Abstract Cores in Implicit Hitting Set MaxSat Solving

Jeremias Berg[1] Fahiem Bacchus[2] Alex Poole[2]

[1]HIIT, Dept. Computer Science, University of Helsinki, Finland

[2]University of Toronto, Department of Computer Science, Canada

SAT 2020
Online

MaxSat

- A declarative optimisation paradigm based on propositional logic
  - i.e. Boolean literals and clauses
  - Builds on the success of SAT
  - Applications in:
    - data analysis
    - error localization in C code
    - haplotyping with pedigrees.
    - and others.

MaxSat

- A declarative optimisation paradigm based on propositional logic
  - i.e. Boolean literals and clauses
  - Builds on the success of SAT
  - Applications in:
    - data analysis
    - error localization in C code
    - haplotyping with pedigrees.
    - and others.

MaxSat

- A declarative optimisation paradigm based on propositional logic
  - i.e. Boolean literals and clauses
  - Builds on the success of SAT
  - Applications in:
    - data analysis
    - error localization in C code
    - haplotyping with pedigrees.
    - and others.

MaxSat

- A declarative optimisation paradigm based on propositional logic
  - i.e. Boolean literals and clauses
  - Builds on the success of SAT
  - Applications in:
    - data analysis
    - error localization in C code
    - haplotyping with pedigrees.
    - and others.

Ghosh and Meel [2019]
Chen et al. [2010]
Zhang and Bacchus [2012]
Berg and Järvisalo [2017]
Demirovic et al. [2019]
Hosokawa et al. [2019]
...

MaxSat

- A declarative optimisation paradigm based on propositional logic
  - i.e. Boolean literals and clauses
  - Builds on the success of SAT
  - Applications in:
    - data analysis
    - error localization in C code
    - haplotyping with pedigrees.
    - and others.
  - 17 new benchmark families from different application areas submitted to the past two MaxSAT Evaluations

Ghosh and Meel [2019]
Chen et al. [2010]
Zhang and Bacchus [2012]
Berg and Järvisalo [2017]
Demirovic et al. [2019]
Hosokawa et al. [2019]
...

Our Contributions

- IHS solvers central in real-world MaxSAT solving
  - Decouple MaxSAT into core-extraction (SAT-solving) and optimization (IP-solving)
  - Avoid increasing complexity of SAT calls,
  - needs to extract a large number of cores on some instances.
- *Abstract cores*: a compact way of representing an exponential number of ordinary cores.
- In this paper we:
  - ... incorporate abstract cores into IHS solvers,
  - ... prove correctness of the algorithm,
  - ... prove that abstract cores improve IHS solvers in theory and
  - ... show significant experimental improvements.

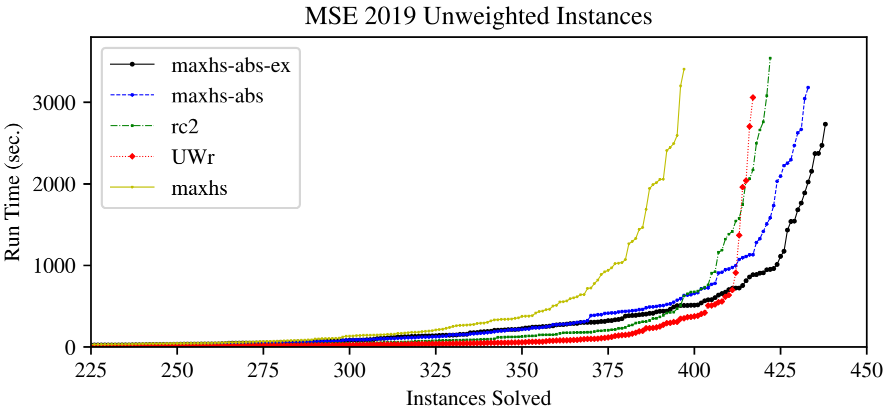
Our Contributions

- IHS solvers central in real-world MaxSAT solving
  - Decouple MaxSAT into core-extraction (SAT-solving) and optimization (IP-solving)
  - Avoid increasing complexity of SAT calls,
  - needs to extract a large number of cores on some instances.
- *Abstract cores*: a compact way of representing an exponential number of ordinary cores.
- In this paper we:
  - ... incorporate abstract cores into IHS solvers,
  - ... prove correctness of the algorithm,
  - ... prove that abstract cores improve IHS solvers in theory and
  - ... show significant experimental improvements.

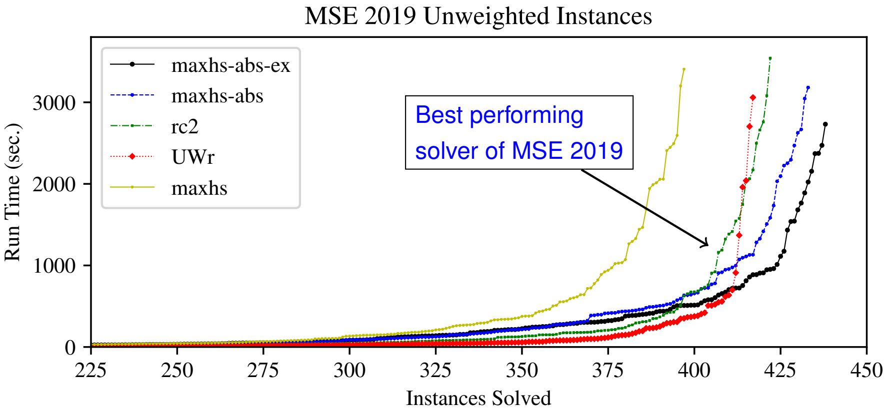
Our Contributions

- IHS solvers central in real-world MaxSAT solving
  - Decouple MaxSAT into core-extraction (SAT-solving) and optimization (IP-solving)
  - Avoid increasing complexity of SAT calls,
  - needs to extract a large number of cores on some instances.
- *Abstract cores*: a compact way of representing an exponential number of ordinary cores.
- In this paper we:
  - ... incorporate abstract cores into IHS solvers,
  - ... prove correctness of the algorithm,
  - ... prove that abstract cores improve IHS solvers in theory and
  - ... show significant experimental improvements.

Our Contributions



Our Contributions



Outline

- Preliminaries
- IHS based MaxSat solving
- Abstract cores
- IHS with abstract cores
- Empirical results.

MaxSat solving with Implicit Hitting Sets

Preliminaries: MaxSat

- Optimisation extension of Boolean Satisfiability (SAT)
- An instance consists of:
  - a set of hard clauses,
  - a set of soft clauses and
  - a weight function *wt* over the soft clauses.
- Find τ that:
  - satisfies all hard clauses and
  - maximises the number of satisfied soft clauses.
- Assume w.l.o.g. that all soft clauses are unit negative literals
  - *Blocking variables:* the variables in soft clauses.
  - $\mathcal{F}_B$ the set of blocking variables

Preliminaries: MaxSat

- Optimisation extension of Boolean Satisfiability (SAT)
- An instance consists of:
  - a set of hard clauses,
  - a set of soft clauses and
  - a weight function *wt* over the soft clauses.

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

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

$$w((\neg b_1)) = w((\neg b_2)) = 1, \quad w((\neg b_3)) = 1$$

- Find τ that:
  - satisfies all hard clauses and
  - maximises the number of satisfied soft clauses.
- Assume w.l.o.g. that all soft clauses are unit negative literals
  - *Blocking variables:* the variables in soft clauses.
  - $\mathcal{F}_B$ the set of blocking variables

Preliminaries: MaxSat

- Optimisation extension of Boolean Satisfiability (SAT)
- An instance consists of:
  - a set of hard clauses,
  - a set of soft clauses and
  - a weight function *wt* over the soft clauses.
  - weights can be any integers
- Find τ that:
  - satisfies all hard clauses and
  - maximises the number of satisfied soft clauses.
- Assume w.l.o.g. that all soft clauses are unit negative literals
  - *Blocking variables:* the variables in soft clauses.
  - $\mathcal{F}_B$ the set of blocking variables

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

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

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

Preliminaries: MaxSat

- Optimisation extension of Boolean Satisfiability (SAT)
- An instance consists of:
  - a set of hard clauses and
  - a set of soft clauses.
  - a weight function *wt* over the soft clauses.
- Find τ that:
  - satisfies all hard clauses and
  - maximises the number of satisfied soft clauses.
- Assume w.l.o.g. that all soft clauses are unit negative literals
  - *Blocking variables:* the variables in soft clauses.
  - $\mathcal{F}_B$ the set of blocking variables

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

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

Preliminaries: MaxSat

- Optimisation extension of Boolean Satisfiability (SAT)
- An instance consists of:
  - a set of hard clauses and
  - a set of soft clauses.
  - a weight function *wt* over the soft clauses.
- Find τ that:
  - satisfies all hard clauses and
  - maximises the number of satisfied soft clauses.
- Assume w.l.o.g. that all soft clauses are unit negative literals
  - *Blocking variables:* the variables in soft clauses.
  - $\mathcal{F}_B$ the set of blocking variables

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

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

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

$$cost(\tau) = \mathbf{1}$$

Preliminaries: MaxSat

- Optimisation extension of Boolean Satisfiability (SAT)
- An instance consists of:
  - a set of hard clauses and
  - a set of soft clauses.
  - a weight function *wt* over the soft clauses.
- Find τ that:
  - satisfies all hard clauses and
  - maximises the number of satisfied soft clauses.
- Assume w.l.o.g. that all soft clauses are unit negative literals
  - *Blocking variables:* the variables in soft clauses.
  - $\mathcal{F}_B$ the set of blocking variables

$$\tau = \{\neg b_1, b_2, \neg b_3\}$$

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

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

$$cost(\tau) = \mathbf{1}$$

Preliminaries: MaxSat

- Optimisation extension of Boolean Satisfiability (SAT)
- An instance consists of:
  - a set of hard clauses and
  - a set of soft clauses.
  - a weight function *wt* over the soft clauses.
- Find τ that:
  - satisfies all hard clauses and
  - maximises the number of satisfied soft clauses.
- Assume w.l.o.g. that all soft clauses are unit negative literals
  - *Blocking variables:* the variables in soft clauses.
  - $\mathcal{F}_B$ the set of blocking variables

$$\tau = \{\neg b_1, b_2, \neg b_3\}$$

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

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

$$cost(\tau) = \mathbf{1}$$

$$\mathcal{F}_B = \{b_1, b_2, b_3\}$$

Preliminaries: UNSAT Cores

- Central in modern MaxSAT solving:
- $\kappa \subset \mathcal{F}_S$ is an *core* if $\mathcal{F}_H \wedge \kappa$ is unsatisfiable
- κ **can be represented as a clause over $\mathcal{F}_B$ that is entailed by $\mathcal{F}_H$**
- or as a linear inequality

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

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

$$\mathcal{F}_B = \{b_1, b_2, b_3\}$$

Preliminaries: UNSAT Cores

- Central in modern MaxSAT solving:
- $\kappa \subset \mathcal{F}_S$ is an *core* if $\mathcal{F}_H \wedge \kappa$ is unsatisfiable
- **κ can be represented as a clause over $\mathcal{F}_B$ that is entailed by $\mathcal{F}_H$**
- or as a linear inequality

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

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

$$\mathcal{F}_B = \{b_1, b_2, b_3\}$$

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

Preliminaries: UNSAT Cores

- Central in modern MaxSAT solving:
- $\kappa \subset \mathcal{F}_S$ is an *core* if $\mathcal{F}_H \wedge \kappa$ is unsatisfiable
- **κ can be represented as a clause over $\mathcal{F}_B$ that is entailed by $\mathcal{F}_H$**
- or as a linear inequality

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

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

$$\mathcal{F}_B = \{b_1, b_2, b_3\}$$

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

$$\kappa = (b_1 \vee b_2)$$

Preliminaries: UNSAT Cores

- Central in modern MaxSAT solving:
- $\kappa \subset \mathcal{F}_S$ is an *core* if $\mathcal{F}_H \wedge \kappa$ is unsatisfiable
- **κ can be represented as a clause over $\mathcal{F}_B$ that is entailed by $\mathcal{F}_H$**
- or as a linear inequality

