DRMaxSAT with MaxHS: First Contact

A.Morgado[1] A.Ignatiev[1,4] M.L.Bonet[2] J.Marques-Silva[1]
S.Buss[3]

[1] Faculty of Science,University of Lisbon, Portugal

[2] Computer Science, Universidad Politécnica de Cataluna, Barcelona, Spain

[3] Department of Mathematics, University of California, San Diego, USA

[4] ISDCT SB RAS, Irkutsk, Russia

SAT 2019

Motivation

- Success of Conflict-Driven Clause Learning (CDCL) demonstrates the reach of the Resolution proof system

Motivation

- Success of Conflict-Driven Clause Learning (CDCL) demonstrates the reach of the Resolution proof system
- From proof complexity point of view, Resolution is regarded as a rather weak proof system

Motivation

- Success of Conflict-Driven Clause Learning (CDCL) demonstrates the reach of the Resolution proof system
- From proof complexity point of view, Resolution is regarded as a rather weak proof system
- Recent efforts for developing efficient implementations of stronger proof systems:
  - Extended Resolution (ExtRes)
  - DRAT
  - Cutting Planes (CP)
  - Dual-Rail Maximum Satisfiability (DRMaxSAT)

DRMaxSAT - General Idea

- Translates a CNF formula $\mathcal{F}$ using the Dual-Rail Encoding
- Uses a MaxSAT algorithm to obtain the cost of the encoded formula
- Determines the satisfiability of $\mathcal{F}$ based on the cost of the encoded formula

DRMaxSAT - Previous work 1/3

Weighted DRMaxSAT simulates general resolution.

DRMaxSAT - Previous work 1/3

Weighted DRMaxSAT simulates general resolution.

DRMaxSAT refutes in polynomial time:

- Pigeonhole Principle (PHP)
- Doubled Pigeonhole principle (2PHP)

DRMaxSAT - Previous work 1/3

Weighted DRMaxSAT simulates general resolution.

DRMaxSAT refutes in polynomial time:

- Pigeonhole Principle (PHP)
- Doubled Pigeonhole principle (2PHP)

MaxSAT algorithms based on:

- MaxSAT Resolution [AAAI18]
- Core-Guided MaxSAT Algorithms [SAT17]

DRMaxSAT - Previous work 1/3

Weighted DRMaxSAT simulates general resolution.

DRMaxSAT refutes in polynomial time:

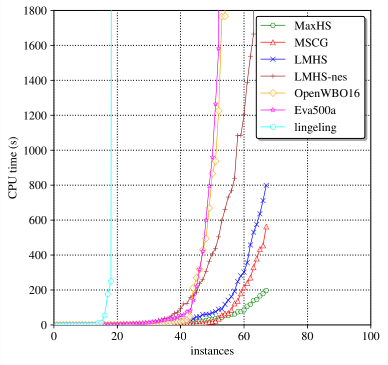
- Pigeonhole Principle (PHP)
- Doubled Pigeonhole principle (2PHP)

MaxSAT algorithms based on:

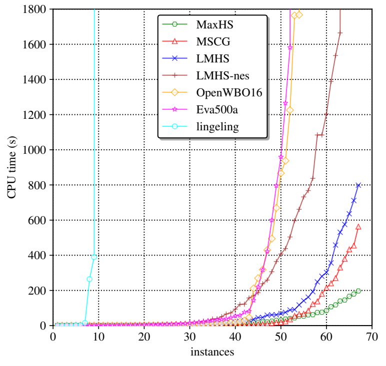
- MaxSAT Resolution [AAAI18]
- Core-Guided MaxSAT Algorithms [SAT17]

- Hitting Set-based MaxSAT Approach: MaxHS-like Algorithm [SAT19]

DRMaxSAT - Previous work 2/3



PHP



2PHP

Outline

Basic MaxHS Algorithm

DRMaxSAT

DRMaxSAT/basic MaxHS vs Pigeonhole Principle

DRMaxSAT/basic MaxHS vs Doubled Pigeonhole Principle

Conclusions

Outline

Basic MaxHS Algorithm

DRMaxSAT

DRMaxSAT/basic MaxHS vs Pigeonhole Principle

DRMaxSAT/basic MaxHS vs Doubled Pigeonhole Principle

Conclusions

SAT

- $\mathcal{F}$ - CNF formula: conjunction of clauses
- Clause: disjunction of literals
- Literal: variable x or its completement $\neg x$
- $\mathcal{A}$ - Assignment: mapping from variables to $\{0, 1\}$

- SAT problem: Given $\mathcal{F}$ determine if there is an assignment $\mathcal{A}$ for $\mathcal{F}$ that satisfies all its clauses, otherwise $\mathcal{F}$ is unsatisfiable.

