SAT-Based Problem Solving

Lecture #2:
Solving Minimal Set Problems with SAT Oracles

Joao Marques-Silva[1,2]

[1]University College Dublin, Ireland

[2]IST/INESC-ID, Lisbon, Portugal

University of Calabria, Italy

February 2015

Part III

Computing Subset Minimal Sets

SAT solving in practice

- SAT is a success story of Computer Science
  - Hundreds (even more?) of practical applications

SAT solving in practice

- SAT is a success story of Computer Science
  - Hundreds (even more?) of practical applications

- Many formulated as decision problems; many others **not**

Decision vs. function problems

Answer	Problem Type

Decision vs. function problems

Answer	Problem Type
Yes/No	Decision Problems

Decision vs. function problems

Answer	Problem Type
Yes/No	Decision Problems
Otherwise	

Decision vs. function problems

Answer	Problem Type
Yes/No	Decision Problems
Otherwise	Function (Search) Problems

Decision vs. function problems

Answer	Problem Type
Yes/No	Decision Problems
Otherwise	Function (Search) Problems

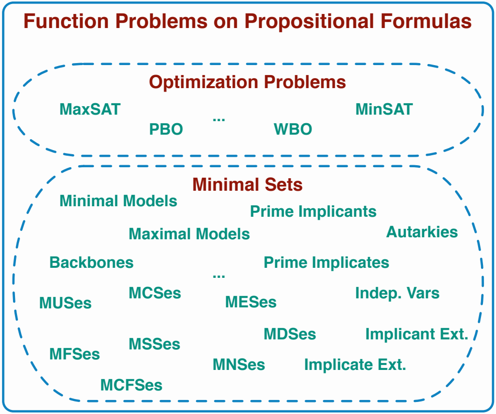
Function Problems on Propositional Formulas

MaxSAT
PBO
...
WBO
MinSAT

Minimal Models
Prime Implicants
Maximal Models
Autarkies
Backbones
Prime Implicates
...
MUSes
MCSes
MESes
Indep. Vars
MDSes
Implicant Ext.
MFSes
MSSes
MNSes
Implicate Ext.
MCFSes

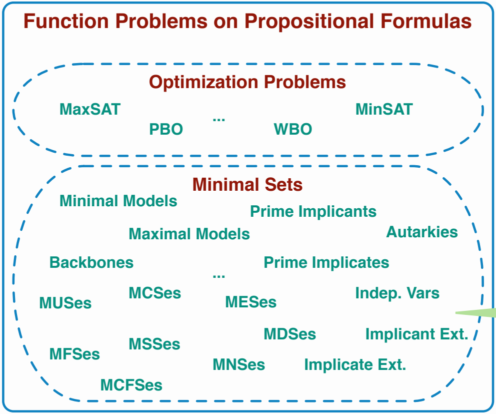
Decision vs. function problems

Answer	Problem Type
Yes/No	Decision Problems
Otherwise	Function (Search) Problems



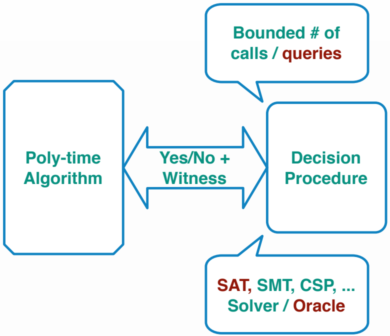
Decision vs. function problems

Answer	Problem Type
Yes/No	Decision Problems
Otherwise	Function (Search) Problems

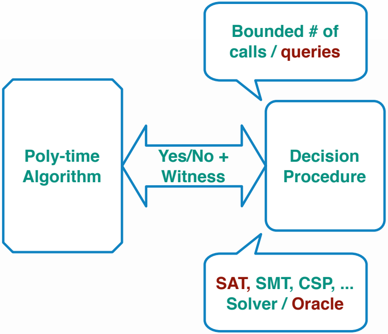


Recap MUS, MCS, MSS, MES, MFS, ...!

How to solve function problems?

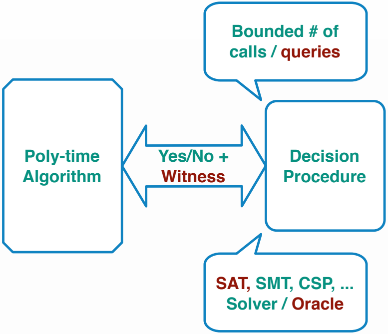


How to solve function problems?



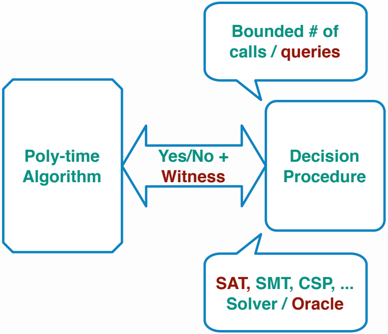
- SAT oracle $\neq$ (standard) NP oracle. **Why?**

How to solve function problems?



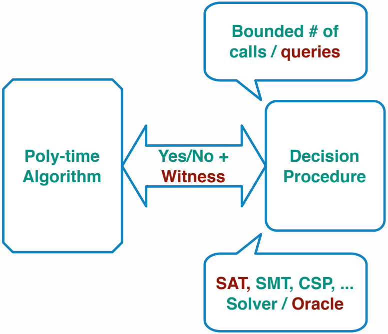
- SAT oracle $\neq$ (standard) NP oracle. **Why?**
  - SAT oracles compute witnesses for **Y** outcomes

How to solve function problems?



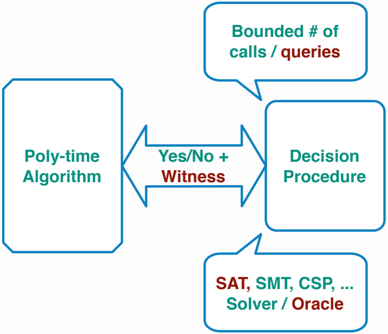
- SAT oracle $\neq$ (standard) NP oracle. **Why?**
  - SAT oracles compute witnesses for **Y** outcomes
  - SAT oracle corresponds to a **witness oracle** (more later) [e.g. BKT93]

How to solve function problems?



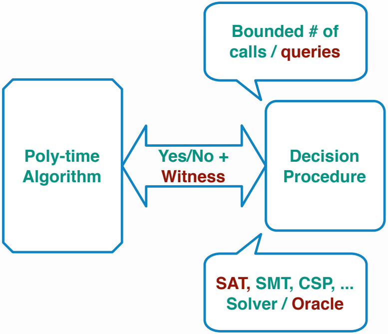
- SAT oracle $\neq$ (standard) NP oracle. **Why?**
  - SAT oracles compute witnesses for **Y** outcomes
  - SAT oracle corresponds to a **witness oracle** (more later) [e.g. BKT93]
- SAT oracle queries can be expensive! **How to minimize queries?**

How to solve function problems?



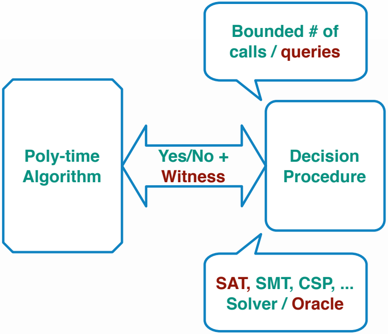
- SAT oracle $\neq$ (standard) NP oracle. **Why?**
  - SAT oracles compute witnesses for **Y** outcomes
  - SAT oracle corresponds to a **witness oracle** (more later) [e.g. BKT93]
- SAT oracle queries can be expensive! **How to minimize queries?**
  - Develop more efficient algorithms, i.e. with **fewer** oracle calls

How to solve function problems?



- SAT oracle $\neq$ (standard) NP oracle. **Why?**
  - SAT oracles compute witnesses for **Y** outcomes
  - SAT oracle corresponds to a **witness oracle** (more later) [e.g. BKT93]
- SAT oracle queries can be expensive! **How to minimize queries?**
  - Develop more efficient algorithms, i.e. with **fewer** oracle calls
  - Characterize **query complexity** of function problems

How to solve function problems?



- SAT oracle $\neq$ (standard) NP oracle. **Why?**
  - SAT oracles compute witnesses for **Y** outcomes
  - SAT oracle corresponds to a **witness oracle** (more later) [e.g. BKT93]
- SAT oracle queries can be expensive! **How to minimize queries?**
  - Develop more efficient algorithms, i.e. with **fewer** oracle calls
  - Characterize **query complexity** of function problems
    - **But**, use **witness oracles** instead of NP oracles

Detour – the FSAT problem

- FSAT: Compute satisfying assignment (model) for formula $\mathcal{F}$

Detour – the FSAT problem

- FSAT: Compute satisfying assignment (model) for formula $\mathcal{F}$
- How to solve FSAT with NP oracle?

Detour – the FSAT problem

- FSAT: Compute satisfying assignment (model) for formula $\mathcal{F}$
- How to solve FSAT with NP oracle?
  - Check whether $\mathcal{F}$ is SAT
  - Build model by iteratively checking literals

Detour – the FSAT problem

- FSAT: Compute satisfying assignment (model) for formula $\mathcal{F}$
- How to solve FSAT with NP oracle?
  - Check whether $\mathcal{F}$ is SAT
  - Build model by iteratively checking literals
- How many calls are necessary to solve FSAT with NP oracle?

Detour – the FSAT problem

- FSAT: Compute satisfying assignment (model) for formula $\mathcal{F}$
- How to solve FSAT with NP oracle?
  - Check whether $\mathcal{F}$ is SAT
  - Build model by iteratively checking literals
- How many calls are necessary to solve FSAT with NP oracle?
  - Basic algorithm requires linear number of NP oracles calls
  - Can we do better than linear?

Detour – the FSAT problem

- FSAT: Compute satisfying assignment (model) for formula $\mathcal{F}$
- How to solve FSAT with NP oracle?
  - Check whether $\mathcal{F}$ is SAT
  - Build model by iteratively checking literals
- How many calls are necessary to solve FSAT with NP oracle?
  - Basic algorithm requires linear number of NP oracles calls
  - Can we do better than linear?
- Result: [GF93]

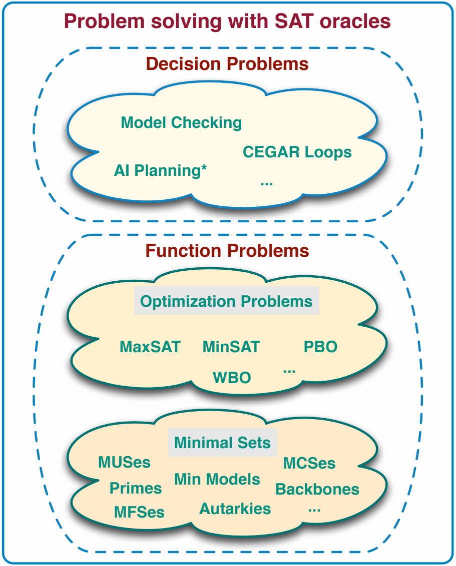
  FSAT cannot be solved with $\mathcal{O}(\log n)$ calls unless $\mathrm{P} = \mathrm{NP}$

Detour – the FSAT problem

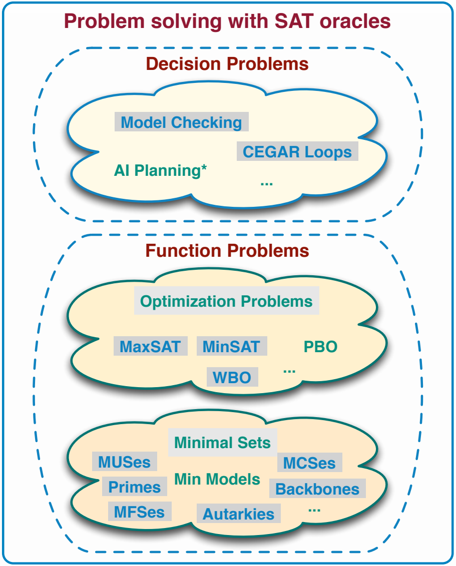
- FSAT: Compute satisfying assignment (model) for formula $\mathcal{F}$
- How to solve FSAT with NP oracle?
  - Check whether $\mathcal{F}$ is SAT
  - Build model by iteratively checking literals
- How many calls are necessary to solve FSAT with NP oracle?
  - Basic algorithm requires linear number of NP oracles calls
  - Can we do better than linear?
- Result: [GF93]

  FSAT cannot be solved with $\mathcal{O}(\log n)$ calls unless $\mathrm{P} = \mathrm{NP}$
- **But with SAT oracle, one call suffices !**
  - Model is given by witness returned by SAT oracle

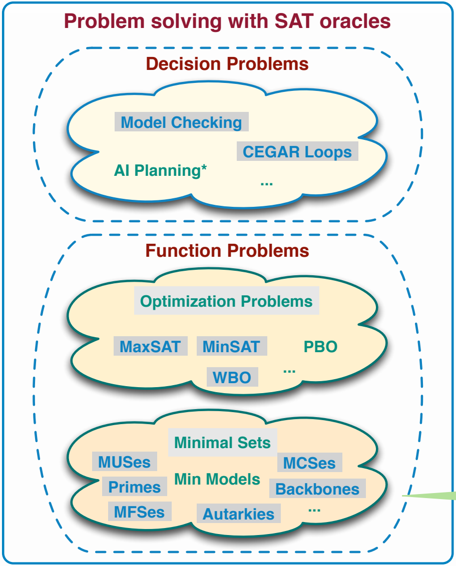
Problem solving with SAT oracles – general case



Problem solving with SAT oracles – our work (2007-...)



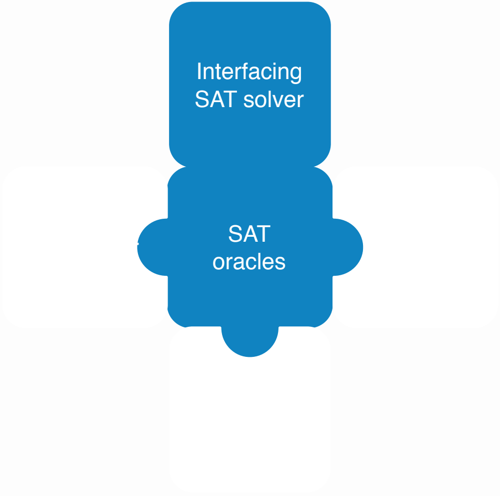
Problem solving with SAT oracles – our work (2007-...)



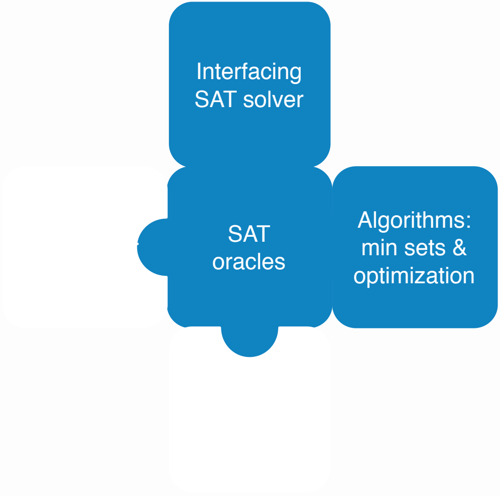
But also, MESes, groups, variables, circuits, SMUS,...

