Clause Redundancy and Preprocessing in Maximum Satisfiability

Hannes Ihalainen Jeremias Berg Matti Järvisalo

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

IJCAR 2022
Haifa

Motivation

- MaxSAT: declarative optimisation based on propositional logic.
  - compute an assignment to Boolean (0-1) variables that:
    - ★ satisfies a set of (hard) clauses.
    - ★ minimizes a linear objective function (equivalently, minimizes the sum of weights of falsified soft 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])

(Complete) SAT-based MaxSAT solving

Algorithms

(Complete) SAT-based MaxSAT solving

Algorithms

Core-Guided

- Lower-bounding search
- with a SAT solver

(Complete) SAT-based MaxSAT solving

Algorithms

Core-Guided

- Lower-bounding search with a SAT solver

Implicit Hitting Set (IHS)

- Lower-bounding search with a SAT and MIP solver

(Complete) SAT-based MaxSAT solving

Algorithms

Core-Guided

- Lower-bounding search with a SAT solver

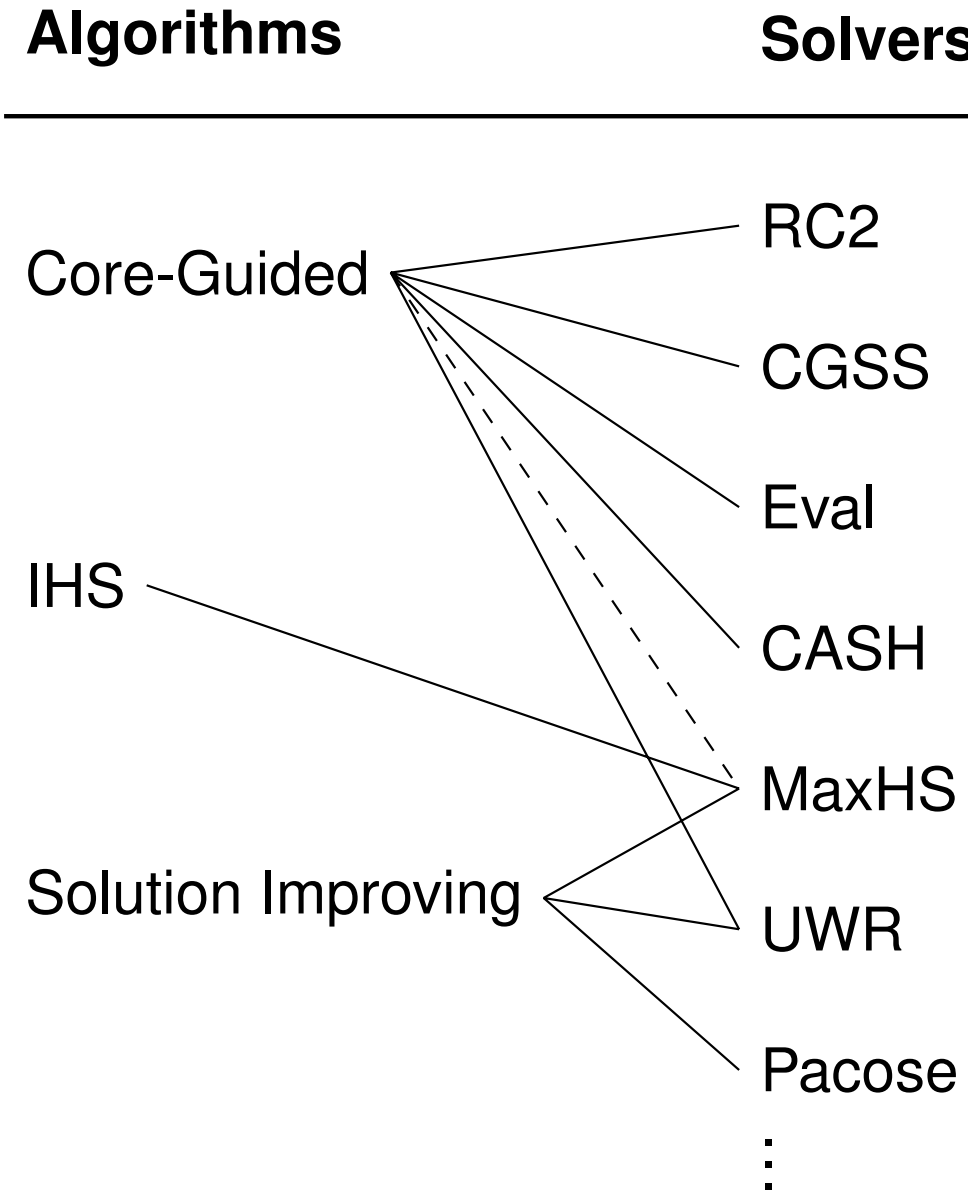
IHS

- Lower-bounding search with a SAT and MIP solver

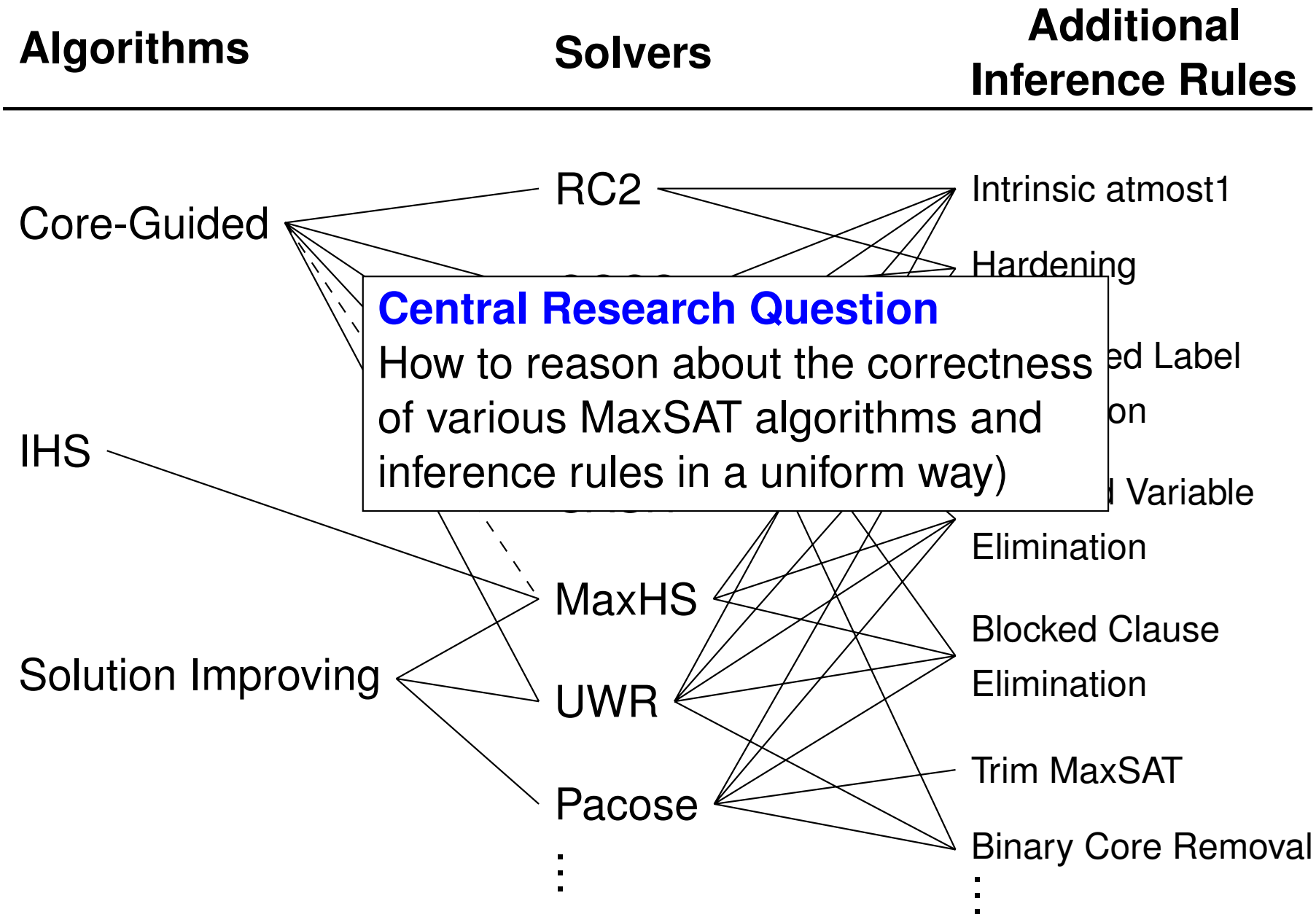
Solution Improving

- Upper-bounding search with a SAT solver

