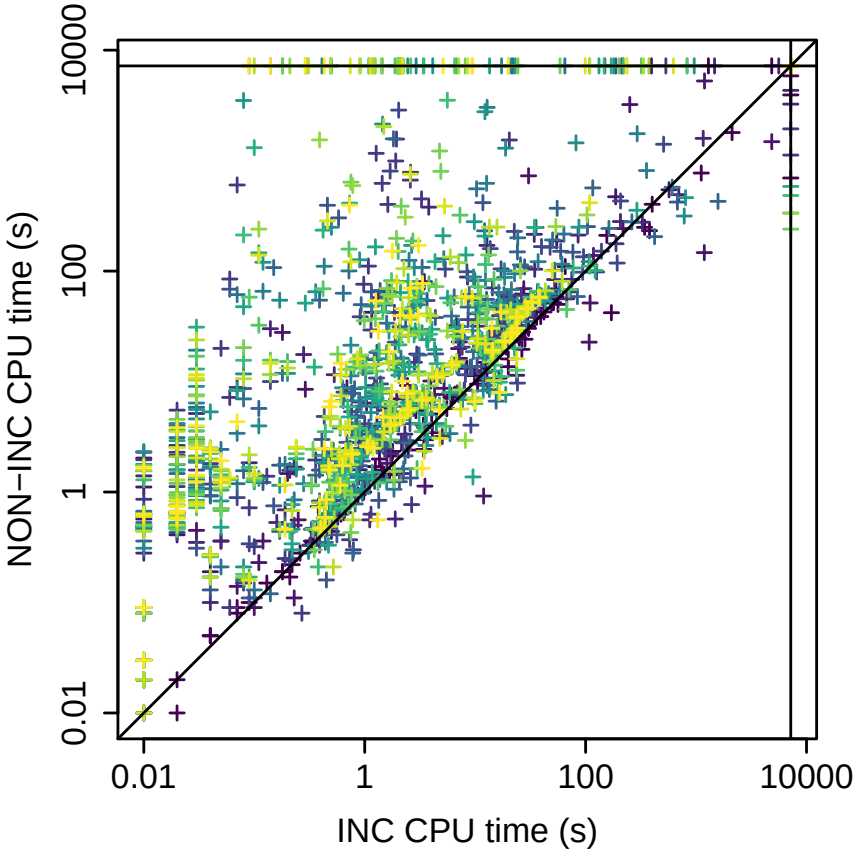
Incremental Optimisation

- *Incremental solving* $\rightarrow$ solving a sequence of similar instances.
- Incremental SAT solving:
  - well established
  - extensively applied (e.g. by MaxSAT solvers).
- Applications for incremental MaxSAT known
  - XAI
  - predict & optimize
  - markov logic networks
  - ...
- So far, very little support from solvers
  - adding clauses.
  - restarting core-guided search when search stagnates.

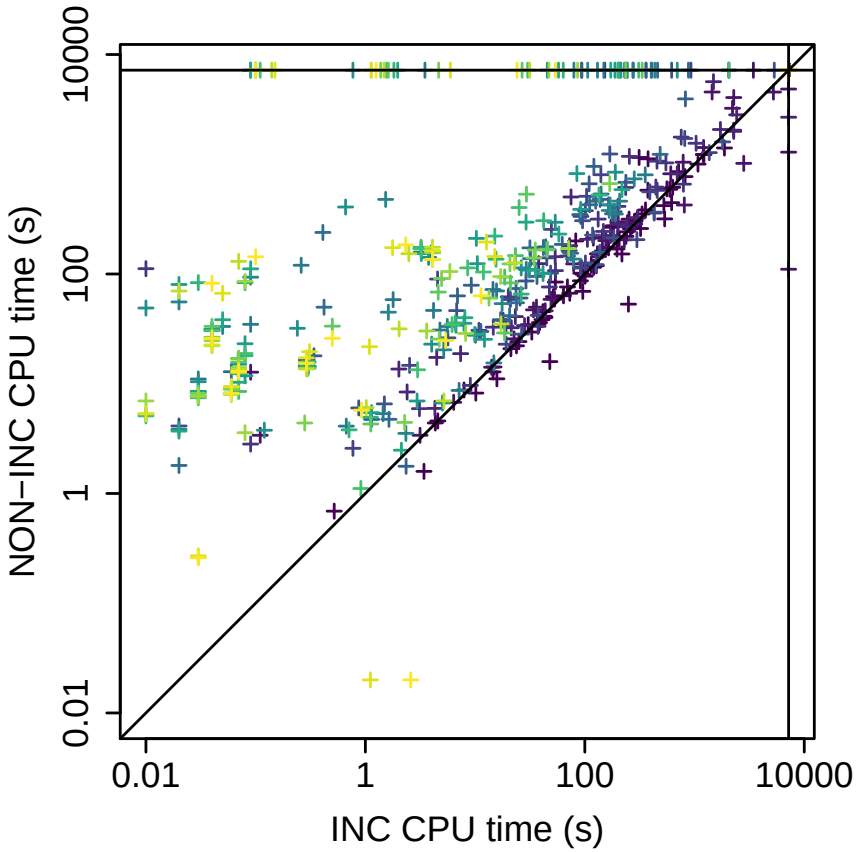
Mangal et al. [2016]
Cimatti et al. [2013]
Zhang et al. [2016]
Richardson and Domingos [2006]
Malioutov and Meel [2018]
Ghosh and Meel [2019]
Hu et al. [2020]
Yu et al. [2020]
Mulamba et al. [2021]
Saikko et al. [2016]
Ignatiev et al. [2019]
Si et al. [2016]
⋮

Our Contributions

An extension of the IHS solver MaxHS that supports changing weights incrementally



decision rules (MLIC)
[Malioutov and Meel, 2018]



decision trees (AdaBoost)
[Hu, Siala, Hebrard, and Huguet, 2020]

- blue points $\rightarrow$ earlier iterations
- yellow points $\rightarrow$ later iterations

Outline

- Preliminaries
- The Implicit Hitting Set (IHS) based approach to MaxSAT
- Incremental IHS
- Case Studies
- Conclusions

Preliminaries: Maximum Satisfiability

- Optimisation extension of Boolean Satisfiability (SAT)
- An instance consists of:
  - a set of hard clauses,
  - a set of soft clauses and
  - a weight function w over the soft clauses.
- Find τ that:
  - satisfies all hard clauses and
  - maximises the sum of weights of satisfied soft clauses.

Preliminaries: Maximum Satisfiability

- Optimisation extension of Boolean Satisfiability (SAT)
- An instance consists of:
  - a set of hard clauses,
  - a set of soft clauses and
  - a weight function w over the soft clauses.
- Find τ that:
  - satisfies all hard clauses and
  - maximises the sum of weights of satisfied soft clauses.

$$\mathcal{F}_H = \{(b_1 \vee b_2), (b_2 \vee b_3)\}$$

$$\mathcal{F}_S = \{(\neg b_1), (\neg b_2), (\neg b_3)\}$$

$$w((\neg b_1)) = 2, w((\neg b_2)) = 4,$$
$$w((\neg b_3)) = 3$$

Preliminaries: Maximum Satisfiability

- Optimisation extension of Boolean Satisfiability (SAT)
- An instance consists of:
  - a set of hard clauses,
  - a set of soft clauses and
  - a weight function w over the soft clauses.
- Find τ that:
  - satisfies all hard clauses and
  - maximises the sum of weights of satisfied soft clauses.

$$\mathcal{F}_H = \{(b_1 \vee b_2), (b_2 \vee b_3)\}$$

$$\mathcal{F}_S = \{(\neg b_1), (\neg b_2), (\neg b_3)\}$$

$$w((\neg b_1)) = 2, w((\neg b_2)) = 4,$$
$$w((\neg b_3)) = 3$$

Preliminaries: Maximum Satisfiability

- Optimisation extension of Boolean Satisfiability (SAT)
- An instance consists of:
  - a set of hard clauses,
  - a set of soft clauses and
  - a weight function w over the soft clauses.
- Find τ that:
  - satisfies all hard clauses and
  - maximises the sum of weights of satisfied soft clauses.

$$\tau(b_1) = \tau(b_3) = 0$$
$$\tau(b_2) = 1$$

$$\mathcal{F}_H = \{(b_1 \vee b_2), (b_2 \vee b_3)\}$$

$$\mathcal{F}_S = \{(\neg b_1), (\neg b_2), (\neg b_3)\}$$

$$w((\neg b_1)) = 2, w((\neg b_2)) = 4,$$
$$w((\neg b_3)) = 3$$

$$cost(\tau) = \mathbf{4}$$

$$cost(\mathcal{F}) = \mathbf{4}$$

Preliminaries: Maximum Satisfiability

- Optimisation extension of Boolean Satisfiability (SAT)
- An instance consists of:
  - a set of hard clauses,
  - a set of soft clauses and
  - a weight function w over the soft clauses.
- Find τ that:
  - satisfies all hard clauses and
  - maximises the sum of weights of satisfied soft clauses.

$$\tau(b_1) = \tau(b_3) = 0$$
$$\tau(b_2) = 1$$

$$\mathcal{F}_H = \{(b_1 \vee b_2), (b_2 \vee b_3)\}$$

$$\mathcal{F}_S = \{(\neg b_1), (\neg b_2), (\neg b_3)\}$$

$$\mathbf{w((\neg b_1)) = 1}, w((\neg b_2)) = 4,$$
$$\mathbf{w((\neg b_3)) = 2}$$

$$cost(\tau) = \mathbf{4}$$

$$cost(\mathcal{F}) = \mathbf{3}$$

Preliminaries: Maximum Satisfiability

- Optimisation extension of Boolean Satisfiability (SAT)
- An instance consists of:
  - a set of hard clauses,
  - a set of soft clauses and
  - a weight function w over the soft clauses.
- Find τ that:
  - satisfies all hard clauses and
  - maximises the sum of weights of satisfied soft clauses.

$$\tau(b_1) = \tau(b_3) = 1$$
$$\tau(b_2) = 0$$

$$\mathcal{F}_H = \{(b_1 \vee b_2), (b_2 \vee b_3)\}$$

$$\mathcal{F}_S = \{(\neg b_1), (\neg b_2), (\neg b_3)\}$$

$$\mathbf{w((\neg b_1)) = 1}, w((\neg b_2)) = 4,$$
$$\mathbf{w((\neg b_3)) = 2}$$

$$cost(\tau) = \mathbf{3}$$

$$cost(\mathcal{F}) = \mathbf{3}$$

Preliminaries: UNSAT Cores

$$\mathcal{F}_H = \{(b_1 \vee b_2), (b_2 \vee b_3)\}$$

$$\mathcal{F}_S = \{(\neg b_1), (\neg b_2), (\neg b_3)\}$$

- Central in modern MaxSAT solving (including IHS).
- $\kappa \subset \mathcal{F}_S$ is an *core* if $\mathcal{F}_H \wedge \kappa$ is unsatisfiable

Cores in IHS MaxSAT

$$\sum_{C \in hs} w(C) \leq cost(\mathcal{F})$$

for a minimum-cost hitting set *hs* over *any* set of core.