MaxSAT

- Partial MaxSAT problem $< \mathcal{H}, \mathcal{S} >$:
  - $\mathcal{H}$ - set of hard clauses
  - $\mathcal{S}$ - set of soft clauses

  Goal: Find an assignment $\mathcal{A}$ that satisfies all clauses in $\mathcal{H}$ and maximizes the number of satisfied clauses in $\mathcal{S}$

MaxSAT

- Partial MaxSAT problem $< \mathcal{H}, \mathcal{S} >$:
  - $\mathcal{H}$ - set of hard clauses
  - $\mathcal{S}$ - set of soft clauses

  Goal: Find an assignment $\mathcal{A}$ that satisfies all clauses in $\mathcal{H}$ and maximizes the number of satisfied clauses in $\mathcal{S}$

- Cost of an assignment: number of unsatisfied clauses in $\mathcal{S}$

Basic MaxHS-like Algorithm

- MaxHS is a relatively recent MaxSAT approach:
  - based on the hitting set duality between MCSes and MUSes
  - results in simpler oracle calls, but at the cost of possibly exponentially larger number of calls

Basic MaxHS-like Algorithm

- MaxHS is a relatively recent MaxSAT approach:
  - based on the hitting set duality between MCSes and MUSes
  - results in simpler oracle calls, but at the cost of possibly exponentially larger number of calls

Input : $< \mathcal{H}, \mathcal{S} >$ WCNF formula

$K \leftarrow \emptyset$

while *true* **do**

  $h \leftarrow \text{MinimumHS}(K)$

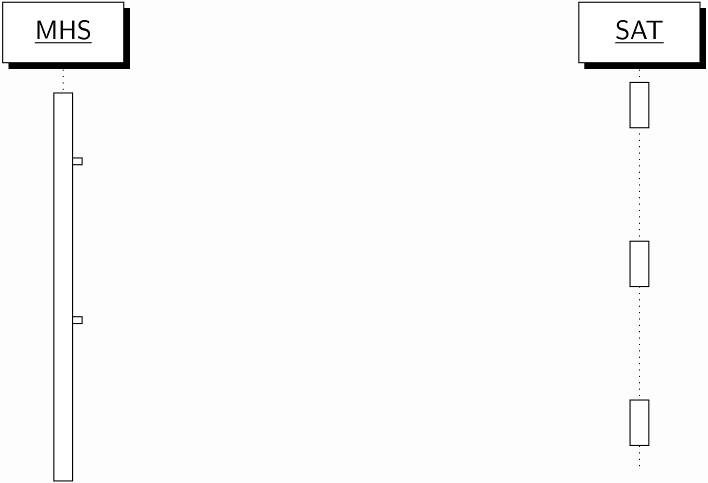
  $(st, \mu) \leftarrow \text{SAT}(\mathcal{H} \cup \mathcal{S} \setminus h)$

  if *st* **then** **return** μ

  $K \leftarrow K \cup \{\mu\}$

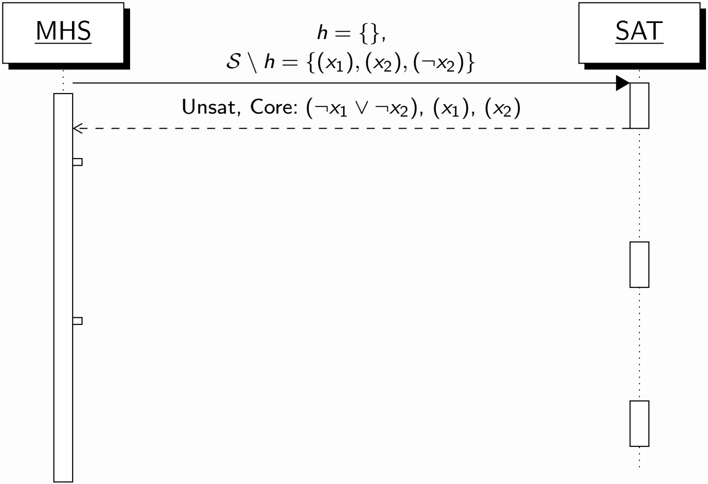
Example Basic MaxHS-like Algorithm

$\mathcal{H} = \{(\neg x_1 \vee \neg x_2)\}$ $\quad$ $\mathcal{S} = \{s_1 : (x_1), s_2 : (x_2), s_3 : (\neg x_2)\}$