(Complete) SAT-based MaxSAT solving



(Complete) SAT-based MaxSAT solving



Our (Theoretical) Contributions(1/2)

Clause Redundancy central base for proofs of correctness of SAT solving techniques.

Our (Theoretical) Contributions(1/2)

Clause Redundancy central base for proofs of correctness of SAT solving techniques.

In this paper we:

- Lift recently proposed redundancy notions from SAT to MaxSAT.
- Reason about the correctness of various solver techniques.
- Analyse the effect of redundant clauses on solutions of MaxSAT instances:
  - Focus especially on the MaxSAT specific concept of minimal correction sets.

[Heule and Kiesl, 2017]
[Heule, Kiesl, and Biere, 2020]
[Järvisalo, Biere, and Heule, 2012]
[Heule, Kiesl, Seidl, and Biere, 2017]
[Biere, Järvisalo, and Kiesl, 2021]
[Järvisalo, Heule, and Biere, 2012]
[Cruz-Filipe, Heule, Hunt Jr., Kaufmann, and Schneider-Kamp, 2017]
[Heule, Järvisalo, Lonsing, Seidl, and Biere, 2015]
[Heule, Kiesl, and Biere, 2019]

Our (Theoretical) Contributions(1/2)

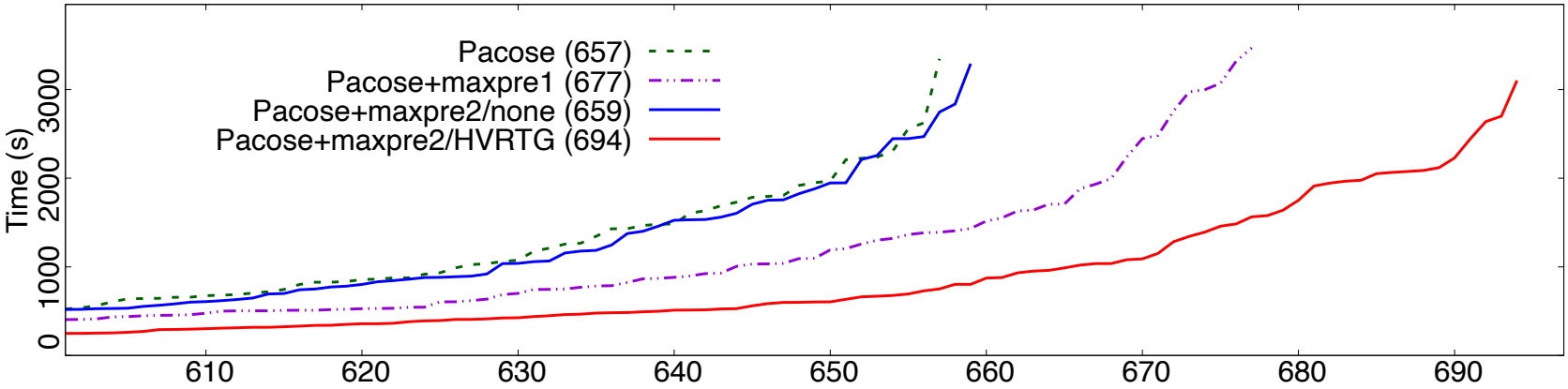
Clause Redundancy central base for proofs of correctness of SAT solving techniques.