Problem solving with SAT oracles – some challenges

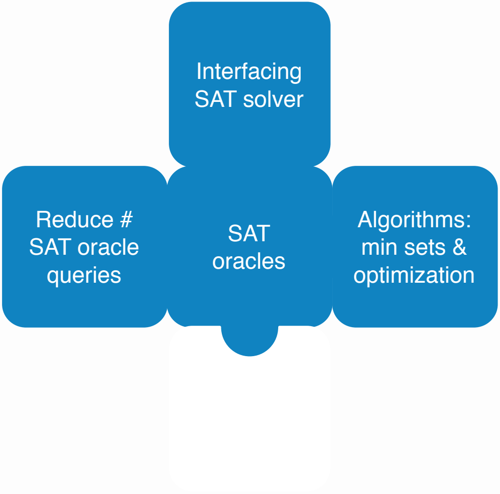
Problem solving with SAT oracles – some challenges



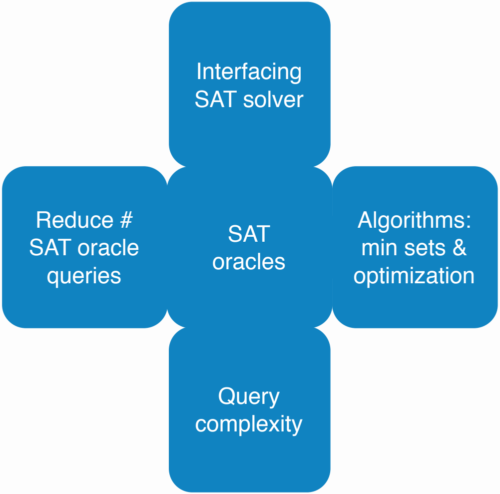
Problem solving with SAT oracles – some challenges



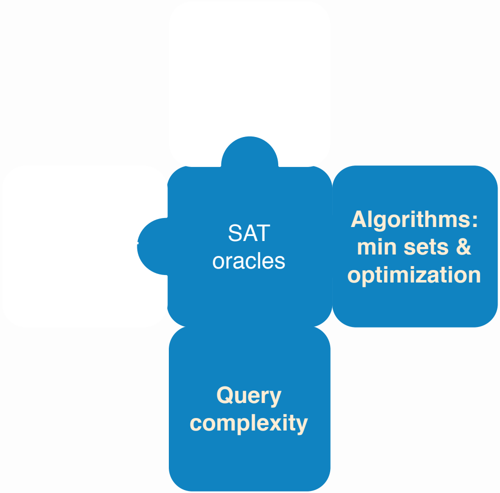
Problem solving with SAT oracles – some challenges



Problem solving with SAT oracles – some challenges



Problem solving with SAT oracles – these talks



Another detour – some challenges

- **MUS**: [e.g. PW88,SP88,CD91,BDTW93,J01,J04,HLSB06,KBK09,K11,MSL11,BMS11,BLMS12,MSJB13]
  - Find $\mathcal{M} \subseteq \mathcal{F}$ s.t. $\mathcal{M}$ is unsatisfiable and $\mathcal{M}$ is irreducible
  - **Q1**: Algorithms for computing one MUS?
  - **Q2**: Worst-case number of queries to NP/SAT oracle to compute one MUS?

Another detour – some challenges

- **MUS**: [e.g. PW88,SP88,CD91,BDTW93,J01,J04,HLSB06,KBK09,K11,MSL11,BMS11,BLMS12,MSJB13]
  - Find $\mathcal{M} \subseteq \mathcal{F}$ s.t. $\mathcal{M}$ is unsatisfiable and $\mathcal{M}$ is irreducible
  - **Q1**: Algorithms for computing one MUS?
  - **Q2**: Worst-case number of queries to NP/SAT oracle to compute one MUS?

- **MCS**: [e.g. R87,BS05,OOF05,LS08,FSZ12,NBE12,MSHJPB13]
  - Find $\mathcal{C} \subseteq \mathcal{F}$ s.t. $\mathcal{F} \setminus \mathcal{C}$ is satisfiable and $\mathcal{C}$ is irreducible
  - **Q1**: Algorithms for computing one MCS?
  - **Q2**: Worst-case number of queries to NP/SAT oracle to compute one MCS?

Another detour – some challenges

- **MUS**: [e.g. PW88,SP88,CD91,BDTW93,J01,J04,HLSB06,KBK09,K11,MSL11,BMS11,BLMS12,MSJB13]
  - Find $\mathcal{M} \subseteq \mathcal{F}$ s.t. $\mathcal{M}$ is unsatisfiable and $\mathcal{M}$ is irreducible
  - **Q1**: Algorithms for computing one MUS?
  - **Q2**: Worst-case number of queries to NP/SAT oracle to compute one MUS?

- **MCS**: [e.g. R87,BS05,OOF05,LS08,FSZ12,NBE12,MSHJPB13]
  - Find $\mathcal{C} \subseteq \mathcal{F}$ s.t. $\mathcal{F} \setminus \mathcal{C}$ is satisfiable and $\mathcal{C}$ is irreducible
  - **Q1**: Algorithms for computing one MCS?
  - **Q2**: Worst-case number of queries to NP/SAT oracle to compute one MCS?

- **Backbone**: [e.g. MZKST99,KK01,SW01,SKK03,KSTW05,MSJL10,ZWSM11,JLMS15]
  - Find set of literals common to all satisfying assignments of $\mathcal{F}$
  - **Q1**: Algorithms for computing the Backbone of $\mathcal{F}$?
  - **Q2**: Worst-case number of queries to NP/SAT oracle to compute the Backbone of $\mathcal{F}$?

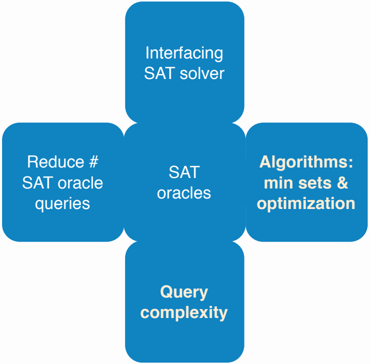
Outline

Revisit MUSes

Minimal Sets

Query Complexity

Conclusions



Outline

Revisit MUSes

Minimal Sets

Query Complexity

Conclusions

Defining MUSes

$x_6 \vee x_2$	$\neg x_6 \vee x_2$	$\neg x_2 \vee x_1$	$\neg x_1$
$\neg x_6 \vee x_8$	$x_6 \vee \neg x_8$	$x_2 \vee x_4$	$\neg x_4 \vee x_5$
$x_7 \vee x_5$	$\neg x_7 \vee x_5$	$\neg x_5 \vee x_3$	$\neg x_3$

- Formula is unsatisfiable but not irreducible

Defining MUSes

$x_6 \vee x_2$	$\neg x_6 \vee x_2$	$\neg x_2 \vee x_1$	$\neg x_1$
$\neg x_6 \vee x_8$	$x_6 \vee \neg x_8$	$x_2 \vee x_4$	$\neg x_4 \vee x_5$
$x_7 \vee x_5$	$\neg x_7 \vee x_5$	$\neg x_5 \vee x_3$	$\neg x_3$

- Formula is unsatisfiable but not irreducible
- Can remove clauses, and formula still unsatisfiable

Defining MUSes

$x_6 \vee x_2$	$\neg x_6 \vee x_2$	$\neg x_2 \vee x_1$	$\neg x_1$
$\neg x_6 \vee x_8$	$x_6 \vee \neg x_8$	$x_2 \vee x_4$	$\neg x_4 \vee x_5$
$x_7 \vee x_5$	$\neg x_7 \vee x_5$	$\neg x_5 \vee x_3$	$\neg x_3$

- Formula is unsatisfiable but not irreducible
- Can remove clauses, and formula still unsatisfiable
- A Minimal Unsatisfiable Subformula (**MUS**) is an unsatisfiable and **irreducible** subformula

Defining MUSes

$x_6 \vee x_2$	$\neg x_6 \vee x_2$	$\neg x_2 \vee x_1$	$\neg x_1$
$\neg x_6 \vee x_8$	$x_6 \vee \neg x_8$	$x_2 \vee x_4$	$\neg x_4 \vee x_5$
$x_7 \vee x_5$	$\neg x_7 \vee x_5$	$\neg x_5 \vee x_3$	$\neg x_3$

- Formula is unsatisfiable but not irreducible
- Can remove clauses, and formula still unsatisfiable
- A Minimal Unsatisfiable Subformula (**MUS**) is an unsatisfiable and **irreducible** subformula

Defining MUSes

$x_6 \vee x_2$	$\neg x_6 \vee x_2$	$\neg x_2 \vee x_1$	$\neg x_1$
$\neg x_6 \vee x_8$	$x_6 \vee \neg x_8$	$x_2 \vee x_4$	$\neg x_4 \vee x_5$
$x_7 \vee x_5$	$\neg x_7 \vee x_5$	$\neg x_5 \vee x_3$	$\neg x_3$

- Formula is unsatisfiable but not irreducible
- Can remove clauses, and formula still unsatisfiable
- A Minimal Unsatisfiable Subformula (**MUS**) is an unsatisfiable and **irreducible** subformula

Defining MUSes

$x_6 \vee x_2$	$\neg x_6 \vee x_2$	$\neg x_2 \vee x_1$	$\neg x_1$
$\neg x_6 \vee x_8$	$x_6 \vee \neg x_8$	$x_2 \vee x_4$	$\neg x_4 \vee x_5$
$x_7 \vee x_5$	$\neg x_7 \vee x_5$	$\neg x_5 \vee x_3$	$\neg x_3$

- Formula is unsatisfiable but not irreducible
- Can remove clauses, and formula still unsatisfiable
- A Minimal Unsatisfiable Subformula (**MUS**) is an unsatisfiable and **irreducible** subformula
- How to compute an MUS?

An Example

$$(\neg x_1 \vee x_2)$$
$$(\neg x_3 \vee x_2)$$
$$(x_1 \vee x_2)$$
$$(\neg x_3)$$
$$(\neg x_2)$$

UNSAT instance; not irreducible

An Example

$(\neg x_1 \vee x_2)$
$(\neg x_3 \vee x_2)$
$(x_1 \vee x_2)$
$(\neg x_3)$
$(\neg x_2)$

Hide clause $(\neg x_1 \vee x_2)$

An Example

$(\neg x_1 \lor x_2)$
$(\neg x_3 \lor x_2)$
$(x_1 \lor x_2)$
$(\neg x_3)$
$(\neg x_2)$

SAT instance $\rightarrow$ keep clause $(\neg x_1 \lor x_2)$

An Example

$(\neg x_1 \vee x_2)$
$(\neg x_3 \vee x_2)$
$(x_1 \vee x_2)$
$(\neg x_3)$
$(\neg x_2)$

Hide clause $(\neg x_3 \vee x_2)$

An Example

$$(\neg x_1 \vee x_2)$$
$$(x_1 \vee x_2)$$
$$(\neg x_3)$$
$$(\neg x_2)$$

UNSAT instance $\rightarrow$ remove clause $(\neg x_3 \vee x_2)$

An Example

$(\neg x_1 \vee x_2)$
$(\neg x_3 \vee x_2)$
$(x_1 \vee x_2)$
$(\neg x_3)$
$(\neg x_2)$

Hide clause $(x_1 \vee x_2)$

An Example

$$(\neg x_1 \vee x_2)$$
$$(\neg x_3 \vee x_2)$$
$$(x_1 \vee x_2)$$
$$(\neg x_3)$$
$$(\neg x_2)$$

SAT instance $\rightarrow$ keep clause $(x_1 \vee x_2)$

An Example

$(\neg x_1 \vee x_2)$
$(\neg x_3 \vee x_2)$
$(x_1 \vee x_2)$
$(\neg x_3)$
$(\neg x_2)$

Hide clause $(\neg x_3)$

An Example

$$(\neg x_1 \vee x_2)$$

$$(x_1 \vee x_2)$$

$$(\neg x_2)$$

UNSAT instance $\rightarrow$ remove clause $(\neg x_3)$

An Example

$$(\neg x_1 \vee x_2)$$

$$(\neg x_3 \vee x_2)$$

$$(x_1 \vee x_2)$$

$$(\neg x_3)$$

$$(\neg x_2)$$

Hide clause $(\neg x_2)$

An Example

$$(\neg x_1 \vee x_2)$$
$$(\neg x_3 \vee x_2)$$
$$(x_1 \vee x_2)$$
$$(\neg x_3)$$
$$(\neg x_2)$$

SAT instance $\rightarrow$ keep clause $(\neg x_2)$

An Example

$$(\neg x_1 \vee x_2)$$
$$(x_1 \vee x_2)$$
$$(\neg x_2)$$

Computed MUS

Deletion-Based MUS Extraction

Input : Unsatisfiable CNF Formula $\mathcal{F}$
Output: MUS $\mathcal{M}$
begin
  $\mathcal{M} \leftarrow \mathcal{F}$ // MUS over-approximation
  foreach $c \in \mathcal{M}$ **do**
    if not SAT$(\mathcal{M} \setminus \{c\})$ **then**
      $\mathcal{M} \leftarrow \mathcal{M} \setminus \{c\}$ // If UNSAT$(\mathcal{M} \setminus \{c\})$, then $c \notin \mathcal{M}$
  return $\mathcal{M}$ // Final $\mathcal{M}$ is MUS
end

- Number of calls to SAT solver: $\mathcal{O}(|\mathcal{F}|)$

Deletion-Based MUS Extraction

Input : Unsatisfiable CNF Formula $\mathcal{F}$
Output: MUS $\mathcal{M}$
begin
  $\mathcal{M} \leftarrow \mathcal{F}$ // MUS over-approximation
  foreach $c \in \mathcal{M}$ **do**
    if not SAT$(\mathcal{M} \setminus \{c\})$ **then**
      $\mathcal{M} \leftarrow \mathcal{M} \setminus \{c\}$ // Remove c from $\mathcal{M}$
  return $\mathcal{M}$ // Final $\mathcal{M}$ is MUS
end

- Number of calls to SAT solver: $\mathcal{O}(|\mathcal{F}|)$

More on MUS Extraction

Algorithm	# Oracle Calls	Reference
Insertion (Default)	$\mathcal{O}(m \times k)$	[SP88]
Deletion (Default)	$\mathcal{O}(m)$	[CD91,BDTW93]
QuickXplain	$\mathcal{O}(k \times (1 + \log \frac{m}{k}))$	[J01,J04]
Dichotomic	$\mathcal{O}(k \times \log m)$	[HLSB06]
Insertion with Relaxation Variables	$\mathcal{O}(m)$	[MSL11]
Deletion with Model Rotation	$\mathcal{O}(m)$	[BLMS12,MSL11]
Progression	$\mathcal{O}(k \times \log(1 + \frac{m}{k}))$	[MSJB13]

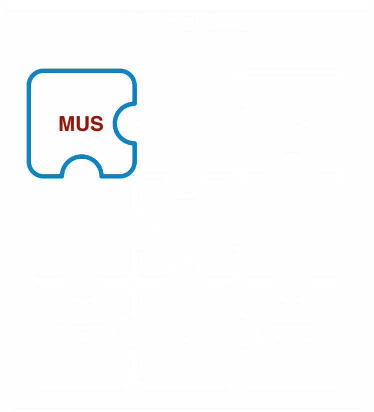
More on MUS Extraction

Algorithm	# Oracle Calls	Reference
Insertion (Default)	$\mathcal{O}(m \times k)$	[SP88]
Deletion (Default)	$\mathcal{O}(m)$	[CD91,BDTW93]
QuickXplain	$\mathcal{O}(k \times (1 + \log \frac{m}{k}))$	[J01,J04]
Dichotomic	$\mathcal{O}(k \times \log m)$	[HLSB06]
Insertion with Relaxation Variables	$\mathcal{O}(m)$	[MSL11]
Deletion with Model Rotation	$\mathcal{O}(m)$	[BLMS12,MSL11]
Progression	$\mathcal{O}(k \times \log(1 + \frac{m}{k}))$	[MSJB13]

- Additional Techniques:
  - Restrict formula to unsatisfiable subsets [BDTW93,HLSB06,DHN06,MSL11]
  - Check redundancy condition [vMW08,MSL11,BLMS12]
  - Model rotation, recursive model rotation, etc. [MSL11,BMS11,BLMS12,W12]

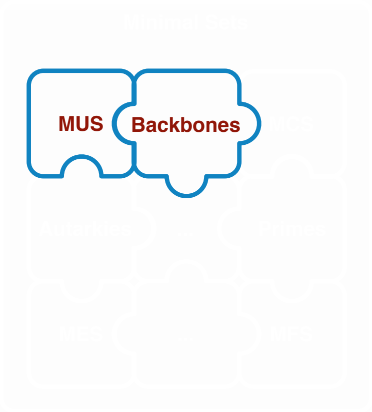
Outline

Computing minimal sets is ubiquitous!