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

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

$$\mathcal{F}_B = \{b_1, b_2, b_3\}$$

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

$$\kappa = (b_1 \vee b_2)$$

$$\kappa = b_1 + b_2 \geq 1$$

Preliminaries: UNSAT Cores

- Central in modern MaxSAT solving:
- $\kappa \subset \mathcal{F}_S$ is an *core* if $\mathcal{F}_H \wedge \kappa$ is unsatisfiable
- κ **can be represented as a clause over** $\mathcal{F}_B$ **that is entailed by** $\mathcal{F}_H$
- or as a linear inequality

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

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

$$\mathcal{F}_B = \{b_1, b_2, b_3\}$$

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

$$\kappa = (b_1, b_2)$$

$$\kappa = b_1 + b_2 \geq 1$$

> In the rest of the presentation, we represent clauses$(b_1 \vee b_2)$ as (b_1, b_2)

Solving (unweighted) MaxSat with IHS

$\mathcal{F}_H = \{(b_1, b_2), (b_2, b_3), (b_3, b_4)\}$

$\mathcal{F}_S = \{(\neg b_1), (\neg b_2), (\neg b_3), (\neg b_4)\}$

$\mathcal{F}_B = \{b_1, b_2, b_3, b_4\}$

`Basic-IHS` $(\mathcal{F})$

Solving (unweighted) MaxSat with IHS

$\mathcal{F}_H = \{(b_1, b_2), (b_2, b_3), (b_3, b_4)\}$

$\mathcal{F}_S = \{(\neg b_1), (\neg b_2), (\neg b_3), (\neg b_4)\}$

$\mathcal{F}_B = \{b_1, b_2, b_3, b_4\}$

`Basic-IHS` $(\mathcal{F})$

Initialize

$UB = \infty$

$LB = 0$

$\text{CORES} = \emptyset$

$\text{BESTSOL} = \varnothing$

Solving (unweighted) MaxSat with IHS

$\mathcal{F}_H = \{(b_1, b_2), (b_2, b_3), (b_3, b_4)\}$

$\mathcal{F}_S = \{(\neg b_1), (\neg b_2), (\neg b_3), (\neg b_4)\}$

$\mathcal{F}_B = \{b_1, b_2, b_3, b_4\}$

`Basic-IHS` $(\mathcal{F})$

Initialize

while $LB < UB$

$UB = \infty$

$LB = 0$

CORES $= \emptyset$

BESTSOL $= \varnothing$

Solving (unweighted) MaxSat with IHS

$\mathcal{F}_H = \{(b_1, b_2), (b_2, b_3), (b_3, b_4)\}$

$\mathcal{F}_S = \{(\neg b_1), (\neg b_2), (\neg b_3), (\neg b_4)\}$

$\mathcal{F}_B = \{b_1, b_2, b_3, b_4\}$

$hs = \texttt{Min-Hs}\ (\mathcal{F}_B, \emptyset)$

$UB = \infty$

$LB = 0$

$\text{CORES} = \emptyset$

$\text{BESTSOL} = \varnothing$

`Basic-IHS` $(\mathcal{F})$

Initialize

while $LB < UB$

Compute min-cost hitting set *hs*

`Min-Hs` $(\mathcal{F}_B, \text{CORES})$:

minimize: $\sum_{b \in \mathcal{F}_B} b$

subject to: $\sum_{b \in \kappa} b \geq 1\ \forall \kappa \in \text{CORES}$

return: $\{b \mid b$ set to 1 in opt. soln$\}$

Solving (unweighted) MaxSat with IHS

$\mathcal{F}_H = \{(b_1, b_2), (b_2, b_3), (b_3, b_4)\}$

$\mathcal{F}_S = \{(\neg b_1), (\neg b_2), (\neg b_3), (\neg b_4)\}$

$\mathcal{F}_B = \{b_1, b_2, b_3, b_4\}$

$hs = \texttt{Min-Hs}\ (\mathcal{F}_B, \emptyset)$

$\texttt{Basic-IHS}\ (\mathcal{F})$

Initialize

while $LB < UB$

Compute min-cost hitting set hs

$UB = \infty$

$LB = 0$

$\text{CORES} = \emptyset$

$\text{BESTSOL} = \varnothing$

Weighted Case →

$\texttt{Min-Hs}\ (\mathcal{F}_B, \text{CORES})$:

minimize: $\sum_{b \in \mathcal{F}_B} \mathbf{wt(b)b}$

subject to: $\sum_{b \in \kappa} b \geq 1\ \forall \kappa \in \text{CORES}$

return: $\{b \mid b \text{ set to 1 in opt. soln}\}$

Solving (unweighted) MaxSat with IHS

$\mathcal{F}_H = \{(b_1, b_2), (b_2, b_3), (b_3, b_4)\}$

$\mathcal{F}_S = \{(\neg b_1), (\neg b_2), (\neg b_3), (\neg b_4)\}$

$\mathcal{F}_B = \{b_1, b_2, b_3, b_4\}$

$hs = \emptyset$

$UB = \infty$

$LB = |\emptyset|$

$\text{CORES} = \emptyset$

$\text{BESTSOL} = \varnothing$

`Basic-IHS` $(\mathcal{F})$

Initialize
while $LB < UB$
  Compute min-cost hitting set *hs*
  Update LB

`Min-Hs` $(\mathcal{F}_B, \text{CORES})$:
minimize: $\sum_{b \in \mathcal{F}_B} b$
subject to: $\sum_{b \in \kappa} b \geq 1 \ \forall \kappa \in \text{CORES}$
return: $\{b \mid b \text{ set to 1 in opt. soln}\}$

Solving (unweighted) MaxSat with IHS

$\mathcal{F}_H = \{(b_1, b_2), (b_2, b_3), (b_3, b_4)\}$

$\mathcal{F}_S = \{(\neg b_1), (\neg b_2), (\neg b_3), (\neg b_4)\}$

$\mathcal{F}_B = \{b_1, b_2, b_3, b_4\}$

$hs = \emptyset$

$\mathcal{A} = \mathcal{F}_B \setminus hs$

$UB = \infty$

$LB = 0$

$\text{CORES} = \emptyset$

$\text{BESTSOL} = \varnothing$

`Basic-IHS` $(\mathcal{F})$

Initialize
while $LB < UB$
 Compute min-cost hitting set *hs*
 Update LB
 Set up assumptions

`Min-Hs` $(\mathcal{F}_B, \text{CORES})$:
minimize: $\sum_{b \in \mathcal{F}_B} b$
subject to: $\sum_{b \in \kappa} b \geq 1 \ \forall \kappa \in \text{CORES}$
return: $\{b \mid b$ set to 1 in opt. soln$\}$

Solving (unweighted) MaxSat with IHS

$\mathcal{F}_H = \{(b_1, b_2), (b_2, b_3), (b_3, b_4)\}$

$\mathcal{F}_S = \{(\neg b_1), (\neg b_2), (\neg b_3), (\neg b_4)\}$

$\mathcal{F}_B = \{b_1, b_2, b_3, b_4\}$

$\texttt{sat-assume}(\mathcal{F}_H, \neg\mathcal{A})$

$\mathcal{A} = \{b_1, b_2, b_3, b_4\}$

$K = \{\}$

$\texttt{Basic-IHS}\ (\mathcal{F})$

Initialize

while $LB < UB$

Compute min-cost hitting set *hs*

Update LB

Set up assumptions

Extract cores until SAT

$UB = \infty$

$LB = 0$

$\text{CORES} = \emptyset$

$\text{BESTSOL} = \varnothing$

$\texttt{Min-Hs}\ (\mathcal{F}_B, \text{CORES})$:

minimize: $\sum_{b \in \mathcal{F}_B} b$

subject to: $\sum_{b \in \kappa} b \geq 1\ \forall \kappa \in \text{CORES}$

return: $\{b \mid b \text{ set to 1 in opt. soln}\}$

Solving (unweighted) MaxSat with IHS

$\mathcal{F}_H = \{(b_1, b_2), (b_2, b_3), (b_3, b_4)\}$

$\mathcal{F}_S = \{(\neg b_1), (\neg b_2), (\neg b_3), (\neg b_4)\}$

$\mathcal{F}_B = \{b_1, b_2, b_3, b_4\}$

$\texttt{sat-assume}(\mathcal{F}_H, \neg\mathcal{A})$

$\mathcal{A} = \{\not{b}_1, \not{b}_2, b_3, b_4\}$

$K = \{(b_1, b_2)\}$

$UB = \infty$

$LB = 0$

$\text{CORES} = \emptyset$

$\text{BESTSOL} = \varnothing$

`Basic-IHS` $(\mathcal{F})$

Initialize

while $LB < UB$

Compute min-cost hitting set *hs*

Update LB

Set up assumptions

Extract cores until SAT

`Min-Hs` $(\mathcal{F}_B, \text{CORES})$:

minimize: $\sum_{b \in \mathcal{F}_B} b$

subject to: $\sum_{b \in \kappa} b \geq 1 \ \forall \kappa \in \text{CORES}$

return: $\{b \mid b \text{ set to 1 in opt. soln}\}$

Solving (unweighted) MaxSat with IHS

$\mathcal{F}_H = \{(b_1, b_2), (b_2, b_3), (b_3, b_4)\}$

$\mathcal{F}_S = \{(\neg b_1), (\neg b_2), (\neg b_3), (\neg b_4)\}$

$\mathcal{F}_B = \{b_1, b_2, b_3, b_4\}$

$\texttt{sat-assume}(\mathcal{F}_H, \neg\mathcal{A})$

$\mathcal{A} = \{b_3, b_4\}$

$K = \{(b_1, b_2)\}$

$UB = \infty$

$LB = 0$

$\text{CORES} = \emptyset$

$\text{BESTSOL} = \varnothing$

`Basic-IHS` $(\mathcal{F})$

Initialize

while $LB < UB$

Compute min-cost hitting set *hs*

Update LB

Set up assumptions

Extract cores until SAT

`Min-Hs` $(\mathcal{F}_B, \text{CORES})$:

minimize: $\sum_{b \in \mathcal{F}_B} b$

subject to: $\sum_{b \in \kappa} b \geq 1 \ \forall \kappa \in \text{CORES}$

return: $\{b \mid b \text{ set to 1 in opt. soln}\}$

Solving (unweighted) MaxSat with IHS

$\mathcal{F}_H = \{(b_1, b_2), (b_2, b_3), (b_3, b_4)\}$

$\mathcal{F}_S = \{(\neg b_1), (\neg b_2), (\neg b_3), (\neg b_4)\}$

$\mathcal{F}_B = \{b_1, b_2, b_3, b_4\}$

$\texttt{sat-assume}(\mathcal{F}_H, \neg\mathcal{A})$

$\mathcal{A} = \{\not{b}_3, \not{b}_4\}$

$K = \{(b_1, b_2), (b_3, b_4)\}$

$UB = \infty$

$LB = 0$

$\text{CORES} = \emptyset$

$\text{BESTSOL} = \varnothing$

$\texttt{Basic-IHS}\,(\mathcal{F})$

Initialize

while $LB < UB$

Compute min-cost hitting set *hs*

Update LB

Set up assumptions

Extract cores until SAT

$\texttt{Min-Hs}\,(\mathcal{F}_B, \text{CORES})$:

minimize: $\sum_{b \in \mathcal{F}_B} b$

subject to: $\sum_{b \in \kappa} b \geq 1 \;\forall \kappa \in \text{CORES}$

return: $\{b \mid b \text{ set to 1 in opt. soln}\}$

Solving (unweighted) MaxSat with IHS

$\mathcal{F}_H = \{(b_1, b_2), (b_2, b_3), (b_3, b_4)\}$

$\mathcal{F}_S = \{(\neg b_1), (\neg b_2), (\neg b_3), (\neg b_4)\}$

$\mathcal{F}_B = \{b_1, b_2, b_3, b_4\}$

$\texttt{sat-assume}(\mathcal{F}_H, \neg\mathcal{A})$

$\mathcal{A} = \{\}$

$K = \{(b_1, b_2), (b_3, b_4)\}$

`Basic-IHS` $(\mathcal{F})$

Initialize
while $LB < UB$
  Compute min-cost hitting set *hs*
  Update LB
  Set up assumptions
  Extract cores until SAT