Preliminaries: UNSAT Cores

- Central in modern MaxSAT solving (including IHS).
- $\kappa \subset \mathcal{F}_S$ is an *core* if $\mathcal{F}_H \wedge \kappa$ is unsatisfiable

$$\mathcal{F}_H = \{(b_1 \vee b_2), (b_2 \vee b_3)\}$$

$$\mathcal{F}_S = \{(\neg b_1), (\neg b_2), (\neg b_3)\}$$

$$\kappa = \{(\neg b_1), (\neg b_2)\}$$

Cores in IHS MaxSAT

$$\sum_{C \in hs} w(C) \leq cost(\mathcal{F})$$

for a minimum-cost hitting set *hs* over *any* set of core.

Preliminaries: UNSAT Cores

- Central in modern MaxSAT solving (including IHS).
- $\kappa \subset \mathcal{F}_S$ is an *core* if $\mathcal{F}_H \wedge \kappa$ is unsatisfiable

$$\mathcal{F}_H = \{(b_1 \vee b_2), (b_2 \vee b_3)\}$$

$$\mathcal{F}_S = \{(\neg b_1), (\neg b_2), (\neg b_3)\}$$

$$\kappa = \{(\neg b_1), (\neg b_2)\}$$

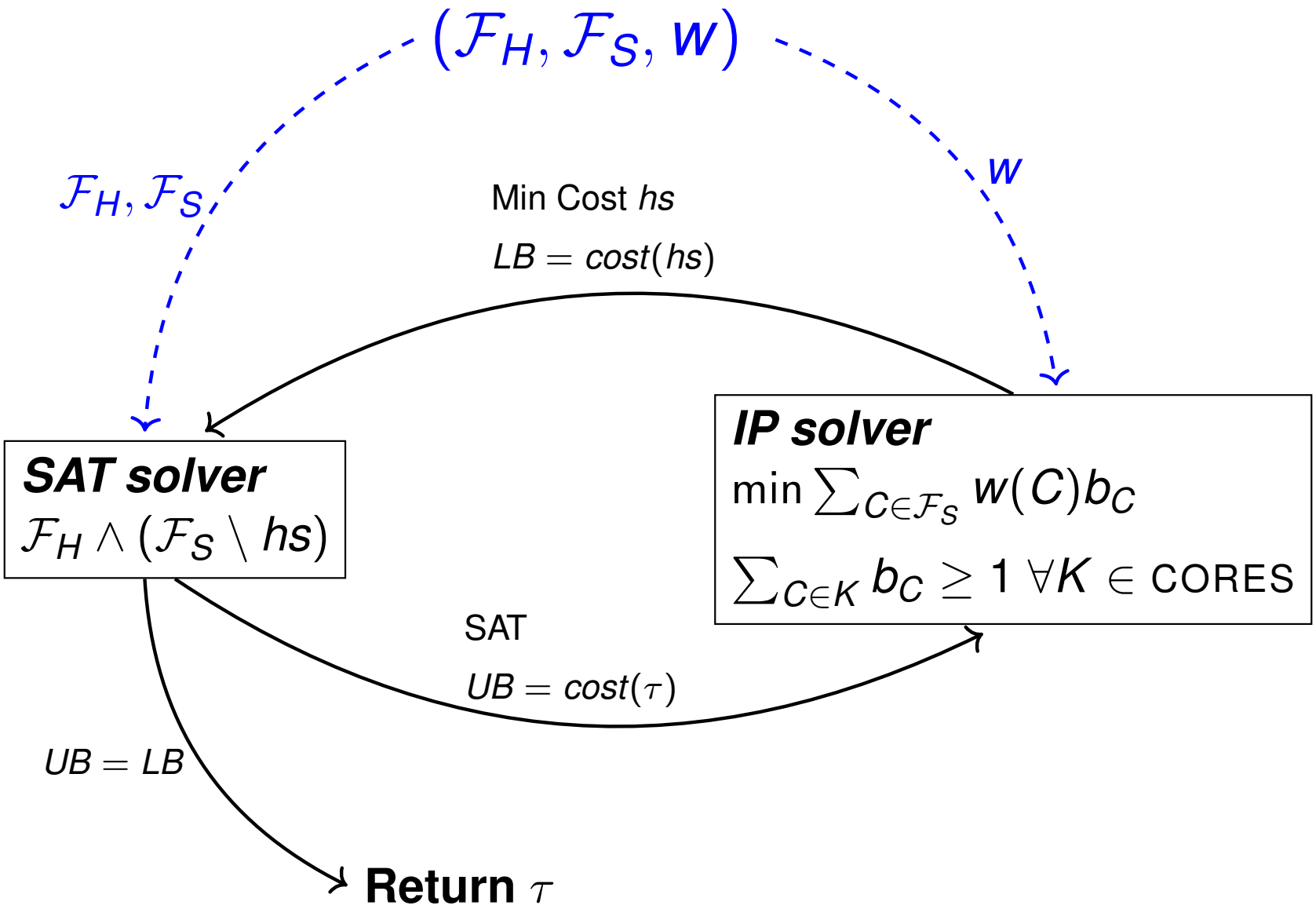
Cores in IHS MaxSAT

$$\sum_{C \in hs} w(C) \leq cost(\mathcal{F})$$

for a minimum-cost hitting set *hs* over *any* set of core.

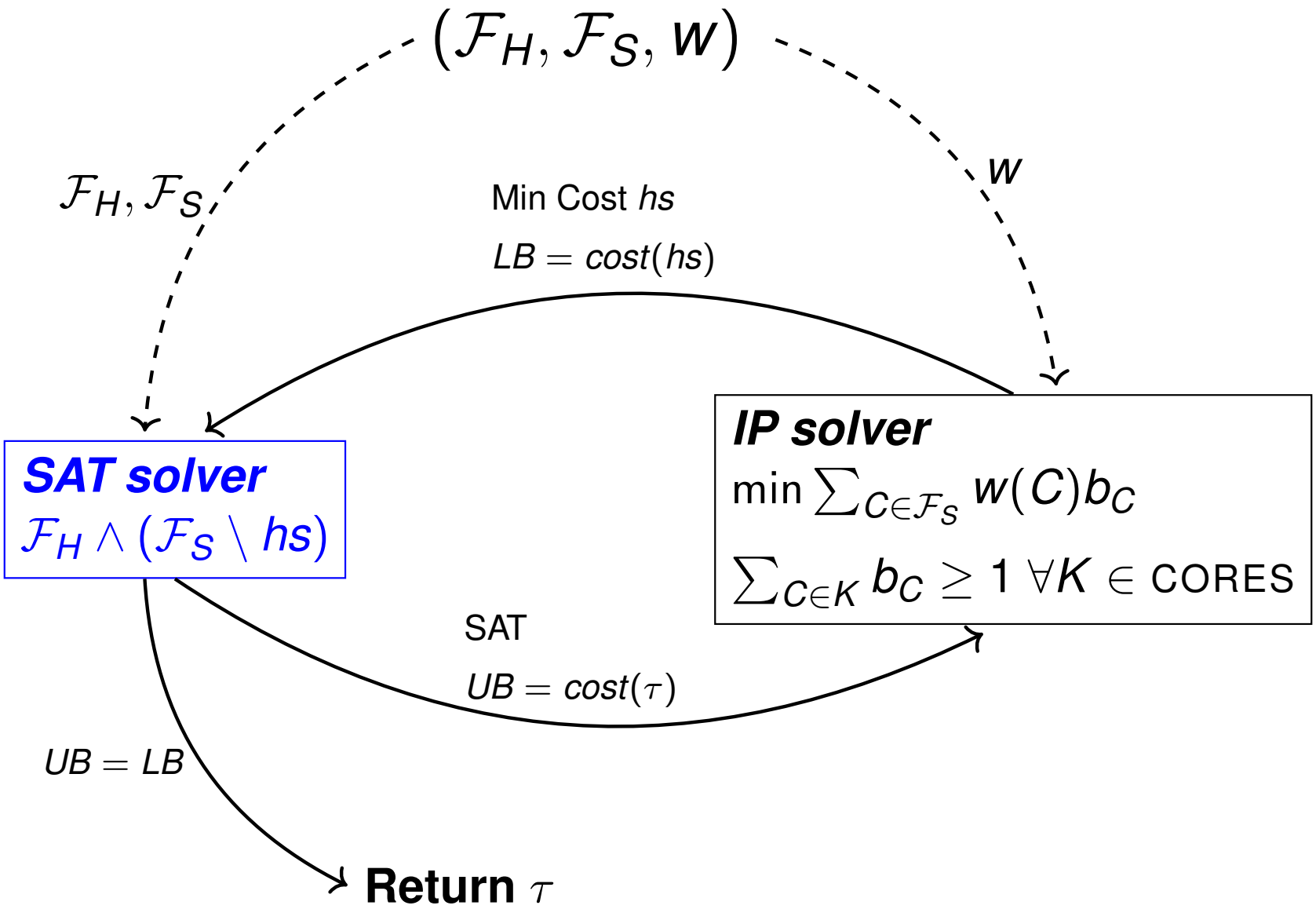
The IHS approach to MaxSAT

[Davies and Bacchus, 2011]