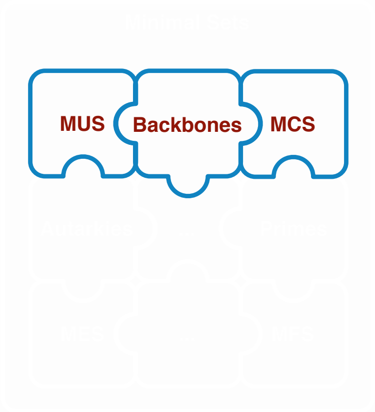
- MUSes are minimal sets
  - Extensive work since the mid 80s

Computing minimal sets is ubiquitous!



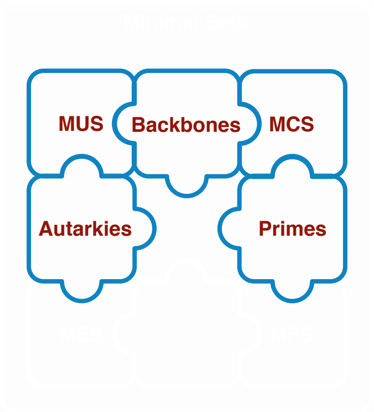
- Backbones(!) are minimal sets
  - Extensive work since the late 90s

Computing minimal sets is ubiquitous!



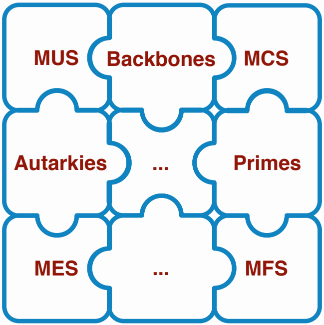
- MCSes are minimal sets
  - Extensive work since the mid 80s

Computing minimal sets is ubiquitous!



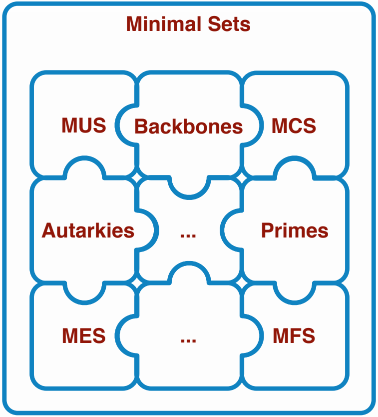
- Autarkies(!) & primes are also minimal sets
  - Extensive work since the 80s & 30s(!), resp.

Computing minimal sets is ubiquitous!



- MESes, MFSes (and many more!) are minimal sets
  - Work since the 00s & 90s, etc.

Computing minimal sets is ubiquitous!



- ∴ **Develop framework for reasoning about minimal sets !**
  - **Why?** Common algorithms & techniques; new insights & results; ...

Example – MUSes as minimal sets

$$(\bar{x}_1 \vee \bar{x}_2) \quad (x_1) \quad (x_5 \vee x_6) \quad (\bar{x}_3 \vee \bar{x}_4) \quad (x_2) \quad (x_3) \quad (x_4)$$

- Formula is unsatisfiable but **not** irreducible

Example – MUSes as minimal sets

$$(\bar{x}_1 \vee \bar{x}_2) \quad (x_1) \quad \boxed{(x_5 \vee x_6)} \quad (\bar{x}_3 \vee \bar{x}_4) \quad (x_2) \quad (x_3) \quad (x_4)$$

- Formula is unsatisfiable but **not** irreducible
- Can remove clauses, and formula still unsatisfiable

Example – MUSes as minimal sets

$\boxed{(\bar{x}_1 \vee \bar{x}_2)}\ \boxed{(x_1)}\ (x_5 \vee x_6)\ (\bar{x}_3 \vee \bar{x}_4)\ \boxed{(x_2)}\ (x_3)\ (x_4)$

- Formula is unsatisfiable but **not** irreducible
- Can remove clauses, and formula still unsatisfiable

- **Minimal Unsatisfiable Subset** (**MUS**):
  - Irreducible subformula that is unsatisfiable
    - MUSes are minimal sets

Example – MUSes as minimal sets

$(\bar{x}_1 \vee \bar{x}_2) \quad (x_1) \quad (x_5 \vee x_6) \quad \boxed{(\bar{x}_3 \vee \bar{x}_4)} \quad (x_2) \quad \boxed{(x_3)}\boxed{(x_4)}$

- Formula is unsatisfiable but **not** irreducible
- Can remove clauses, and formula still unsatisfiable
- **Minimal Unsatisfiable Subset** (**MUS**):
  - Irreducible subformula that is unsatisfiable
    - MUSes are minimal sets

Example – MUSes as minimal sets

$(\bar{x}_1 \vee \bar{x}_2) \quad (x_1) \quad (x_5 \vee x_6) \quad \boxed{(\bar{x}_3 \vee \bar{x}_4)} \quad (x_2) \quad \boxed{(x_3)}\boxed{(x_4)}$

- Formula is unsatisfiable but **not** irreducible
- Can remove clauses, and formula still unsatisfiable
- **Minimal Unsatisfiable Subset** (**MUS**):
  - Irreducible subformula that is unsatisfiable
    - MUSes are minimal sets
- Complexity results:
  - Decision problem: D^P-complete [PW88]
  - Function problem: in FP^{NP} with lower bound in $FP^{NP}_{||}$ [CT95]

Example – MCSes as minimal sets

$$(\bar{x}_1 \vee \bar{x}_2) \quad (x_1) \quad (x_5 \vee x_6) \quad (\bar{x}_3 \vee \bar{x}_4) \quad (x_2) \quad (x_3) \quad (x_4)$$

- Formula is unsatisfiable with satisfiable subformulas

Example – MCSes as minimal sets

$(\bar{x}_1 \vee \bar{x}_2)$ (x_1) $(x_5 \vee x_6)$ $(\bar{x}_3 \vee \bar{x}_4)$ (x_2) (x_3) (x_4)

- Formula is unsatisfiable with satisfiable subformulas
- Can remove clauses such that remaining clauses are satisfiable

Example – MCSes as minimal sets

$(\bar{x}_1 \vee \bar{x}_2) \quad (x_1) \quad (x_5 \vee x_6) \quad (\bar{x}_3 \vee \bar{x}_4) \quad (x_2) \quad (x_3) \quad (x_4)$

- Formula is unsatisfiable with satisfiable subformulas
- Can remove clauses such that remaining clauses are satisfiable
- **Minimal Correction Subset** (**MCS**):
  - Irreducible subformula such that the complement is satisfiable
    - MCSes are minimal sets

Example – MCSes as minimal sets

$(\bar{x}_1 \vee \bar{x}_2) \quad (x_1) \quad (x_5 \vee x_6) \quad (\bar{x}_3 \vee \bar{x}_4) \quad \boxed{(x_2)}\boxed{(x_3)} \quad (x_4)$

- Formula is unsatisfiable with satisfiable subformulas
- Can remove clauses such that remaining clauses are satisfiable
- **Minimal Correction Subset** (**MCS**):
  - Irreducible subformula such that the complement is satisfiable
    - MCSes are minimal sets

Example – MCSes as minimal sets

$$(\bar{x}_1 \vee \bar{x}_2) \quad (x_1) \quad (x_5 \vee x_6) \quad (\bar{x}_3 \vee \bar{x}_4) \quad \boxed{(x_2)} \quad (x_3) \quad \boxed{(x_4)}$$

- Formula is unsatisfiable with satisfiable subformulas
- Can remove clauses such that remaining clauses are satisfiable

- **Minimal Correction Subset** (**MCS**):
  - Irreducible subformula such that the complement is satisfiable
    - MCSes are minimal sets

Example – MCSes as minimal sets

$$(\bar{x}_1 \vee \bar{x}_2) \quad (x_1) \quad (x_5 \vee x_6) \quad (\bar{x}_3 \vee \bar{x}_4) \quad \boxed{(x_2)} \quad (x_3) \quad \boxed{(x_4)}$$

- Formula is unsatisfiable with satisfiable subformulas
- Can remove clauses such that remaining clauses are satisfiable
- **Minimal Correction Subset (MCS)**:
  - Irreducible subformula such that the complement is satisfiable
    - MCSes are minimal sets
- Complexity results:
  - Function problem: can be solved with $\mathcal{O}(\log n)$ calls to a SAT oracle

Example – MCSes as minimal sets

$$(\bar{x}_1 \vee \bar{x}_2) \quad (x_1) \quad (x_5 \vee x_6) \quad (\bar{x}_3 \vee \bar{x}_4) \quad \boxed{(x_2)} \quad (x_3) \quad \boxed{(x_4)}$$

- Formula is unsatisfiable with satisfiable subformulas
- Can remove clauses such that remaining clauses are satisfiable
- **Minimal Correction Subset** (**MCS**):
  - Irreducible subformula such that the complement is satisfiable
    - MCSes are minimal sets
- Complexity results:
  - Function problem: can be solved with $\mathcal{O}(\log n)$ calls to a SAT oracle. **Why?**

Monotone predicates

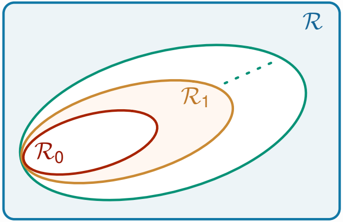
- Set of elements $\mathcal{R}$
- Predicate $P : 2^{\mathcal{R}} \to \{0, 1\}$

Monotone predicates

- Set of elements $\mathcal{R}$
- Predicate $P : 2^{\mathcal{R}} \to \{0, 1\}$
- P is **monotone** iff P has the following property: [BM07]

$\Longrightarrow$ If $P(\mathcal{R}_0) = 1$ holds and $\mathcal{R}_0 \subseteq \mathcal{R}_1 \subseteq \mathcal{R}$, then $P(\mathcal{R}_1) = 1$ also holds

  - Note: $P(\mathcal{R}) = 1$ must hold; otherwise no minimal set



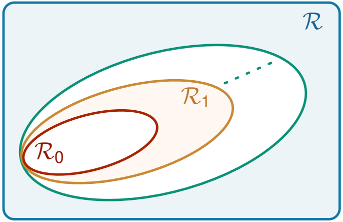
Monotone predicates

- Set of elements $\mathcal{R}$
- Predicate $P : 2^{\mathcal{R}} \to \{0, 1\}$

- P is **monotone** iff P has the following property: [BM07]

> $\Longrightarrow$ If $P(\mathcal{R}_0) = 1$ holds and $\mathcal{R}_0 \subseteq \mathcal{R}_1 \subseteq \mathcal{R}$, then $P(\mathcal{R}_1) = 1$ also holds

  - Note: $P(\mathcal{R}) = 1$ must hold; otherwise no minimal set



- Minimal Set over Monotone Predicate (MSMP) problem: [MSJB13]
  1. Given $\mathcal{R}$ and monotone predicate P over $\mathcal{R}$,
  2. compute **minimal set** $\mathcal{M} \subseteq \mathcal{R}$ such that $P(\mathcal{M}) = 1$ holds

Example reductions to MSMP

	MUS	MCS
$\mathcal{R}$	$\mathcal{F}$	$\mathcal{F}$
$P(\mathcal{W}), \mathcal{W} \subseteq \mathcal{R}$	$\neg$**SAT**$(\mathcal{W})$	SAT$(\mathcal{F} \setminus \mathcal{W})$
Min. set $\mathcal{M}$, $P(\mathcal{M})$	$\neg$SAT$(\mathcal{M})$ true	SAT$(\mathcal{F} \setminus \mathcal{M})$ true
$\forall_{\mathcal{M}' \subset \mathcal{M}}, P(\mathcal{M}')$	$\neg$SAT$(\mathcal{M}')$ false	SAT$(\mathcal{F} \setminus \mathcal{M}')$ false

Example reductions to MSMP

	MUS	MCS
$\mathcal{R}$	$\mathcal{F}$	$\mathcal{F}$
$P(\mathcal{W}), \mathcal{W} \subseteq \mathcal{R}$	$\neg\mathbf{SAT}(\mathcal{W})$	$\text{SAT}(\mathcal{F} \setminus \mathcal{W})$
Min. set $\mathcal{M}$, $P(\mathcal{M})$	$\neg\text{SAT}(\mathcal{M})$ true	$\text{SAT}(\mathcal{F} \setminus \mathcal{M})$ true
$\forall_{\mathcal{M}' \subset \mathcal{M}}, P(\mathcal{M}')$	$\neg\text{SAT}(\mathcal{M}')$ false	$\text{SAT}(\mathcal{F} \setminus \mathcal{M}')$ false

c_1	c_2	c_3	c_4	c_5	c_6	c_7
$(\bar{x}_1 \vee \bar{x}_2)$	(x_1)	$(x_5 \vee x_6)$	$(\bar{x}_3 \vee \bar{x}_4)$	(x_2)	(x_3)	(x_4)

MUS:	$\mathcal{W}$	$P(\mathcal{W}) \triangleq \neg\text{SAT}(\mathcal{W})$	
	$c_1\, c_2\, c_3\, c_4\, c_5\, c_6\, c_7$	1	
	$c_2\, c_3\, c_4\, c_5\, c_6\, c_7$	1	
	$c_3\, c_4\, c_5\, c_6\, c_7$	1	
	$c_4\, c_5\, c_6\, c_7$	1	
	$c_4\, c_6\, c_7$	1	$\mathcal{M}$
	$c_4\, c_7$	0	
	$c_6\, c_7$	0	
	...		