$UB = \infty$

$LB = 0$

$\text{CORES} = \emptyset$

$\text{BESTSOL} = \varnothing$

`Min-Hs` $(\mathcal{F}_B, \text{CORES})$:
minimize: $\sum_{b \in \mathcal{F}_B} b$
subject to: $\sum_{b \in \kappa} b \geq 1 \ \forall \kappa \in \text{CORES}$
return: $\{b \mid b \text{ set to 1 in opt. soln}\}$

Solving (unweighted) MaxSat with IHS

$\mathcal{F}_H = \{(b_1, b_2), (b_2, b_3), (b_3, b_4)\}$

$\mathcal{F}_S = \{(\neg b_1), (\neg b_2), (\neg b_3), (\neg b_4)\}$

$\mathcal{F}_B = \{b_1, b_2, b_3, b_4\}$

$\texttt{sat-assume}(\mathcal{F}_H, \neg\mathcal{A})$

$\mathcal{A} = \{\}$

$K = \{(b_1, b_2), (b_3, b_4)\}$

$\tau = \{\neg b_1, b_2, \neg b_3, b_4\}$

$UB = \infty$

$LB = 0$

$\text{CORES} = \emptyset$

$\text{BESTSOL} = \varnothing$

`Basic-IHS` $(\mathcal{F})$

Initialize
while $LB < UB$
  Compute min-cost hitting set *hs*
  Update LB
  Set up assumptions
  Extract cores until SAT

`Min-Hs` $(\mathcal{F}_B, \text{CORES})$:
minimize: $\sum_{b \in \mathcal{F}_B} b$
subject to: $\sum_{b \in \kappa} b \geq 1 \ \forall \kappa \in \text{CORES}$
return: $\{b \mid b \text{ set to 1 in opt. soln}\}$

Solving (unweighted) MaxSat with IHS

$\mathcal{F}_H = \{(b_1, b_2), (b_2, b_3), (b_3, b_4)\}$

$\mathcal{F}_S = \{(\neg b_1), (\neg b_2), (\neg b_3), (\neg b_4)\}$

$\mathcal{F}_B = \{b_1, b_2, b_3, b_4\}$

$K = \{(b_1, b_2), (b_3, b_4)\}$

$\tau = \{\neg b_1, b_2, \neg b_3, b_4\}$

UB $= cost(\tau)$

$LB = 0$

CORES $= \emptyset$

BESTSOL $= \{\neg b_1, b_2, \neg b_3, b_4\}$

`Basic-IHS` $(\mathcal{F})$

Initialize
while $LB < UB$
- Compute min-cost hitting set *hs*
- Update LB
- Set up assumptions
- Extract cores until SAT
- Update UB

`Min-Hs` $(\mathcal{F}_B, \text{CORES})$:
minimize: $\sum_{b \in \mathcal{F}_B} b$
subject to: $\sum_{b \in \kappa} b \geq 1 \ \forall \kappa \in \text{CORES}$
return: $\{b \mid b \text{ set to 1 in opt. soln}\}$

Solving (unweighted) MaxSat with IHS

$\mathcal{F}_H = \{(b_1, b_2), (b_2, b_3), (b_3, b_4)\}$

$\mathcal{F}_S = \{(\neg b_1), (\neg b_2), (\neg b_3), (\neg b_4)\}$

$\mathcal{F}_B = \{b_1, b_2, b_3, b_4\}$

$K = \{(b_1, b_2), (b_3, b_4)\}$

$\tau = \{\neg b_1, b_2, \neg b_3, b_4\}$

$UB = 2$

$LB = 0$

$\text{CORES} = \{(b_1, b_2), (b_3, b_4)\}$

$\text{BESTSOL} = \{\neg b_1, b_2, \neg b_3, b_4\}$

`Basic-IHS` $(\mathcal{F})$

Initialize
while $LB < UB$
- Compute min-cost hitting set *hs*
- Update LB
- Set up assumptions
- Extract cores until SAT
- Update UB
- Add cores to CORES

`Min-Hs` $(\mathcal{F}_B, \text{CORES})$:
minimize: $\sum_{b \in \mathcal{F}_B} b$
subject to: $\sum_{b \in \kappa} b \geq 1 \ \forall \kappa \in \text{CORES}$
return: $\{b \mid b \text{ set to 1 in opt. soln}\}$

Solving (unweighted) MaxSat with IHS

$\mathcal{F}_H = \{(b_1, b_2), (b_2, b_3), (b_3, b_4)\}$

$\mathcal{F}_S = \{(\neg b_1), (\neg b_2), (\neg b_3), (\neg b_4)\}$

$\mathcal{F}_B = \{b_1, b_2, b_3, b_4\}$

`Basic-IHS` $(\mathcal{F})$

Initialize
while *LB* $<$ *UB*
  Compute min-cost hitting set *hs*
  Update LB
  Set up assumptions
  Extract cores until SAT
  Update UB
  Add cores to CORES

$UB = 2$

$LB = 0$

CORES $= \{(b_1, b_2), (b_3, b_4)\}$

BESTSOL $= \{\neg b_1, b_2, \neg b_3, b_4\}$

Solving (unweighted) MaxSat with IHS

$\mathcal{F}_H = \{(b_1, b_2), (b_2, b_3), (b_3, b_4)\}$

$\mathcal{F}_S = \{(\neg b_1), (\neg b_2), (\neg b_3), (\neg b_4)\}$

$\mathcal{F}_B = \{b_1, b_2, b_3, b_4\}$

$hs = \texttt{Min-Hs}\ (\mathcal{F}_B, \{(b_1, b_2), (b_3, b_4)\})$

$\texttt{Basic-IHS}\ (\mathcal{F})$

Initialize
while $LB < UB$
  Compute min-cost hitting set *hs*
  Update LB
  Set up assumptions
  Extract cores until SAT
  Update UB
  Add cores to CORES

$UB = 2$

$LB = 0$

$\text{CORES} = \{(b_1, b_2), (b_3, b_4)\}$

$\text{BESTSOL} = \{\neg b_1, b_2, \neg b_3, b_4\}$

$\texttt{Min-Hs}\ (\mathcal{F}_B, \text{CORES})$:
minimize: $\sum_{b \in \mathcal{F}_B} b$
subject to: $\sum_{b \in \kappa} b \geq 1\ \forall \kappa \in \text{CORES}$
return: $\{b \mid b \text{ set to 1 in opt. soln}\}$

Solving (unweighted) MaxSat with IHS

$\mathcal{F}_H = \{(b_1, b_2), (b_2, b_3), (b_3, b_4)\}$

$\mathcal{F}_S = \{(\neg b_1), (\neg b_2), (\neg b_3), (\neg b_4)\}$

$\mathcal{F}_B = \{b_1, b_2, b_3, b_4\}$

$hs = \{b_1, b_4\}$

`Basic-IHS` $(\mathcal{F})$

Initialize
while $LB < UB$
  Compute min-cost hitting set *hs*
  Update LB
  Set up assumptions
  Extract cores until SAT
  Update UB
  Add cores to CORES

$UB = 2$

$LB = |\{b_1, b_4\}|$

$\text{CORES} = \{(b_1, b_2), (b_3, b_4)\}$

$\text{BESTSOL} = \{\neg b_1, b_2, \neg b_3, b_4\}$

Solving (unweighted) MaxSat with IHS

$\mathcal{F}_H = \{(b_1, b_2), (b_2, b_3), (b_3, b_4)\}$

$\mathcal{F}_S = \{(\neg b_1), (\neg b_2), (\neg b_3), (\neg b_4)\}$

$\mathcal{F}_B = \{b_1, b_2, b_3, b_4\}$

$hs = \{b_1, b_4\}$

$UB = 2$

$LB = 2$

$\textsc{cores} = \{(b_1, b_2), (b_3, b_4)\}$

$\textsc{bestsol} = \{\neg b_1, b_2, \neg b_3, b_4\}$

`Basic-IHS` $(\mathcal{F})$

Initialize
while $LB < UB$
  Compute min-cost hitting set *hs*
  Update LB
  Set up assumptions
  Extract cores until SAT
  Update UB
  Add cores to CORES
return BESTSOL

Solving (unweighted) MaxSat with IHS

$\mathcal{F}_H = \{(b_1, b_2), (b_2, b_3), (b_3, b_4)\}$

$\mathcal{F}_S = \{(\neg b_1), (\neg b_2), (\neg b_3), (\neg b_4)\}$

$\mathcal{F}_B = \{b_1, b_2, b_3, b_4\}$

$hs = \{b_1, b_4\}$

`Basic-IHS` $(\mathcal{F})$

Initialize
while $LB < UB$
  Compute min-cost hitting set *hs*
  Update LB
  Set up assumptions
  Extract cores until SAT
  Update UB
  Add cores to CORES
return BESTSOL

$UB = 2$

$LB = 2$ ← LB need to be increased to optimum before termination

$\text{CORES} = \{(b_1, b_2), (b_3, b_4)\}$

$\text{BESTSOL} = \{\neg b_1, b_2, \neg b_3, b_4\}$

Abstract Cores

Drawbacks of IHS Davies [2013]

Main motivation for our work:
There exists MaxSAT instances on which IHS needs an exponential number of cores.

Drawbacks of IHS Davies [2013]

Main motivation for our work:
There exists MaxSAT instances on which IHS needs an exponential number of cores.

$$\mathcal{F}_H = \left\{ \text{CNF}(\sum_{i=1}^{n} b_i \geq r) \right\}$$

$$\mathcal{F}_S = \{(\neg b_1), \ldots, (\neg b_n)\}$$

$$\mathcal{F}_B = \{b_1, \ldots, b_n\}$$

Drawbacks of IHS Davies [2013]

Main motivation for our work:
There exists MaxSAT instances on which IHS needs an exponential number of cores.

$$\mathcal{F}_H = \left\{ \text{CNF}(\sum_{i=1}^{n} b_i \geq r) \right\}$$

$$\mathcal{F}_S = \{(\neg b_1), \ldots, (\neg b_n)\}$$

$$\mathcal{F}_B = \{b_1, \ldots, b_n\}$$

$\mathbf{n = 8}, \quad \mathbf{r = 4}$

$\kappa_1 = (b_{i_1}, b_{i_2}, b_{i_3}, b_{i_4}, b_{i_5})$

is a core for any i_1, i_2, i_3, i_4, i_5

Intuition:

Any $\kappa \subset \mathcal{F}_S$ s.t $|\kappa| = (n - r) + 1$ is a core.

Drawbacks of IHS Davies [2013]

Main motivation for our work:
There exists MaxSAT instances on which IHS needs an exponential number of cores.

$$\mathcal{F}_H = \left\{ \text{CNF}(\sum_{i=1}^{n} b_i \geq r) \right\}$$

$$\mathcal{F}_S = \{(\neg b_1), \ldots, (\neg b_n)\}$$

$$\mathcal{F}_B = \{b_1, \ldots, b_n\}$$

$\mathbf{n = 8}, \quad \mathbf{r = 4}$

$\kappa_1 = (b_{i_1}, b_{i_2}, b_{i_3}, b_{i_4}, b_{i_5})$

is a core for any i_1, i_2, i_3, i_4, i_5

$\kappa_2 = (b_{i_1}, b_{i_2}, b_{i_3}, b_{i_4})$

is **not** a core for any i_1, i_2, i_3, i_4

Intuition:

Any $\kappa \subset \mathcal{F}_S$ s.t $|\kappa| = (n - r) + 1$ is a core.