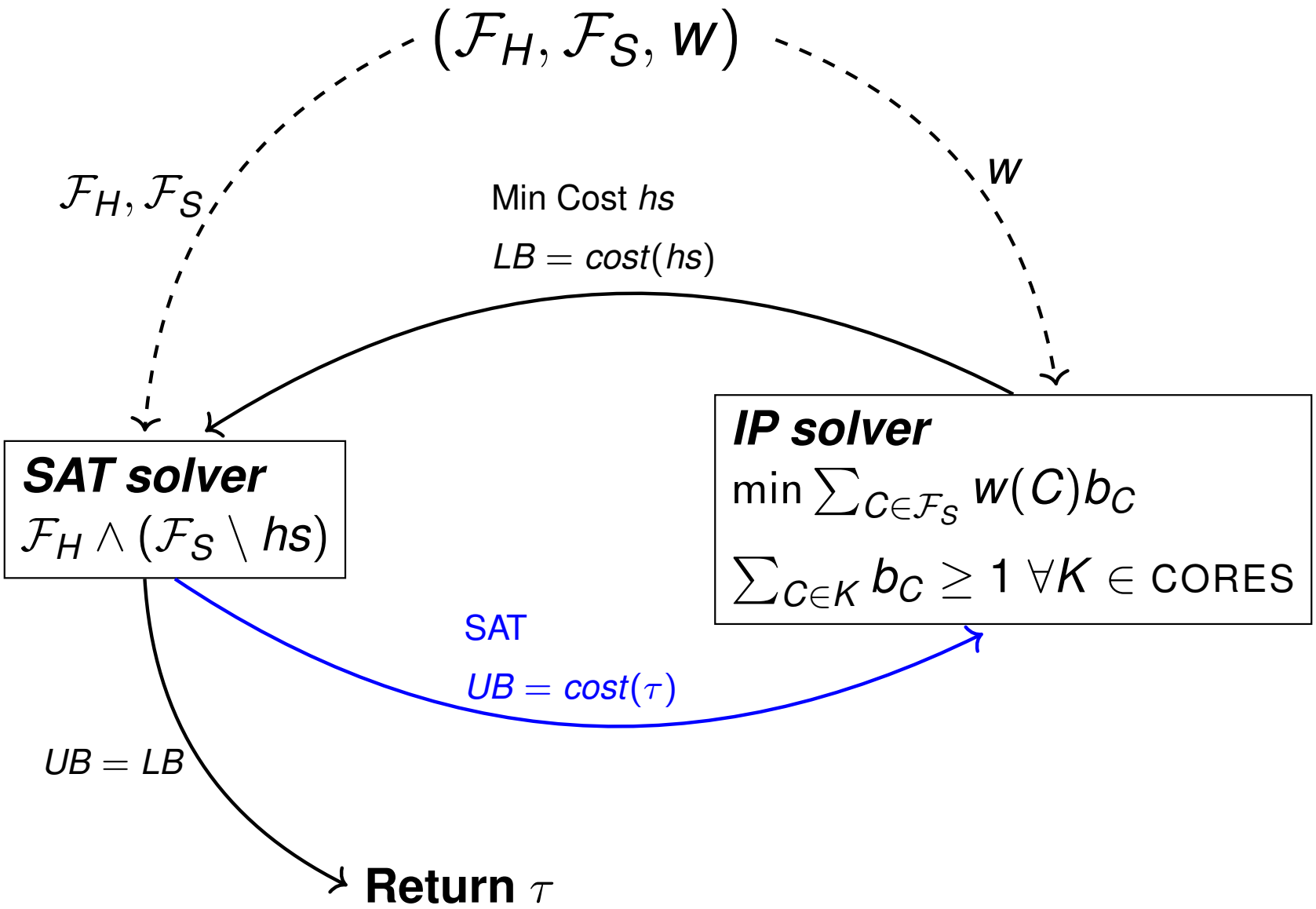
The IHS approach to MaxSAT

[Davies and Bacchus, 2011]



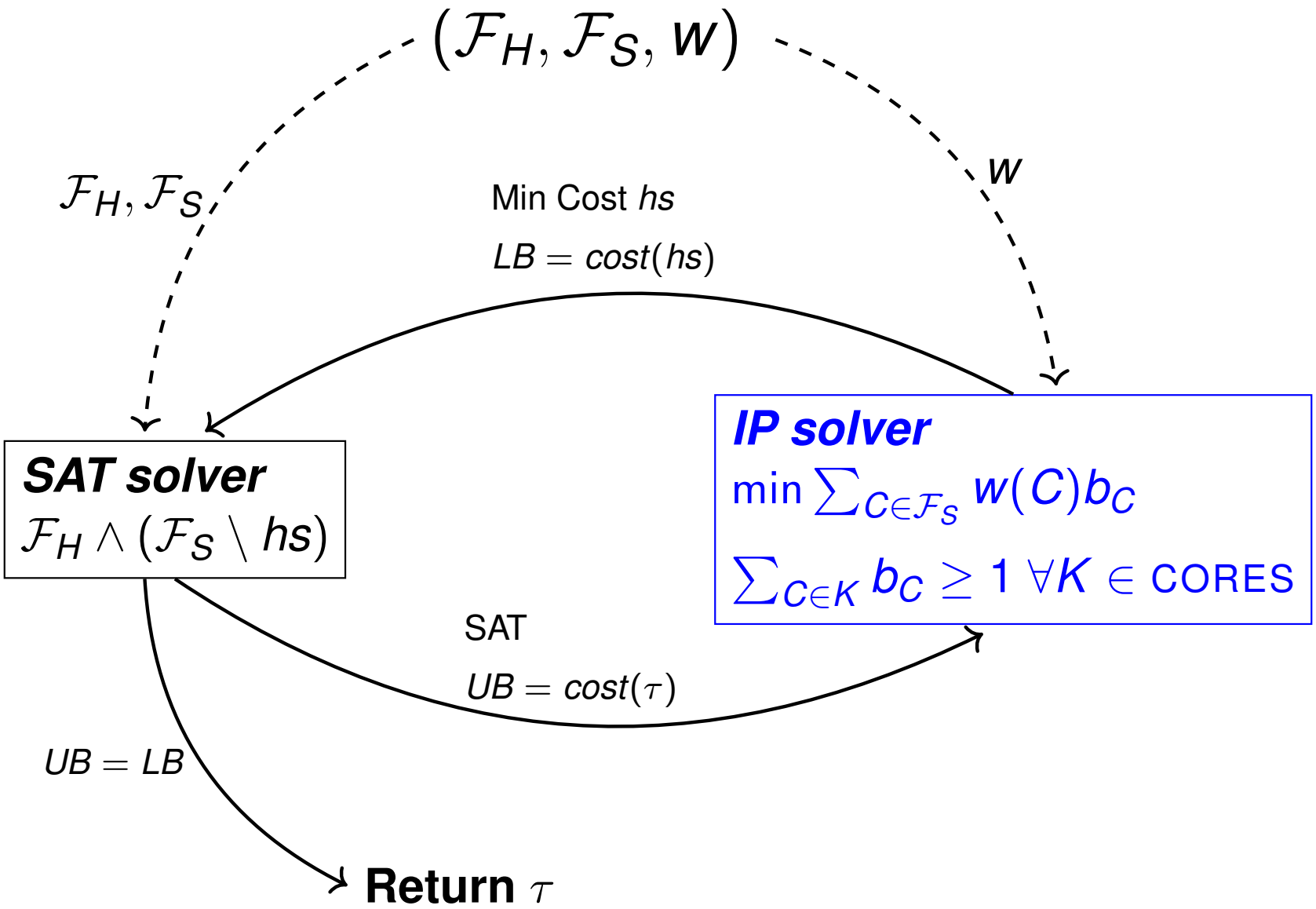
The IHS approach to MaxSAT

[Davies and Bacchus, 2011]



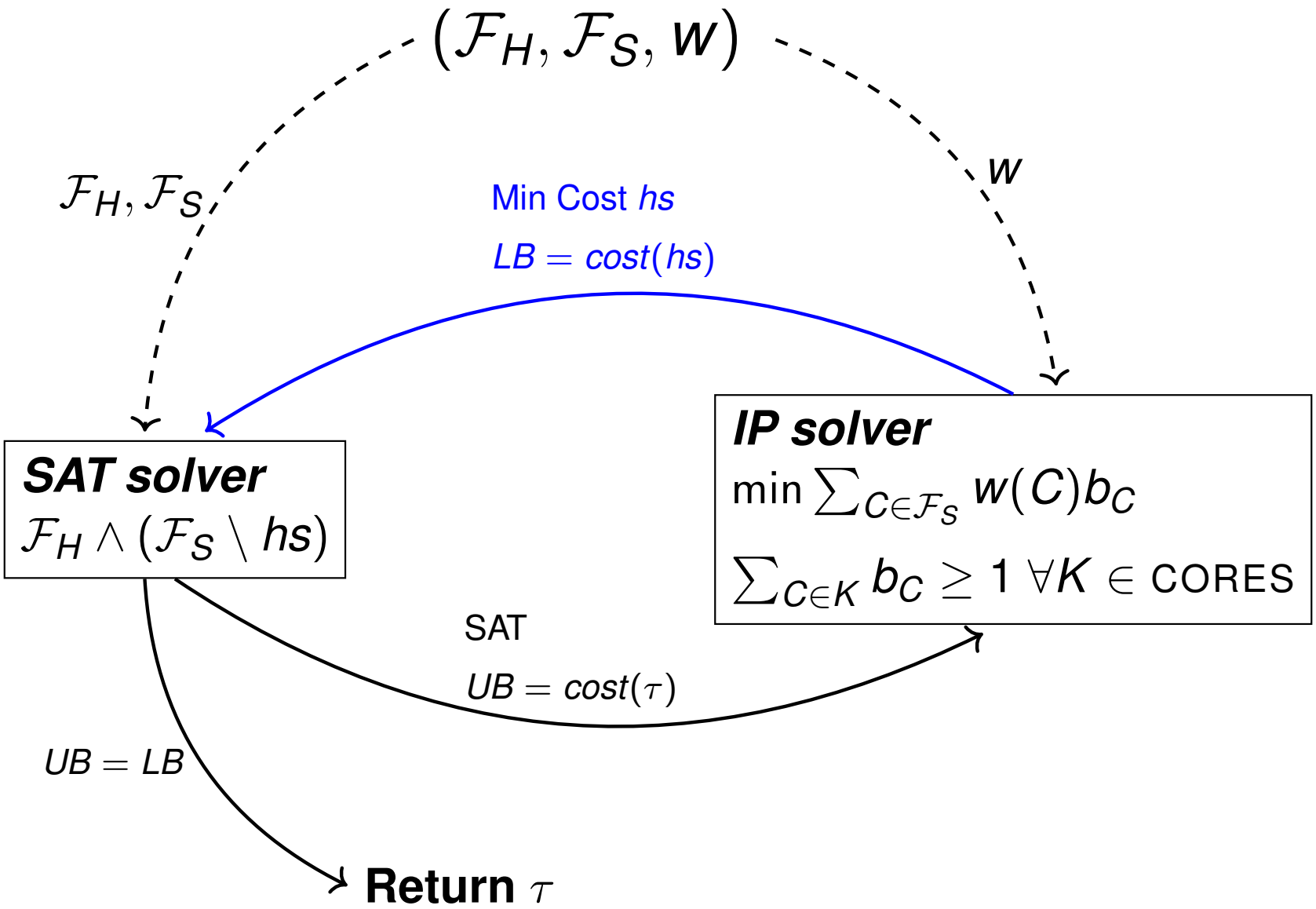
The IHS approach to MaxSAT

[Davies and Bacchus, 2011]