Example reductions to MSMP

	MUS	MCS
$\mathcal{R}$	$\mathcal{F}$	$\boldsymbol{\mathcal{F}}$
$P(\mathcal{W}), \mathcal{W} \subseteq \mathcal{R}$	$\neg$SAT($\mathcal{W}$)	**SAT($\mathcal{F} \setminus \mathcal{W}$)**
Min. set $\mathcal{M}$, $P(\mathcal{M})$	$\neg$SAT($\mathcal{M}$) true	SAT($\mathcal{F} \setminus \mathcal{M}$) true
$\forall_{\mathcal{M}' \subset \mathcal{M}}, P(\mathcal{M}')$	$\neg$SAT($\mathcal{M}'$) false	SAT($\mathcal{F} \setminus \mathcal{M}'$) false

Example reductions to MSMP

	MUS	MCS
$\mathcal{R}$	$\mathcal{F}$	$\mathcal{F}$
$P(\mathcal{W}), \mathcal{W} \subseteq \mathcal{R}$	$\neg\text{SAT}(\mathcal{W})$	$\text{SAT}(\mathcal{F} \setminus \mathcal{W})$
Min. set $\mathcal{M}$, $P(\mathcal{M})$	$\neg\text{SAT}(\mathcal{M})$ true	$\text{SAT}(\mathcal{F} \setminus \mathcal{M})$ true
$\forall_{\mathcal{M}' \subset \mathcal{M}}, P(\mathcal{M}')$	$\neg\text{SAT}(\mathcal{M}')$ false	$\text{SAT}(\mathcal{F} \setminus \mathcal{M}')$ false

c_1	c_2	c_3	c_4	c_5	c_6	c_7
$(\bar{x}_1 \vee \bar{x}_2)$	(x_1)	$(x_5 \vee x_6)$	$(\bar{x}_3 \vee \bar{x}_4)$	(x_2)	(x_3)	(x_4)

MCS:	$\mathcal{W}$	$\mathcal{F} \setminus \mathcal{W}$	$P(\mathcal{W}) \triangleq \text{SAT}(\mathcal{F} \setminus \mathcal{W})$	
	$c_1\, c_2\, c_3\, c_4\, c_5\, c_6\, c_7$	$\emptyset$	1	
	$c_1\, c_2\, c_3\, c_4\, c_5\, c_7$	c_6	1	
	$c_1\, c_2\, c_3\, c_5\, c_7$	$c_4\, c_6$	1	
	$c_1\, c_2\, c_5\, c_7$	$c_3\, c_4\, c_6$	1	
	$c_1\, c_5\, c_7$	$c_2\, c_3\, c_4\, c_6$	1	
	$c_5\, c_7$	$c_1\, c_2\, c_3\, c_4\, c_6$	1	$\mathcal{M}$
	c_5	$c_1\, c_2\, c_3\, c_4\, c_6\, c_7$	0	
	c_7	$c_1\, c_2\, c_3\, c_4\, c_5\, c_6$	0	

Reductions to MSMP – a glimpse

Problem	$\mathcal{R}$	$P(\mathcal{W}), \mathcal{W} \subseteq \mathcal{R}$
FMUS	$\mathcal{F}$	$\neg\mathrm{SAT}(\wedge_{c\in\mathcal{W}}(c))$
FMCS	$\mathcal{F}$	$\mathrm{SAT}(\wedge_{c\in\mathcal{R}\setminus\mathcal{W}}(c))$
FMES	$\mathcal{F}$	$\neg\mathrm{SAT}(\neg\mathcal{F} \wedge \wedge_{c\in\mathcal{W}}(c))$
FMDS	$\mathcal{F}$	$\mathrm{SAT}(\neg\mathcal{F} \wedge \wedge_{c\in\mathcal{R}\setminus\mathcal{W}}(c))$
FCMFS	$\mathcal{F}$	$\mathrm{SAT}(\wedge_{c\in\mathcal{R}\setminus\mathcal{W}}(\neg c))$
FMnM	X	$\mathrm{SAT}(\mathcal{F} \wedge \wedge_{x\in\mathcal{R}\setminus\mathcal{W}}(\neg x))$
FPIt	$L(t)$	$\neg\mathrm{SAT}(\neg\mathcal{F} \wedge \wedge_{l\in\mathcal{W}}(l))$
FPIc	$L(c)$	$\neg\mathrm{SAT}(\mathcal{F} \wedge \wedge_{l\in\mathcal{W}}(\neg l))$
FLEIt	$\mathcal{L}_t$	$\neg\mathrm{SAT}(\mathcal{F}^{\mathrm{ItX}} \wedge (\vee_{l\in\mathcal{R}\setminus\mathcal{W}} \neg l))$
FLEIc	$\mathcal{L}_c$	$\neg\mathrm{SAT}(\mathcal{F}^{\mathrm{IcX}} \wedge (\vee_{l\in\mathcal{R}\setminus\mathcal{W}} l))$
FMnES	$\mathcal{J}$	$\neg\mathrm{SAT}(\neg\mathcal{I} \wedge \wedge_{c\in\mathcal{W}}(c))$
FMxES	$\mathcal{N}$	$\neg\mathrm{SAT}(\mathcal{J} \wedge (\vee_{c\in\mathcal{R}\setminus\mathcal{W}} \neg c))$
FBBr	$\mathcal{V}$	$\neg\mathrm{SAT}(\mathcal{F} \wedge (\vee_{l\in\mathcal{R}\setminus\mathcal{W}} \neg l))$
FBB	X	$\neg\mathrm{SAT}(\mathcal{F}^{\mathrm{BB}} \wedge (\vee_{x\in\mathcal{R}\setminus\mathcal{W}}\, x \wedge \neg x'))$
FVInd	X	$\neg\mathrm{SAT}(\mathcal{F}^{\mathrm{VInd}} \wedge \wedge_{x_i\in\mathcal{W}}(x_i \leftrightarrow y_i))$
FAut	X^+	$\mathrm{SAT}(\mathcal{F}^{\mathrm{Aut}} \wedge \wedge_{x^+\in\mathcal{R}\setminus\mathcal{W}}(x^+))$
...		...

Reductions to MSMP – a glimpse

Problem	$\mathcal{R}$	$P(\mathcal{W}), \mathcal{W} \subseteq \mathcal{R}$
FMUS	$\mathcal{F}$	$\neg\mathrm{SAT}(\bigwedge_{c\in\mathcal{W}}(c))$
FMCS	$\mathcal{F}$	$\mathrm{SAT}(\bigwedge_{c\in\mathcal{R}\setminus\mathcal{W}}(c))$
FMES	$\mathcal{F}$	$\neg\mathrm{SAT}(\neg\mathcal{F} \wedge \bigwedge_{c\in\mathcal{W}}(c))$
FMDS	$\mathcal{F}$	$\mathrm{SAT}(\neg\mathcal{F} \wedge \bigwedge_{c\in\mathcal{R}\setminus\mathcal{W}}(c))$
FCMFS	$\mathcal{F}$	$\mathrm{SAT}(\bigwedge_{c\in\mathcal{R}\setminus\mathcal{W}}(\neg c))$
FMnM	X	$\mathrm{SAT}(\mathcal{F} \wedge \bigwedge_{x\in\mathcal{R}\setminus\mathcal{W}}(\neg x))$
FPIt	$L(t)$	$\neg\mathrm{SAT}(\neg\mathcal{F} \wedge \bigwedge_{l\in\mathcal{W}}(l))$
FPIc	$L(c)$	$\neg\mathrm{SAT}(\mathcal{F} \wedge \bigwedge_{l\in\mathcal{W}}(\neg l))$
FLEIt	$\mathcal{L}_t$	$\neg\mathrm{SAT}(\mathcal{F}^{\mathrm{ltX}} \wedge (\bigvee_{l\in\mathcal{R}\setminus\mathcal{W}} \neg l))$
FLEIc	$\mathcal{L}_c$	$\neg\mathrm{SAT}(\mathcal{F}^{\mathrm{lcX}} \wedge (\bigvee_{l\in\mathcal{R}\setminus\mathcal{W}} l))$
FMnES	$\mathcal{J}$	$\neg\mathrm{SAT}(\neg\mathcal{I} \wedge \bigwedge_{c\in\mathcal{W}}(c))$
FMxES	$\mathcal{N}$	$\neg\mathrm{SAT}(\mathcal{J} \wedge (\bigvee_{c\in\mathcal{R}\setminus\mathcal{W}}\neg c))$
FBBr	$\mathcal{V}$	$\neg\mathrm{SAT}(\mathcal{F} \wedge (\bigvee_{l\in\mathcal{R}\setminus\mathcal{W}}\neg l))$
FBB	X	$\neg\mathrm{SAT}(\mathcal{F}^{\mathrm{BB}} \wedge (\bigvee_{x\in\mathcal{R}\setminus\mathcal{W}} x \wedge \neg x'))$
FVInd	X	$\neg\mathrm{SAT}(\mathcal{F}^{\mathrm{VInd}} \wedge \bigwedge_{x_i\in\mathcal{W}}(x_i \leftrightarrow y_i))$
FAut	X^+	$\mathrm{SAT}(\mathcal{F}^{\mathrm{Aut}} \wedge \bigwedge_{x^+\in\mathcal{R}\setminus\mathcal{W}}(x^+))$
...		...

MSMP algorithms

- Why MSMP algorithms?

MSMP algorithms

- Why MSMP algorithms? Common algorithms & techniques, ...

MSMP algorithms

- Why MSMP algorithms? Common algorithms & techniques, ...
- Adapt algorithms for MUS extraction
  - Insertion; Deletion; Dichotomic; QuickXplain; Progression
- Worst-case number of predicate tests:
  - Set $\mathcal{R}$ with m elements and k the size of largest minimal subset

Algorithm	# Predicate tests	Reference
Insertion (Default)	$\mathcal{O}(m \times k)$	[SP88,vMW08]
Deletion (Default)	$\mathcal{O}(m)$	[CD91,BDTW93]
Dichotomic	$\mathcal{O}(k \times \log m)$	[HLSB06]
QuickXplain	$\mathcal{O}(k \times (1 + \log \frac{m}{k}))$	[J01,J04]
Progression	$\mathcal{O}(k \times \log(1 + \frac{m}{k}))$	[MSJB13]

  - For MUSes/MCSes/PIs/MMs/MESes/etc. each predicate test represents one query to a SAT oracle

MSMP algorithms

- Why MSMP algorithms? Common algorithms & techniques, ...
- Adapt algorithms for MUS extraction
  - Insertion; Deletion; Dichotomic; QuickXplain; Progression
- Worst-case number of predicate tests:
  - Set $\mathcal{R}$ with m elements and k the size of largest minimal subset

Algorithm	# Predicate tests	Reference
Insertion (Default)	$\mathcal{O}(m \times k)$	[SP88,vMW08]
Deletion (Default)	$\mathcal{O}(m)$	[CD91,BDTW93]
Dichotomic	$\mathcal{O}(k \times \log m)$	[HLSB06]
QuickXplain	$\mathcal{O}(k \times (1 + \log \frac{m}{k}))$	[J01,J04]
Progression	$\mathcal{O}(k \times \log(1 + \frac{m}{k}))$	[MSJB13]

  - For MUSes/MCSes/PIs/MMs/MESes/etc. each predicate test represents one query to a SAT oracle

$\mathcal{O}(m)$ calls for **last 4**!

MSMP algorithms

- Why MSMP algorithms? Common algorithms & techniques, ...
- Adapt algorithms for MUS extraction
  - Insertion; Deletion; Dichotomic; QuickXplain; Progression
- Worst-case number of predicate tests:
  - Set $\mathcal{R}$ with m elements and k the size of largest minimal subset

Algorithm	# Predicate tests	Reference
Insertion (Default)	$\mathcal{O}(m \times k)$	[SP88,vMW08]
Deletion (Default)	$\mathcal{O}(m)$	[CD91,BDTW93]
Dichotomic	$\mathcal{O}(k \times \log m)$	[HLSB06]
QuickXplain	$\mathcal{O}(k \times (1 + \log \frac{m}{k}))$	[J01,J04]
Progression	$\mathcal{O}(k \times \log(1 + \frac{m}{k}))$	[MSJB13]

  - For MUSes/MCSes/PIs/MMs/MESes/etc. each predicate test represents one query to a SAT oracle

$\mathcal{O}(m)$ calls for **last 4**!

- MSMP algorithms can integrate well-known pruning techniques
  - Clause set refinement; Model rotation; etc.* [BDTW93,DHN06,MSL11,BLMS12]

Deletion algorithm – revisited

Input : Target set $\mathcal{T}$
Output: Minimal subset $\mathcal{M}$
begin
    $\mathcal{M} \leftarrow \mathcal{T}$ // Precondition: $P(\mathcal{T})$ holds
    foreach $u \in \mathcal{M}$ **do** // *Inv:* $P(\mathcal{M})$
        if $P(\mathcal{M} \setminus \{u\})$ **then** // P holds without element
            $\mathcal{M} \leftarrow \mathcal{M} \setminus \{u\}$ // Drop element
    return $\mathcal{M}$ // Postcondition: $\mathcal{M}$ is minimal set s.t. $P(\mathcal{M})$ holds
end

- Number of predicate tests: $\mathcal{O}(m)$

Deletion – MUS example

c_1	c_2	c_3	c_4	c_5	c_6	c_7
$(\bar{x}_1 \vee \bar{x}_2)$	(x_1)	$(x_5 \vee x_6)$	$(\bar{x}_3 \vee \bar{x}_4)$	(x_2)	(x_3)	(x_4)

- MUS predicate test: $\mathcal{W} \triangleq \mathcal{M} \setminus \{c_i\}$, $P(\mathcal{W}) \triangleq \neg\text{SAT}(\mathcal{W})$

c_i	$\mathcal{M}$	$\mathcal{M} \setminus \{c_i\}$	$P(\mathcal{W})$	Outcome

Deletion – MUS example

c_1	c_2	c_3	c_4	c_5	c_6	c_7
$(\bar{x}_1 \vee \bar{x}_2)$	(x_1)	$(x_5 \vee x_6)$	$(\bar{x}_3 \vee \bar{x}_4)$	(x_2)	(x_3)	(x_4)

- MUS predicate test: $\mathcal{W} \triangleq \mathcal{M} \setminus \{c_i\}$, $P(\mathcal{W}) \triangleq \neg\mathrm{SAT}(\mathcal{W})$

c_i	$\mathcal{M}$	$\mathcal{M} \setminus \{c_i\}$	$P(\mathcal{W})$	Outcome
c_1	$c_1..c_7$	$c_2..c_7$	1	Drop c_1

Deletion – MUS example

c_1	c_2	c_3	c_4	c_5	c_6	c_7
$(\bar{x}_1 \vee \bar{x}_2)$	(x_1)	$(x_5 \vee x_6)$	$(\bar{x}_3 \vee \bar{x}_4)$	(x_2)	(x_3)	(x_4)

- MUS predicate test: $\mathcal{W} \triangleq \mathcal{M} \setminus \{c_i\}$, $P(\mathcal{W}) \triangleq \neg\mathrm{SAT}(\mathcal{W})$

c_i	$\mathcal{M}$	$\mathcal{M} \setminus \{c_i\}$	$P(\mathcal{W})$	Outcome
c_1	$c_1..c_7$	$c_2..c_7$	1	Drop c_1
c_2	$c_2..c_7$	$c_3..c_7$	1	Drop c_2

Deletion – MUS example

c_1	c_2	c_3	c_4	c_5	c_6	c_7
$(\bar{x}_1 \vee \bar{x}_2)$	(x_1)	$(x_5 \vee x_6)$	$(\bar{x}_3 \vee \bar{x}_4)$	(x_2)	(x_3)	(x_4)

- MUS predicate test: $\mathcal{W} \triangleq \mathcal{M} \setminus \{c_i\}$, $P(\mathcal{W}) \triangleq \neg\mathrm{SAT}(\mathcal{W})$

c_i	$\mathcal{M}$	$\mathcal{M} \setminus \{c_i\}$	$P(\mathcal{W})$	Outcome
c_1	$c_1..c_7$	$c_2..c_7$	1	Drop c_1
c_2	$c_2..c_7$	$c_3..c_7$	1	Drop c_2
c_3	$c_3..c_7$	$c_4..c_7$	1	Drop c_3