Drawbacks of IHS Davies [2013]

Main motivation for our work:
There exists MaxSAT instances on which IHS needs an exponential number of cores.

$$\mathcal{F}_H = \left\{ \text{CNF}(\sum_{i=1}^{n} b_i \geq r) \right\}$$

$$\mathcal{F}_S = \{(\neg b_1), \ldots, (\neg b_n)\}$$

$$\mathcal{F}_B = \{b_1, \ldots, b_n\}$$

$\mathbf{n = 8}, \quad \mathbf{r = 4}$

$\kappa_1 = (b_{i_1}, b_{i_2}, b_{i_3}, b_{i_4}, b_{i_5})$

is a core for any i_1, i_2, i_3, i_4, i_5

$\kappa_2 = (b_{i_1}, b_{i_2}, b_{i_3}, b_{i_4})$

is **not** a core for any i_1, i_2, i_3, i_4

Intuition:

Any $\kappa \subset \mathcal{F}_S$ s.t $|\kappa| = (n - r) + 1$ is a core.

IHS needs to extract all $\binom{n}{(n-r)+1}$

of them.

Drawbacks of IHS Davies [2013]

Main motivation for our work:
There exists MaxSAT instances on which IHS needs an exponential number of cores.

$$\mathcal{F}_H = \left\{ \text{CNF}(\sum_{i=1}^{n} b_i \geq r) \right\}$$

$$\mathcal{F}_S = \{(\neg b_1), \ldots, (\neg b_n)\}$$

$$\mathcal{F}_B = \{b_1, \ldots, b_n\}$$

Blocking variables are **exchangeable**:
cores are defined by the number of them,
not the identity of them

$\mathbf{n = 8}, \quad \mathbf{r = 4}$

$\kappa_1 = (b_{i_1}, b_{i_2}, b_{i_3}, b_{i_4}, b_{i_5})$

is a core for any i_1, i_2, i_3, i_4, i_5

$\kappa_2 = (b_{i_1}, b_{i_2}, b_{i_3}, b_{i_4})$

is **not** a core for any i_1, i_2, i_3, i_4

Intuition:

Any $\kappa \subset \mathcal{F}_S$ s.t $|\kappa| = (n - r) + 1$ is a core.
IHS needs to extract all $\binom{n}{(n-r)+1}$
of them.

Our Focus

Research Question(s)

Does there exists a compact representation of large sets of cores that IHS can reason over?

Abstract cores

Idea: What happens if we introduce literals that count the number of blocking variables set to true?

(similar to variables that have been successfully used in core-guided solvers)

Abstract cores

Idea: What happens if we introduce literals that count the number of blocking variables set to true?

(similar to variables that have been successfully used in core-guided solvers)

$$AB = \{b_1, \ldots, b_5\} \subset \mathcal{F}_B$$

$$\mathcal{F}_H = \left\{ \text{CNF}(\sum_{i=1}^{n} b_i \geq r) \right\}$$

$$\mathcal{F}_S = \{(\neg b_1), \ldots, (\neg b_n)\}$$

$$\mathcal{F}_B = \{b_1, \ldots, b_n\}$$

Abstract cores

Idea: What happens if we introduce literals that count the number of blocking variables set to true?

(similar to variables that have been successfully used in core-guided solvers)

$$\mathcal{F}_H = \left\{ \text{CNF}(\sum_{i=1}^{n} b_i \geq r) \right\}$$

$$\mathcal{F}_S = \{(\neg b_1), \ldots, (\neg b_n)\}$$

$$\mathcal{F}_B = \{b_1, \ldots, b_n\}$$

$$s^{AB}[i] \leftrightarrow \left(\sum_{b \in AB} b \geq i\right)$$

$$AB = \{b_1, \ldots, b_5\} \subset \mathcal{F}_B$$

Define $s^{AB}[i]$

Abstract cores

Idea: What happens if we introduce literals that count the number of blocking variables set to true?

(similar to variables that have been successfully used in core-guided solvers)

$$\mathcal{F}_H = \left\{ \text{CNF}(\sum_{i=1}^{n} b_i \geq r) \right\}$$

$$\mathcal{F}_S = \{(\neg b_1), \ldots, (\neg b_n)\}$$

$$\mathcal{F}_B = \{b_1, \ldots, b_n\}$$

$$s^{AB}[i] \leftrightarrow \left(\sum_{b \in AB} b \geq i\right)$$

↑

Note: Can be encoded as CNF

$AB = \{b_1, \ldots, b_5\} \subset \mathcal{F}_B$

Define $s^{AB}[i]$

Abstract cores

Idea: What happens if we introduce literals that count the number of blocking variables set to true?

(similar to variables that have been successfully used in core-guided solvers)

$$\mathcal{F}_H = \left\{ \text{CNF}(\sum_{i=1}^{n} b_i \geq r) \right\}$$

$$\mathcal{F}_S = \{(\neg b_1), \ldots, (\neg b_n)\}$$

$$\mathcal{F}_B = \{b_1, \ldots, b_n\}$$

$$s^{AB}[i] \leftrightarrow (\textstyle\sum_{b \in AB} b \geq i)$$

$AB = \{b_1, \ldots, b_5\} \subset \mathcal{F}_B$

Define $s^{AB}[i]$

Consider: $(b_7, s^{AB}[3], b_n)$

Abstract cores

Idea: What happens if we introduce literals that count the number of blocking variables set to true?

(similar to variables that have been successfully used in core-guided solvers)

$$\mathcal{F}_H = \left\{ \text{CNF}(\sum_{i=1}^{n} b_i \geq r) \right\}$$

$$\mathcal{F}_S = \{(\neg b_1), \ldots, (\neg b_n)\}$$

$$\mathcal{F}_B = \{b_1, \ldots, b_n\}$$

$$s^{AB}[i] \leftrightarrow \left(\sum_{b \in AB} b \geq i\right)$$

$AB = \{b_1, \ldots, b_5\} \subset \mathcal{F}_B$

Define $s^{AB}[i]$

Consider: $(b_7, s^{AB}[3], b_n)$

$(b_7, b_1, b_2, b_3, b_n)$

Abstract cores

Idea: What happens if we introduce literals that count the number of blocking variables set to true?

(similar to variables that have been successfully used in core-guided solvers)

$$\mathcal{F}_H = \left\{ \text{CNF}(\sum_{i=1}^{n} b_i \geq r) \right\}$$

$$\mathcal{F}_S = \{(\neg b_1), \ldots, (\neg b_n)\}$$

$$\mathcal{F}_B = \{b_1, \ldots, b_n\}$$

$$s^{AB}[i] \leftrightarrow (\textstyle\sum_{b \in AB} b \geq i)$$

$AB = \{b_1, \ldots, b_5\} \subset \mathcal{F}_B$

Define $s^{AB}[i]$

Consider: $(b_7, s^{AB}[3], b_n)$

$(b_7, b_1, b_2, b_3, b_n)$

$(b_7, b_3, b_4, b_2, b_n)$

Abstract cores

Idea: What happens if we introduce literals that count the number of blocking variables set to true?

(similar to variables that have been successfully used in core-guided solvers)

$$\mathcal{F}_H = \left\{ \text{CNF}(\sum_{i=1}^{n} b_i \geq r) \right\}$$

$$\mathcal{F}_S = \{(\neg b_1), \ldots, (\neg b_n)\}$$

$$\mathcal{F}_B = \{b_1, \ldots, b_n\}$$

$$s^{AB}[i] \leftrightarrow \left(\sum_{b \in AB} b \geq i\right)$$

$AB = \{b_1, \ldots, b_5\} \subset \mathcal{F}_B$

Define $s^{AB}[i]$

Consider: $(b_7, s^{AB}[3], b_n)$

$(b_7, b_1, b_2, b_3, b_n)$

$(b_7, b_4, b_2, b_1, b_n)$

$(b_7, b_3, b_4, b_2, b_n)$

Abstract cores

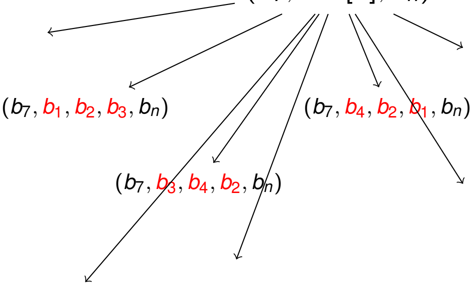
Idea: What happens if we introduce literals that count the number of blocking variables set to true?

(similar to variables that have been successfully used in core-guided solvers)

$AB = \{b_1, \ldots, b_5\} \subset \mathcal{F}_B$

Define $s^{AB}[i]$

Consider: $(b_7, s^{AB}[3], b_n)$



$$\mathcal{F}_H = \left\{ \text{CNF}\left(\sum_{i=1}^{n} b_i \geq r\right) \right\}$$

$$\mathcal{F}_S = \{(\neg b_1), \ldots, (\neg b_n)\}$$

$$\mathcal{F}_B = \{b_1, \ldots, b_n\}$$

$$s^{AB}[i] \leftrightarrow \left(\sum_{b \in AB} b \geq i\right)$$

Abstract cores

Idea: What happens if we introduce literals that count the number of blocking variables set to true?

(similar to variables that have been successfully used in core-guided solvers)

Terminology:
AB is an *abstraction set*

$s^{AB}[i]$ is an *abstraction variable*

The *definition* of $s^{AB}[i]$ is
$s^{AB}[i] \leftrightarrow (\sum_{b \in AB} b \geq i)$

$AB = \{b_1, \ldots, b_5\} \subset \mathcal{F}_B$

Define $s^{AB}[i]$

Consider: $(b_7, s^{AB}[3], b_n)$

Abstract cores

Idea: What happens if we introduce literals that count the number of blocking variables set to true?

(similar to variables that have been successfully used in core-guided solvers)

Terminology:
AB is an *abstraction set*

$s^{AB}[i]$ is an *abstraction variable*

The *definition* of $s^{AB}[i]$ is
$s^{AB}[i] \leftrightarrow \left(\sum_{b \in AB} b \geq i\right)$

$AB = \{b_1, \ldots, b_5\} \subset \mathcal{F}_B$

Define $s^{AB}[i]$

Consider: $(b_7, s^{AB}[3], b_n)$

Abstract Core:
a clause over abstraction and blocking variables that is entailed by $\mathcal{F}_H$ and the definitions of abstraction variables

Abstract cores

Idea: What happens if we introduce literals that count the number of blocking variables set to true?

(similar to variables that have been successfully used in core-guided solvers)

Terminology:
AB is an *abstraction set*

$s^{AB}[i]$ is an *abstraction variable*

The *definition* of $s^{AB}[i]$ is
$s^{AB}[i] \leftrightarrow \left(\sum_{b \in AB} b \geq i\right)$

$AB = \{b_1, \ldots, b_5\} \subset \mathcal{F}_B$

Define $s^{AB}[i]$

Could use other definitions
Summations successfull
in core-guided solvers.

Abstract Core:
a clause over abstraction and blocking variables that is entailed by $\mathcal{F}_H$ and the definitions of abstraction variables

Abstract cores are expressive

Proposition

An abstract core containing the abstraction variables $\{s^{AB^1}[j_1], \ldots, s^{AB^k}[j_k]\}$ is equivalent to the conjunction of

$$\prod_{i=1}^{k} \binom{|AB^i|}{|AB^i| - j_i + 1}$$

regular cores.

Abstract cores are expressive

Proposition

An abstract core containing the abstraction variables $\{s^{AB^1}[j_1], \ldots, s^{AB^k}[j_k]\}$ is equivalent to the conjunction of

$$\prod_{i=1}^{k} \binom{|AB^i|}{|AB^i| - j_i + 1}$$

regular cores.

Two Questions remain:

1. How to compute abstraction sets?
2. How to extract and reason over abstract cores in IHS?

Computing Abstraction Sets

Ideally

Identify a set $S \subset \mathcal{F}_B$ of exchangeable blocking variables.

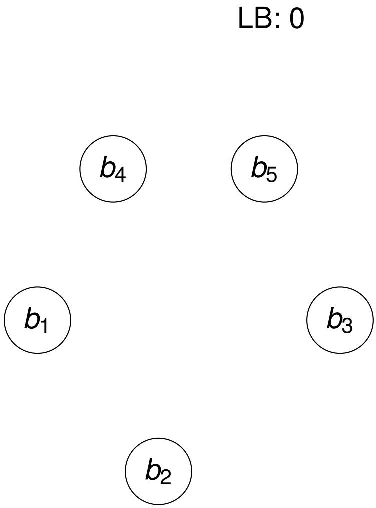
Computing Abstraction Sets

Ideally

Identify a set $S \subset \mathcal{F}_B$ of exchangeable blocking variables.