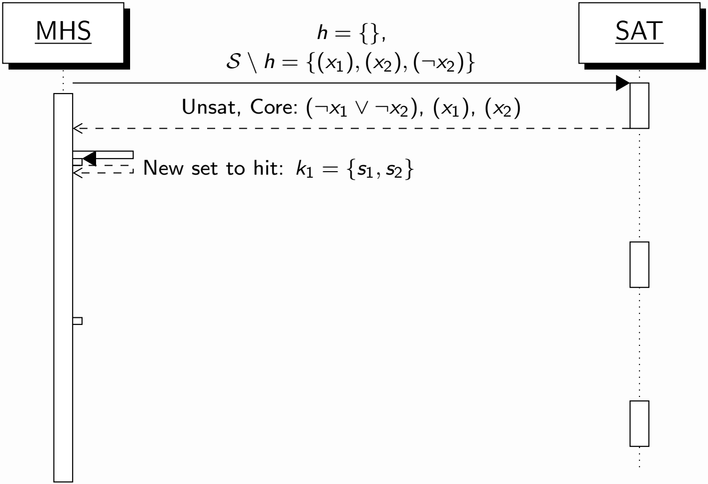
Example Basic MaxHS-like Algorithm

$$\mathcal{H} = \{(\neg x_1 \vee \neg x_2)\} \qquad \mathcal{S} = \{s_1 : (x_1), s_2 : (x_2), s_3 : (\neg x_2)\}$$



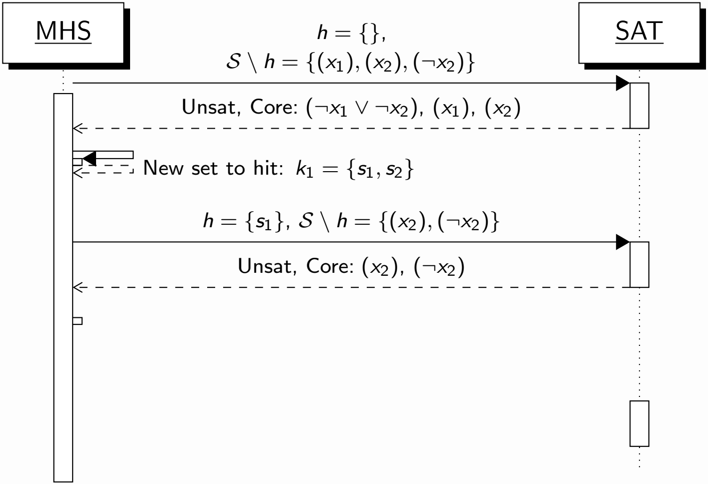
Example Basic MaxHS-like Algorithm

$$\mathcal{H} = \{(\neg x_1 \vee \neg x_2)\} \qquad \mathcal{S} = \{s_1 : (x_1), s_2 : (x_2), s_3 : (\neg x_2)\}$$



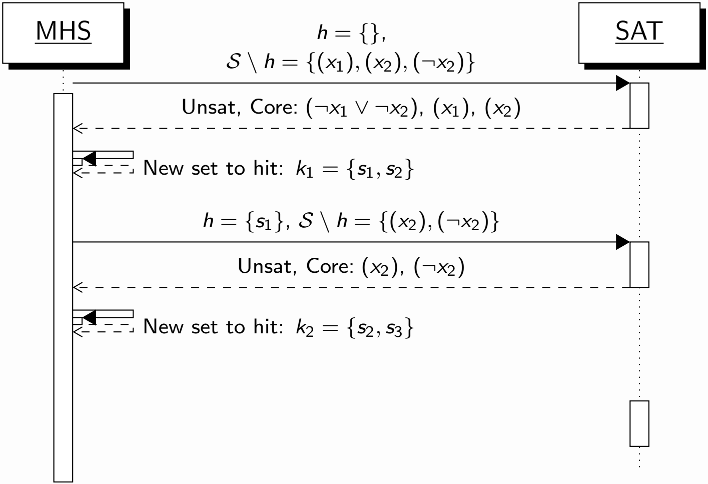
Example Basic MaxHS-like Algorithm

$\mathcal{H} = \{(\neg x_1 \vee \neg x_2)\} \qquad \mathcal{S} = \{s_1 : (x_1), s_2 : (x_2), s_3 : (\neg x_2)\}$