Deletion – MUS example

c_1	c_2	c_3	c_4	c_5	c_6	c_7
$(\bar{x}_1 \vee \bar{x}_2)$	(x_1)	$(x_5 \vee x_6)$	$(\bar{x}_3 \vee \bar{x}_4)$	(x_2)	(x_3)	(x_4)

- MUS predicate test: $\mathcal{W} \triangleq \mathcal{M} \setminus \{c_i\}$, $P(\mathcal{W}) \triangleq \neg\text{SAT}(\mathcal{W})$

c_i	$\mathcal{M}$	$\mathcal{M} \setminus \{c_i\}$	$P(\mathcal{W})$	Outcome
c_1	$c_1..c_7$	$c_2..c_7$	1	Drop c_1
c_2	$c_2..c_7$	$c_3..c_7$	1	Drop c_2
c_3	$c_3..c_7$	$c_4..c_7$	1	Drop c_3
c_4	$c_4..c_7$	$c_5..c_7$	0	Keep c_4

Deletion – MUS example

c_1	c_2	c_3	c_4	c_5	c_6	c_7
$(\bar{x}_1 \vee \bar{x}_2)$	(x_1)	$(x_5 \vee x_6)$	$(\bar{x}_3 \vee \bar{x}_4)$	(x_2)	(x_3)	(x_4)

- MUS predicate test: $\mathcal{W} \triangleq \mathcal{M} \setminus \{c_i\}$, $P(\mathcal{W}) \triangleq \neg\text{SAT}(\mathcal{W})$

c_i	$\mathcal{M}$	$\mathcal{M} \setminus \{c_i\}$	$P(\mathcal{W})$	Outcome
c_1	$c_1..c_7$	$c_2..c_7$	1	Drop c_1
c_2	$c_2..c_7$	$c_3..c_7$	1	Drop c_2
c_3	$c_3..c_7$	$c_4..c_7$	1	Drop c_3
c_4	$c_4..c_7$	$c_5..c_7$	0	Keep c_4
c_5	$c_4..c_7$	c_4, c_6, c_7	1	Drop c_5

Deletion – MUS example

c_1	c_2	c_3	c_4	c_5	c_6	c_7
$(\bar{x}_1 \vee \bar{x}_2)$	(x_1)	$(x_5 \vee x_6)$	$(\bar{x}_3 \vee \bar{x}_4)$	(x_2)	(x_3)	(x_4)

- MUS predicate test: $\mathcal{W} \triangleq \mathcal{M} \setminus \{c_i\}$, $P(\mathcal{W}) \triangleq \neg\mathrm{SAT}(\mathcal{W})$

c_i	$\mathcal{M}$	$\mathcal{M} \setminus \{c_i\}$	$P(\mathcal{W})$	Outcome
c_1	$c_1..c_7$	$c_2..c_7$	1	Drop c_1
c_2	$c_2..c_7$	$c_3..c_7$	1	Drop c_2
c_3	$c_3..c_7$	$c_4..c_7$	1	Drop c_3
c_4	$c_4..c_7$	$c_5..c_7$	0	Keep c_4
c_5	$c_4..c_7$	c_4, c_6, c_7	1	Drop c_5
c_6	c_4, c_6, c_7	c_4, c_7	0	Keep c_6

Deletion – MUS example

c_1	c_2	c_3	c_4	c_5	c_6	c_7
$(\bar{x}_1 \vee \bar{x}_2)$	(x_1)	$(x_5 \vee x_6)$	$(\bar{x}_3 \vee \bar{x}_4)$	(x_2)	(x_3)	(x_4)

- MUS predicate test: $\mathcal{W} \triangleq \mathcal{M} \setminus \{c_i\}$, $P(\mathcal{W}) \triangleq \neg\text{SAT}(\mathcal{W})$

c_i	$\mathcal{M}$	$\mathcal{M} \setminus \{c_i\}$	$P(\mathcal{W})$	Outcome
c_1	$c_1..c_7$	$c_2..c_7$	1	Drop c_1
c_2	$c_2..c_7$	$c_3..c_7$	1	Drop c_2
c_3	$c_3..c_7$	$c_4..c_7$	1	Drop c_3
c_4	$c_4..c_7$	$c_5..c_7$	0	Keep c_4
c_5	$c_4..c_7$	c_4, c_6, c_7	1	Drop c_5
c_6	c_4, c_6, c_7	c_4, c_7	0	Keep c_6
c_7	c_4, c_6, c_7	c_4, c_6	0	Keep c_7

Deletion – MUS example

c_1	c_2	c_3	c_4	c_5	c_6	c_7
$(\bar{x}_1 \vee \bar{x}_2)$	(x_1)	$(x_5 \vee x_6)$	$(\bar{x}_3 \vee \bar{x}_4)$	(x_2)	(x_3)	(x_4)

- MUS predicate test: $\mathcal{W} \triangleq \mathcal{M} \setminus \{c_i\}$, $P(\mathcal{W}) \triangleq \neg\mathrm{SAT}(\mathcal{W})$

c_i	$\mathcal{M}$	$\mathcal{M} \setminus \{c_i\}$	$P(\mathcal{W})$	Outcome
c_1	$c_1..c_7$	$c_2..c_7$	1	Drop c_1
c_2	$c_2..c_7$	$c_3..c_7$	1	Drop c_2
c_3	$c_3..c_7$	$c_4..c_7$	1	Drop c_3
c_4	$c_4..c_7$	$c_5..c_7$	0	Keep c_4
c_5	$c_4..c_7$	c_4, c_6, c_7	1	Drop c_5
c_6	c_4, c_6, c_7	c_4, c_7	0	Keep c_6
c_7	c_4, c_6, c_7	c_4, c_6	0	Keep c_7

- MUS: $\{c_4, c_6, c_7\}$

Deletion – MCS example

c_1	c_2	c_3	c_4	c_5	c_6	c_7
$(\bar{x}_1 \vee \bar{x}_2)$	(x_1)	$(x_5 \vee x_6)$	$(\bar{x}_3 \vee \bar{x}_4)$	(x_2)	(x_3)	(x_4)

- MCS predicate test: $\mathcal{W} \triangleq \mathcal{M} \setminus \{c_i\}$, $P(\mathcal{W}) \triangleq \text{SAT}(\mathcal{F} \setminus \mathcal{W})$

c_i	$\mathcal{M}$	$\mathcal{M} \setminus \{c_i\}$	$\mathcal{F} \setminus (\mathcal{M} \setminus \{c_i\})$	$P(\mathcal{W})$	Outcome

Deletion – MCS example

c_1	c_2	c_3	c_4	c_5	c_6	c_7
$(\bar{x}_1 \vee \bar{x}_2)$	(x_1)	$(x_5 \vee x_6)$	$(\bar{x}_3 \vee \bar{x}_4)$	(x_2)	(x_3)	(x_4)

- MCS predicate test: $\mathcal{W} \triangleq \mathcal{M} \setminus \{c_i\}$, $P(\mathcal{W}) \triangleq \text{SAT}(\mathcal{F} \setminus \mathcal{W})$

c_i	$\mathcal{M}$	$\mathcal{M} \setminus \{c_i\}$	$\mathcal{F} \setminus (\mathcal{M} \setminus \{c_i\})$	$P(\mathcal{W})$	Outcome
c_1	$c_1..c_7$	$c_2..c_7$	c_1	1	Drop c_1

Deletion – MCS example

c_1	c_2	c_3	c_4	c_5	c_6	c_7
$(\bar{x}_1 \vee \bar{x}_2)$	(x_1)	$(x_5 \vee x_6)$	$(\bar{x}_3 \vee \bar{x}_4)$	(x_2)	(x_3)	(x_4)

- MCS predicate test: $\mathcal{W} \triangleq \mathcal{M} \setminus \{c_i\}$, $P(\mathcal{W}) \triangleq \mathrm{SAT}(\mathcal{F} \setminus \mathcal{W})$

c_i	$\mathcal{M}$	$\mathcal{M} \setminus \{c_i\}$	$\mathcal{F} \setminus (\mathcal{M} \setminus \{c_i\})$	$P(\mathcal{W})$	Outcome
c_1	$c_1..c_7$	$c_2..c_7$	c_1	1	Drop c_1
c_2	$c_2..c_7$	$c_3..c_7$	c_1, c_2	1	Drop c_2

Deletion – MCS example

c_1	c_2	c_3	c_4	c_5	c_6	c_7
$(\bar{x}_1 \vee \bar{x}_2)$	(x_1)	$(x_5 \vee x_6)$	$(\bar{x}_3 \vee \bar{x}_4)$	(x_2)	(x_3)	(x_4)

- MCS predicate test: $\mathcal{W} \triangleq \mathcal{M} \setminus \{c_i\}$, $P(\mathcal{W}) \triangleq \text{SAT}(\mathcal{F} \setminus \mathcal{W})$

c_i	$\mathcal{M}$	$\mathcal{M} \setminus \{c_i\}$	$\mathcal{F} \setminus (\mathcal{M} \setminus \{c_i\})$	$P(\mathcal{W})$	Outcome
c_1	$c_1..c_7$	$c_2..c_7$	c_1	1	Drop c_1
c_2	$c_2..c_7$	$c_3..c_7$	c_1, c_2	1	Drop c_2
c_3	$c_3..c_7$	$c_4..c_7$	$c_1..c_3$	1	Drop c_3

Deletion – MCS example

c_1	c_2	c_3	c_4	c_5	c_6	c_7
$(\bar{x}_1 \vee \bar{x}_2)$	(x_1)	$(x_5 \vee x_6)$	$(\bar{x}_3 \vee \bar{x}_4)$	(x_2)	(x_3)	(x_4)

- MCS predicate test: $\mathcal{W} \triangleq \mathcal{M} \setminus \{c_i\}$, $P(\mathcal{W}) \triangleq \mathrm{SAT}(\mathcal{F} \setminus \mathcal{W})$

c_i	$\mathcal{M}$	$\mathcal{M} \setminus \{c_i\}$	$\mathcal{F} \setminus (\mathcal{M} \setminus \{c_i\})$	$P(\mathcal{W})$	Outcome
c_1	$c_1..c_7$	$c_2..c_7$	c_1	1	Drop c_1
c_2	$c_2..c_7$	$c_3..c_7$	c_1, c_2	1	Drop c_2
c_3	$c_3..c_7$	$c_4..c_7$	$c_1..c_3$	1	Drop c_3
c_4	$c_4..c_7$	$c_5..c_7$	$c_1..c_4$	1	Drop c_4

Deletion – MCS example

$$\underbrace{(\bar{x}_1 \vee \bar{x}_2)}_{c_1} \quad \underbrace{(x_1)}_{c_2} \quad \underbrace{(x_5 \vee x_6)}_{c_3} \quad \underbrace{(\bar{x}_3 \vee \bar{x}_4)}_{c_4} \quad \underbrace{(x_2)}_{c_5} \quad \underbrace{(x_3)}_{c_6} \quad \underbrace{(x_4)}_{c_7}$$

- MCS predicate test: $\boxed{\mathcal{W} \triangleq \mathcal{M} \setminus \{c_i\} \ , \ P(\mathcal{W}) \triangleq \mathrm{SAT}(\mathcal{F} \setminus \mathcal{W})}$

c_i	$\mathcal{M}$	$\mathcal{M} \setminus \{c_i\}$	$\mathcal{F} \setminus (\mathcal{M} \setminus \{c_i\})$	$P(\mathcal{W})$	Outcome
c_1	$c_1..c_7$	$c_2..c_7$	c_1	1	Drop c_1
c_2	$c_2..c_7$	$c_3..c_7$	c_1, c_2	1	Drop c_2
c_3	$c_3..c_7$	$c_4..c_7$	$c_1..c_3$	1	Drop c_3
c_4	$c_4..c_7$	$c_5..c_7$	$c_1..c_4$	1	Drop c_4
c_5	$c_5..c_7$	c_6, c_7	$c_1..c_5$	0	Keep c_5

Deletion – MCS example

c_1	c_2	c_3	c_4	c_5	c_6	c_7
$(\bar{x}_1 \vee \bar{x}_2)$	(x_1)	$(x_5 \vee x_6)$	$(\bar{x}_3 \vee \bar{x}_4)$	(x_2)	(x_3)	(x_4)

- MCS predicate test: $\mathcal{W} \triangleq \mathcal{M} \setminus \{c_i\}\ ,\ P(\mathcal{W}) \triangleq \mathrm{SAT}(\mathcal{F} \setminus \mathcal{W})$

c_i	$\mathcal{M}$	$\mathcal{M} \setminus \{c_i\}$	$\mathcal{F} \setminus (\mathcal{M} \setminus \{c_i\})$	$P(\mathcal{W})$	Outcome
c_1	$c_1..c_7$	$c_2..c_7$	c_1	1	Drop c_1
c_2	$c_2..c_7$	$c_3..c_7$	c_1, c_2	1	Drop c_2
c_3	$c_3..c_7$	$c_4..c_7$	$c_1..c_3$	1	Drop c_3
c_4	$c_4..c_7$	$c_5..c_7$	$c_1..c_4$	1	Drop c_4
c_5	$c_5..c_7$	c_6, c_7	$c_1..c_5$	0	Keep c_5
c_6	c_5, c_6, c_7	c_5, c_7	$c_1..c_4, c_6$	1	Drop c_6

Deletion – MCS example

c_1	c_2	c_3	c_4	c_5	c_6	c_7
$(\bar{x}_1 \vee \bar{x}_2)$	(x_1)	$(x_5 \vee x_6)$	$(\bar{x}_3 \vee \bar{x}_4)$	(x_2)	(x_3)	(x_4)

- MCS predicate test: $\mathcal{W} \triangleq \mathcal{M} \setminus \{c_i\}$, $P(\mathcal{W}) \triangleq \text{SAT}(\mathcal{F} \setminus \mathcal{W})$

c_i	$\mathcal{M}$	$\mathcal{M} \setminus \{c_i\}$	$\mathcal{F} \setminus (\mathcal{M} \setminus \{c_i\})$	$P(\mathcal{W})$	Outcome
c_1	$c_1..c_7$	$c_2..c_7$	c_1	1	Drop c_1
c_2	$c_2..c_7$	$c_3..c_7$	c_1, c_2	1	Drop c_2
c_3	$c_3..c_7$	$c_4..c_7$	$c_1..c_3$	1	Drop c_3
c_4	$c_4..c_7$	$c_5..c_7$	$c_1..c_4$	1	Drop c_4
c_5	$c_5..c_7$	c_6, c_7	$c_1..c_5$	0	Keep c_5
c_6	c_5, c_6, c_7	c_5, c_7	$c_1..c_4, c_6$	1	Drop c_6
c_7	c_5, c_7	c_5	$c_1..c_4, c_6, c_7$	0	Keep c_7

Deletion – MCS example

c_1	c_2	c_3	c_4	c_5	c_6	c_7
$(\bar{x}_1 \vee \bar{x}_2)$	(x_1)	$(x_5 \vee x_6)$	$(\bar{x}_3 \vee \bar{x}_4)$	(x_2)	(x_3)	(x_4)