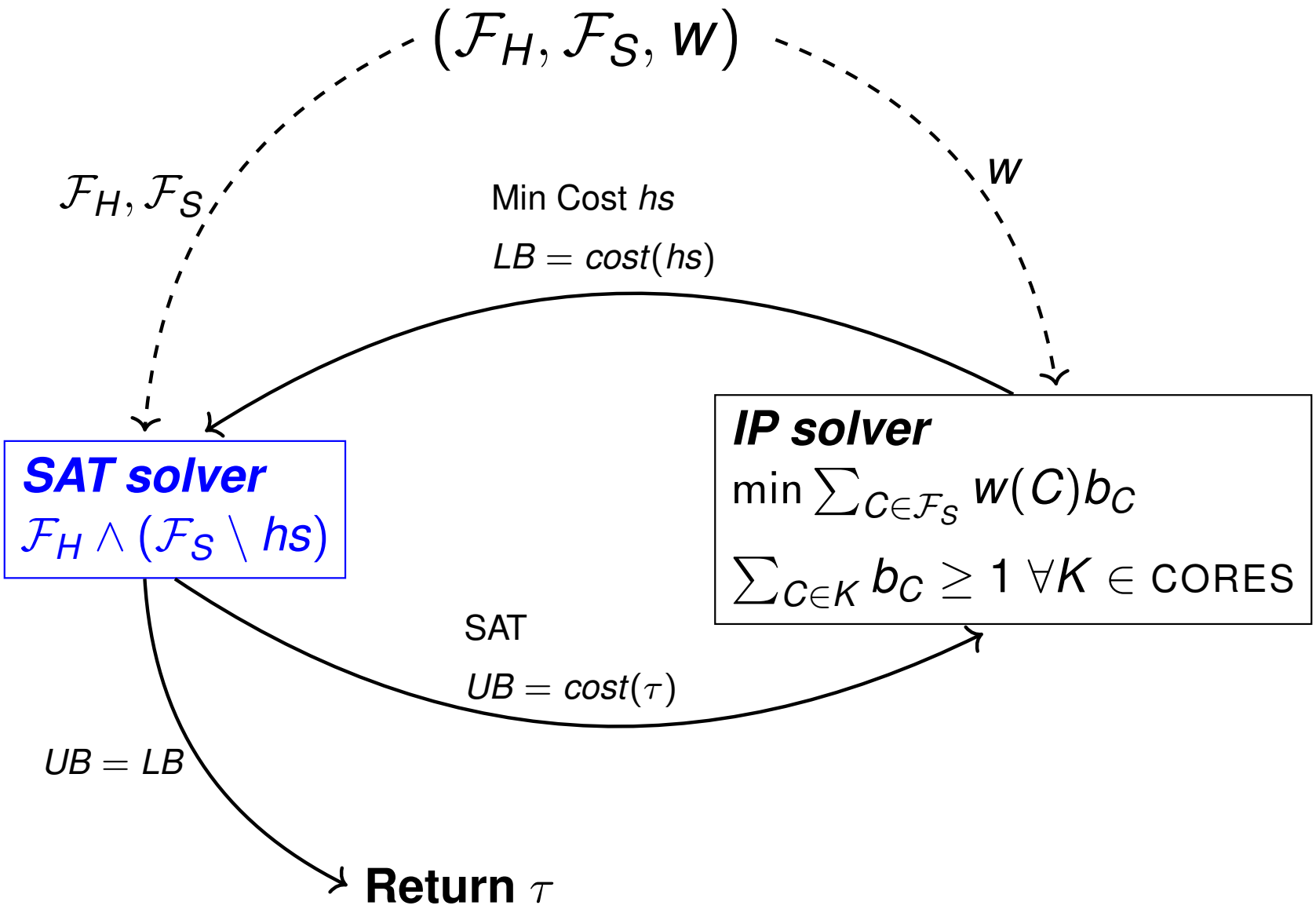
The IHS approach to MaxSAT

[Davies and Bacchus, 2011]



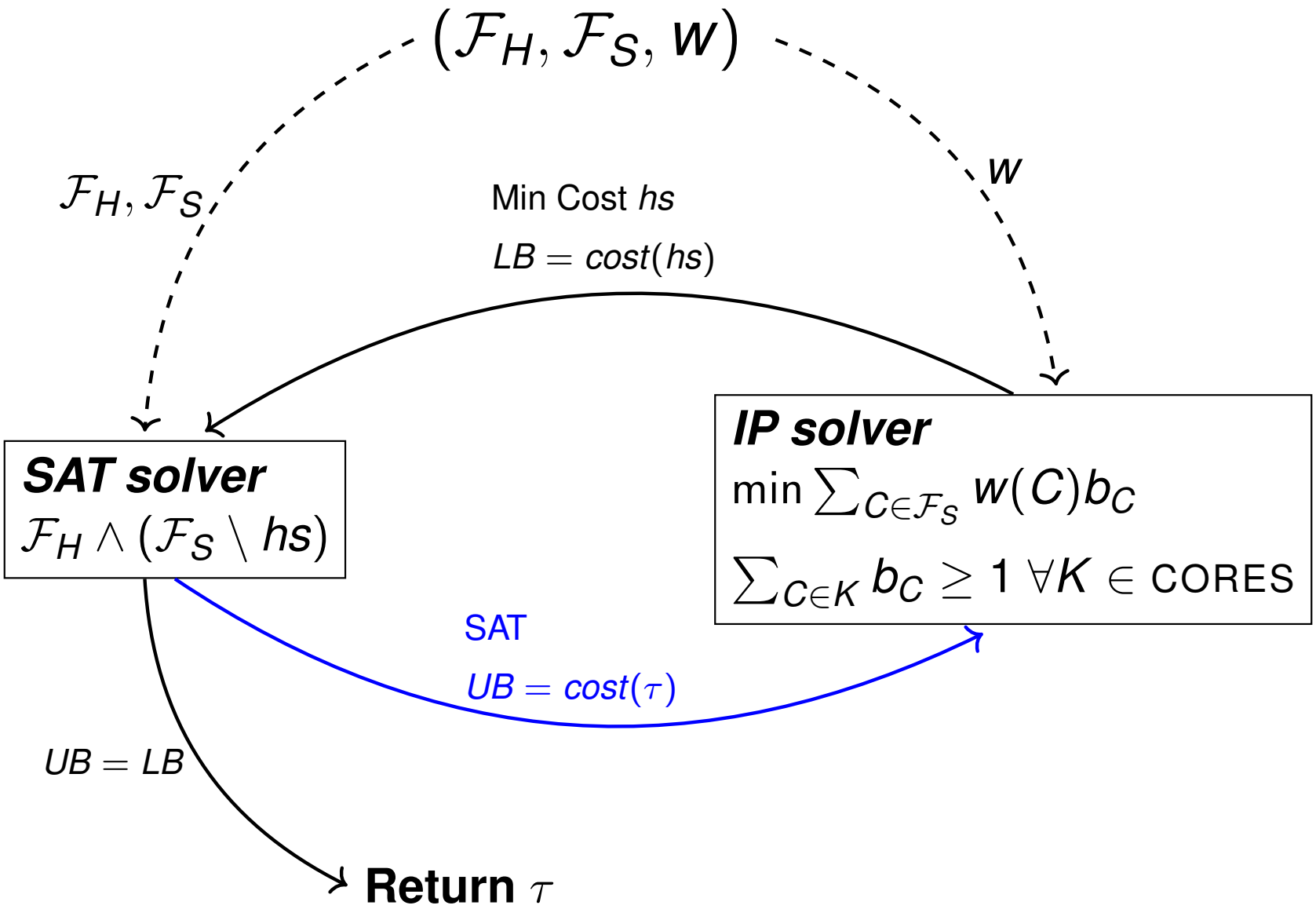
The IHS approach to MaxSAT

[Davies and Bacchus, 2011]



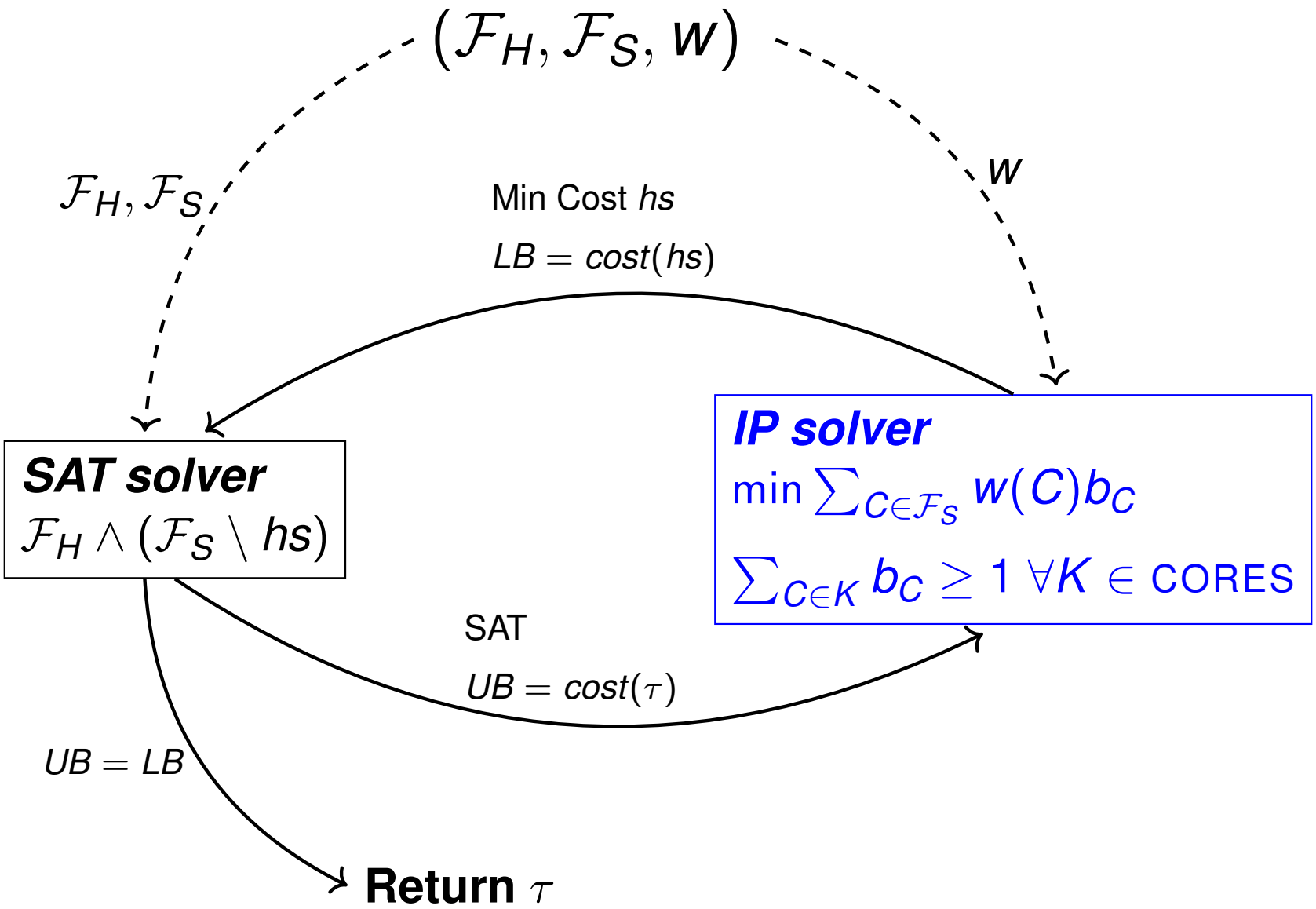
The IHS approach to MaxSAT

[Davies and Bacchus, 2011]