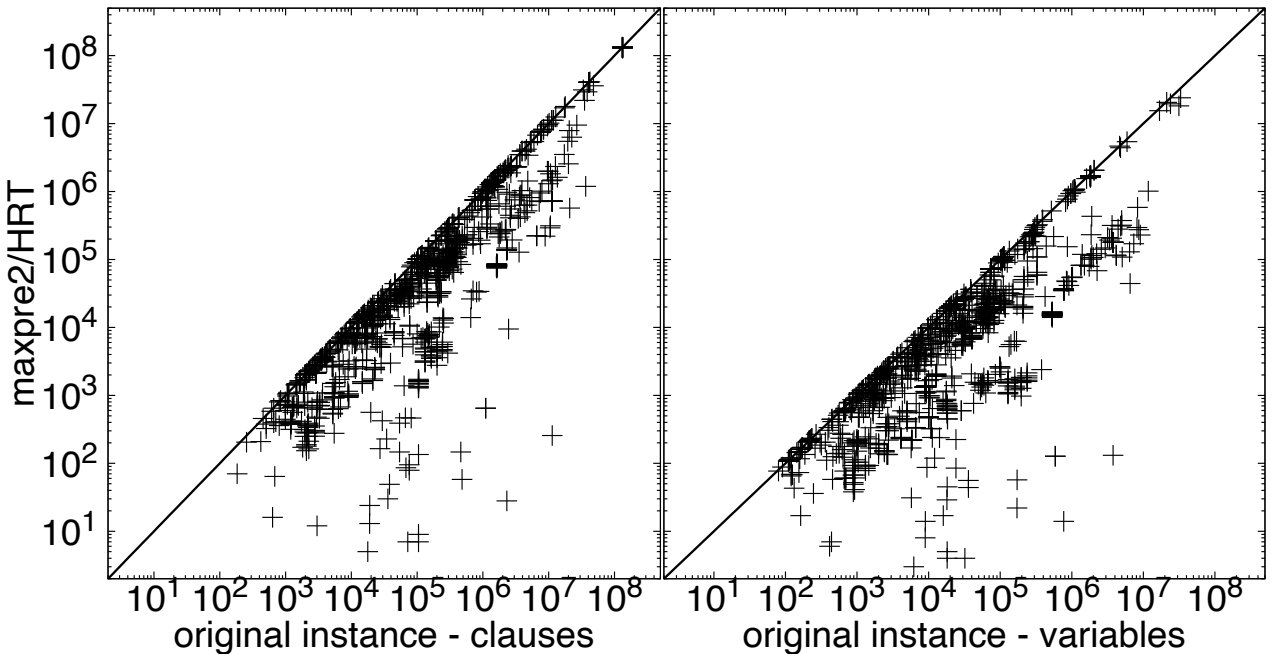
In this paper we:

- Lift recently proposed redundancy notions from SAT to MaxSAT.
- Reason about the correctness of various solver techniques.
- Analyse the effect of redundant clauses on solutions of MaxSAT instances:
  - Focus especially on the MaxSAT specific concept of minimal correction sets.

[Heule and Kiesl, 2017]
[Heule, Kiesl, and Biere, 2020]
[Järvisalo, Biere, and Heule, 2012]
[Heule, Kiesl, Seidl, and Biere, 2017]
[Biere, Järvisalo, and Kiesl, 2021]
[Järvisalo, Heule, and Biere, 2012]
[Cruz-Filipe, Heule, Hunt Jr., Kaufmann, and Schneider-Kamp, 2017]
[Heule, Järvisalo, Lonsing, Seidl, and Biere, 2015]
[Heule, Kiesl, and Biere, 2019]

Expressive preprocessing in practice





Our (Practical) Contributions(2/2)

- **MaxPre 2**: Stand-alone implementation of reasoning techniques previously only appearing in solvers.
- Empirical evaluation on effect of preprocessing on several types of state-of-the-art solvers.

[Korhonen, Berg, Saikko, and Järvisalo, 2017]
[Ansótegui, Bonet, Gabàs, and Levy, 2012]
[Paxian, Raiola, and Becker, 2021]
[Marques-Silva and Planes, 2008]
[Berg, Demirovic, and Stuckey, 2019]
[Morgado, Dodaro, and Marques-Silva, 2014]
[Davies and Bacchus, 2013]
[Martins, Joshi, Manquinho, and Lynce, 2014a]
[Saikko, Berg, and Järvisalo, 2016]
[Martins, Manquinho, and Lynce, 2014b]
[Koshimura, Zhang, Fujita, and Hasegawa, 2012]
[Ignatiev, Morgado, and Marques-Silva, 2019]
[Korhonen, Berg, Saikko, and Järvisalo, 2017]

Outline

1. Definitions.
2. Clause Redundancy in MaxSAT
3. Effects of Redundant Clauses in MaxSAT.
4. Empirical Evaluation.

Preliminaries: Maximum Satisfiability

- Optimisation extension of Boolean Satisfiability (SAT)
- An instance:
  - a set of hard clauses,
  - a linear objective function *cost*
- Find τ that:
  - satisfies all hard clauses and
  - minimizes cost
  - $cost(\mathcal{F})$ = cost of its optimal solutions.
- $\mathcal{F}_B$: set of objective variables.
- **Note:** All results extend to weights!

Preliminaries: Maximum Satisfiability

- Optimisation extension of Boolean Satisfiability (SAT)
- An instance:
  - a set of hard clauses,
  - a linear objective function *cost*
  - equivalent to a set of soft clauses.
- Find τ that:
  - satisfies all hard clauses and
  - minimizes cost
  - $cost(\mathcal{F}) =$ cost of its optimal solutions.
- $\mathcal{F}_B$: set of objective variables.
- **Note:** All results extend to weights!

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

$$cost \equiv b_1 + b_2 + b_3$$

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

Preliminaries: Maximum Satisfiability

- Optimisation extension of Boolean Satisfiability (SAT)
- An instance:
  - a set of hard clauses,
  - a linear objective function *cost*
- Find τ that:
  - satisfies all hard clauses and
  - minimizes cost
  - $cost(\mathcal{F})$ = cost of its optimal solutions.
- $\mathcal{F}_B$: set of objective variables.
- **Note:** All results extend to weights!