- MCS predicate test: $\mathcal{W} \triangleq \mathcal{M} \setminus \{c_i\}$, $P(\mathcal{W}) \triangleq \mathrm{SAT}(\mathcal{F} \setminus \mathcal{W})$

c_i	$\mathcal{M}$	$\mathcal{M} \setminus \{c_i\}$	$\mathcal{F} \setminus (\mathcal{M} \setminus \{c_i\})$	$P(\mathcal{W})$	Outcome
c_1	$c_1..c_7$	$c_2..c_7$	c_1	1	Drop c_1
c_2	$c_2..c_7$	$c_3..c_7$	c_1, c_2	1	Drop c_2
c_3	$c_3..c_7$	$c_4..c_7$	$c_1..c_3$	1	Drop c_3
c_4	$c_4..c_7$	$c_5..c_7$	$c_1..c_4$	1	Drop c_4
c_5	$c_5..c_7$	c_6, c_7	$c_1..c_5$	0	Keep c_5
c_6	c_5, c_6, c_7	c_5, c_7	$c_1..c_4, c_6$	1	Drop c_6
c_7	c_5, c_7	c_5	$c_1..c_4, c_6, c_7$	0	Keep c_7

- MCS: $\{c_5, c_7\}$

Deletion – MCS example

c_1	c_2	c_3	c_4	c_5	c_6	c_7
$(\bar{x}_1 \vee \bar{x}_2)$	(x_1)	$(x_5 \vee x_6)$	$(\bar{x}_3 \vee \bar{x}_4)$	(x_2)	(x_3)	(x_4)

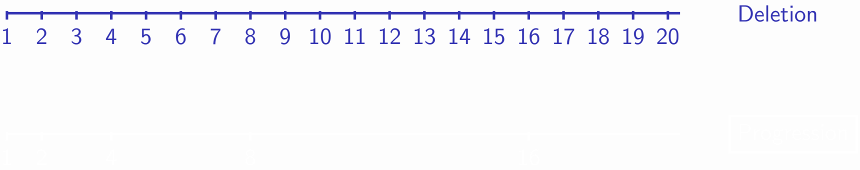
- MCS predicate test: $\mathcal{W} \triangleq \mathcal{M} \setminus \{c_i\}$, $P(\mathcal{W}) \triangleq \mathrm{SAT}(\mathcal{F} \setminus \mathcal{W})$

c_i	$\mathcal{M}$	$\mathcal{M} \setminus \{c_i\}$	$\mathcal{F} \setminus (\mathcal{M} \setminus \{c_i\})$	$P(\mathcal{W})$	Outcome
c_1	$c_1..c_7$	$c_2..c_7$	c_1	1	Drop c_1
c_2	$c_2..c_7$	$c_3..c_7$	c_1, c_2	1	Drop c_2
c_3	$c_3..c_7$	$c_4..c_7$	$c_1..c_3$	1	Drop c_3
c_4	$c_4..c_7$	$c_5..c_7$	$c_1..c_4$	1	Drop c_4
c_5	$c_5..c_7$	c_6, c_7	$c_1..c_5$	0	Keep c_5
c_6	c_5, c_6, c_7	c_5, c_7	$c_1..c_4, c_6$	1	Drop c_6
c_7	c_5, c_7	c_5	$c_1..c_4, c_6, c_7$	0	Keep c_7

- MCS: $\{c_5, c_7\}$

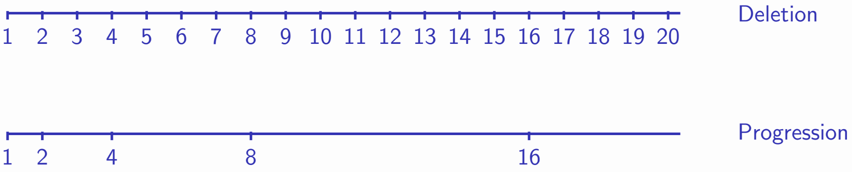
Compare with std MSS grow procedure!

From deletion to progression



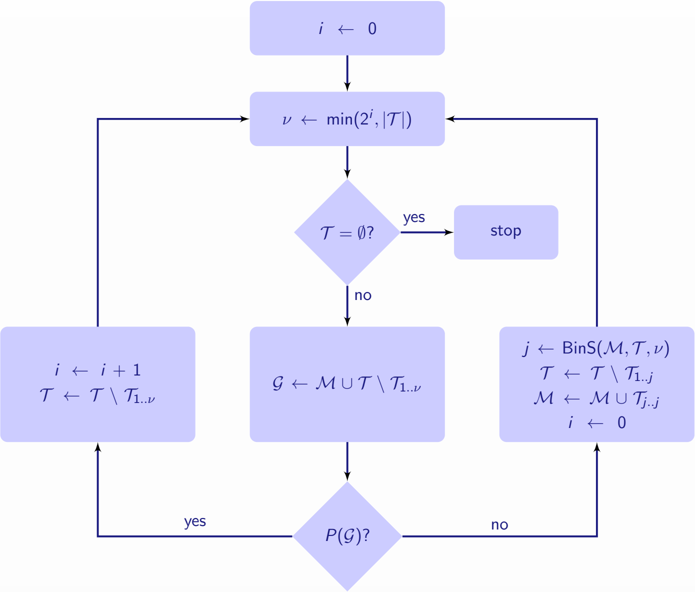
- Deletion: Check (& remove?) one element at a time

From deletion to progression

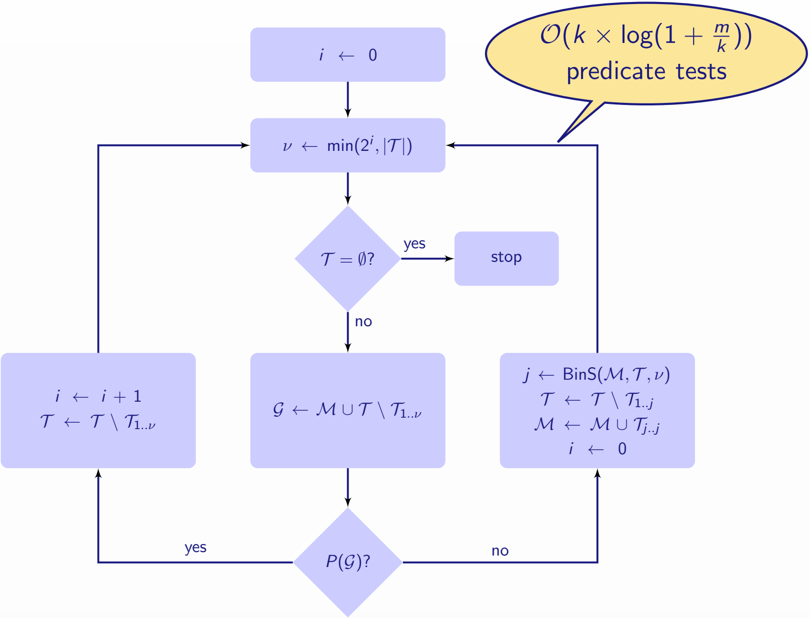


- Deletion: Check (& remove?) one element at a time
  - Pick set of elements given by **arithmetic** progression
- Progression: Check (& remove) exponentially growing set of elements
  - Pick set of elements given by **geometric** progression

Progression algorithm



Progression algorithm



Progression – MUS example

c_1	c_2	c_3	c_4	c_5	c_6	c_7
$(\bar{x}_1 \vee \bar{x}_2)$	(x_1)	$(x_5 \vee x_6)$	$(\bar{x}_3 \vee \bar{x}_4)$	(x_2)	(x_3)	(x_4)

- MUS predicate test: $\mathcal{W} \triangleq \mathcal{M} \cup \mathcal{T} \setminus \mathcal{T}_{1..\nu}$, $P(\mathcal{W}) \triangleq \neg\mathrm{SAT}(\mathcal{W})$

i	$\nu = \min(2^i, \lvert\mathcal{T}\rvert)$	$\mathcal{M}$	$\mathcal{T}$	$\mathcal{T} \setminus \mathcal{T}_{1..\nu}$	$P(\mathcal{W})$	BinSearch

Progression – MUS example

c_1	c_2	c_3	c_4	c_5	c_6	c_7
$(\bar{x}_1 \vee \bar{x}_2)$	(x_1)	$(x_5 \vee x_6)$	$(\bar{x}_3 \vee \bar{x}_4)$	(x_2)	(x_3)	(x_4)

- MUS predicate test: $\mathcal{W} \triangleq \mathcal{M} \cup \mathcal{T} \setminus \mathcal{T}_{1..\nu}$, $P(\mathcal{W}) \triangleq \neg\mathrm{SAT}(\mathcal{W})$

i	$\nu = \min(2^i, \|\mathcal{T}\|)$	$\mathcal{M}$	$\mathcal{T}$	$\mathcal{T} \setminus \mathcal{T}_{1..\nu}$	$P(\mathcal{W})$	BinSearch
0	1	$\emptyset$	$c_1..c_7$	$c_2..c_7$	1	–

Progression – MUS example

c_1	c_2	c_3	c_4	c_5	c_6	c_7
$(\bar{x}_1 \vee \bar{x}_2)$	(x_1)	$(x_5 \vee x_6)$	$(\bar{x}_3 \vee \bar{x}_4)$	(x_2)	(x_3)	(x_4)

- MUS predicate test: $\mathcal{W} \triangleq \mathcal{M} \cup \mathcal{T} \setminus \mathcal{T}_{1..\nu}$, $P(\mathcal{W}) \triangleq \neg\mathrm{SAT}(\mathcal{W})$

i	$\nu = \min(2^i, \lvert\mathcal{T}\rvert)$	$\mathcal{M}$	$\mathcal{T}$	$\mathcal{T} \setminus \mathcal{T}_{1..\nu}$	$P(\mathcal{W})$	BinSearch
0	1	$\emptyset$	$c_1..c_7$	$c_2..c_7$	1	–
1	2	$\emptyset$	$c_2..c_7$	$c_4..c_7$	1	–

Progression – MUS example

c_1	c_2	c_3	c_4	c_5	c_6	c_7
$(\bar{x}_1 \vee \bar{x}_2)$	(x_1)	$(x_5 \vee x_6)$	$(\bar{x}_3 \vee \bar{x}_4)$	(x_2)	(x_3)	(x_4)

- MUS predicate test: $\mathcal{W} \triangleq \mathcal{M} \cup \mathcal{T} \setminus \mathcal{T}_{1..\nu}$, $P(\mathcal{W}) \triangleq \neg\mathrm{SAT}(\mathcal{W})$

i	$\nu = \min(2^i, \lvert\mathcal{T}\rvert)$	$\mathcal{M}$	$\mathcal{T}$	$\mathcal{T} \setminus \mathcal{T}_{1..\nu}$	$P(\mathcal{W})$	BinSearch
0	1	$\emptyset$	$c_1..c_7$	$c_2..c_7$	1	–
1	2	$\emptyset$	$c_2..c_7$	$c_4..c_7$	1	–
2	4	$\emptyset$	$c_4..c_7$	$\emptyset$	0	c_4

Progression – MUS example

c_1	c_2	c_3	c_4	c_5	c_6	c_7
$(\bar{x}_1 \vee \bar{x}_2)$	(x_1)	$(x_5 \vee x_6)$	$(\bar{x}_3 \vee \bar{x}_4)$	(x_2)	(x_3)	(x_4)

- MUS predicate test: $\mathcal{W} \triangleq \mathcal{M} \cup \mathcal{T} \setminus \mathcal{T}_{1..\nu}$, $P(\mathcal{W}) \triangleq \neg \mathrm{SAT}(\mathcal{W})$

i	$\nu = \min(2^i, \lvert\mathcal{T}\rvert)$	$\mathcal{M}$	$\mathcal{T}$	$\mathcal{T} \setminus \mathcal{T}_{1..\nu}$	$P(\mathcal{W})$	BinSearch
0	1	$\emptyset$	$c_1..c_7$	$c_2..c_7$	1	–
1	2	$\emptyset$	$c_2..c_7$	$c_4..c_7$	1	–
2	4	$\emptyset$	$c_4..c_7$	$\emptyset$	0	c_4
0	1	c_4	$c_5..c_7$	$c_6..c_7$	1	–

Progression – MUS example

c_1	c_2	c_3	c_4	c_5	c_6	c_7
$(\bar{x}_1 \vee \bar{x}_2)$	(x_1)	$(x_5 \vee x_6)$	$(\bar{x}_3 \vee \bar{x}_4)$	(x_2)	(x_3)	(x_4)

- MUS predicate test: $\mathcal{W} \triangleq \mathcal{M} \cup \mathcal{T} \setminus \mathcal{T}_{1..\nu} \quad , \ P(\mathcal{W}) \triangleq \neg\mathrm{SAT}(\mathcal{W})$

i	$\nu = \min(2^i, \lvert\mathcal{T}\rvert)$	$\mathcal{M}$	$\mathcal{T}$	$\mathcal{T} \setminus \mathcal{T}_{1..\nu}$	$P(\mathcal{W})$	BinSearch
0	1	$\emptyset$	$c_1..c_7$	$c_2..c_7$	1	–
1	2	$\emptyset$	$c_2..c_7$	$c_4..c_7$	1	–
2	4	$\emptyset$	$c_4..c_7$	$\emptyset$	0	c_4
0	1	c_4	$c_5..c_7$	$c_6..c_7$	1	–
1	2	c_4	$c_6..c_7$	$\emptyset$	0	c_6

Progression – MUS example

c_1	c_2	c_3	c_4	c_5	c_6	c_7
$(\bar{x}_1 \vee \bar{x}_2)$	(x_1)	$(x_5 \vee x_6)$	$(\bar{x}_3 \vee \bar{x}_4)$	(x_2)	(x_3)	(x_4)

- MUS predicate test: $\mathcal{W} \triangleq \mathcal{M} \cup \mathcal{T} \setminus \mathcal{T}_{1..\nu}$, $P(\mathcal{W}) \triangleq \neg \mathrm{SAT}(\mathcal{W})$

i	$\nu = \min(2^i, \lvert\mathcal{T}\rvert)$	$\mathcal{M}$	$\mathcal{T}$	$\mathcal{T} \setminus \mathcal{T}_{1..\nu}$	$P(\mathcal{W})$	BinSearch
0	1	$\emptyset$	$c_1..c_7$	$c_2..c_7$	1	–
1	2	$\emptyset$	$c_2..c_7$	$c_4..c_7$	1	–
2	4	$\emptyset$	$c_4..c_7$	$\emptyset$	0	c_4
0	1	c_4	$c_5..c_7$	$c_6..c_7$	1	–
1	2	c_4	$c_6..c_7$	$\emptyset$	0	c_6
0	1	c_4, c_6	c_7	$\emptyset$	0	c_7

Progression – MUS example

c_1	c_2	c_3	c_4	c_5	c_6	c_7
$(\bar{x}_1 \vee \bar{x}_2)$	(x_1)	$(x_5 \vee x_6)$	$(\bar{x}_3 \vee \bar{x}_4)$	(x_2)	(x_3)	(x_4)

- MUS predicate test: $\mathcal{W} \triangleq \mathcal{M} \cup \mathcal{T} \setminus \mathcal{T}_{1..\nu}$, $P(\mathcal{W}) \triangleq \neg\text{SAT}(\mathcal{W})$

i	$\nu = \min(2^i, \|\mathcal{T}\|)$	$\mathcal{M}$	$\mathcal{T}$	$\mathcal{T} \setminus \mathcal{T}_{1..\nu}$	$P(\mathcal{W})$	BinSearch
0	1	$\emptyset$	$c_1..c_7$	$c_2..c_7$	1	–
1	2	$\emptyset$	$c_2..c_7$	$c_4..c_7$	1	–
2	4	$\emptyset$	$c_4..c_7$	$\emptyset$	0	c_4
0	1	c_4	$c_5..c_7$	$c_6..c_7$	1	–
1	2	c_4	$c_6..c_7$	$\emptyset$	0	c_6
0	1	c_4, c_6	c_7	$\emptyset$	0	c_7
0	–	c_4, c_6, c_7	$\emptyset$	–	–	–