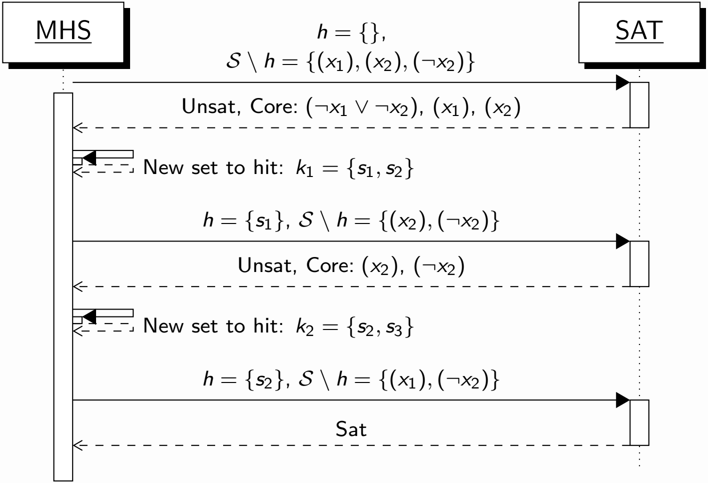
Example Basic MaxHS-like Algorithm

$$\mathcal{H} = \{(\neg x_1 \vee \neg x_2)\} \qquad \mathcal{S} = \{s_1 : (x_1), s_2 : (x_2), s_3 : (\neg x_2)\}$$



Example Basic MaxHS-like Algorithm

$$\mathcal{H} = \{(\neg x_1 \vee \neg x_2)\} \qquad \mathcal{S} = \{s_1 : (x_1), s_2 : (x_2), s_3 : (\neg x_2)\}$$



Outline

Basic MaxHS Algorithm

DRMaxSAT

DRMaxSAT/basic MaxHS vs Pigeonhole Principle

DRMaxSAT/basic MaxHS vs Doubled Pigeonhole Principle

Conclusions

DRMaxSAT: DRE

Dual-Rail Encoding (DRE)

[DAC87, AI99]

Input: $\mathcal{F}$ CNF formula with N variables $X = \{x_1, \ldots, x_N\}$

DRMaxSAT: DRE

Dual-Rail Encoding (DRE)

[DAC87, AI99]

Input: $\mathcal{F}$ CNF formula with N variables $X = \{x_1, \ldots, x_N\}$

Output: MaxSAT problem $< \mathcal{H}, \mathcal{S} >$:

DRMaxSAT: DRE

Dual-Rail Encoding (DRE)

[DAC87, AI99]

Input: $\mathcal{F}$ CNF formula with N variables $X = \{x_1, \ldots, x_N\}$

Output: MaxSAT problem $< \mathcal{H}, \mathcal{S} >$:

- for each $x_i \in X$:
  - associate new variables p_i and n_i

$$x_i = 1 \text{ iff } p_i = 1, \text{ and } x_i = 0 \text{ iff } n_i = 1$$

  - add to $\mathcal{S}$ the clauses (p_i) and (n_i)
  - add to $\mathcal{H}$ the clause $(\neg p_i \vee \neg n_i)$ ($\mathcal{P}$ clauses)

DRMaxSAT: DRE

Dual-Rail Encoding (DRE)

[DAC87, AI99]

Input: $\mathcal{F}$ CNF formula with N variables $X = \{x_1, \ldots, x_N\}$

Output: MaxSAT problem $< \mathcal{H}, \mathcal{S} >$:

- for each $x_i \in X$:
  - associate new variables p_i and n_i

$$x_i = 1 \text{ iff } p_i = 1, \text{ and } x_i = 0 \text{ iff } n_i = 1$$

  - add to $\mathcal{S}$ the clauses (p_i) and (n_i)
  - add to $\mathcal{H}$ the clause $(\neg p_i \vee \neg n_i)$ ($\mathcal{P}$ clauses)

- for each clause $c \in \mathcal{F}$ add to $\mathcal{H}$ the clause c':
  - if $x_i \in c$ then $\neg n_i \in c'$
  - if $\neg x_i \in c$ then $\neg p_i \in c'$

DRMaxSAT: DRE Example

$\mathcal{F} = \{(\neg x_1 \vee \neg x_2), (x_1), (x_2), (\neg x_2)\}$

DRMaxSAT: DRE Example

$\mathcal{F} = \{(\neg x_1 \vee \neg x_2), (x_1), (x_2), (\neg x_2)\}$

- MaxSAT problem $< \mathcal{H}, \mathcal{S} >$

DRMaxSAT: DRE Example

$\mathcal{F} = \{(\neg x_1 \vee \neg x_2), (x_1), (x_2), (\neg x_2)\}$

- MaxSAT problem $< \mathcal{H}, \mathcal{S} >$
- for x_1:
  - create p_1 and n_1
  - add (p_1), (n_1) to $\mathcal{S}$
  - add $(\neg p_1 \vee \neg n_1)$ to $\mathcal{H}$
- for x_2:
  - create p_2 and n_2
  - add (p_2), (n_2) to $\mathcal{S}$
  - add $(\neg p_2 \vee \neg n_2)$ to $\mathcal{H}$

DRMaxSAT: DRE Example

$\mathcal{F} = \{(\neg x_1 \vee \neg x_2), (x_1), (x_2), (\neg x_2)\}$

- MaxSAT problem $< \mathcal{H}, \mathcal{S} >$
- for $(\neg x_1 \vee \neg x_2)$:
  - add $(\neg p_1 \vee \neg p_2)$ to $\mathcal{H}$
- for (x_1), (x_2), $(\neg x_2)$:
  - add $(\neg n_1)$, $(\neg n_2)$, $(\neg p_2)$ to $\mathcal{H}$

DRMaxSAT: DRE Example

$\mathcal{F} = \{(\neg x_1 \lor \neg x_2), (x_1), (x_2), (\neg x_2)\}$

MaxSAT problem $< \mathcal{H}, \mathcal{S} >$

DRMaxSAT: DRE Example

$\mathcal{F} = \{(\neg x_1 \vee \neg x_2), (x_1), (x_2), (\neg x_2)\}$

MaxSAT problem $< \mathcal{H}, \mathcal{S} >$:

$$\mathcal{S} = \{(p_1), (n_1), (p_2), (n_1)\}$$

$$\begin{aligned}\mathcal{H} = \{&(\neg p_1 \vee \neg n_1), (\neg p_2 \vee \neg n_2),\\ &(\neg p_1 \vee \neg p_2),\\ &(\neg n_1), (\neg n_2), (\neg p_2)\}\end{aligned}$$

DRMaxSAT: DRE Example

$\mathcal{F} = \{(\neg x_1 \vee \neg x_2), (x_1), (x_2), (\neg x_2)\}$

MaxSAT problem $< \mathcal{H}, \mathcal{S} >$:

$$\mathcal{S} = \{(p_1), (n_1), (p_2), (n_1)\}$$

$$\begin{aligned}\mathcal{H} = \{&(\neg p_1 \vee \neg n_1), (\neg p_2 \vee \neg n_2),\\ &(\neg p_1 \vee \neg p_2),\\ &(\neg n_1), (\neg n_2), (\neg p_2)\}\end{aligned}$$

MaxSAT Cost: 3

DRMaxSAT

Theorem

$\mathcal{F}$ is satisfiable iff there is a truth assignment satisfying $\mathcal{H}$ that satisfies at least N clauses in $\mathcal{S}$. [SAT17]

DRMaxSAT

Theorem

$\mathcal{F}$ is satisfiable iff there is a truth assignment satisfying $\mathcal{H}$ that satisfies at least N clauses in $\mathcal{S}$. *[SAT17]*

Example: $N = 2$ and MaxSAT cost 3, thus $\mathcal{F}$ is unsatisfiable.