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

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

$$cost \equiv b_1 + b_2 + b_3$$

$$cost(\tau) = 2$$

Preliminaries: Maximum Satisfiability

- Optimisation extension of Boolean Satisfiability (SAT)
- An instance:
  - a set of hard clauses,
  - a linear objective function *cost*
- Find τ that:
  - satisfies all hard clauses and
  - minimizes cost
  - $cost(\mathcal{F})$ = cost of its optimal solutions.
- $\mathcal{F}_B$: set of objective variables.
- **Note:** All results extend to weights!

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

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

$$cost \equiv b_1 + b_2 + b_3$$

$$cost(\tau) = 1$$

$$cost(\mathcal{F}) = 1$$

Preliminaries: Maximum Satisfiability

- Optimisation extension of Boolean Satisfiability (SAT)
- An instance:
  - a set of hard clauses,
  - a linear objective function *cost*
- Find τ that:
  - satisfies all hard clauses and
  - minimizes cost
  - $cost(\mathcal{F}) =$ cost of its optimal solutions.
- $\mathcal{F}_B$: set of objective variables.
- **Note:** All results extend to weights!

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

$$cost \equiv \sum_{b \in \mathcal{F}_B} b$$

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

Redundant Clauses in MaxSAT

Clause Redundancy

In MaxSAT

C redundant for $\mathcal{F}$ if

$$cost(\mathcal{F}) = cost(\mathcal{F} \wedge C).$$

where $\mathcal{F} \wedge C = (\mathcal{F}_H \wedge C, \mathcal{F}_B)$

In SAT [Järvisalo, Heule, and Biere, 2012; Heule, Järvisalo, Lonsing, Seidl, and Biere, 2015]

C redundant for F if F and $F \wedge C$ are *equisatisfiable**.

* i.e. if F is satisfiable if and only if $F \wedge C$ is

Clause Redundancy

In MaxSAT

C redundant for $\mathcal{F}$ if

$$cost(\mathcal{F}) = cost(\mathcal{F} \wedge C).$$

where $\mathcal{F} \wedge C = (\mathcal{F}_H \wedge C, \mathcal{F}_B)$

In SAT [Järvisalo, Heule, and Biere, 2012; Heule, Järvisalo, Lonsing, Seidl, and Biere, 2015]

C redundant for F if F and $F \wedge C$ are *equisatisfiable**.

* i.e. if F is satisfiable if and only if $F \wedge C$ is

Note: that the definitions coincide if $\mathcal{F}_B = \emptyset$

Clause Redundancy via Semantic Implication

(Informal) Theorem for SAT [Heule, Kiesl, and Biere, 2020]

Assume $\tau(F) = 1$ and $\tau(C) = 0$.
C is redundant if ω for which $\omega(F \wedge C) = 1$ can be constructed.

Formally, if an ω for which $\omega(C) = 1$, and $\mathcal{F}_H\big|_{\neg C} \models \mathcal{F}_H\big|_{\omega}$ exists.

Clause Redundancy via Semantic Implication

(Informal) Theorem for SAT [Heule, Kiesl, and Biere, 2020]

Assume $\tau(F) = 1$ and $\tau(C) = 0$.
C is redundant if ω for which $\omega(F \wedge C) = 1$ can be constructed.

Our extension for MaxSAT

Assume $\tau(\mathcal{F}_H) = 1$ and $\tau(C) = 0$.
C is redundant an ω for which:

- $\omega(\mathcal{F}_H \wedge C) = 1$, and
- $cost(\omega) \leq cost(\tau)$

can be constructed.

Clause Redundancy via Semantic Implication

Our extension for MaxSAT

Assume $\tau(\mathcal{F}_H) = 1$ and $\tau(C) = 0$.
C is redundant if ω for which:

- $\omega(\mathcal{F}_H \wedge C) = 1$, and
- $cost(\omega) \leq cost(\tau)$

can be constructed.

More specifically C is:

- cost literal propagation redundant (CLPR) if
  ω flips a single $l \in C \setminus \mathcal{F}_B$.
- cost set propagation redundant (CSPR) if
  ω flips a subset $L \subset C \setminus \mathcal{F}_B$.
- cost propagation redundant (CPR) if
  ω is constructed based on a given third assignment δ.

Example

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

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

Example

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

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

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

$(\neg x \vee \neg b_1)$

1. $(\neg x \vee \neg b_1)$ is redundant,
2. $(\neg b_1)$ is redundant,
3. $(\neg b_2)$ is not redundant since the witness would have higher cost

Example

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

$$(\neg x \vee \neg b_1)$$

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

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

$$\omega(b_2) = \omega(x) = 1$$
$$\omega(b_1) = \omega(b_3) = \omega(y) = 0$$
$$cost(\omega) = 1 \leq 2 = cost(\tau)$$

1. $(\neg x \vee \neg b_1)$ is redundant, alter by $b_1 = 0$.
2. $(\neg b_1)$ is redundant,
3. $(\neg b_2)$ is not redundant since the witness would have higher cost

Example

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

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

$$(\neg b_1)$$

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

1. $(\neg x \vee \neg b_1)$ is redundant, alter by $b_1 = 0$.
2. $(\neg b_1)$ is redundant,
3. $(\neg b_2)$ is not redundant since the witness would have higher cost

Example

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

$$(\neg b_1)$$

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

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

$$\omega(b_2) = \omega(b_3) = \omega(y) = \omega(x) = 1$$
$$\omega(b_1) = 0$$

$$cost(\omega) = 2 \leq 2 = cost(\tau)$$

1. $(\neg x \vee \neg b_1)$ is redundant, alter by $b_1 = 0$.
2. $(\neg b_1)$ is redundant, alter by $b_1 = 0$ and $x = b_2 = 1$
3. $(\neg b_2)$ is not redundant since the witness would have higher cost

Example

$$\mathcal{F}_H = \{(b_1 \vee x), (\neg x \vee b_2), (b_2 \vee y), (\neg y \vee b_3)\} \qquad (\neg b_2)$$