Progression – MUS example

c_1	c_2	c_3	c_4	c_5	c_6	c_7
$(\bar{x}_1 \vee \bar{x}_2)$	(x_1)	$(x_5 \vee x_6)$	$(\bar{x}_3 \vee \bar{x}_4)$	(x_2)	(x_3)	(x_4)

- MUS predicate test: $\mathcal{W} \triangleq \mathcal{M} \cup \mathcal{T} \setminus \mathcal{T}_{1..\nu}$, $P(\mathcal{W}) \triangleq \neg\mathrm{SAT}(\mathcal{W})$

i	$\nu = \min(2^i, \lvert\mathcal{T}\rvert)$	$\mathcal{M}$	$\mathcal{T}$	$\mathcal{T} \setminus \mathcal{T}_{1..\nu}$	$P(\mathcal{W})$	BinSearch
0	1	$\emptyset$	$c_1..c_7$	$c_2..c_7$	1	–
1	2	$\emptyset$	$c_2..c_7$	$c_4..c_7$	1	–
2	4	$\emptyset$	$c_4..c_7$	$\emptyset$	0	c_4
0	1	c_4	$c_5..c_7$	$c_6..c_7$	1	–
1	2	c_4	$c_6..c_7$	$\emptyset$	0	c_6
0	1	c_4, c_6	c_7	$\emptyset$	0	c_7
0	–	c_4, c_6, c_7	$\emptyset$	–	–	–

- MUS: $\{c_4, c_6, c_7\}$

Progression – MUS example

c_1	c_2	c_3	c_4	c_5	c_6	c_7
$(\bar{x}_1 \vee \bar{x}_2)$	(x_1)	$(x_5 \vee x_6)$	$(\bar{x}_3 \vee \bar{x}_4)$	(x_2)	(x_3)	(x_4)

- MUS predicate test: $\mathcal{W} \triangleq \mathcal{M} \cup \mathcal{T} \setminus \mathcal{T}_{1..\nu}$, $P(\mathcal{W}) \triangleq \neg\mathrm{SAT}(\mathcal{W})$

i	$\nu = \min(2^i, \lvert\mathcal{T}\rvert)$	$\mathcal{M}$	$\mathcal{T}$	$\mathcal{T} \setminus \mathcal{T}_{1..\nu}$	$P(\mathcal{W})$	BinSearch
0	1	$\emptyset$	$c_1..c_7$	$c_2..c_7$	1	–
1	2	$\emptyset$	$c_2..c_7$	$c_4..c_7$	1	–
2	4	$\emptyset$	$c_4..c_7$	$\emptyset$	0	c_4
0	1	c_4	$c_5..c_7$	$c_6..c_7$	1	–
1	2	c_4	$c_6..c_7$	$\emptyset$	0	c_6
0	1	c_4, c_6	c_7	$\emptyset$	0	c_7
0	–	c_4, c_6, c_7	$\emptyset$	–	–	–

- MUS: $\{c_4, c_6, c_7\}$

BinSearch gets elements of $\mathcal{M}$

Progression – MCS example

c_1	c_2	c_3	c_4	c_5	c_6	c_7
$(\bar{x}_1 \vee \bar{x}_2)$	(x_1)	$(x_5 \vee x_6)$	$(\bar{x}_3 \vee \bar{x}_4)$	(x_2)	(x_3)	(x_4)

- MCS predicate test: $\mathcal{W} \triangleq \mathcal{M} \cup \mathcal{T} \setminus \mathcal{T}_{1..\nu}$, $P(\mathcal{W}) \triangleq \mathrm{SAT}(\mathcal{F} \setminus \mathcal{W})$

i	$\nu = (\cdot)$	$\mathcal{M}$	$\mathcal{T}$	$\mathcal{F} \setminus (\mathcal{M} \cup \mathcal{T} \setminus \mathcal{T}_{1..\nu})$	$P(\mathcal{W})$	BinSearch

Progression – MCS example

c_1	c_2	c_3	c_4	c_5	c_6	c_7
$(\bar{x}_1 \vee \bar{x}_2)$	(x_1)	$(x_5 \vee x_6)$	$(\bar{x}_3 \vee \bar{x}_4)$	(x_2)	(x_3)	(x_4)

- MCS predicate test: $\mathcal{W} \triangleq \mathcal{M} \cup \mathcal{T} \setminus \mathcal{T}_{1..\nu}$, $P(\mathcal{W}) \triangleq \mathrm{SAT}(\mathcal{F} \setminus \mathcal{W})$

i	$\nu = (\cdot)$	$\mathcal{M}$	$\mathcal{T}$	$\mathcal{F} \setminus (\mathcal{M} \cup \mathcal{T} \setminus \mathcal{T}_{1..\nu})$	$P(\mathcal{W})$	BinSearch
0	1	$\emptyset$	$c_1..c_7$	c_1	1	–

Progression – MCS example

c_1	c_2	c_3	c_4	c_5	c_6	c_7
$(\bar{x}_1 \vee \bar{x}_2)$	(x_1)	$(x_5 \vee x_6)$	$(\bar{x}_3 \vee \bar{x}_4)$	(x_2)	(x_3)	(x_4)

- MCS predicate test: $\mathcal{W} \triangleq \mathcal{M} \cup \mathcal{T} \setminus \mathcal{T}_{1..\nu}$, $P(\mathcal{W}) \triangleq \text{SAT}(\mathcal{F} \setminus \mathcal{W})$

i	$\nu = (\cdot)$	$\mathcal{M}$	$\mathcal{T}$	$\mathcal{F} \setminus (\mathcal{M} \cup \mathcal{T} \setminus \mathcal{T}_{1..\nu})$	$P(\mathcal{W})$	BinSearch
0	1	$\emptyset$	$c_1..c_7$	c_1	1	–
1	2	$\emptyset$	$c_2..c_7$	c_1, c_2, c_3	1	–

Progression – MCS example

c_1	c_2	c_3	c_4	c_5	c_6	c_7
$(\bar{x}_1 \vee \bar{x}_2)$	(x_1)	$(x_5 \vee x_6)$	$(\bar{x}_3 \vee \bar{x}_4)$	(x_2)	(x_3)	(x_4)

- MCS predicate test: $\mathcal{W} \triangleq \mathcal{M} \cup \mathcal{T} \setminus \mathcal{T}_{1..\nu}$, $P(\mathcal{W}) \triangleq \mathrm{SAT}(\mathcal{F} \setminus \mathcal{W})$

i	$\nu = (\cdot)$	$\mathcal{M}$	$\mathcal{T}$	$\mathcal{F} \setminus (\mathcal{M} \cup \mathcal{T} \setminus \mathcal{T}_{1..\nu})$	$P(\mathcal{W})$	BinSearch
0	1	$\emptyset$	$c_1..c_7$	c_1	1	–
1	2	$\emptyset$	$c_2..c_7$	c_1, c_2, c_3	1	–
2	4	$\emptyset$	$c_4..c_7$	$c_1..c_7$	0	c_5

Progression – MCS example

c_1	c_2	c_3	c_4	c_5	c_6	c_7
$(\bar{x}_1 \vee \bar{x}_2)$	(x_1)	$(x_5 \vee x_6)$	$(\bar{x}_3 \vee \bar{x}_4)$	(x_2)	(x_3)	(x_4)

- MCS predicate test: $\mathcal{W} \triangleq \mathcal{M} \cup \mathcal{T} \setminus \mathcal{T}_{1..\nu}$, $P(\mathcal{W}) \triangleq \mathrm{SAT}(\mathcal{F} \setminus \mathcal{W})$

i	$\nu = (\cdot)$	$\mathcal{M}$	$\mathcal{T}$	$\mathcal{F} \setminus (\mathcal{M} \cup \mathcal{T} \setminus \mathcal{T}_{1..\nu})$	$P(\mathcal{W})$	BinSearch
0	1	$\emptyset$	$c_1..c_7$	c_1	1	–
1	2	$\emptyset$	$c_2..c_7$	c_1, c_2, c_3	1	–
2	4	$\emptyset$	$c_4..c_7$	$c_1..c_7$	0	c_5
0	1	c_5	$c_6..c_7$	$c_1..c_4, c_6$	1	–

Progression – MCS example

c_1	c_2	c_3	c_4	c_5	c_6	c_7
$(\bar{x}_1 \vee \bar{x}_2)$	(x_1)	$(x_5 \vee x_6)$	$(\bar{x}_3 \vee \bar{x}_4)$	(x_2)	(x_3)	(x_4)

- MCS predicate test: $\mathcal{W} \triangleq \mathcal{M} \cup \mathcal{T} \setminus \mathcal{T}_{1..\nu}$, $P(\mathcal{W}) \triangleq \mathrm{SAT}(\mathcal{F} \setminus \mathcal{W})$

i	$\nu = (\cdot)$	$\mathcal{M}$	$\mathcal{T}$	$\mathcal{F} \setminus (\mathcal{M} \cup \mathcal{T} \setminus \mathcal{T}_{1..\nu})$	$P(\mathcal{W})$	BinSearch
0	1	$\emptyset$	$c_1..c_7$	c_1	1	–
1	2	$\emptyset$	$c_2..c_7$	c_1, c_2, c_3	1	–
2	4	$\emptyset$	$c_4..c_7$	$c_1..c_7$	0	c_5
0	1	c_5	$c_6..c_7$	$c_1..c_4, c_6$	1	–
1	1	c_5	c_7	$c_1..c_4, c_6, c_7$	0	c_7

Progression – MCS example

c_1	c_2	c_3	c_4	c_5	c_6	c_7
$(\bar{x}_1 \vee \bar{x}_2)$	(x_1)	$(x_5 \vee x_6)$	$(\bar{x}_3 \vee \bar{x}_4)$	(x_2)	(x_3)	(x_4)

- MCS predicate test: $\mathcal{W} \triangleq \mathcal{M} \cup \mathcal{T} \setminus \mathcal{T}_{1..\nu} \quad , \ P(\mathcal{W}) \triangleq \text{SAT}(\mathcal{F} \setminus \mathcal{W})$

i	$\nu = (\cdot)$	$\mathcal{M}$	$\mathcal{T}$	$\mathcal{F} \setminus (\mathcal{M} \cup \mathcal{T} \setminus \mathcal{T}_{1..\nu})$	$P(\mathcal{W})$	BinSearch
0	1	$\emptyset$	$c_1..c_7$	c_1	1	–
1	2	$\emptyset$	$c_2..c_7$	c_1, c_2, c_3	1	–
2	4	$\emptyset$	$c_4..c_7$	$c_1..c_7$	0	c_5
0	1	c_5	$c_6..c_7$	$c_1..c_4, c_6$	1	–
1	1	c_5	c_7	$c_1..c_4, c_6, c_7$	0	c_7
0	–	c_5, c_7	$\emptyset$	–	–	–

Progression – MCS example

c_1	c_2	c_3	c_4	c_5	c_6	c_7
$(\bar{x}_1 \vee \bar{x}_2)$	(x_1)	$(x_5 \vee x_6)$	$(\bar{x}_3 \vee \bar{x}_4)$	(x_2)	(x_3)	(x_4)

- MCS predicate test: $\mathcal{W} \triangleq \mathcal{M} \cup \mathcal{T} \setminus \mathcal{T}_{1..\nu} \;\;,\; P(\mathcal{W}) \triangleq \mathrm{SAT}(\mathcal{F} \setminus \mathcal{W})$

i	$\nu = (\cdot)$	$\mathcal{M}$	$\mathcal{T}$	$\mathcal{F} \setminus (\mathcal{M} \cup \mathcal{T} \setminus \mathcal{T}_{1..\nu})$	$P(\mathcal{W})$	BinSearch
0	1	$\emptyset$	$c_1..c_7$	c_1	1	–
1	2	$\emptyset$	$c_2..c_7$	c_1, c_2, c_3	1	–
2	4	$\emptyset$	$c_4..c_7$	$c_1..c_7$	0	c_5
0	1	c_5	$c_6..c_7$	$c_1..c_4, c_6$	1	–
1	1	c_5	c_7	$c_1..c_4, c_6, c_7$	0	c_7
0	–	c_5, c_7	$\emptyset$	–	–	–

- MCS: $\{c_5, c_7\}$

Progression – MCS example

c_1	c_2	c_3	c_4	c_5	c_6	c_7
$(\bar{x}_1 \vee \bar{x}_2)$	(x_1)	$(x_5 \vee x_6)$	$(\bar{x}_3 \vee \bar{x}_4)$	(x_2)	(x_3)	(x_4)

- MCS predicate test: $\mathcal{W} \triangleq \mathcal{M} \cup \mathcal{T} \setminus \mathcal{T}_{1..\nu}$, $P(\mathcal{W}) \triangleq \mathrm{SAT}(\mathcal{F} \setminus \mathcal{W})$

i	$\nu = (\cdot)$	$\mathcal{M}$	$\mathcal{T}$	$\mathcal{F} \setminus (\mathcal{M} \cup \mathcal{T} \setminus \mathcal{T}_{1..\nu})$	$P(\mathcal{W})$	BinSearch
0	1	$\emptyset$	$c_1..c_7$	c_1	1	–
1	2	$\emptyset$	$c_2..c_7$	c_1, c_2, c_3	1	–
2	4	$\emptyset$	$c_4..c_7$	$c_1..c_7$	0	c_5
0	1	c_5	$c_6..c_7$	$c_1..c_4, c_6$	1	–
1	1	c_5	c_7	$c_1..c_4, c_6, c_7$	0	c_7
0	–	c_5, c_7	$\emptyset$	–	–	–

- MCS: $\{c_5, c_7\}$

BinSearch gets elements of $\mathcal{M}$

Outline

Revisit MUSes

Minimal Sets

Query Complexity

Conclusions

Oracles and query complexity

- Disclaimer: Ongoing work; comments welcome!