In practice

Form abstraction sets over blocking variables that appear frequently in cores together.



Computing Abstraction Sets

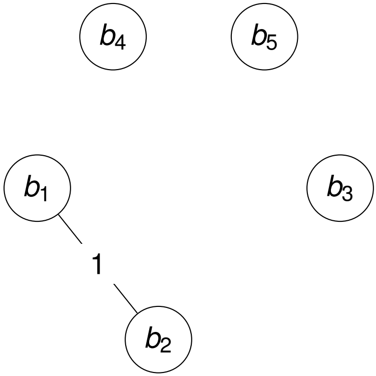
Ideally

Identify a set $S \subset \mathcal{F}_B$ of exchangeable blocking variables.

In practice

Form abstraction sets over blocking variables that appear frequently in cores together.

Core: (b_1, b_2) LB: 0



Computing Abstraction Sets

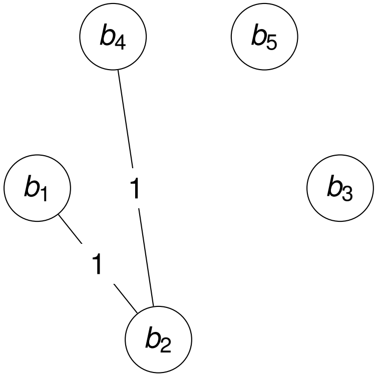
Core: (b_2, b_4) LB: 0

Ideally

Identify a set $S \subset \mathcal{F}_B$ of exchangeable blocking variables.

In practice

Form abstraction sets over blocking variables that appear frequently in cores together.



Computing Abstraction Sets

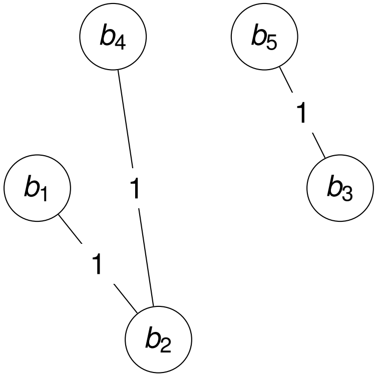
Core: (b_3, b_5) LB: 0

Ideally

Identify a set $S \subset \mathcal{F}_B$ of exchangeable blocking variables.

In practice

Form abstraction sets over blocking variables that appear frequently in cores together.



Computing Abstraction Sets

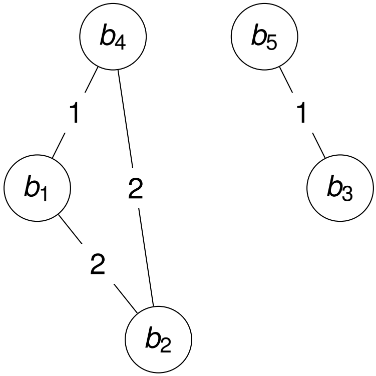
Core: (b_1, b_2, b_4) LB: 0

Ideally

Identify a set $S \subset \mathcal{F}_B$ of exchangeable blocking variables.

In practice

Form abstraction sets over blocking variables that appear frequently in cores together.



Computing Abstraction Sets

Ideally

Identify a set $S \subset \mathcal{F}_B$ of exchangeable blocking variables.

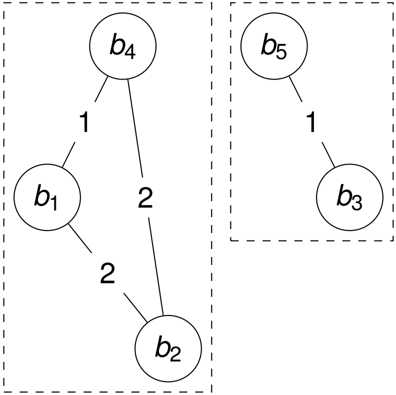
In practice

Form abstraction sets over blocking variables that appear frequently in cores together.

Recall: IHS needs to increase LB to optimum

Clustering

LB: 0



Computing Abstraction Sets

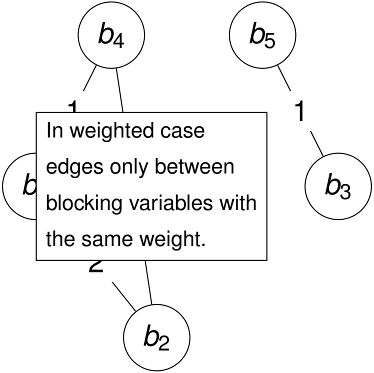
Ideally

Identify a set $S \subset \mathcal{F}_B$ of exchangeable blocking variables.

In practice

Form abstraction sets over blocking variables that appear frequently in cores together.

Recall: IHS needs to increase LB to optimum



IHS with abstract core reasoning

$\mathcal{F}_H = \{(b_1, b_2), (b_2, b_3), (b_3, b_4)\}$

`Abstract-IHS` $(\mathcal{F})$

$\mathcal{F}_B = \{b_1, b_2, b_3, b_4\}$

IHS with abstract core reasoning

$\mathcal{F}_H = \{(b_1, b_2), (b_2, b_3), (b_3, b_4)\}$

$\mathcal{F}_B = \{b_1, b_2, b_3, b_4\}$

`Abstract-IHS` $(\mathcal{F})$

Initialize

$UB = \infty$ $LB = 0$

$\mathcal{AB} = \emptyset$

$\text{CORES} = \emptyset$

$\text{BESTSOL} = \varnothing$

IHS with abstract core reasoning

$\mathcal{F}_H = \{(b_1, b_2), (b_2, b_3), (b_3, b_4)\}$

$\mathcal{F}_B = \{b_1, b_2, b_3, b_4\}$

`Abstract-IHS` $(\mathcal{F})$

Initialize

while $LB < UB$

$UB = \infty$ $LB = 0$

$\mathcal{AB} = \emptyset$

$\text{CORES} = \emptyset$

$\text{BESTSOL} = \varnothing$

IHS with abstract core reasoning

$\mathcal{F}_H = \{(b_1, b_2), (b_2, b_3), (b_3, b_4),$
$\bigwedge_{i=1}^{2} \text{CNF}\left((b_2 + b_3 \geq i) \rightarrow s^{AB}[i]\right)\}$

$\mathcal{F}_B = \{b_1, b_2, b_3, b_4\}$

`Abstract-IHS` $(\mathcal{F})$

Initialize
while $LB < UB$
  Update $\mathcal{AB}$

$UB = \infty$ $LB = 0$

$\mathcal{AB} = \{AB = \{b_2, b_3\}\}$

$\text{CORES} = \emptyset$

$\text{BESTSOL} = \varnothing$

IHS with abstract core reasoning

$\mathcal{F}_H = \{(b_1, b_2), (b_2, b_3), (b_3, b_4),$
$\bigwedge_{i=1}^{2} \text{CNF}\left((b_2 + b_3 \geq i) \rightarrow s^{AB}[i]\right)\}$

$\mathcal{F}_B = \{b_1, b_2, b_3, b_4\}$

$hs = \texttt{Min-Abs}\ (\mathcal{F}_B, \emptyset, \mathcal{AB})$

$\texttt{Abstract-IHS}\ (\mathcal{F})$

Initialize
while $LB < UB$
  Update $\mathcal{AB}$
  Compute min-cost hitting set *hs*

$\sum_{b \in AB} b - k \cdot s^{AB}[k] \geq 0$

$\sum_{b \in AB} b - |AB| \cdot s^{AB}[k] < k$

$\texttt{Min-Abs}\ (\mathcal{F}_B, \text{CORES}, \mathcal{AB})$:
minimize: $\sum_{b \in \mathcal{F}_B} b$
subject to: $\sum_{b \in \kappa} b \geq 1\ \forall \kappa \in \text{CORES}$
$\left(\sum_{\mathbf{b} \in \mathbf{AB}} \mathbf{b} \geq \mathbf{k}\right) \leftrightarrow \mathbf{s}^{\mathbf{AB}}[\mathbf{k}]\ \forall \mathbf{AB} \in \mathcal{AB}$
return: $\{b \mid b \text{ set to 1 in opt. soln}\}$

IHS with abstract core reasoning

$\mathcal{F}_H = \{(b_1, b_2), (b_2, b_3), (b_3, b_4),$
$\bigwedge_{i=1}^{2} \text{CNF}\left((b_2 + b_3 \geq i) \to s^{AB}[i]\right)\}$

$\mathcal{F}_B = \{b_1, b_2, b_3, b_4\}$

$hs = \emptyset$

$UB = \infty$ $LB = |\emptyset|$

$\mathcal{AB} = \{AB = \{b_2, b_3\}\}$

$\text{CORES} = \emptyset$

$\text{BESTSOL} = \varnothing$

`Abstract-IHS` $(\mathcal{F})$

Initialize
while $LB < UB$
  Update $\mathcal{AB}$
  Compute min-cost hitting set hs
  Update LB

`Min-Abs` $(\mathcal{F}_B, \text{CORES}, \mathcal{AB})$:
minimize: $\sum_{b \in \mathcal{F}_B} b$
subject to: $\sum_{b \in \kappa} b \geq 1 \;\forall \kappa \in \text{CORES}$
$\left(\sum_{b \in AB} b \geq k\right) \leftrightarrow s^{AB}[k] \;\forall AB \in \mathcal{AB}$
return: $\{b \mid b \text{ set to 1 in opt. soln}\}$

IHS with abstract core reasoning

$\mathcal{F}_H = \{(b_1, b_2), (b_2, b_3), (b_3, b_4),$
$\bigwedge_{i=1}^{2} \text{CNF}\left((b_2 + b_3 \geq i) \rightarrow s^{AB}[i]\right)\}$

$\mathcal{F}_B = \{b_1, b_2, b_3, b_4\}$

$hs = \emptyset$

$\mathcal{A} = \text{ABSTRACT}(\mathcal{F}_B, hs, \mathcal{AB})$
$= \{b_1, s^{AB}[1], b_4\}$

$AB = \{b_2, b_3\}$

`Abstract-IHS` $(\mathcal{F})$

Initialize
while $LB < UB$
  Update $\mathcal{AB}$
  Compute min-cost hitting set hs
  Update LB
  Set up assumptions

ABSTRACT$(\mathcal{F}_B, hs, \mathcal{AB})$
  $\mathcal{A} \leftarrow \{b \mid b \in \mathcal{F}_B - hs\}$
  foreach $AB \in \mathcal{AB}$ **do**
    $\mathcal{A} \leftarrow \mathcal{A} - \{b \mid b \in AB\}$
    $\mathcal{A} \leftarrow \mathcal{A} \cup \{s^{AB}[|AB \cap hs| + 1]\}$
  return $\mathcal{A}$

IHS with abstract core reasoning

$\mathcal{F}_H = \{(b_1, b_2), (b_2, b_3), (b_3, b_4),$
$\bigwedge_{i=1}^{2} \text{CNF}\left((b_2 + b_3 \geq i) \rightarrow s^{AB}[i]\right)\}$

$\mathcal{F}_B = \{b_1, b_2, b_3, b_4\}$

$\texttt{sat-assume}(\mathcal{F}_H, \neg\mathcal{A})$

$\mathcal{A} = \{b_1, s^{AB}[1], b_4\}$

$K = \{\}$

$UB = \infty$ $LB = 0$

$\mathcal{AB} = \{AB = \{b_2, b_3\}\}$

$\text{CORES} = \emptyset$

$\text{BESTSOL} = \varnothing$

`Abstract-IHS` $(\mathcal{F})$

Initialize
while $LB < UB$
  Update $\mathcal{AB}$
  Compute min-cost hitting set *hs*
  Update LB
  Set up assumptions
  Extract cores until SAT