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

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

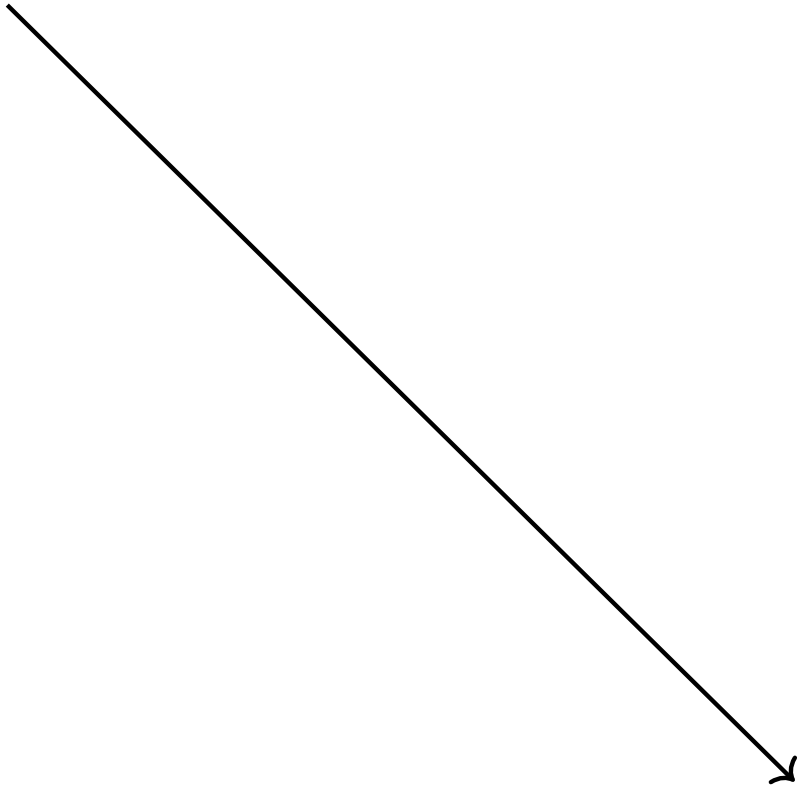
$$\omega(b_1) = \omega(b_3) = \omega(y) = 1$$
$$\omega(x) = \omega(b_2) = 0$$
$$cost(\omega) = 2 \geq 1 = cost(\tau)$$

1. $(\neg x \vee \neg b_1)$ is redundant, alter by $b_1 = 0$.
2. $(\neg b_1)$ is redundant, alter by $b_1 = 0$ and $x = b_2 = 1$
3. $(\neg b_2)$ is not redundant since the witness would have higher cost

Why does this matter?

Example: Bounded Variable Elimination

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



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

Why does this matter?

Example: Bounded Variable Elimination

$$\mathcal{F}_H = \{(b_1 \vee x), (\neg x \vee b_2), (b_2 \vee y), (\neg y, b_3)\} \longrightarrow \mathcal{F}_H^1 = \{(b_1 \vee x), (\neg x \vee b_2), (b_2 \vee y), (\neg y \vee b_3), (b_1 \vee b_2)\}$$

$$\mathcal{F}_H^2 = \{\cancel{(b_1 \vee x)}, (\neg x \vee b_2), (b_2 \vee y), (\neg y \vee b_3), (b_1 \vee b_2)\} \longrightarrow \mathcal{F}_H^3 = \{\cancel{(b_1 \vee x)}, \cancel{(\neg x \vee b_2)}, (b_2 \vee y), (\neg y \vee b_3), (b_1 \vee b_2)\}$$

$$\ldots \longrightarrow \mathcal{F}_H^P = \{(b_1 \vee b_2), (b_2 \vee b_3)\}$$

Effect of Redundant Clauses in MaxSAT

Definition: MCSes

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

$$cost \equiv b_1 + b_2 + b_3$$

- MCS: subset-minimal set of blocking variables set to 1 by some solution.
  - equivalently, a minimal set of soft clauses falsified by some solution.
- Tight connections between MCSes and MaxSAT solutions.
- MCS($\mathcal{F}$): set of all MCSes of $\mathcal{F}$.

Definition: MCSes

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

$cost \equiv b_1 + b_2 + b_3$

$hs^1 = \{b_1, b_3\}$ $(\equiv \{(\neg b_1), (\neg b_3)\})$

$\tau^1(b_1) = \tau^1(b_3) = \tau^1(y) = 1$

$\tau^1(b_2) = \tau^1(x) = 0$

- MCS: subset-minimal set of blocking variables set to 1 by some solution.
  - equivalently, a minimal set of soft clauses falsified by some solution.
- Tight connections between MCSes and MaxSAT solutions.
- MCS($\mathcal{F}$): set of all MCSes of $\mathcal{F}$.

Definition: MCSes

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

$$cost \equiv b_1 + b_2 + b_3$$

$hs^1 = \{b_1, b_3\}$ $(\equiv \{(\neg b_1), (\neg b_3)\})$

$\tau^1(b_1) = \tau^1(b_3) = \tau^1(y) = 1$

$\tau^1(b_2) = \tau^1(x) = 0$

$hs^2 = \{b_2\}$ $(\equiv \{(\neg b_2)\})$

$\tau(b_2) = \tau(x) = 1$

$\tau(b_1) = \tau(b_3) = \tau(y) = 0$

- MCS: subset-minimal set of blocking variables set to 1 by some solution.
  - equivalently, a minimal set of soft clauses falsified by some solution.
- Tight connections between MCSes and MaxSAT solutions.
- MCS($\mathcal{F}$): set of all MCSes of $\mathcal{F}$.

Definition: MCSes

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

$cost \equiv b_1 + b_2 + b_3$

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

$hs^1 = \{b_1, b_3\}$ $(\equiv \{(\neg b_1), (\neg b_3)\})$

$\tau^1(b_1) = \tau^1(b_3) = \tau^1(y) = 1$

$\tau^1(b_2) = \tau^1(x) = 0$

$hs^2 = \{b_2\}$ $(\equiv \{(\neg b_2)\})$

$\tau(b_2) = \tau(x) = 1$

$\tau(b_1) = \tau(b_3) = \tau(y) = 0$

- MCS: subset-minimal set of blocking variables set to 1 by some solution.
  - equivalently, a minimal set of soft clauses falsified by some solution.
- Tight connections between MCSes and MaxSAT solutions.
- $\text{MCS}(\mathcal{F})$: set of all MCSes of $\mathcal{F}$.