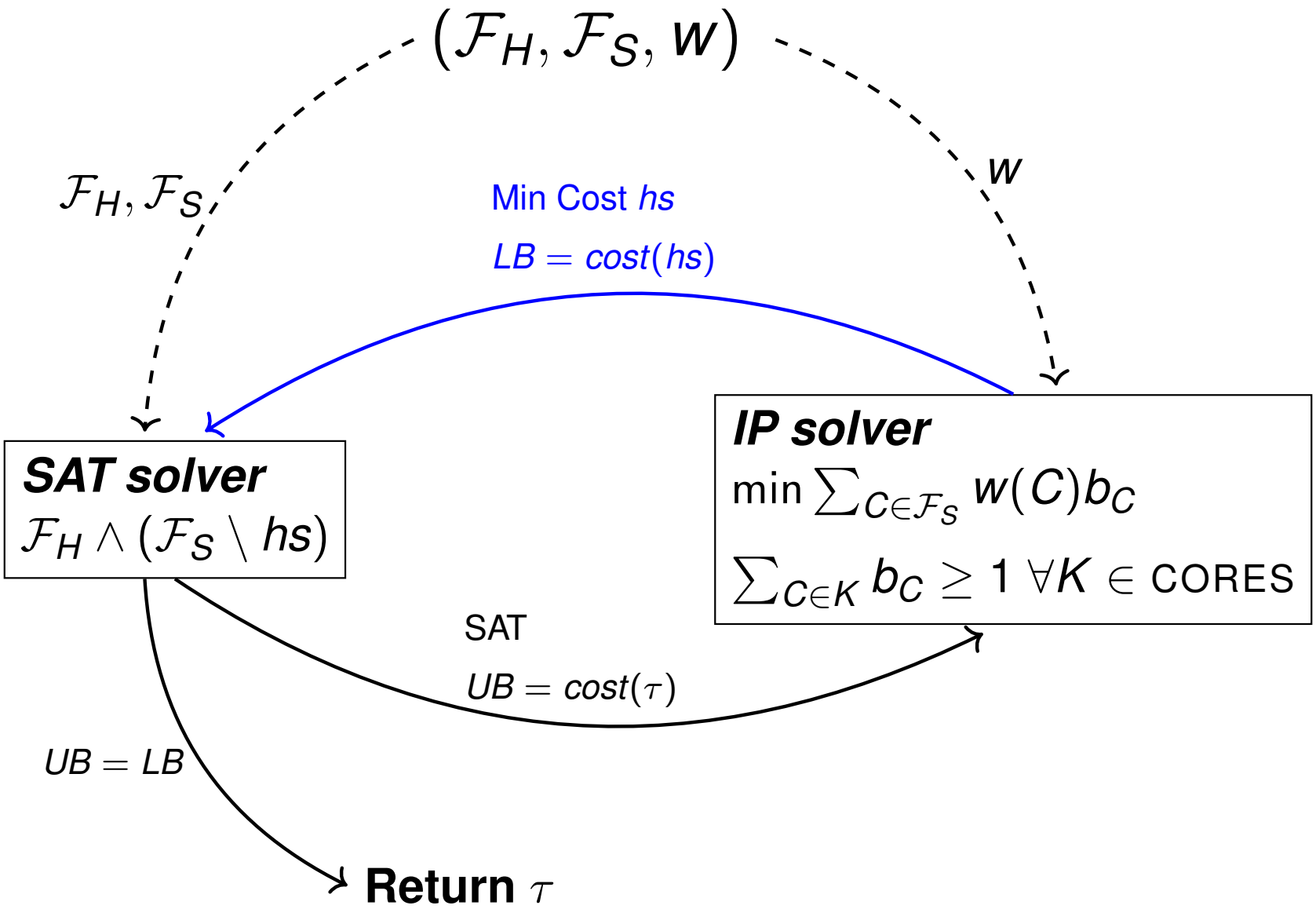
The IHS approach to MaxSAT

[Davies and Bacchus, 2011]



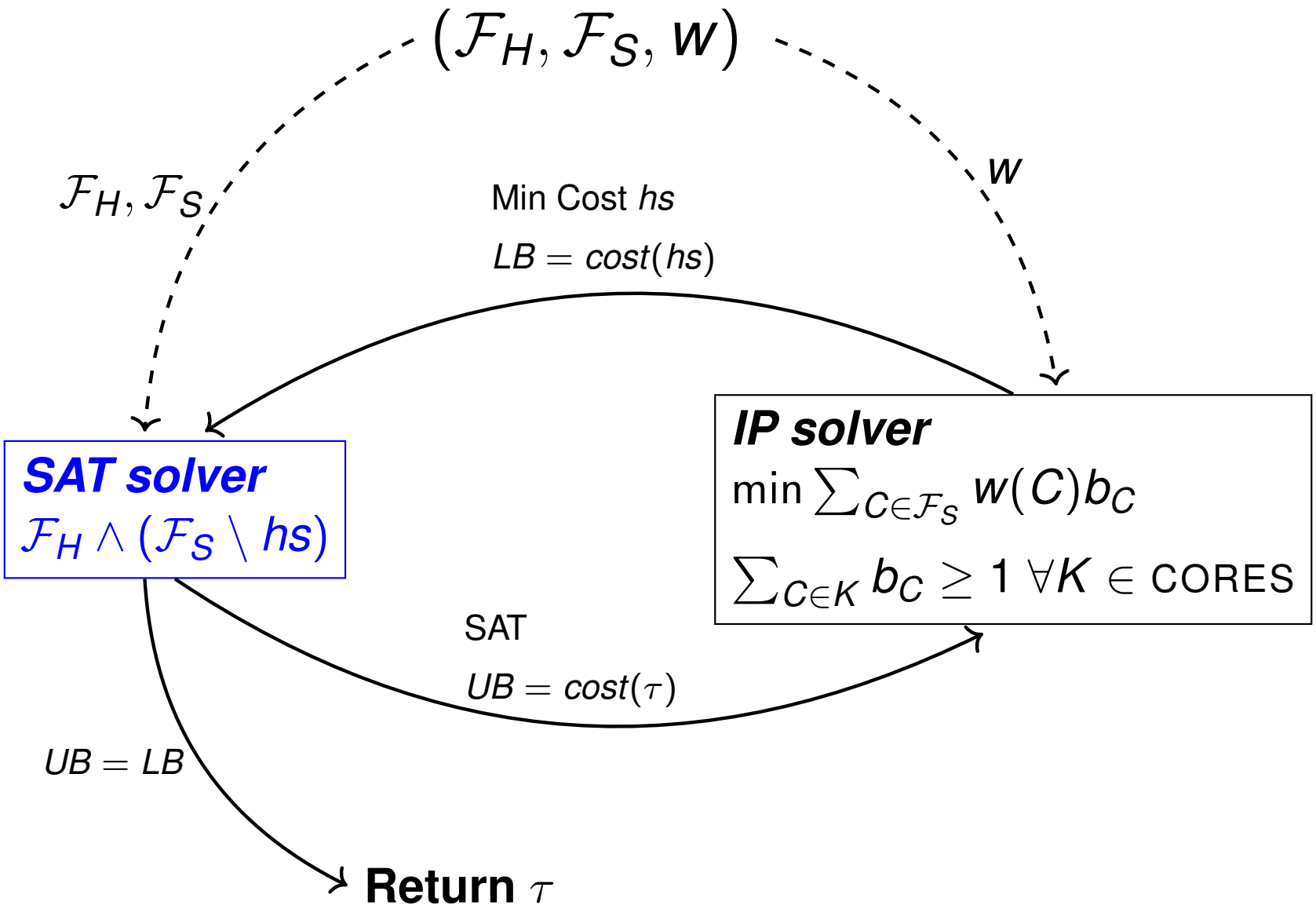
The IHS approach to MaxSAT

[Davies and Bacchus, 2011]



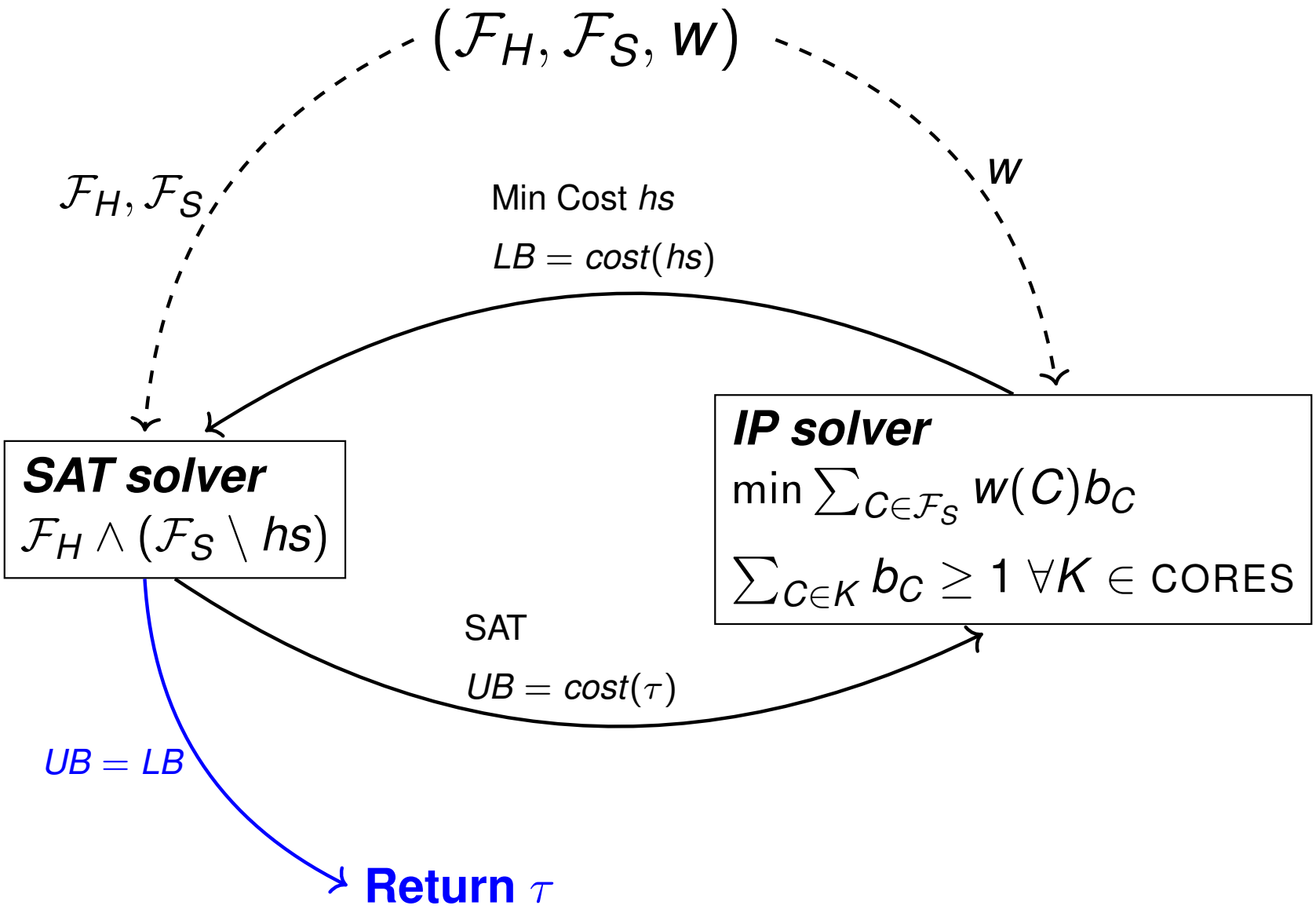
The IHS approach to MaxSAT

[Davies and Bacchus, 2011]