`Min-Abs` $(\mathcal{F}_B, \text{CORES}, \mathcal{AB})$:
minimize: $\sum_{b \in \mathcal{F}_B} b$
subject to: $\sum_{b \in \kappa} b \geq 1 \ \forall \kappa \in \text{CORES}$
$\left(\sum_{b \in AB} b \geq k\right) \leftrightarrow s^{AB}[k] \ \forall AB \in \mathcal{AB}$
return: $\{b \mid b \text{ set to 1 in opt. soln}\}$

IHS with abstract core reasoning

$\mathcal{F}_H = \{(b_1, b_2), (b_2, b_3), (b_3, b_4),$
$\bigwedge_{i=1}^{2} \text{CNF}\left((b_2 + b_3 \geq i) \rightarrow s^{AB}[i]\right)\}$

$\mathcal{F}_B = \{b_1, b_2, b_3, b_4\}$

`sat-assume`$(\mathcal{F}_H, \neg\mathcal{A})$

$\mathcal{A} = \{b_1, \cancel{s^{AB}[1]}, b_4\}$

$K = \{(s^{AB}[1])\}$

`Abstract-IHS` $(\mathcal{F})$

Initialize
while $LB < UB$
- Update $\mathcal{AB}$
- Compute min-cost hitting set *hs*
- Update LB
- Set up assumptions
- Extract cores until SAT

$UB = \infty$ $LB = 0$

$\mathcal{AB} = \{AB = \{b_2, b_3\}\}$

$\text{CORES} = \emptyset$

$\text{BESTSOL} = \varnothing$

`Min-Abs` $(\mathcal{F}_B, \text{CORES}, \mathcal{AB})$:
minimize: $\sum_{b \in \mathcal{F}_B} b$
subject to: $\sum_{b \in \kappa} b \geq 1 \; \forall \kappa \in \text{CORES}$
$\left(\sum_{b \in AB} b \geq k\right) \leftrightarrow s^{AB}[k] \; \forall AB \in \mathcal{AB}$
return: $\{b \mid b \text{ set to 1 in opt. soln}\}$

IHS with abstract core reasoning

$\mathcal{F}_H = \{(b_1, b_2), (b_2, b_3), (b_3, b_4),$
$\bigwedge_{i=1}^{2} \text{CNF}\left((b_2 + b_3 \geq i) \to s^{AB}[i]\right)\}$

$\mathcal{F}_B = \{b_1, b_2, b_3, b_4\}$

`sat-assume`$(\mathcal{F}_H, \neg\mathcal{A})$

$\mathcal{A} = \{b_1, b_4\}$

$K = \{(s^{AB}[1])\}$

$UB = \infty$ $LB = 0$

$\mathcal{AB} = \{AB = \{b_2, b_3\}\}$

$\text{CORES} = \emptyset$

$\text{BESTSOL} = \varnothing$

`Abstract-IHS` $(\mathcal{F})$

Initialize
while $LB < UB$
Update $\mathcal{AB}$
Compute min-cost hitting set *hs*
Update LB
Set up assumptions
Extract cores until SAT

`Min-Abs` $(\mathcal{F}_B, \text{CORES}, \mathcal{AB})$:
minimize: $\sum_{b \in \mathcal{F}_B} b$
subject to: $\sum_{b \in \kappa} b \geq 1 \ \forall \kappa \in \text{CORES}$
$\left(\sum_{b \in AB} b \geq k\right) \leftrightarrow s^{AB}[k] \ \forall AB \in \mathcal{AB}$
return: $\{b \mid b \text{ set to 1 in opt. soln}\}$

IHS with abstract core reasoning

$\mathcal{F}_H = \{(b_1, b_2), (b_2, b_3), (b_3, b_4),$
$\bigwedge_{i=1}^{2} \text{CNF}\left((b_2 + b_3 \geq i) \to s^{AB}[i]\right)\}$

$\mathcal{F}_B = \{b_1, b_2, b_3, b_4\}$

$\texttt{sat-assume}(\mathcal{F}_H, \neg\mathcal{A})$

$\mathcal{A} = \{b_1, b_4\}$

$K = \{(s^{AB}[1])\}$

$\tau = \{\neg b_1, b_2, b_3, \neg b_4\}$

$UB = \infty \qquad LB = 0$

$\mathcal{AB} = \{AB = \{b_2, b_3\}\}$

$\text{CORES} = \emptyset$

$\text{BESTSOL} = \varnothing$

`Abstract-IHS` $(\mathcal{F})$

Initialize
while $LB < UB$
  Update $\mathcal{AB}$
  Compute min-cost hitting set *hs*
  Update LB
  Set up assumptions
  Extract cores until SAT

`Min-Abs` $(\mathcal{F}_B, \text{CORES}, \mathcal{AB})$:
minimize: $\sum_{b \in \mathcal{F}_B} b$
subject to: $\sum_{b \in \kappa} b \geq 1 \ \forall \kappa \in \text{CORES}$
$\left(\sum_{b \in AB} b \geq k\right) \leftrightarrow s^{AB}[k] \ \forall AB \in \mathcal{AB}$
return: $\{b \mid b \text{ set to 1 in opt. soln}\}$

IHS with abstract core reasoning

$\mathcal{F}_H = \{(b_1, b_2), (b_2, b_3), (b_3, b_4),$
$\bigwedge_{i=1}^{2} \text{CNF}\left((b_2 + b_3 \geq i) \to s^{AB}[i]\right)\}$

$\mathcal{F}_B = \{b_1, b_2, b_3, b_4\}$

$K = \{(s^{AB}[1])\}$

$\tau = \{\neg b_1, b_2, b_3, \neg b_4\}$

UB $= cost(\tau)$ $\quad LB = 0$

$\mathcal{AB} = \{AB = \{b_2, b_3\}\}$

CORES $= \emptyset$

BESTSOL $= \{\neg b_1, b_2, b_3, \neg b_4\}$

`Abstract-IHS` $(\mathcal{F})$

Initialize
while $LB < UB$
- Update $\mathcal{AB}$
- Compute min-cost hitting set *hs*
- Update LB
- Set up assumptions
- Extract cores until SAT
- Update UB

`Min-Abs` $(\mathcal{F}_B, \text{CORES}, \mathcal{AB})$:
minimize: $\sum_{b \in \mathcal{F}_B} b$
subject to: $\sum_{b \in \kappa} b \geq 1 \ \forall \kappa \in \text{CORES}$
$\left(\sum_{b \in AB} b \geq k\right) \leftrightarrow s^{AB}[k] \ \forall AB \in \mathcal{AB}$
return: $\{b \mid b \text{ set to 1 in opt. soln}\}$

IHS with abstract core reasoning

$\mathcal{F}_H = \{(b_1, b_2), (b_2, b_3), (b_3, b_4),$
$\bigwedge_{i=1}^{2} \text{CNF}\left((b_2 + b_3 \geq i) \rightarrow s^{AB}[i]\right)\}$

$\mathcal{F}_B = \{b_1, b_2, b_3, b_4\}$

$K = \{(s^{AB}[1])\}$

$\tau = \{\neg b_1, b_2, b_3, \neg b_4\}$

$UB = 2$ $LB = 0$

$\mathcal{AB} = \{AB = \{b_2, b_3\}\}$

$\text{CORES} = \{(s^{AB}[1])\}$

$\text{BESTSOL} = \{\neg b_1, b_2, b_3, \neg b_4\}$

`Abstract-IHS` $(\mathcal{F})$

Initialize
while $LB < UB$
- Update $\mathcal{AB}$
- Compute min-cost hitting set *hs*
- Update LB
- Set up assumptions
- Extract cores until SAT
- Update UB
- Add cores to CORES

`Min-Abs` $(\mathcal{F}_B, \text{CORES}, \mathcal{AB})$:
minimize: $\sum_{b \in \mathcal{F}_B} b$
subject to: $\sum_{b \in \kappa} b \geq 1 \ \forall \kappa \in \text{CORES}$
$(\sum_{b \in AB} b \geq k) \leftrightarrow s^{AB}[k] \ \forall AB \in \mathcal{AB}$
return: $\{b \mid b \text{ set to 1 in opt. soln}\}$

IHS with abstract core reasoning

$\mathcal{F}_H = \{(b_1, b_2), (b_2, b_3), (b_3, b_4),$
$\bigwedge_{i=1}^{2} \text{CNF}\left((b_2 + b_3 \geq i) \rightarrow s^{AB}[i]\right)\}$

$\mathcal{F}_B = \{b_1, b_2, b_3, b_4\}$

`Abstract-IHS` $(\mathcal{F})$

Initialize
while $LB < UB$
- Update $\mathcal{AB}$
- Compute min-cost hitting set *hs*
- Update LB
- Set up assumptions
- Extract cores until SAT
- Update UB
- Add cores to CORES

$UB = 2$ $LB = 0$

$\mathcal{AB} = \{AB = \{b_2, b_3\}\}$

$\text{CORES} = \{(s^{AB}[1])\}$

$\text{BESTSOL} = \{\neg b_1, b_2, b_3, \neg b_4\}$

`Min-Abs` $(\mathcal{F}_B, \text{CORES}, \mathcal{AB})$:
minimize: $\sum_{b \in \mathcal{F}_B} b$
subject to: $\sum_{b \in \kappa} b \geq 1 \; \forall \kappa \in \text{CORES}$
$\left(\sum_{b \in AB} b \geq k\right) \leftrightarrow s^{AB}[k] \; \forall AB \in \mathcal{AB}$
return: $\{b \mid b \text{ set to 1 in opt. soln}\}$

IHS with abstract core reasoning

$\mathcal{F}_H = \{(b_1, b_2), (b_2, b_3), (b_3, b_4),$
$\bigwedge_{i=1}^{2} \text{CNF}\left((b_2 + b_3 \geq i) \rightarrow s^{AB}[i]\right)\}$

$\mathcal{F}_B = \{b_1, b_2, b_3, b_4\}$

`Abstract-IHS` $(\mathcal{F})$

Initialize
while $LB < UB$
  Update $\mathcal{AB}$
  Compute min-cost hitting set *hs*
  Update LB
  Set up assumptions
  Extract cores until SAT
  Update UB
  Add cores to CORES

$UB = 2$ $LB = 0$

$\mathcal{AB} = \{AB = \{b_2, b_3\}\}$

$\text{CORES} = \{(s^{AB}[1])\}$

$\text{BESTSOL} = \{\neg b_1, b_2, b_3, \neg b_4\}$

`Min-Abs` $(\mathcal{F}_B, \text{CORES}, \mathcal{AB})$:
minimize: $\sum_{b \in \mathcal{F}_B} b$
subject to: $\sum_{b \in \kappa} b \geq 1 \; \forall \kappa \in \text{CORES}$
$\left(\sum_{b \in AB} b \geq k\right) \leftrightarrow s^{AB}[k] \; \forall AB \in \mathcal{AB}$
return: $\{b \mid b \text{ set to 1 in opt. soln}\}$

IHS with abstract core reasoning

$\mathcal{F}_H = \{(b_1, b_2), (b_2, b_3), (b_3, b_4),$
$\bigwedge_{i=1}^{2} \text{CNF}\left((b_2 + b_3 \geq i) \to s^{AB}[i]\right)\}$

$\mathcal{F}_B = \{b_1, b_2, b_3, b_4\}$

$hs = \texttt{Min-Abs}\ (\mathcal{F}_B, \{(s^{AB}[1])\}, \mathcal{AB})$

$\texttt{Abstract-IHS}\ (\mathcal{F})$

Initialize
while $LB < UB$
  Update $\mathcal{AB}$
  Compute min-cost hitting set *hs*
  Update LB
  Set up assumptions
  Extract cores until SAT
  Update UB
  Add cores to CORES

$UB = 2$ $LB = 0$

$\mathcal{AB} = \{AB = \{b_2, b_3\}\}$

$\text{CORES} = \{(s^{AB}[1])\}$

$\text{BESTSOL} = \{\neg b_1, b_2, b_3, \neg b_4\}$

$\texttt{Min-Abs}\ (\mathcal{F}_B, \text{CORES}, \mathcal{AB})$:
minimize: $\sum_{b \in \mathcal{F}_B} b$
subject to: $\sum_{b \in \kappa} b \geq 1\ \forall \kappa \in \text{CORES}$
$\left(\sum_{\mathbf{b} \in \mathbf{AB}} \mathbf{b} \geq \mathbf{k}\right) \leftrightarrow \mathbf{s^{AB}}[\mathbf{k}]\ \forall \mathbf{AB} \in \mathcal{AB}$
return: $\{b \mid b \text{ set to 1 in opt. soln}\}$