Effect of redundant Clauses on MCSes

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

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

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

- Adding CLPR clause $(\neg x \vee \neg b_1)$ does not affect MCSes.
- Adding CPR clause $(\neg b_1)$ does.

Effect of redundant Clauses on MCSes

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

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

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

$$\mathcal{F}_H^P = \{(b_1 \vee x), (\neg x \vee b_2), (b_2 \vee y), (\neg y \vee b_3), (\neg x \vee \neg b_1)\}$$

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

$$\text{MCS}(\mathcal{F}^P) = \{\{b_1, b_3\}, \{b_2\}\}$$

- Adding CLPR clause $(\neg x \vee \neg b_1)$ does not affect MCSes.
- Adding CPR clause $(\neg b_1)$ does.

Effect of redundant Clauses on MCSes

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

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

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

$$\mathcal{F}_H^P = \{(b_1 \vee x), (\neg x \vee b_2), (b_2 \vee y), (\neg y \vee b_3), (\neg b_1)\}$$

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

$$\text{MCS}(\mathcal{F}^P) = \{\{b_2\}\}$$

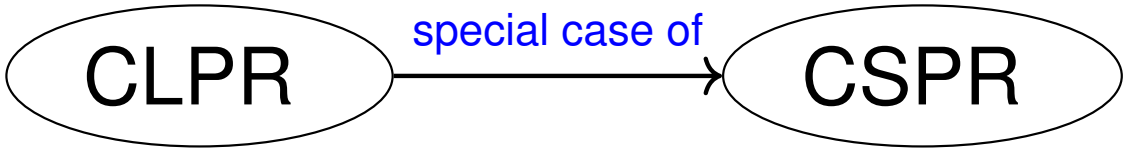
- Adding CLPR clause $(\neg x \vee \neg b_1)$ does not affect MCSes.
- Adding CPR clause $(\neg b_1)$ does.

Analysis of Redundant Clauses

Summary

Analysis of Redundant Clauses

Summary



Analysis of Redundant Clauses

Summary

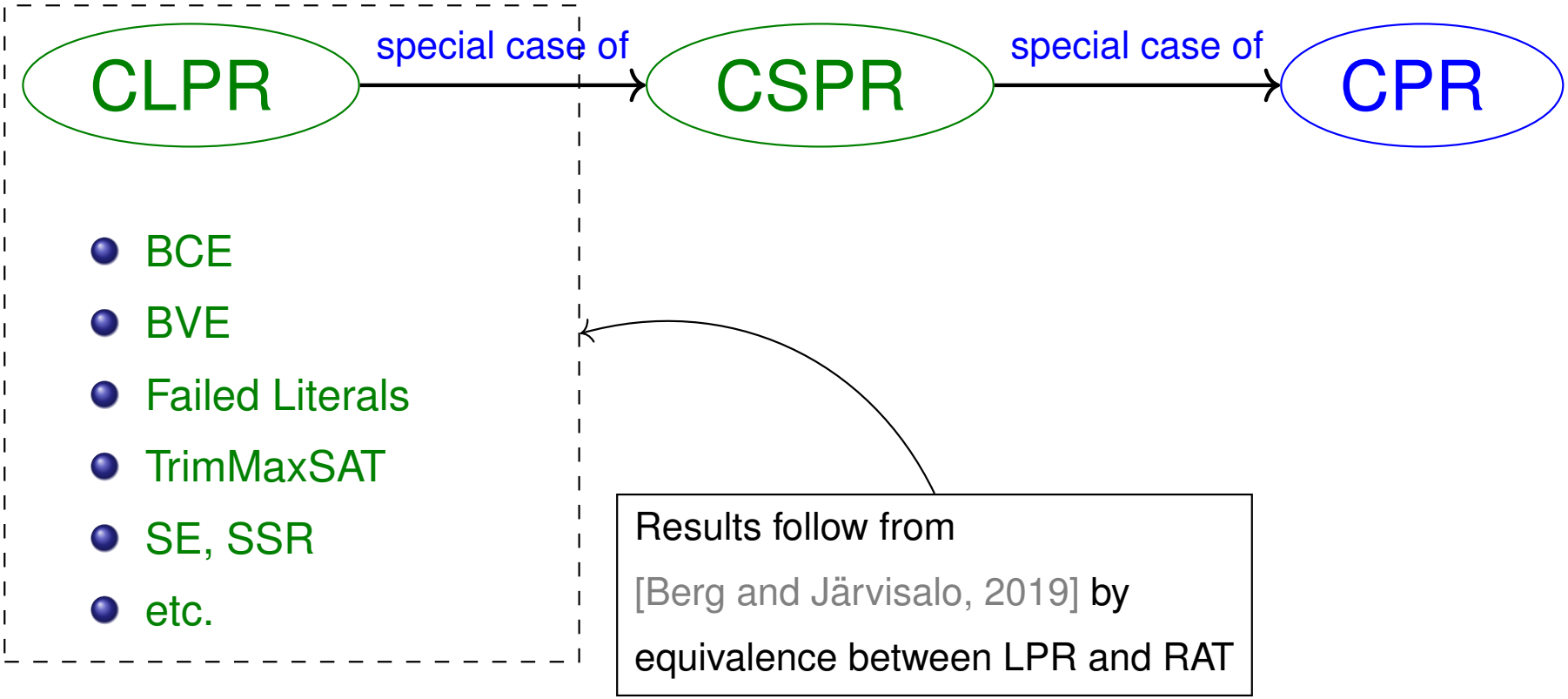
Analysis of Redundant Clauses

Summary

- Adding or removing preserves all MCSes
- Adding or removing is guaranteed to only preserve one minimum cost MCS

Analysis of Redundant Clauses

Summary



- Adding or removing preserves all MCSes
- Adding or removing is guaranteed to only preserve one minimum cost MCS

Analysis of Redundant Clauses

Summary

- BCE
- BVE
- Failed Literals
- TrimMaxSAT
- SE, SSR
- etc.