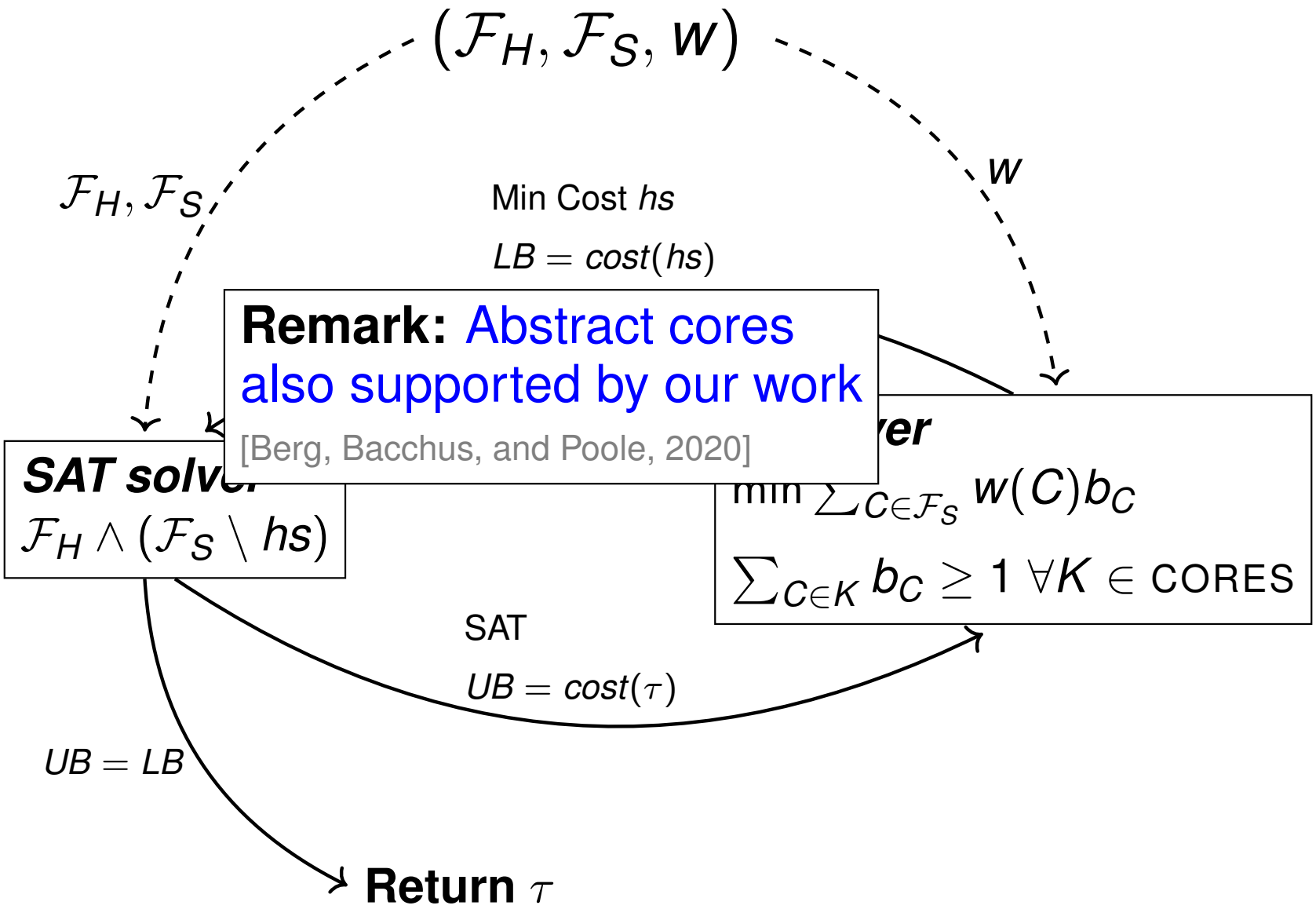
The IHS approach to MaxSAT

[Davies and Bacchus, 2011]



The IHS approach to MaxSAT

[Davies and Bacchus, 2011]



Towards Incrementality

Incremental Solving

Changing weights

$$\mathcal{F}_H = \{(b_1 \vee b_2), (b_2 \vee b_3)\}$$

$$\mathcal{F}_S = \{(\neg b_1), (\neg b_2), (\neg b_3)\}$$

Focus of our work — incremental weight changes

Iteratively compute optimal solutions to instances with the same constraints but different weights.

The ”naive” solution: NON-INC

Independently invoke an IHS MaxSAT solver on each instance.

Incremental Solving

Changing weights

$$\mathcal{F}_H = \{(b_1 \vee b_2), (b_2 \vee b_3)\}$$

$$\mathcal{F}_S = \{(\neg b_1), (\neg b_2), (\neg b_3)\}$$

$w_1((\neg b_1)) = 2, w_1((\neg b_2)) = 4, w_1((\neg b_3)) = 3$
$\tau_1(b_1) = \tau_1(b_3) = 0, \tau_1(b_2) = 1$
$cost(\tau) = 4$

Focus of our work — incremental weight changes

Iteratively compute optimal solutions to instances with the same constraints but different weights.

The ”naive” solution: NON-INC

Independently invoke an IHS MaxSAT solver on each instance.

Incremental Solving

Changing weights

$$\mathcal{F}_H = \{(b_1 \vee b_2), (b_2 \vee b_3)\}$$

$$\mathcal{F}_S = \{(\neg b_1), (\neg b_2), (\neg b_3)\}$$

$w_1((\neg b_1)) = 2, w_1((\neg b_2)) = 4, w_1((\neg b_3)) = 3$
$\tau_1(b_1) = \tau_1(b_3) = 0, \tau_1(b_2) = 1$
$cost(\tau) = 4$

$w_2((\neg b_1)) = 1, w_2((\neg b_2)) = 3, w_2((\neg b_3)) = 1$
$\tau_2(b_1) = \tau_2(b_3) = 1, \tau_2(b_2) = 0$
$cost(\tau) = 2$

Focus of our work — incremental weight changes

Iteratively compute optimal solutions to instances with the same constraints but different weights.

The ”naive” solution: NON-INC

Independently invoke an IHS MaxSAT solver on each instance.

Incremental Solving

Changing weights

$$\mathcal{F}_H = \{(b_1 \vee b_2), (b_2 \vee b_3)\}$$

$$\mathcal{F}_S = \{(\neg b_1), (\neg b_2), (\neg b_3)\}$$

$w_1((\neg b_1)) = 2, w_1((\neg b_2)) = 4, w_1((\neg b_3)) = 3$
$\tau_1(b_1) = \tau_1(b_3) = 0, \tau_1(b_2) = 1$
$cost(\tau) = 4$

$w_2((\neg b_1)) = 1, w_2((\neg b_2)) = 3, w_2((\neg b_3)) = 1$
$\tau_2(b_1) = \tau_2(b_3) = 1, \tau_2(b_2) = 0$
$cost(\tau) = 2$

Focus of our work — incremental weight changes

Iteratively compute optimal solutions to instances with the same constraints but different weights.

The ”naive” solution: NON-INC

Independently invoke an IHS MaxSAT solver on each instance.

Why IHS for incrementality?

Observations:

- Formula not altered during search
- Definition of core does not depend on weights.
  - $\rightarrow$ Cores can be preserved.
  - $\rightarrow$ Only objective needs to be altered in the IP-solver.
- The SAT solver knows nothing about the weights.
  - $\rightarrow$ No need to reinitialise.

Why IHS for incrementality?

Observations:

- Formula not altered during search
- Definition of core does not depend on weights.
  - $\rightarrow$ Cores can be preserved.
  - $\rightarrow$ Only objective needs to be altered in the IP-solver.
- The SAT solver knows nothing about the weights.
  - $\rightarrow$ No need to reinitialise.

Why IHS for incrementality?

Observations:

- Formula not altered during search
- Definition of core does not depend on weights.
  - $\rightarrow$ Cores can be preserved.
  - $\rightarrow$ Only objective needs to be altered in the IP-solver.
- The SAT solver knows nothing about the weights.
  - $\rightarrow$ No need to reinitialise.

Incremental IHS

in theory

IHS-INC($\mathcal{F}, k$)
  CORES $\leftarrow \emptyset$
  for $i = 1, \ldots, k$ **do**
    $w =$ next-w()
    $UB \leftarrow cost(\mathcal{F}, w, \tau_{best})$
    $\tau_{best} \leftarrow$ IHS($\mathcal{F}$, CORES, UB)
    output τ_{best}

Note:

- Abstract cores need (slightly) more care
  - details in paper...

Incremental IHS

in theory

IHS-INC($\mathcal{F}, k$)
- CORES $\leftarrow \emptyset$
- **for** $i = 1, \ldots, k$ **do**
  - $w =$ next-w()
  - $UB \leftarrow cost(\mathcal{F}, w, \tau_{best})$
  - $\tau_{best} \leftarrow$ IHS($\mathcal{F}$, CORES, UB)
  - **output** τ_{best}

Note:

- Abstract cores need (slightly) more care
  - details in paper...

Incremental IHS

On top of MaxHS [Davies and Bacchus, 2011]

In practice: A formula $\mathcal{F}$ is simplified to $\textsc{simp}(\mathcal{F})$ prior to search
$\rightarrow$Realizing `IHS-INC` needs non-trivial amount of engineering.

Incremental IHS

On top of MaxHS [Davies and Bacchus, 2011]

In practice: A formula $\mathcal{F}$ is simplified to $\text{SIMP}(\mathcal{F})$ prior to search
$\rightarrow$Realizing `IHS-INC` needs non-trivial amount of engineering.

Removal of dublicates: $(C, w_1), (C, w_2) \in \mathcal{F}_S \rightarrow (C, w_1 + w_2) \in \text{SIMP}(\mathcal{F})_S$

Need to remember original indexes.

Contradictory softs: $((x), w_1), ((\neg x), w_2) \in \mathcal{F}_S \rightarrow \begin{cases} ((x), w_1 - w_2) \in \text{SIMP}(\mathcal{F})_S \\ \text{if } w_1 > w_2 \\ ((\neg x), w_2 - w_1) \in \text{SIMP}(\mathcal{F})_S \\ \text{else.} \end{cases}$

Need to remember reification variables and change if needed.

$\vdots$

Not all techniques can be kept

Not all techniques can be kept

Hardening: $(C, w_1) \in \mathcal{F}_S \quad \exists cost(\tau) < w_1 \rightarrow C \in \text{SIMP}(\mathcal{F})_H$

Can not (yet) be used in the incremental setting.

Reduced cost fixing: [Bacchus, Hyttinen, Järvisalo, and Saikko, 2017]

Can not (yet) be used in the incremental setting.