IHS with abstract core reasoning

$\mathcal{F}_H = \{(b_1, b_2), (b_2, b_3), (b_3, b_4),$
$\bigwedge_{i=1}^{2} \text{CNF}\left((b_2 + b_3 \geq i) \rightarrow s^{AB}[i]\right)\}$

$\mathcal{F}_B = \{b_1, b_2, b_3, b_4\}$

$hs = \{b_2\}$

`Abstract-IHS` $(\mathcal{F})$

Initialize
while $LB < UB$
  Update $\mathcal{AB}$
  Compute min-cost hitting set hs
  Update LB
  Set up assumptions
  Extract cores until SAT
  Update UB
  Add cores to CORES

$UB = 2$ $LB = |\{b_2\}|$

$\mathcal{AB} = \{AB = \{b_2, b_3\}\}$

$\text{CORES} = \{(s^{AB}[1])\}$

$\text{BESTSOL} = \{\neg b_1, b_2, b_3, \neg b_4\}$

`Min-Abs` $(\mathcal{F}_B, \text{CORES}, \mathcal{AB})$:
minimize: $\sum_{b \in \mathcal{F}_B} b$
subject to: $\sum_{b \in \kappa} b \geq 1 \ \forall \kappa \in \text{CORES}$
$\left(\sum_{b \in AB} b \geq k\right) \leftrightarrow s^{AB}[k] \ \forall AB \in \mathcal{AB}$
return: $\{b \mid b \text{ set to 1 in opt. soln}\}$

IHS with abstract core reasoning

$\mathcal{F}_H = \{(b_1, b_2), (b_2, b_3), (b_3, b_4),$
$\bigwedge_{i=1}^{2} \text{CNF}\left((b_2 + b_3 \geq i) \rightarrow s^{AB}[i]\right)\}$

$\mathcal{F}_B = \{b_1, b_2, b_3, b_4\}$

$hs = \{b_2\}$

$\mathcal{A} = \{b_1, s^{AB}[2], b_4\}$

`Abstract-IHS` $(\mathcal{F})$

Initialize
while $LB < UB$
Update $\mathcal{AB}$
Compute min-cost hitting set hs
Update LB
Set up assumptions
Extract cores until SAT
Update UB
Add cores to CORES

ABSTRACT$(\mathcal{F}_B, hs, \mathcal{AB})$
$\mathcal{A} \leftarrow \{b \mid b \in \mathcal{F}_B - hs\}$
foreach $AB \in \mathcal{AB}$ **do**
$\mathcal{A} \leftarrow \mathcal{A} - \{b \mid b \in AB\}$
$\mathcal{A} \leftarrow \mathcal{A} \cup \{s^{AB}[|AB \cap hs| + 1]\}$
return $\mathcal{A}$

IHS with abstract core reasoning

$\mathcal{F}_H = \{(b_1, b_2), (b_2, b_3), (b_3, b_4),$
$\bigwedge_{i=1}^{2} \text{CNF}\left((b_2 + b_3 \geq i) \rightarrow s^{AB}[i]\right)\}$

$\mathcal{F}_B = \{b_1, b_2, b_3, b_4\}$

$\texttt{sat-assume}(\mathcal{F}_H, \neg\mathcal{A})$

$\mathcal{A} = \{b_1, s^{AB}[2], b_4\}$

$K = \{\}$

$\texttt{Abstract-IHS}\ (\mathcal{F})$

Initialize
while $LB < UB$
  Update $\mathcal{AB}$
  Compute min-cost hitting set *hs*
  Update LB
  Set up assumptions
  Extract cores until SAT
  Update UB
  Add cores to CORES

$UB = 2$ $LB = 1$

$\mathcal{AB} = \{AB = \{b_2, b_3\}\}$

$\text{CORES} = \{(s^{AB}[1])\}$

$\text{BESTSOL} = \{\neg b_1, b_2, b_3, \neg b_4\}$

$\texttt{Min-Abs}\ (\mathcal{F}_B, \text{CORES}, \mathcal{AB})$:
minimize: $\sum_{b \in \mathcal{F}_B} b$
subject to: $\sum_{b \in \kappa} b \geq 1\ \forall \kappa \in \text{CORES}$
$\left(\sum_{b \in AB} b \geq k\right) \leftrightarrow s^{AB}[k]\ \forall AB \in \mathcal{AB}$
return: $\{b \mid b \text{ set to 1 in opt. soln}\}$

IHS with abstract core reasoning

$\mathcal{F}_H = \{(b_1, b_2), (b_2, b_3), (b_3, b_4),$
$\bigwedge_{i=1}^{2} \text{CNF}\left((b_2 + b_3 \geq i) \to s^{AB}[i]\right)\}$

$\mathcal{F}_B = \{b_1, b_2, b_3, b_4\}$

`sat-assume`$(\mathcal{F}_H, \neg\mathcal{A})$

$\mathcal{A} = \{\cancel{b_1}, \cancel{s^{AB}[2]}, \cancel{b_4}\}$

$K = \{(b_1, s^{AB}[2], b_4)\}$

`Abstract-IHS` $(\mathcal{F})$

Initialize
while $LB < UB$
  Update $\mathcal{AB}$
  Compute min-cost hitting set *hs*
  Update LB
  Set up assumptions
  Extract cores until SAT
  Update UB
  Add cores to CORES

$UB = 2$ $LB = 1$

$\mathcal{AB} = \{AB = \{b_2, b_3\}\}$

$\text{CORES} = \{(s^{AB}[1])\}$

$\text{BESTSOL} = \{\neg b_1, b_2, b_3, \neg b_4\}$

`Min-Abs` $(\mathcal{F}_B, \text{CORES}, \mathcal{AB})$:
minimize: $\sum_{b \in \mathcal{F}_B} b$
subject to: $\sum_{b \in \kappa} b \geq 1 \ \forall \kappa \in \text{CORES}$
$\left(\sum_{b \in AB} b \geq k\right) \leftrightarrow s^{AB}[k] \ \forall AB \in \mathcal{AB}$
return: $\{b \mid b \text{ set to 1 in opt. soln}\}$

IHS with abstract core reasoning

$\mathcal{F}_H = \{(b_1, b_2), (b_2, b_3), (b_3, b_4),$
$\bigwedge_{i=1}^{2} \text{CNF}\left((b_2 + b_3 \geq i) \to s^{AB}[i]\right)\}$

$\mathcal{F}_B = \{b_1, b_2, b_3, b_4\}$

$K = \{(b_1, s^{AB}[2], b_4)\}$

$\tau = \{\neg b_1, b_2, b_3, \neg b_4\}$

$UB = 2$ $LB = 1$

$\mathcal{AB} = \{AB = \{b_2, b_3\}\}$

$\text{CORES} = \{(s^{AB}[1])\}$

$\text{BESTSOL} = \{\neg b_1, b_2, b_3, \neg b_4\}$

`Abstract-IHS` $(\mathcal{F})$

Initialize
while $LB < UB$
- Update $\mathcal{AB}$
- Compute min-cost hitting set *hs*
- Update LB
- Set up assumptions
- Extract cores until SAT
- Update UB
- Add cores to CORES

`Min-Abs` $(\mathcal{F}_B, \text{CORES}, \mathcal{AB})$:
minimize: $\sum_{b \in \mathcal{F}_B} b$
subject to: $\sum_{b \in \kappa} b \geq 1 \ \forall \kappa \in \text{CORES}$
$(\sum_{b \in AB} b \geq k) \leftrightarrow s^{AB}[k] \ \forall AB \in \mathcal{AB}$
return: $\{b \mid b \text{ set to 1 in opt. soln}\}$

IHS with abstract core reasoning

$\mathcal{F}_H = \{(b_1, b_2), (b_2, b_3), (b_3, b_4),$
$\bigwedge_{i=1}^{2} \text{CNF}\left((b_2 + b_3 \geq i) \rightarrow s^{AB}[i]\right)\}$

$\mathcal{F}_B = \{b_1, b_2, b_3, b_4\}$

$K = \{(b_1, s^{AB}[2], b_4)\}$

$\tau = \{\neg b_1, b_2, b_3, \neg b_4\}$

$UB = 2$ $LB = 1$

$\mathcal{AB} = \{AB = \{b_2, b_3\}\}$

$\text{CORES} = \{(s^{AB}[1]), (b_1, s^{AB}[2], b_4)\}$

$\text{BESTSOL} = \{\neg b_1, b_2, b_3, \neg b_4\}$

`Abstract-IHS` $(\mathcal{F})$

Initialize
while $LB < UB$
- Update $\mathcal{AB}$
- Compute min-cost hitting set *hs*
- Update LB
- Set up assumptions
- Extract cores until SAT
- Update UB
- Add cores to CORES

`Min-Abs` $(\mathcal{F}_B, \text{CORES}, \mathcal{AB})$:
minimize: $\sum_{b \in \mathcal{F}_B} b$
subject to: $\sum_{b \in \kappa} b \geq 1 \;\forall \kappa \in \text{CORES}$
$\left(\sum_{b \in AB} b \geq k\right) \leftrightarrow s^{AB}[k] \;\forall AB \in \mathcal{AB}$
return: $\{b \mid b \text{ set to 1 in opt. soln}\}$

IHS with abstract core reasoning

$\mathcal{F}_H = \{(b_1, b_2), (b_2, b_3), (b_3, b_4),$
$\bigwedge_{i=1}^{2} \text{CNF}\left((b_2 + b_3 \geq i) \to s^{AB}[i]\right)\}$

$\mathcal{F}_B = \{b_1, b_2, b_3, b_4\}$

`Abstract-IHS` $(\mathcal{F})$

Initialize
while $LB < UB$
  Update $\mathcal{AB}$
  Compute min-cost hitting set *hs*
  Update LB
  Set up assumptions
  Extract cores until SAT
  Update UB
  Add cores to CORES

$UB = 2$ $LB = 1$

$\mathcal{AB} = \{AB = \{b_2, b_3\}\}$

$\text{CORES} = \{(s^{AB}[1]), (b_1, s^{AB}[2], b_4)\}$

$\text{BESTSOL} = \{\neg b_1, b_2, b_3, \neg b_4\}$

`Min-Abs` $(\mathcal{F}_B, \text{CORES}, \mathcal{AB})$:
minimize: $\sum_{b \in \mathcal{F}_B} b$
subject to: $\sum_{b \in \kappa} b \geq 1 \ \forall \kappa \in \text{CORES}$
$\left(\sum_{b \in AB} b \geq k\right) \leftrightarrow s^{AB}[k] \ \forall AB \in \mathcal{AB}$
return: $\{b \mid b \text{ set to 1 in opt. soln}\}$

IHS with abstract core reasoning

$\mathcal{F}_H = \{(b_1, b_2), (b_2, b_3), (b_3, b_4),$
$\bigwedge_{i=1}^{2} \text{CNF}\left((b_2 + b_3 \geq i) \rightarrow s^{AB}[i]\right)\}$

$\mathcal{F}_B = \{b_1, b_2, b_3, b_4\}$

$hs = \texttt{Min-Abs}\,(\mathcal{F}_B, \text{CORES}, \mathcal{AB})$

`Abstract-IHS` $(\mathcal{F})$

Initialize
while $LB < UB$
- Update $\mathcal{AB}$
- Compute min-cost hitting set hs
- Update LB
- Set up assumptions
- Extract cores until SAT
- Update UB
- Add cores to CORES