Outline

Basic MaxHS Algorithm

DRMaxSAT

DRMaxSAT/basic MaxHS vs Pigeonhole Principle

DRMaxSAT/basic MaxHS vs Doubled Pigeonhole Principle

Conclusions

Pigeonhole Principle

Pigeonhole Principle: If $m + 1$ pigeons are distributed by m holes, then at least one hole contains more than one pigeon.

Pigeonhole Principle

Pigeonhole Principle: If $m+1$ pigeons are distributed by m holes, then at least one hole contains more than one pigeon.

Propositonal encoding of PHP_m^{m+1}

- Variables: x_{ij}, $i \in [m+1]$, $j \in [m]$

$$x_{ij} = 1 \text{ iff pigeon } i \text{ is place in hole } j$$

$N = (m+1)m$

Pigeonhole Principle

Pigeonhole Principle: If $m+1$ pigeons are distributed by m holes, then at least one hole contains more than one pigeon.

Propositonal encoding of PHP_m^{m+1}

- Variables: x_{ij}, $i \in [m+1]$, $j \in [m]$

$$x_{ij} = 1 \text{ iff pigeon } i \text{ is place in hole } j$$

  $N = (m+1)m$

- Contraints:

$$\bigwedge_{i=1}^{m+1} (x_{i1} \vee \ldots \vee x_{im})$$

$$\bigwedge_{j=1}^{m} \quad \bigwedge_{i_1=1}^{m} \bigwedge_{i_2=1}^{m+1} (\neg x_{i_1 j} \vee \neg x_{i_2 j})$$

DRE of Pigeonhole Principle

DRE(PHP_m^{m+1}):

- for each x_{ij}, $i \in [m+1]$, $j \in [m]$:
  - associate variables p_{ij} and n_{ij}
  - add soft clauses (p_{ij}), (n_{ij})
  - add hard $\mathcal{P}$-clause $(\neg p_{ij} \vee \neg n_{ij})$

DRE of Pigeonhole Principle

DRE(PHP_m^{m+1}):

- for each x_{ij}, $i \in [m+1]$, $j \in [m]$:
  - associate variables p_{ij} and n_{ij}
  - add soft clauses (p_{ij}), (n_{ij})
  - add hard $\mathcal{P}$-clause $(\neg p_{ij} \vee \neg n_{ij})$
- for each $i \in [m+1]$:

$$\mathcal{L}_i = (\neg n_{i1} \vee \ldots \vee \neg n_{im})$$

DRE of Pigeonhole Principle

DRE(PHP_m^{m+1}):

- for each x_{ij}, $i \in [m+1]$, $j \in [m]$:
  - associate variables p_{ij} and n_{ij}
  - add soft clauses (p_{ij}), (n_{ij})
  - add hard $\mathcal{P}$-clause $(\neg p_{ij} \vee \neg n_{ij})$
- for each $i \in [m+1]$:

$$\mathcal{L}_i = (\neg n_{i1} \vee \ldots \vee \neg n_{im})$$

- for each $j \in [m]$:

$$\mathcal{M}_j = \bigwedge_{i_1=1}^{m} \bigwedge_{i_2=1}^{m+1} (\neg p_{i_1 j} \vee \neg p_{i_2 j})$$

DRE of Pigeonhole Principle

DRE(PHP_m^{m+1}):

- for each x_{ij}, $i \in [m+1]$, $j \in [m]$:
  - associate variables p_{ij} and n_{ij}
  - add soft clauses (p_{ij}), (n_{ij})
  - add hard $\mathcal{P}$-clause $(\neg p_{ij} \vee \neg n_{ij})$
- for each $i \in [m+1]$:

$$\mathcal{L}_i = (\neg n_{i1} \vee \ldots \vee \neg n_{im})$$

- for each $j \in [m]$:

$$\mathcal{M}_j = \bigwedge_{i_1=1}^{m} \bigwedge_{i_2=1}^{m+1} (\neg p_{i_1 j} \vee \neg p_{i_2 j})$$

PHP_m^{m+1} unsatisfiable if cost $\geq N + 1 = (m+1)m + 1$

Computing MaxSAT cost of DRE(PHP_m^{m+1})

Proposition

Given $< \mathcal{L}_i, \mathcal{S} >$ there is an execution of the basic MaxHS algorithm that computes a MaxSAT cost of 1 in polynomial time.