Case Studies

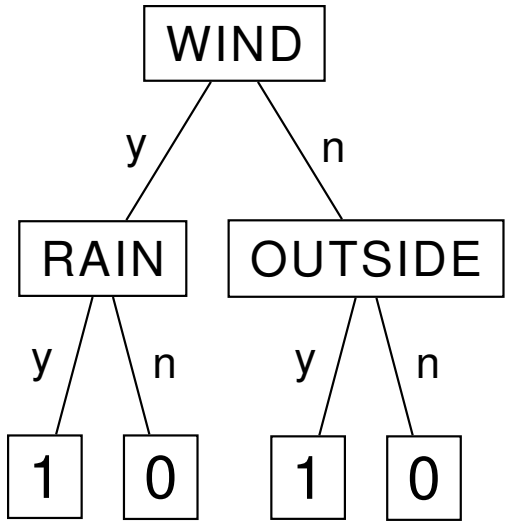
Learning (boosted) decision trees

[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:
  - $\text{DEPTH}(\mathcal{T}) \leq U$
  - $|\{e_i \mid c_i = \mathcal{T}(e_i)\}|$ 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

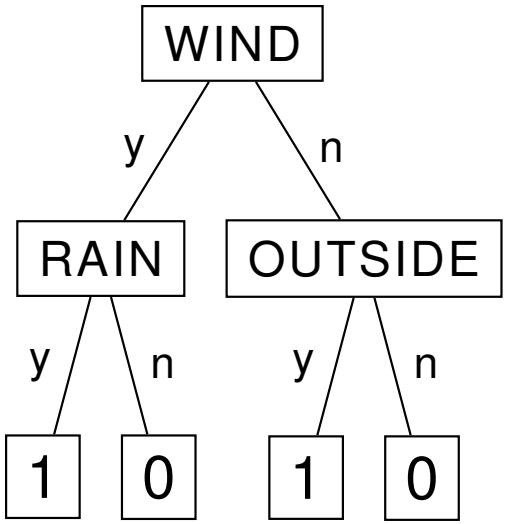
Learning (boosted) decision trees

[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

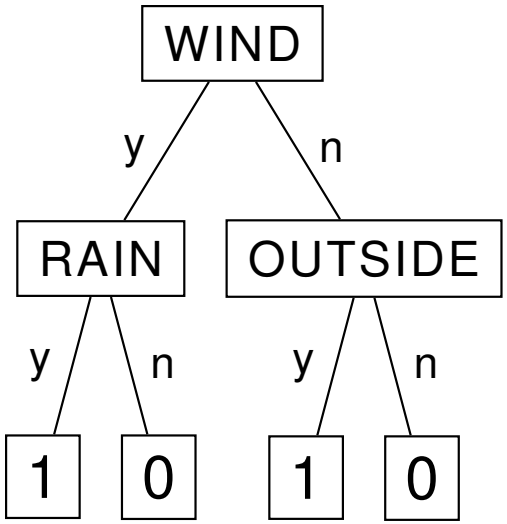
Learning (boosted) decision trees

[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

Learning (boosted) decision trees

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

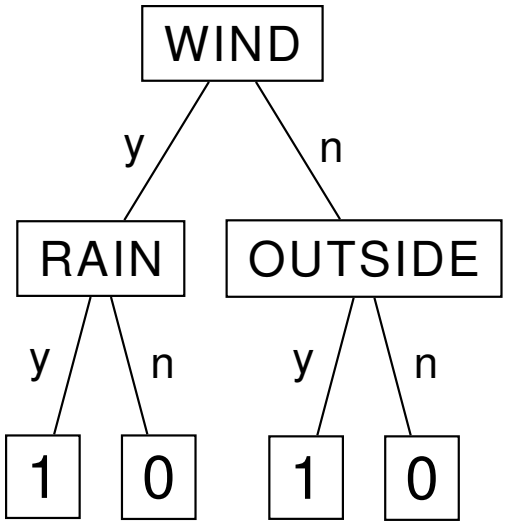
Decision Trees:

- 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

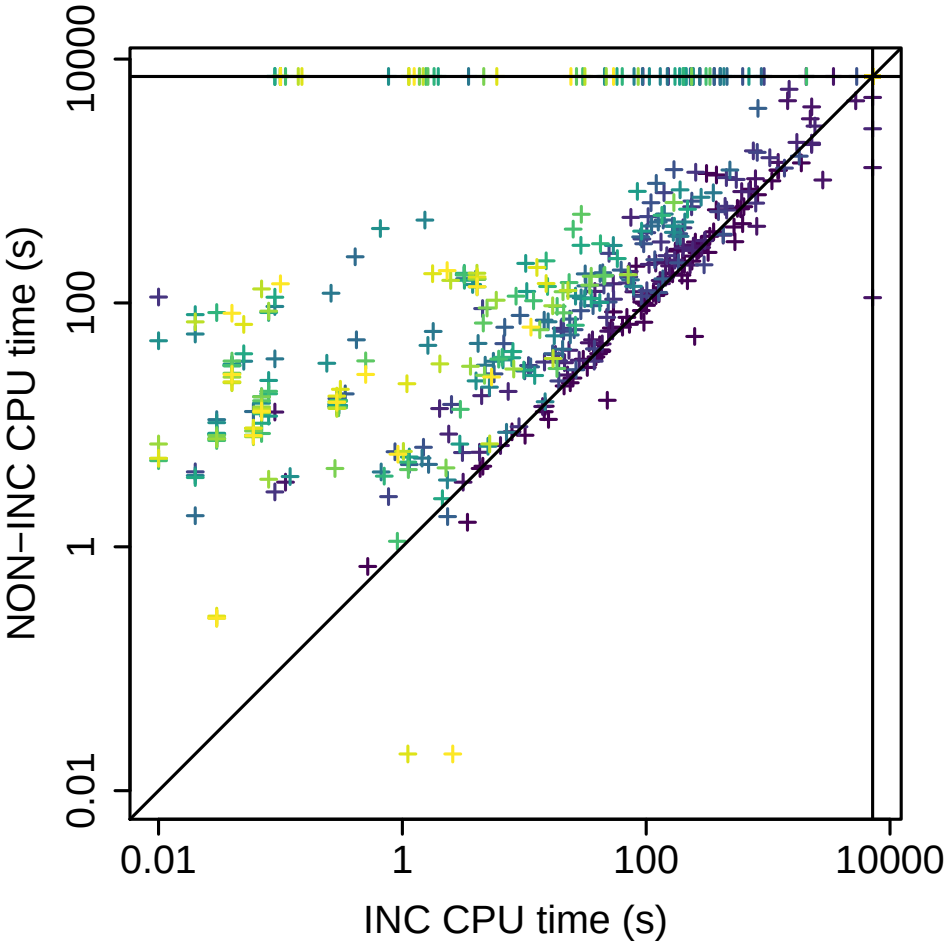
Incrementality: AdaBoost

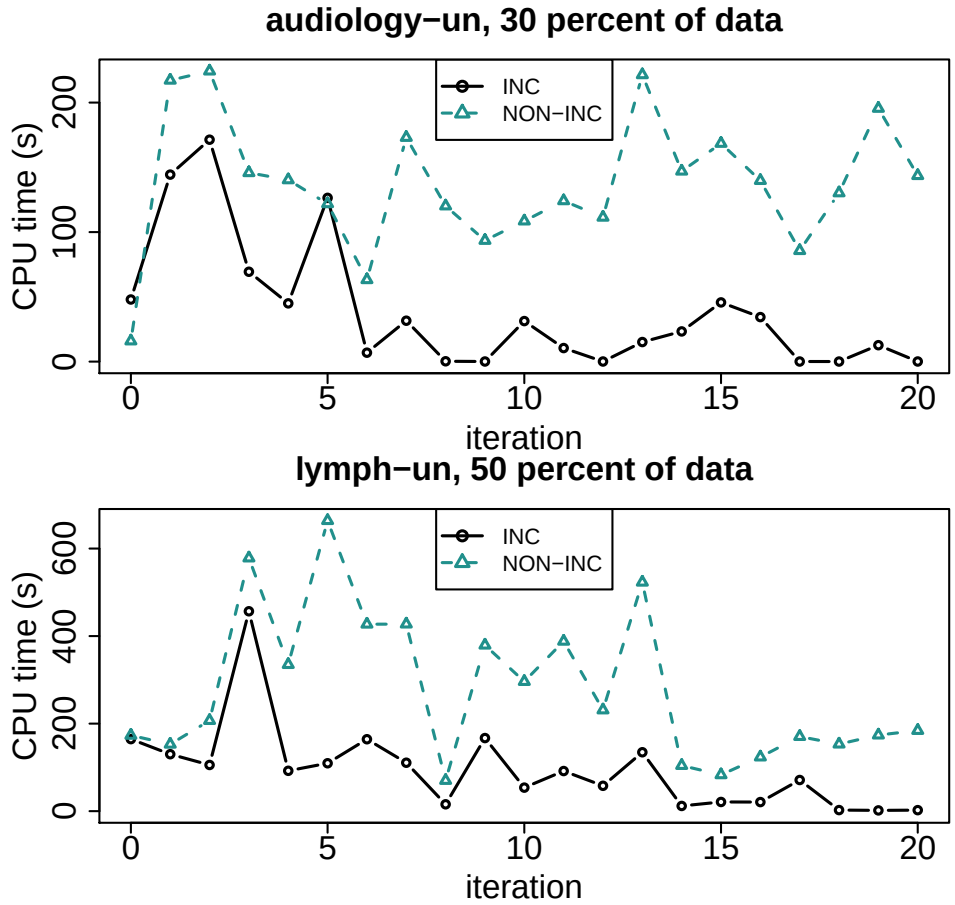
weights $=$ priorities of examples
learn diverse trees that prioritize different examples



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

Incrementality

λ = trade-off between interpretability and accuracy. Learn rules for different λ.

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

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

Incrementality

λ = trade-off between interpretability and accuracy. Learn rules for different λ.