$UB = 2$ $LB = 1$

$\mathcal{AB} = \{AB = \{b_2, b_3\}\}$

$\text{CORES} = \{(s^{AB}[1]), (b_1, s^{AB}[2], b_4)\}$

$\text{BESTSOL} = \{\neg b_1, b_2, b_3, \neg b_4\}$

`Min-Abs` $(\mathcal{F}_B, \text{CORES}, \mathcal{AB})$:
minimize: $\sum_{b \in \mathcal{F}_B} b$
subject to: $\sum_{b \in \kappa} b \geq 1 \ \forall \kappa \in \text{CORES}$
$\left(\sum_{b \in AB} b \geq k\right) \leftrightarrow s^{AB}[k] \ \forall AB \in \mathcal{AB}$
return: $\{b \mid b \text{ set to 1 in opt. soln}\}$

IHS with abstract core reasoning

$\mathcal{F}_H = \{(b_1, b_2), (b_2, b_3), (b_3, b_4),$
$\bigwedge_{i=1}^{2} \text{CNF}\left((b_2 + b_3 \geq i) \rightarrow s^{AB}[i]\right)\}$

$\mathcal{F}_B = \{b_1, b_2, b_3, b_4\}$

$hs = \{b_2, b_3\}$

`Abstract-IHS` $(\mathcal{F})$

Initialize
while $LB < UB$
  Update $\mathcal{AB}$
  Compute min-cost hitting set *hs*
  Update LB
  Set up assumptions
  Extract cores until SAT
  Update UB
  Add cores to CORES

$UB = 2$ $LB = |\{b_2, b_3\}|$

$\mathcal{AB} = \{AB = \{b_2, b_3\}\}$

$\text{CORES} = \{(s^{AB}[1]), (b_1, s^{AB}[2], b_4)\}$

$\text{BESTSOL} = \{\neg b_1, b_2, b_3, \neg b_4\}$

`Min-Abs` $(\mathcal{F}_B, \text{CORES}, \mathcal{AB})$:
minimize: $\sum_{b \in \mathcal{F}_B} b$
subject to: $\sum_{b \in \kappa} b \geq 1 \; \forall \kappa \in \text{CORES}$
$\left(\sum_{b \in AB} b \geq k\right) \leftrightarrow s^{AB}[k] \; \forall AB \in \mathcal{AB}$
return: $\{b \mid b \text{ set to 1 in opt. soln}\}$

IHS with abstract core reasoning

$\mathcal{F}_H = \{(b_1, b_2), (b_2, b_3), (b_3, b_4),$
$\bigwedge_{i=1}^{2} \text{CNF}\left((b_2 + b_3 \geq i) \rightarrow s^{AB}[i]\right)\}$

$\mathcal{F}_B = \{b_1, b_2, b_3, b_4\}$

$hs = \{b_2, b_3\}$

$\texttt{Abstract-IHS}\ (\mathcal{F})$

Initialize
while $LB < UB$
  Update $\mathcal{AB}$
  Compute min-cost hitting set hs
  Update LB
  Set up assumptions
  Extract cores until SAT
  Update UB
  Add cores to CORES
return BESTSOL

$UB = 2$ $LB = 2$

$\mathcal{AB} = \{AB = \{b_2, b_3\}\}$

$\text{CORES} = \{(s^{AB}[1]), (b_1, s^{AB}[2], b_4)\}$

$\text{BESTSOL} = \{\neg b_1, b_2, b_3, \neg b_4\}$

$\texttt{Min-Abs}\ (\mathcal{F}_B, \text{CORES}, \mathcal{AB})$:
minimize: $\sum_{b \in \mathcal{F}_B} b$
subject to: $\sum_{b \in \kappa} b \geq 1\ \forall \kappa \in \text{CORES}$
$\left(\sum_{b \in AB} b \geq k\right) \leftrightarrow s^{AB}[k]\ \forall AB \in \mathcal{AB}$
return: $\{b \mid b \text{ set to 1 in opt. soln}\}$

Effects of Abstract Cores

Abstract cores improve IHS in theory

In theory

For each (unweighted) MaxSAT instance, there exists an abstraction set with which `Abstract-IHS` terminates with a polynomial number of cores.

Abstract cores improve IHS in theory

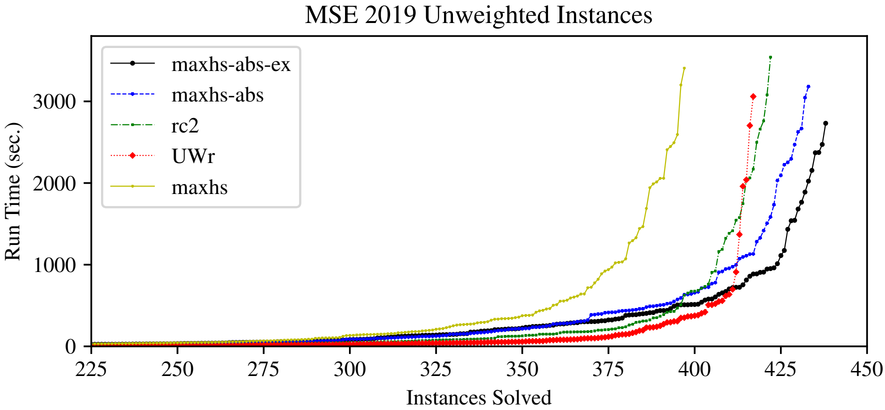
In theory

For each (unweighted) MaxSAT instance, there exists an abstraction set with which `Abstract-IHS` terminates with a polynomial number of cores.

..however

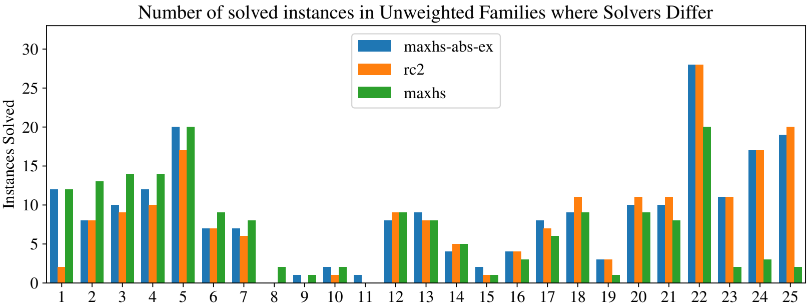
- trade of between expressivity and overhead
- abstraction sets should be large enough to benefit IHS without inducing a lot of overhead.

..and practice



- MAXHS: basic IHS (MaxHS Davies and Bacchus [2013, 2011])
- MAXHS-ABS: MAXHS with abstract core reasoning
- MAXHS-ABS-EX: MAXHS-ABS with additional heuristics.
- RC2 and UWR best performing solvers in 2019 MSE Ignatiev et al. [2019]; Karpinski and Piotrów [2019]; Bacchus et al. [2019]

by benchmark family



Conclusions

- *Abstract Cores:*
  - Cores containing variables representing logical constraints of soft clauses
  - can represent a large set of regular cores
  - We consider summations, other instantiations remain as future work
- In this paper we
  - show how to incorporate abstract cores into the the IHS MaxSAT algorithm,
  - show that the resulting algorithm improves over basic IHS in theory,
  - instantiate the framework within the MaxHS solver
  - show that the resulting solver improves over the standard MaxHS in practice.

Conclusions

- *Abstract Cores:*
  - Cores containing variables representing logical constraints of soft clauses
  - can represent a large set of regular cores
  - We consider summations, other instantiations remain as future work
- In this paper we
  - show how to incorporate abstract cores into the the IHS MaxSAT algorithm,
  - show that the resulting algorithm improves over basic IHS in theory,
  - instantiate the framework within the MaxHS solver
  - show that the resulting solver improves over the standard MaxHS in practice.

Conclusions

- *Abstract Cores:*
  - Cores containing variables representing logical constraints of soft clauses
  - can represent a large set of regular cores
  - We consider summations, other instantiations remain as future work
- In this paper we
  - show how to incorporate abstract cores into the the IHS MaxSAT algorithm,
  - show that the resulting algorithm improves over basic IHS in theory,
  - instantiate the framework within the MaxHS solver
  - show that the resulting solver improves over the standard MaxHS in practice.

Conclusions

- *Abstract Cores:*
    - Cores containing variables representing logical constraints of soft clauses
    - can represent a large set of regular cores
    - We consider summations, other instantiations remain as future work
- In this paper we
    - show how to incorporate abstract cores into the the IHS MaxSAT algorithm,
    - show that the resulting algorithm improves over basic IHS in theory,
    - instantiate the framework within the MaxHS solver
    - show that the resulting solver improves over the standard MaxHS in practice.

Conclusions

- *Abstract Cores:*
  - Cores containing variables representing logical constraints of soft clauses
  - can represent a large set of regular cores
  - We consider summations, other instantiations remain as future work
- In this paper we
  - show how to incorporate abstract cores into the the IHS MaxSAT algorithm,
  - show that the resulting algorithm improves over basic IHS in theory,
  - instantiate the framework within the MaxHS solver
  - show that the resulting solver improves over the standard MaxHS in practice.

Conclusions

- *Abstract Cores:*
  - Cores containing variables representing logical constraints of soft clauses
  - can represent a large set of regular cores
  - We consider summations, other instantiations remain as future work
- In this paper we
  - show how to incorporate abstract cores into the the IHS MaxSAT algorithm,
  - show that the resulting algorithm improves over basic IHS in theory,
  - instantiate the framework within the MaxHS solver
  - show that the resulting solver improves over the standard MaxHS in practice.

Conclusions

- *Abstract Cores:*
  - Cores containing variables representing logical constraints of soft clauses
  - can represent a large set of regular cores
  - We consider summations, other instantiations remain as future work
- In this paper we
  - show how to incorporate abstract cores into the the IHS MaxSAT algorithm,
  - show that the resulting algorithm improves over basic IHS in theory,
  - instantiate the framework within the MaxHS solver
  - show that the resulting solver improves over the standard MaxHS in practice.

Bibliography I

Fahiem Bacchus, Matti Järvisalo, and Ruben Martins. Maxsat evaluation 2018: New developments and detailed results. *J. Satisf. Boolean Model. Comput.*, 11(1):99–131, 2019. URL https://doi.org/10.3233/SAT190119.

Jeremias Berg and Matti Järvisalo. Cost-optimal constrained correlation clustering via weighted partial maximum satisfiability. *Artif. Intell.*, 244:110–142, 2017. URL https://doi.org/10.1016/j.artint.2015.07.001.

Yibin Chen, Sean Safarpour, João Marques-Silva, and Andreas G. Veneris. Automated design debugging with maximum satisfiability. *IEEE Trans. on CAD of Integrated Circuits and Systems*, 29(11):1804–1817, 2010. URL https://doi.org/10.1109/TCAD.2010.2061270.

Jessica Davies. *Solving MAXSAT by Decoupling Optimization and Satisfaction*. PhD thesis, University of Toronto, 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.

Jessica Davies and Fahiem Bacchus. Exploiting the power of mip solvers in maxsat. In Matti Järvisalo and Allen Van Gelder, editors, *Proc SAT*, volume 7962 of *LNCS*, pages 166–181. Springer, 2013. URL https://doi.org/10.1007/978-3-642-39071-5_13.

Emir Demirovic, Nysret Musliu, and Felix Winter. Modeling and solving staff scheduling with partial weighted maxsat. *Annals OR*, 275(1):79–99, 2019. URL https://doi.org/10.1007/s10479-017-2693-y.

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.

Toshinori Hosokawa, Hiroshi Yamazaki, Kenichiro Misawa, Masayoshi Yoshimura, Yuki Hirama, and Masavuki Arai. A low capture power oriented x-filling method using partial maxsat iteratively. In *Proc IEEE International Symposium on Defect and Fault Tolerance in VLSI and Nanotechnology Systems, DFT*, pages 1–6. IEEE, 2019. URL https://doi.org/10.1109/DFT.2019.8875434.

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.

Michal Karpinski and Marek Piotrów. Encoding cardinality constraints using multiway merge selection networks. *Constraints*, 24 (3-4):234–251, 2019. URL https://doi.org/10.1007/s10601-019-09302-0.

Lei Zhang and Fahiem Bacchus. MAXSAT heuristics for cost optimal planning. In Jörg Hoffmann and Bart Selman, editors, *Proc AAAI*. AAAI Press, 2012. URL http://www.aaai.org/ocs/index.php/AAAI/AAAI12/paper/view/5190.

Results - Weighted