Computing MaxSAT cost of DRE(PHP_m^{m+1})

Proposition

Given $< \mathcal{L}_i, \mathcal{S} >$ there is an execution of the basic MaxHS algorithm that computes a MaxSAT cost of 1 in polynomial time.

Proof Idea:

- Core obtained by unit propagation
- Only one set to hit
- cost is 1

Computing MaxSAT cost of DRE(PHP_m^{m+1})

Proposition

Given $< \mathcal{M}_j, \mathcal{S} >$ there is an execution of the basic MaxHS algorithm that computes a MaxSAT cost of m in polynomial time.

Computing MaxSAT cost of DRE(PHP_m^{m+1})

Proposition

Given $<\mathcal{M}_j, \mathcal{S}>$ there is an execution of the basic MaxHS algorithm that computes a MaxSAT cost of m in polynomial time.

Proof Idea:

- Order the clauses in the SAT solver (via an ordering of the variables)

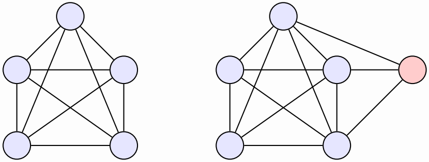
Computing MaxSAT cost of DRE(PHP_m^{m+1})

Proposition

Given $< \mathcal{M}_j, \mathcal{S} >$ there is an execution of the basic MaxHS algorithm that computes a MaxSAT cost of m in polynomial time.

Proof Idea:

- Order the clauses in the SAT solver (via an ordering of the variables)
- Cores induce sets to hit in MHS that correspond to a known graph (a clique or a clique plus one extra vertex)



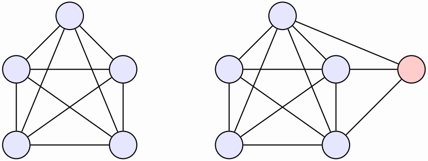
Computing MaxSAT cost of DRE(PHP_m^{m+1})

Proposition

Given $< \mathcal{M}_j, \mathcal{S} >$ *there is an execution of the basic MaxHS algorithm that computes a MaxSAT cost of* m *in polynomial time.*

Proof Idea:

- Order the clauses in the SAT solver (via an ordering of the variables)
- Cores induce sets to hit in MHS that correspond to a known graph (a clique or a clique plus one extra vertex)



- Last iteration corresponds to a clique of size $m + 1$, with minimum hitting set of size m (cost)

Outline

Basic MaxHS Algorithm

DRMaxSAT

DRMaxSAT/basic MaxHS vs Pigeonhole Principle

DRMaxSAT/basic MaxHS vs Doubled Pigeonhole Principle

Conclusions

Doubled Pigeonhole Principle

Doubled Pigeonhole Principle: If $2m + 1$ pigeons are distributed by m holes, then at least one hole contains more than two pigeons.

Doubled Pigeonhole Principle

Doubled Pigeonhole Principle: If $2m+1$ pigeons are distributed by m holes, then at least one hole contains more than two pigeons.

Propositonal encoding of $2PHP_m^{2m+1}$

- Variables: x_{ij}, $i \in [2m+1]$, $j \in [m]$

$$x_{ij} = 1 \text{ iff pigeon } i \text{ is place in hole } j$$

$N = (2m+1)m$

Doubled Pigeonhole Principle

Doubled Pigeonhole Principle: If $2m + 1$ pigeons are distributed by m holes, then at least one hole contains more than two pigeons.

Propositonal encoding of $2PHP_m^{2m+1}$

- Variables: x_{ij}, $i \in [2m + 1]$, $j \in [m]$

$$x_{ij} = 1 \text{ iff pigeon } i \text{ is place in hole } j$$

  $N = (2m + 1)m$

- Contraints:

$$\bigwedge_{i=1}^{2m+1} (x_{i1} \vee \ldots \vee x_{im})$$

$$\bigwedge_{j=1}^{m} \bigwedge_{i_1=1}^{2m-1} \bigwedge_{i_2=1}^{2m} \bigwedge_{i_3=3}^{2m+1} (\neg x_{i_1 j} \vee \neg x_{i_2 j} \vee \neg x_{i_3 j})$$

DRE of Doubled Pigeonhole Principle

DRE($2PHP_m^{2m+1}$):