- Group Subsumed Label Elimination
- Hardening
- etc.

- Adding or removing preserves all MCSes
- Adding or removing is guaranteed to only preserve one minimum cost MCS

MaxPre 2 - Expressive preprocessing in practice

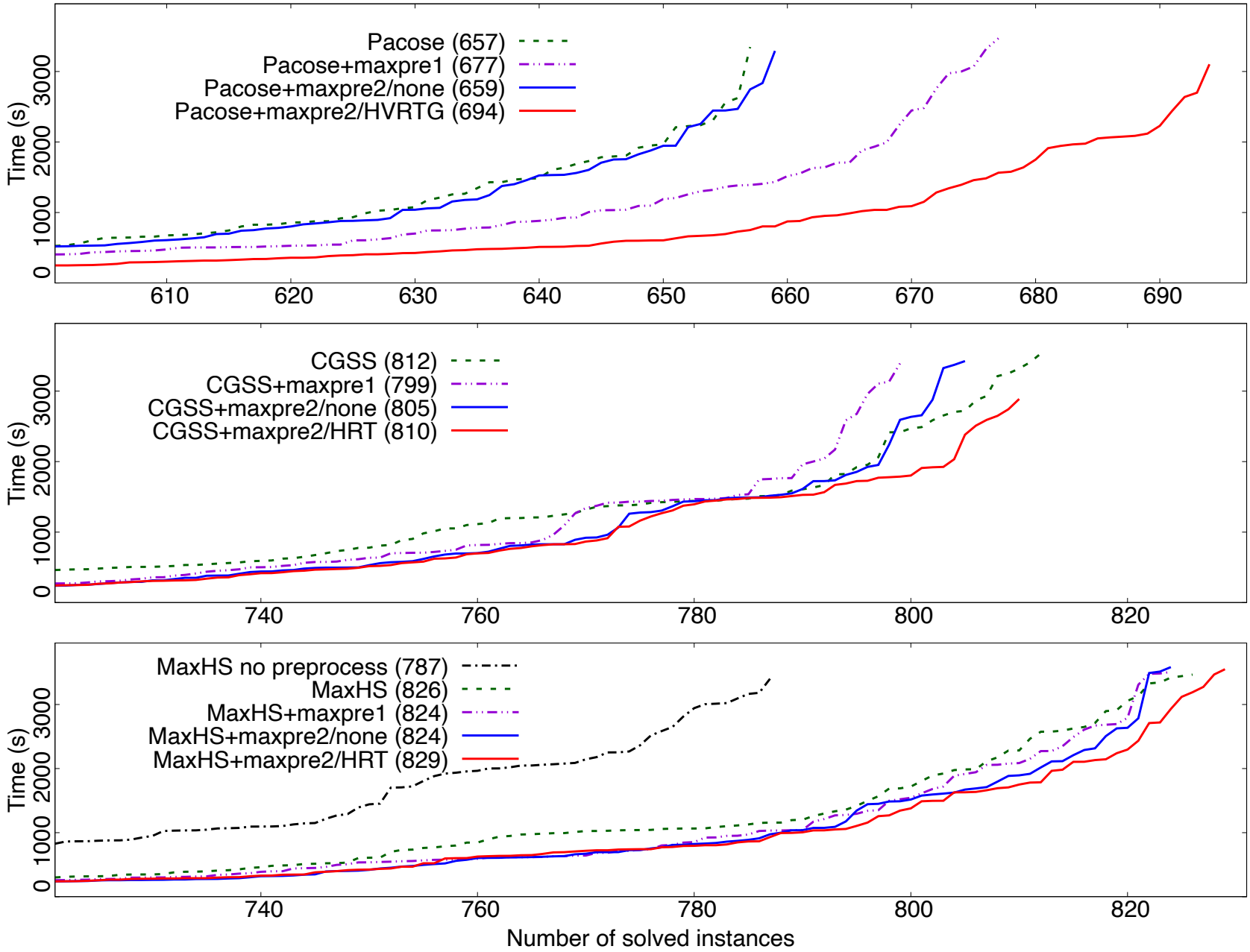
MaxPre 2

- Stand-alone tool implementing several heuristics previously only appearing in solvers.
- Allows applying heuristics as a separate solving step.
- Available in open source
  `https://bitbucket.org/coreo-group/maxpre2/`

Set Up of Experiments

- All distinct weighted benchmarks from MaxSAT Evaluations 2020 and 2021.
- Experiments with three state-of-the-art complete solvers representing the three types of algorithms:
  - Core guided: CGSS
  - IHS-based: MaxHS
  - Model improving: Pacose
- Also experiments with incomplete Loandra

Complete Solving



Incomplete Solving

#Wins	base (maxpre1)	no-prepro	maxpre2/ none	maxpre2/ VG
base (maxpre1)	—			
no-prepro		—		
maxpre2/none			—	
maxpre2/VG				—
Score:	0.852	0.840	0.863	0.870

- Results for the state-of-the-art Loandra solver.
- Wins = #instances on which found solution has lower cost.
- Score = average ratio between best-known cost and cost of found solution.

Incomplete Solving

#Wins	base (maxpre1)	no-prepro	maxpre2/ none	maxpre2/ VG
base (maxpre1)	—		135	
no-prepro		—		
maxpre2/none	105		—	
maxpre2/VG				—
Score:	0.852	0.840	0.863	0.870

- Results for the state-of-the-art Loandra solver.
- Wins = #instances on which found solution has lower cost.
- Score = average ratio between best-known cost and cost of found solution.

Incomplete Solving

#Wins	base (maxpre1)	no-prepro	maxpre2/ none	maxpre2/ VG
base (maxpre1)	—		135	152
no-prepro		—		
maxpre2/none	105		—	
maxpre2/VG	110			—
Score:	0.852	0.840	0.863	0.870

- Results for the state-of-the-art Loandra solver.
- Wins = #instances on which found solution has lower cost.
- Score = average ratio between best-known cost and cost of found solution.