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

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

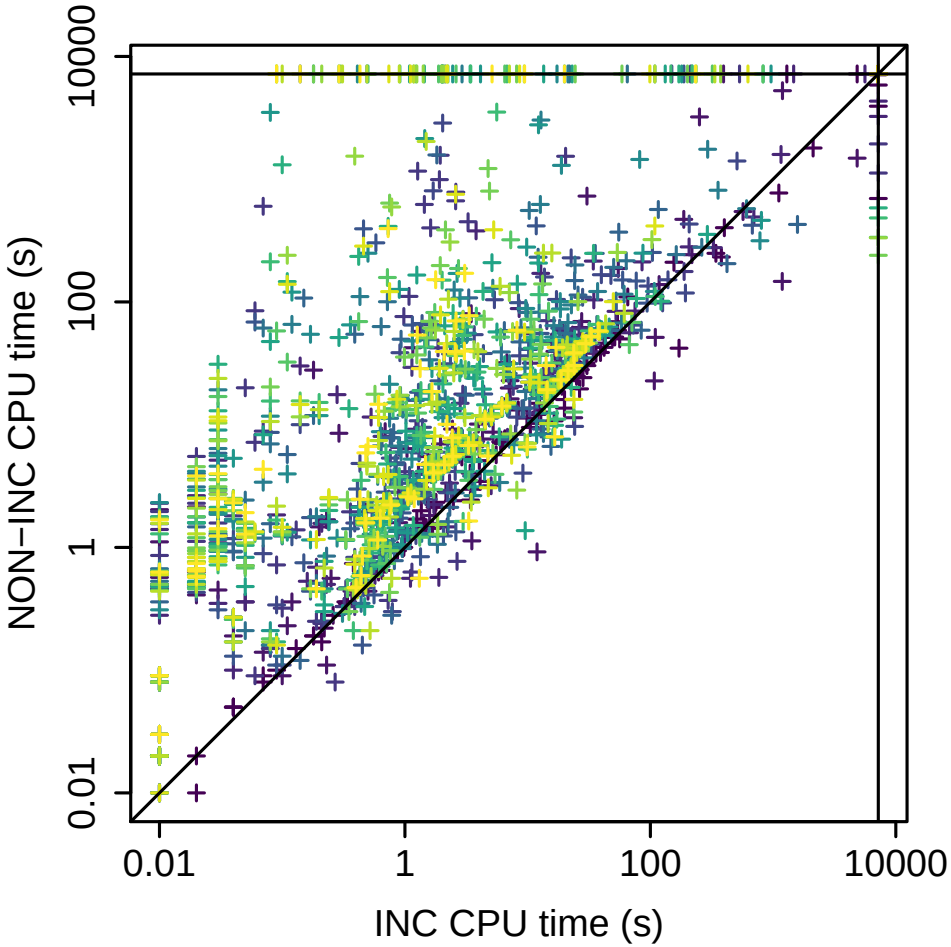
Incrementality

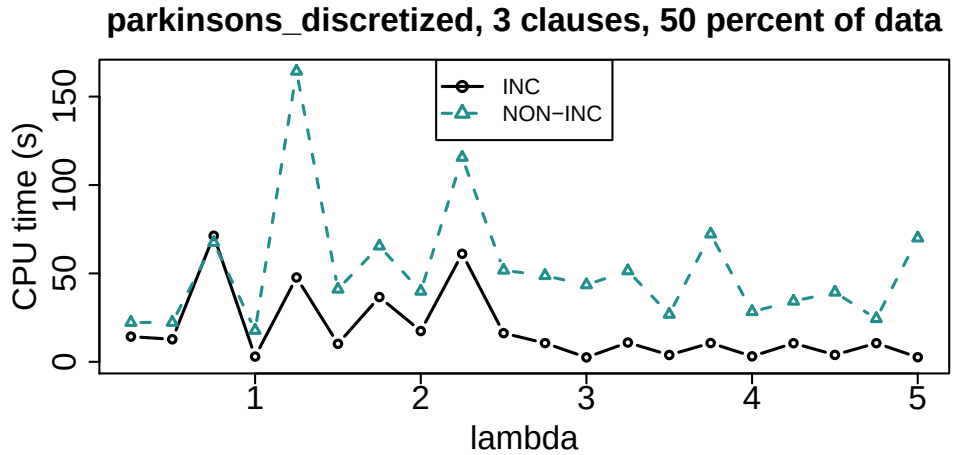
$\lambda =$ trade-off between interpretability and accuracy.
Learn rules for different λ.

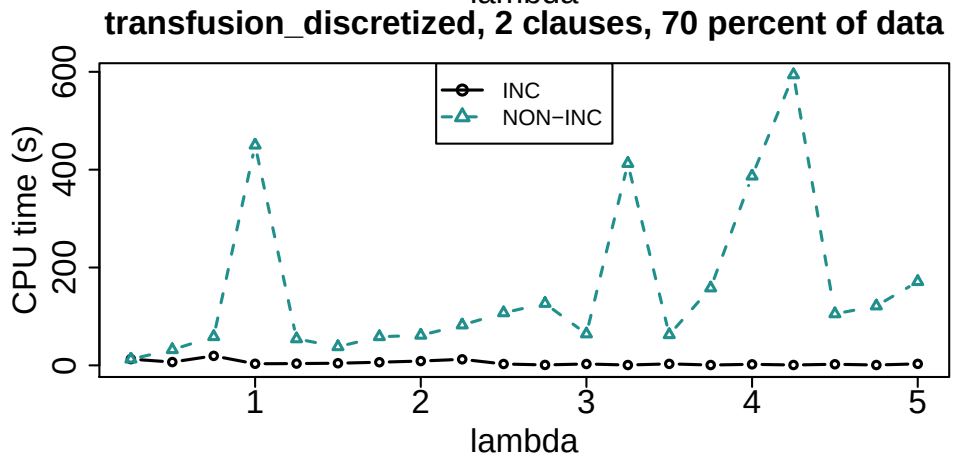
$$\mathcal{R} = (x_{\text{RAIN}}) \wedge (x_{\text{WIND}} \vee x_{\text{OUTSIDE}})$$

Results

Decision Rules







Results

Decision Rules

Take Home Message:
The increased performance of preserving cores far outweighs having to leave out (some) simplification techniques.

Conclusions

- Development of applications of incremental MaxSAT needs incremental solvers.
- In this paper we:
  - Extend the IHS MaxSAT algorithm to changing weights
  - Show potential of resulting algorithm when learning interpretable classifiers.
- Future work:
  - Further forms of incrementality (e.g. assumptions)
  - Other solver types

Conclusions

- Development of applications of incremental MaxSAT needs incremental solvers.
- In this paper we:
  - Extend the IHS MaxSAT algorithm to changing weights
  - Show potential of resulting algorithm when learning interpretable classifiers.
- Future work:
  - Further forms of incrementality (e.g. assumptions)
  - Other solver types

Conclusions

- Development of applications of incremental MaxSAT needs incremental solvers.
- In this paper we:
  - Extend the IHS MaxSAT algorithm to changing weights
  - Show potential of resulting algorithm when learning interpretable classifiers.
- Future work:
  - Further forms of incrementality (e.g. assumptions)
  - Other solver types

Conclusions

- Development of applications of incremental MaxSAT needs incremental solvers.
- In this paper we:
  - Extend the IHS MaxSAT algorithm to changing weights
  - Show potential of resulting algorithm when learning interpretable classifiers.
- Future work:
  - Further forms of incrementality (e.g. assumptions)
  - Other solver types

Bibliography I

Fahiem Bacchus, Antti Hyttinen, Matti Järvisalo, and Paul Saikko. Reduced cost fixing in maxsat. In *Proc. CP*, volume 10416 of *LNCS*, pages 641–651. Springer, 2017. doi: 10.1007/978-3-319-66158-2_41. URL https://doi.org/10.1007/978-3-319-66158-2_41.

Fahiem Bacchus, Matti Järvisalo, and Ruben Martins. *Maximum Satisfiability*, chapter 24, pages 929–991. Frontiers in Artificial Intelligence and Applications. IOS Press, 2021. doi: 10.3233/FAIA201008.

Jeremias Berg, Fahiem Bacchus, and Alex Poole. Abstract cores in implicit hitting set MaxSat solving. In *Proc. SAT*, volume 12178 of *Lecture Notes in Computer Science*, pages 277–294. Springer, 2020.

Alessandro Cimatti, Alberto Griggio, Bastiaan Joost Schaafsma, and Roberto Sebastiani. A modular approach to maxsat modulo theories. In *SAT*, volume 7962 of *Lecture Notes in Computer Science*, pages 150–165. Springer, 2013.

Jessica Davies and Fahiem Bacchus. Solving MAXSAT by solving a sequence of simpler SAT instances. In Jimmy Ho-Man Lee, editor, *Proc. CP*, volume 6876 of *LNCS*, pages 225–239. Springer, 2011. URL https://doi.org/10.1007/978-3-642-23786-7_19.

Bishwamittra Ghosh and Kuldeep S. Meel. IMLI: an incremental framework for maxsat-based learning of interpretable classification rules. In Vincent Conitzer, Gillian K. Hadfield, and Shannon Vallor, editors, *Proc. AIES*, pages 203–210. ACM, 2019. URL https://doi.org/10.1145/3306618.3314283.

Hao Hu, Mohamed Siala, Emmanuel Hebrard, and Marie-José Huguet. Learning optimal decision trees with MaxSAT and its integration in AdaBoost. In *Proc. IJCAI*, pages 1170–1176. ijcai.org, 2020.

Alexey Ignatiev, António Morgado, and João Marques-Silva. RC2: an efficient maxsat solver. *J. Satisf. Boolean Model. Comput.*, 11(1):53–64, 2019. URL https://doi.org/10.3233/SAT190116.

Dmitry Malioutov and Kuldeep S. Meel. MLIC: A MaxSAT-based framework for learning interpretable classification rules. In *Proc. CP*, volume 11008 of *Lecture Notes in Computer Science*, pages 312–327. Springer, 2018.

Ravi Mangal, Xin Zhang, Aditya Kamath, Aditya V. Nori, and Mayur Naik. Scaling relational inference using proofs and refutations. In *AAAI*, pages 3278–3286. AAAI Press, 2016.

Maxime Mulamba, Jayanta Mandi, Michelangelo Diligenti, Michele Lombardi, Victor Bucarey, and Tias Guns. Contrastive losses and solution caching for predict-and-optimize. In *IJCAI*, pages 2833–2840. ijcai.org, 2021.

Nina Narodytska, Alexey Ignatiev, Filipe Pereira, and João Marques-Silva. Learning optimal decision trees with SAT. In *Proc. IJCAI*, pages 1362–1368. ijcai.org, 2018.

Bibliography II

Matthew Richardson and Pedro M. Domingos. Markov logic networks. *Mach. Learn.*, 62(1-2):107–136, 2006.

Paul Saikko, Jeremias Berg, and Matti Järvisalo. LMHS: A SAT-IP hybrid MaxSAT solver. In *Proc. SAT*, volume 9710 of *Lecture Notes in Computer Science*, pages 539–546. Springer, 2016. doi: 10.1007/978-3-319-40970-2_34. URL https://doi.org/10.1007/978-3-319-40970-2_34.

Xujie Si, Xin Zhang, Vasco M. Manquinho, Mikolás Janota, Alexey Ignatiev, and Mayur Naik. On incremental core-guided MaxSAT solving. In *Proc. CP*, volume 9892 of *Lecture Notes in Computer Science*, pages 473–482. Springer, 2016. doi: 10.1007/978-3-319-44953-1_30. URL https://doi.org/10.1007/978-3-319-44953-1_30.

Jinqiang Yu, Alexey Ignatiev, Peter J. Stuckey, and Pierre Le Bodic. Computing optimal decision sets with SAT. In *Proc. CP*, volume 12333 of *Lecture Notes in Computer Science*, pages 952–970. Springer, 2020.

Xin Zhang, Ravi Mangal, Aditya V. Nori, and Mayur Naik. Query-guided maximum satisfiability. In *POPL*, pages 109–122. ACM, 2016.