- for each x_{ij}, $i \in [2m+1]$, $j \in [m]$:
  - associate variables p_{ij} and n_{ij}
  - add soft clauses (p_{ij}), (n_{ij})
  - add hard $\mathcal{P}$-clause $(\neg p_{ij} \vee \neg n_{ij})$

DRE of Doubled Pigeonhole Principle

DRE($2PHP_m^{2m+1}$):

- for each x_{ij}, $i \in [2m+1]$, $j \in [m]$:
  - associate variables p_{ij} and n_{ij}
  - add soft clauses (p_{ij}), (n_{ij})
  - add hard $\mathcal{P}$-clause $(\neg p_{ij} \vee \neg n_{ij})$
- for each $i \in [2m+1]$:

$$\mathcal{L}_i = (\neg n_{i1} \vee \ldots \vee \neg n_{im})$$

DRE of Doubled Pigeonhole Principle

DRE($2PHP_m^{2m+1}$):

- for each x_{ij}, $i \in [2m+1]$, $j \in [m]$:
  - associate variables p_{ij} and n_{ij}
  - add soft clauses (p_{ij}), (n_{ij})
  - add hard $\mathcal{P}$-clause $(\neg p_{ij} \vee \neg n_{ij})$
- for each $i \in [2m+1]$:

$$\mathcal{L}_i = (\neg n_{i1} \vee \ldots \vee \neg n_{im})$$

- for each $j \in [m]$:

$$\mathcal{M}_j = \bigwedge_{i_1=1}^{2m-1} \bigwedge_{i_2=1}^{2m} \bigwedge_{i_3=3}^{2m+1} (\neg p_{i_1 j} \vee \neg p_{i_2 j} \vee \neg p_{i_2 j})$$

DRE of Doubled Pigeonhole Principle

DRE($2PHP_m^{2m+1}$):

- for each x_{ij}, $i \in [2m+1]$, $j \in [m]$:
  - associate variables p_{ij} and n_{ij}
  - add soft clauses (p_{ij}), (n_{ij})
  - add hard $\mathcal{P}$-clause $(\neg p_{ij} \vee \neg n_{ij})$
- for each $i \in [2m+1]$:

$$\mathcal{L}_i = (\neg n_{i1} \vee \ldots \vee \neg n_{im})$$

- for each $j \in [m]$:

$$\mathcal{M}_j = \bigwedge_{i_1=1}^{2m-1} \bigwedge_{i_2=1}^{2m} \bigwedge_{i_3=3}^{2m+1} (\neg p_{i_1 j} \vee \neg p_{i_2 j} \vee \neg p_{i_2 j})$$

$2PHP_m^{m+1}$ unsatisfiable if cost $\geq N + 1 = (2m+1)m + 1$

Computing MaxSAT cost of DRE($2PHP_m^{2m+1}$)

Proposition

Given $<\mathcal{M}_j, \mathcal{S}>$ there is an execution of the basic MaxHS algorithm that computes a MaxSAT cost of $2m-1$ in polynomial time.

Computing MaxSAT cost of DRE($2PHP_m^{2m+1}$)

Proposition

Given $< \mathcal{M}_j, \mathcal{S} >$ there is an execution of the basic MaxHS algorithm that computes a MaxSAT cost of $2m - 1$ in polynomial time.

Proof Idea:

- Order the clauses in the SAT solver (via an ordering of the variables)
- Cores induce sets to hit in MHS that correspond to a known structure (a set of all triplets, or a set of all triplets plus some triplets containing and additional element)
- Last iteration corresponds to a minimum hitting set of size $2m - 1$ (cost)

Outline

Basic MaxHS Algorithm

DRMaxSAT

DRMaxSAT/basic MaxHS vs Pigeonhole Principle

DRMaxSAT/basic MaxHS vs Doubled Pigeonhole Principle

Conclusions

Conclusions

- MaxHS-like MaxSAT algorithms show good performance on dual-rail encoded families of benchmarks
- Showed that DRMaxSAT using Basic MaxHS Algorithm can refute in polynomial time:
  - Pigeonhole Principle
  - Doubled Pigeonhole Principle

- Future work will seek to :
  - understand how MaxHS-like algorithms compare with core-guided algortihms
  - search for other principles (hard for resolution) for which DRMaxSAT may be beneficial