Incomplete Solving

#Wins	base (maxpre1)	no-prepro	maxpre2/ none	maxpre2/ VG
base (maxpre1)	—	154	135	152
no-prepro	208	—	216	218
maxpre2/none	105	143	—	77
maxpre2/VG	110	140	80	—
Score:	0.852	0.840	0.863	0.870

- Results for the state-of-the-art Loandra solver.
- Wins = #instances on which found solution has lower cost.
- Score = average ratio between best-known cost and cost of found solution.

Conclusions

- Lift recently proposed clause redundancy notions from SAT to MaxSAT
- Complete analysis on the effect of redundant clauses on MCSes
- Provide empirical evaluation on the effect of preprocessing on state-of-the-art MaxSAT solvers.

Future Work

- Develop new inference rules based on the redundancy notions.
  - Especially rules on the ”CPR level”.
- Further extend framework to include e.g. core-guided reasoning

Bibliography I

Carlos Ansótegui, Maria Luisa Bonet, Joel Gabàs, and Jordi Levy. Improving SAT-based weighted MaxSAT solvers. In *Proc. CP*, volume 7514 of *Lecture Notes in Computer Science*, pages 86–101. Springer, 2012.

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

Jeremias Berg and Matti Järvisalo. Unifying reasoning and core-guided search for maximum satisfiability. In *JELIA*, volume 11468 of *Lecture Notes in Computer Science*, pages 287–303. Springer, 2019.

Jeremias Berg, Emir Demirovic, and Peter J. Stuckey. Core-boosted linear search for incomplete MaxSAT. In *Proc. CPAIOR*, volume 11494 of *Lecture Notes in Computer Science*, pages 39–56. Springer, 2019.

Armin Biere, Matti Järvisalo, and Benjamin Kiesl. Preprocessing in sat solving. In Armin Biere, Marijn Heule, Hans van Maaren, and Toby Walsh, editors, *Handbook of Satisfiability*, Frontiers in Artificial Intelligence and Applications, chapter 9, pages 391–435. IOS Press, 2021.

Luís Cruz-Filipe, Marijn J. H. Heule, Warren A. Hunt Jr., Matt Kaufmann, and Peter Schneider-Kamp. Efficient certified RAT verification. In *Proc. CADE*, volume 10395 of *Lecture Notes in Computer Science*, pages 220–236. Springer, 2017.

J. Davies and F. Bacchus. Exploiting the power of MIP solvers in MaxSAT. In *Proc. SAT*, volume 7962 of *Lecture Notes in Computer Science*, pages 166–181. Springer, 2013.

Marijn Heule and Benjamin Kiesl. The potential of interference-based proof systems. In *Proc. ARCADE@CADE*, volume 51 of *EPiC Series in Computing*, pages 51–54. EasyChair, 2017.

Marijn Heule, Matti Järvisalo, Florian Lonsing, Martina Seidl, and Armin Biere. Clause Elimination for SAT and QSAT. *Journal of Artificial Intelligence Research*, 53:127–168, 2015.

Marijn J. H. Heule, Benjamin Kiesl, Martina Seidl, and Armin Biere. PRuning through satisfaction. In Ofer Strichman and Rachel Tzoref-Brill, editors, *Proc. HVC*, volume 10629 of *Lecture Notes in Computer Science*, pages 179–194. Springer, 2017.

Marijn J. H. Heule, Benjamin Kiesl, and Armin Biere. Clausal proofs of mutilated chessboards. In *NFM*, volume 11460 of *Lecture Notes in Computer Science*, pages 204–210. Springer, 2019.

Marijn J. H. Heule, Benjamin Kiesl, and Armin Biere. Strong extension-free proof systems. *J. Autom. Reasoning*, 64(3):533–554, 2020.

Bibliography II

Alexey Ignatiev, António Morgado, and João Marques-Silva. RC2: an efficient MaxSAT solver. *J. Satisf. Boolean Model. Comput.*, 11(1):53–64, 2019.

Matti Järvisalo, Armin Biere, and Marijn Heule. Simulating circuit-level simplifications on CNF. *J. Autom. Reason.*, 49(4):583–619, 2012.

Matti Järvisalo, Marijn Heule, and Armin Biere. Inprocessing rules. In *Proc. IJCAR*, volume 7364 of *Lecture Notes in Computer Science*, pages 355–370. Springer, 2012.

Tuukka Korhonen, Jeremias Berg, Paul Saikko, and Matti Järvisalo. MaxPre: An extended MaxSAT preprocessor. In *Proc. SAT*, volume 10491 of *Lecture Notes in Computer Science*, pages 449–456, 2017.

M. Koshimura, T. Zhang, H. Fujita, and R. Hasegawa. QMaxSAT: A Partial Max-SAT Solver. *Journal of Satisfiability, Boolean Modeling and Computation*, 8(1/2):95–100, 2012.

João Marques-Silva and Jordi Planes. Algorithms for Maximum Satisfiability using Unsatisfiable Cores. In *Proc. DATE*, pages 408–413, 2008.

R. Martins, S. Joshi, V.M. Manquinho, and I. Lynce. Incremental Cardinality Constraints for MaxSAT. In *Proc. CP*, volume 8656 of *Lecture Notes in Computer Science*, pages 531–548. Springer, 2014a.

R. Martins, V.M. Manquinho, and I. Lynce. Open-WBO: A modular MaxSAT solver. In *Proc. SAT*, volume 8561 of *Lecture Notes in Computer Science*, pages 438–445. Springer, 2014b.

A. Morgado, C. Dodaro, and J. Marques-Silva. Core-Guided MaxSAT with Soft Cardinality Constraints. In *Proc. CP*, volume 8656 of *Lecture Notes in Computer Science*, pages 564–573. Springer, 2014.

Tobias Paxian, Pascal Raiola, and Bernd Becker. On preprocessing for weighted maxsat. In *Proc. VMCAI*, volume 12597 of *Lecture Notes in Computer Science*, pages 556–577. Springer, 2021.

P. Saikko, J. Berg, and M. Järvisalo. LMHS: A SAT-IP hybrid MaxSAT solver. In *Proc. SAT*, volume 9710 of *Lecture Notes in Computer Science*, pages 539–546. Springer, 2016.