Oracles and query complexity

- NP oracles vs. witness oracles
  - Given instance:

	NP oracle	witness oracle
Accepts(**Y**) / Rejects(**N**)	✓	✓
Returns poly-size **Y** witness	✗	✓

Oracles and query complexity

- NP oracles vs. witness oracles
  - Given instance:

	NP oracle	witness oracle
Accepts(**Y**) / Rejects(**N**)	✓	✓
Returns poly-size **Y** witness	✗	✓

- SAT solvers produce witnesses for **Y** outcomes
  - SAT solvers correspond to witness oracles, i.e. **SAT oracles**

Oracles and query complexity

- NP oracles vs. witness oracles
  - Given instance:

	NP oracle	witness oracle
Accepts(**Y**) / Rejects(**N**)	✓	✓
Returns poly-size **Y** witness	✗	✓

- SAT solvers produce witnesses for **Y** outcomes
  - SAT solvers correspond to witness oracles, i.e. **SAT oracles**

- Some complexity classes for function problems: [e.g. P94,BKT93,JT95]

NP oracles	$\mathrm{FP}^{\mathrm{NP}}[\log n] \subseteq \mathrm{FP}^{\mathrm{NP}}_{\parallel} \subseteq \mathrm{FP}^{\mathrm{NP}}$
witness oracles	$\mathrm{FP}^{\mathrm{NP}}[\mathrm{wit}, \log n]$

Oracles and query complexity

- NP oracles vs. witness oracles
  - Given instance:

	NP oracle	witness oracle
Accepts(**Y**) / Rejects(**N**)	✓	✓
Returns poly-size **Y** witness	✗	✓

- SAT solvers produce witnesses for **Y** outcomes
  - SAT solvers correspond to witness oracles, i.e. **SAT oracles**

- Some complexity classes for function problems: [e.g. P94,BKT93,JT95]

NP oracles	$\mathrm{FP}^{\mathrm{NP}}[\log n] \overset{?}{\subsetneq} \mathrm{FP}^{\mathrm{NP}}_{\vert\vert} \overset{?}{\subsetneq} \mathrm{FP}^{\mathrm{NP}}$
witness oracles	$\mathrm{FP}^{\mathrm{NP}}[\mathrm{wit}, \log n]$

Preliminary results

- # of queries to SAT/NP oracle for solving selected (possibly restricted) function problems:

Preliminary results

- # of queries to SAT/NP oracle for solving selected (possibly restricted) function problems:
  - Backbones: literals common to all models of $\mathcal{F}$ [MZKST99,MSJL10,ZWSM11]
    - Assume reference model
    - Algorithm: use one query to check each literal

Preliminary results

- $\#$ of queries to SAT/NP oracle for solving selected (possibly restricted) function problems:
  - Backbones: literals common to all models of $\mathcal{F}$ [MZKST99,MSJL10,ZWSM11]
    - Assume reference model
    - Algorithm: use one query to check each literal

Best: $\mathcal{O}(V - B)$ calls [ZWSM11]

Preliminary results

- # of queries to SAT/NP oracle for solving selected (possibly restricted) function problems:
  - Backbones: literals common to all models of $\mathcal{F}$ [MZKST99,MSJL10,ZWSM11]
    - Assume reference model
    - Algorithm: use one query to check each literal
  - $\text{MUS}_{\#1}$: compute MUS for formulas with exactly 1 MUS

Preliminary results

- # of queries to SAT/NP oracle for solving selected (possibly restricted) function problems:
  - Backbones: literals common to all models of $\mathcal{F}$ [MZKST99,MSJL10,ZWSM11]
    - Assume reference model
    - Algorithm: use one query to check each literal
  - $\text{MUS}_{\#1}$: compute MUS for formulas with exactly 1 MUS

Best $\mathcal{O}(|\mathcal{F}|)$?

Preliminary results

- # of queries to SAT/NP oracle for solving selected (possibly restricted) function problems:
  - Backbones: literals common to all models of $\mathcal{F}$ [MZKST99,MSJL10,ZWSM11]
    - Assume reference model
    - Algorithm: use one query to check each literal
  - $\text{MUS}_{\#1}$: compute MUS for formulas with exactly 1 MUS

Problem	NP Oracles	SAT Oracles
MCS		

Preliminary results

- $\#$ of queries to SAT/NP oracle for solving selected (possibly restricted) function problems:
  - Backbones: literals common to all models of $\mathcal{F}$ [MZKST99,MSJL10,ZWSM11]
    - Assume reference model
    - Algorithm: use one query to check each literal
  - $\text{MUS}_{\#1}$: compute MUS for formulas with exactly 1 MUS

Problem	NP Oracles	SAT Oracles
MCS	$\mathcal{O}(n)$	

Preliminary results

- # of queries to SAT/NP oracle for solving selected (possibly restricted) function problems:
  - Backbones: literals common to all models of $\mathcal{F}$ [MZKST99,MSJL10,ZWSM11]
    - Assume reference model
    - Algorithm: use one query to check each literal
  - $\text{MUS}_{\#1}$: compute MUS for formulas with exactly 1 MUS

Problem	NP Oracles	SAT Oracles
MCS	$\mathcal{O}(n)$	$\mathcal{O}(\log n)$
MUS		

Preliminary results

- # of queries to SAT/NP oracle for solving selected (possibly restricted) function problems:
  - Backbones: literals common to all models of $\mathcal{F}$ [MZKST99,MSJL10,ZWSM11]
    - Assume reference model
    - Algorithm: use one query to check each literal
  - $\text{MUS}_{\#1}$: compute MUS for formulas with exactly 1 MUS

Problem	NP Oracles	SAT Oracles
MCS	$\mathcal{O}(n)$	$\mathcal{O}(\log n)$
MUS	$\mathcal{O}(n)$	

Preliminary results

- # of queries to SAT/NP oracle for solving selected (possibly restricted) function problems:
  - Backbones: literals common to all models of $\mathcal{F}$ [MZKST99,MSJL10,ZWSM11]
    - Assume reference model
    - Algorithm: use one query to check each literal
  - $\text{MUS}_{\#1}$: compute MUS for formulas with exactly 1 MUS

Problem	NP Oracles	SAT Oracles
MCS	$\mathcal{O}(n)$	$\mathcal{O}(\log n)$
MUS	$\mathcal{O}(n)$	$\mathcal{O}(n)$

Preliminary results

- \# of queries to SAT/NP oracle for solving selected (possibly restricted) function problems:
  - Backbones: literals common to all models of $\mathcal{F}$ [MZKST99,MSJL10,ZWSM11]
    - Assume reference model
    - Algorithm: use one query to check each literal
  - $\text{MUS}_{\#1}$: compute MUS for formulas with exactly 1 MUS

Problem	NP Oracles	SAT Oracles
MCS	$\mathcal{O}(n)$	$\mathcal{O}(\log n)$
MUS	$\mathcal{O}(n)$	$\mathcal{O}(n)$

Best $\mathcal{O}(|\mathcal{F}|)$?

Preliminary results

- $\#$ of queries to SAT/NP oracle for solving selected (possibly restricted) function problems:
  - Backbones: literals common to all models of $\mathcal{F}$ [MZKST99,MSJL10,ZWSM11]
    - Assume reference model
    - Algorithm: use one query to check each literal
  - $\text{MUS}_{\#1}$: compute MUS for formulas with exactly 1 MUS

Problem	NP Oracles	SAT Oracles
MCS	$\mathcal{O}(n)$	$\mathcal{O}(\log n)$
MUS	$\mathcal{O}(n)$	$\mathcal{O}(n)$
Backbones		

Preliminary results

- \# of queries to SAT/NP oracle for solving selected (possibly restricted) function problems:
  - Backbones: literals common to all models of $\mathcal{F}$ [MZKST99,MSJL10,ZWSM11]
    - Assume reference model
    - Algorithm: use one query to check each literal
  - $\text{MUS}_{\#1}$: compute MUS for formulas with exactly 1 MUS

Problem	NP Oracles	SAT Oracles
MCS	$\mathcal{O}(n)$	$\mathcal{O}(\log n)$
MUS	$\mathcal{O}(n)$	$\mathcal{O}(n)$
Backbones	$\mathcal{O}(n)$, $\|\|$	

Preliminary results

- $\#$ of queries to SAT/NP oracle for solving selected (possibly restricted) function problems:
  - Backbones: literals common to all models of $\mathcal{F}$ [MZKST99,MSJL10,ZWSM11]
    - Assume reference model
    - Algorithm: use one query to check each literal
  - $\text{MUS}_{\#1}$: compute MUS for formulas with exactly 1 MUS

Problem	NP Oracles	SAT Oracles
MCS	$\mathcal{O}(n)$	$\mathcal{O}(\log n)$
MUS	$\mathcal{O}(n)$	$\mathcal{O}(n)$
Backbones	$\mathcal{O}(n)$, $\|\|$	
$\text{MUS}_{\#1}$		

Preliminary results

- # of queries to SAT/NP oracle for solving selected (possibly restricted) function problems:
  - Backbones: literals common to all models of $\mathcal{F}$ [MZKST99,MSJL10,ZWSM11]
    - Assume reference model
    - Algorithm: use one query to check each literal
  - $\text{MUS}_{\#1}$: compute MUS for formulas with exactly 1 MUS

Problem	NP Oracles	SAT Oracles
MCS	$\mathcal{O}(n)$	$\mathcal{O}(\log n)$
MUS	$\mathcal{O}(n)$	$\mathcal{O}(n)$
Backbones	$\mathcal{O}(n)$, $\|\|$	
$\text{MUS}_{\#1}$	$\mathcal{O}(n)$, $\|\|$	

Preliminary results

- # of queries to SAT/NP oracle for solving selected (possibly restricted) function problems:
  - Backbones: literals common to all models of $\mathcal{F}$ [MZKST99,MSJL10,ZWSM11]
    - Assume reference model
    - Algorithm: use one query to check each literal
  - $\text{MUS}_{\#1}$: compute MUS for formulas with exactly 1 MUS

Problem	NP Oracles	SAT Oracles
MCS	$\mathcal{O}(n)$	$\mathcal{O}(\log n)$
MUS	$\mathcal{O}(n)$	$\mathcal{O}(n)$
Backbones	$\mathcal{O}(n)$, $\|\|$	$\mathcal{O}(\log n)$
$\text{MUS}_{\#1}$	$\mathcal{O}(n)$, $\|\|$	$\mathcal{O}(\log n)$

- **Why?**

Preliminary results

- # of queries to SAT/NP oracle for solving selected (possibly restricted) function problems:
  - Backbones: literals common to all models of $\mathcal{F}$ [MZKST99,MSJL10,ZWSM11]
    - Assume reference model
    - Algorithm: use one query to check each literal
  - $\text{MUS}_{\#1}$: compute MUS for formulas with exactly 1 MUS

Problem	NP Oracles	SAT Oracles
MCS	FP^{NP}	$\text{FP}^{\text{NP}}[\text{wit}, \log n]$
MUS	FP^{NP}	FP^{NP}
Backbones	$\text{FP}^{\text{NP}}_{\|\|}$	$\text{FP}^{\text{NP}}[\text{wit}, \log n]$
$\text{MUS}_{\#1}$	$\text{FP}^{\text{NP}}_{\|\|}$	$\text{FP}^{\text{NP}}[\text{wit}, \log n]$

- **Why?**

Preliminary results

- $\#$ of queries to SAT/NP oracle for solving selected (possibly restricted) function problems:
  - Backbones: literals common to all models of $\mathcal{F}$ [MZKST99,MSJL10,ZWSM11]
    - Assume reference model
    - Algorithm: use one query to check each literal
  - $\text{MUS}_{\#1}$: compute MUS for formulas with exactly 1 MUS

Problem	NP Oracles	SAT Oracles
MCS	FP^{NP}	$\text{FP}^{\text{NP}}[\text{wit}, \log n]$
MUS	FP^{NP}	FP^{NP}
Backbones	$\text{FP}^{\text{NP}}_{\|\|}$	$\text{FP}^{\text{NP}}[\text{wit}, \log n]$
$\text{MUS}_{\#1}$	$\text{FP}^{\text{NP}}_{\|\|}$	$\text{FP}^{\text{NP}}[\text{wit}, \log n]$

$\text{FP}^{\text{NP}}[\log n]$ if goal is number

- **Why?**

Preliminary results

- # of queries to SAT/NP oracle for solving selected (possibly restricted) function problems:
  - Backbones: literals common to all models of $\mathcal{F}$ [MZKST99,MSJL10,ZWSM11]
    - Assume reference model
    - Algorithm: use one query to check each literal
  - $\text{MUS}_{\#1}$: compute MUS for formulas with exactly 1 MUS

Problem	NP Oracles	SAT Oracles
MCS	FP^{NP}	$\text{FP}^{\text{NP}}[\text{wit}, \log n]$
MUS	FP^{NP}	FP^{NP}
Backbones	$\text{FP}^{\text{NP}}_{\|\|}$	$\text{FP}^{\text{NP}}[\text{wit}, \log n]$
$\text{MUS}_{\#1}$	$\text{FP}^{\text{NP}}_{\|\|}$	$\text{FP}^{\text{NP}}[\text{wit}, \log n]$

- **Why?**

"Easier" than computing SAT witness!

Outline

Revisit MUSes

Minimal Sets

Query Complexity

Conclusions

Conclusions

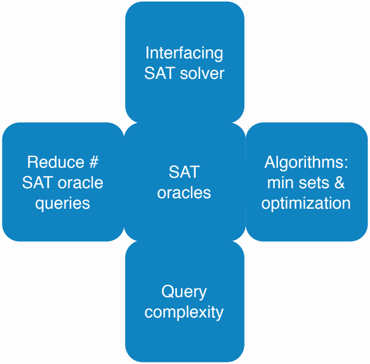
- Significant progress in SAT-based (function) problem solving
  - MUSes, MCSes, MaxSAT, MinSAT, backbones, autarkies, minimal models, prime implicants & implicates
  - But also, MESes, MFSes, etc. etc.
- Categorized function problems on Boolean formulas:
  - Optimization problems
  - Computation of minimal sets
- Introduced the MSMP problem
  - Framework for reasoning about (many) minimal sets problems
- Overviewed algorithms for optimization problems and for minimal set computation
  - E.g. refine UB, refine LB, binary search, core-guided, etc.
  - Insertion, Deletion, Dichotomic, QuickXplain, Progression
- Developed some preliminary query complexity results with witness oracles
  - MCSes, Backbones, $\text{MUS}_{\#1}$

Research directions

- New minimal set problems?
  - And new optimization problems?
- New algorithms?
- New pruning techniques?
- New implementation techniques?
  - How about preprocessing ?
  - How about parallelization ?
- Query complexity results?
  - Also, FPT reductions to SAT?
- How about enumeration problems?
  - MUSes, MCSes, MaxSAT, etc.
- ...

Research directions

- New minimal set problems?
  - And new optimization problems?
- New algorithms?
- New pruning techniques?
- New implementation techniques?
  - How about preprocessing ?
  - How about parallelization ?
- Query complexity results?
  - Also, FPT reductions to SAT?
- How about enumeration problems?
  - MUSes, MCSes, MaxSAT, etc.
- ...



Backbones — proof sketch

- $\nu(x_i)$: truth assignment given to x_i in given reference model (optional, but simpler)

Backbones — proof sketch

- $\nu(x_i)$: truth assignment given to x_i in given reference model (optional, but simpler)
- $\mathcal{F}[X/Y_i]$: formula with fresh set of variables Y_i, associated with each x_i

Backbones — proof sketch

- $\nu(x_i)$: truth assignment given to x_i in given reference model (optional, but simpler)
- $\mathcal{F}[X/Y_i]$: formula with fresh set of variables Y_i, associated with each x_i
- Introduce new variable $z_i \leftrightarrow (\mathcal{F}[X/Y_i] \wedge (y_i \leftrightarrow \neg\nu(x_i)))$

Backbones — proof sketch

- $\nu(x_i)$: truth assignment given to x_i in given reference model (optional, but simpler)
- $\mathcal{F}[X/Y_i]$: formula with fresh set of variables Y_i, associated with each x_i
- Introduce new variable $z_i \leftrightarrow (\mathcal{F}[X/Y_i] \wedge (y_i \leftrightarrow \neg\nu(x_i)))$
  - $z_i = 1$ iff $\mathcal{F}[X/Y_i]$ satisfied with $y_i = \neg\nu(x_i)$

Backbones — proof sketch

- $\nu(x_i)$: truth assignment given to x_i in given reference model (optional, but simpler)
- $\mathcal{F}[X/Y_i]$: formula with fresh set of variables Y_i, associated with each x_i
- Introduce new variable $z_i \leftrightarrow (\mathcal{F}[X/Y_i] \wedge (y_i \leftrightarrow \neg\nu(x_i)))$
  - $z_i = 1$ iff $\mathcal{F}[X/Y_i]$ satisfied with $y_i = \neg\nu(x_i)$
    - i.e. $z_i = 1$ iff x_i is not a backbone variable

Backbones — proof sketch

- $\nu(x_i)$: truth assignment given to x_i in given reference model (optional, but simpler)
- $\mathcal{F}[X/Y_i]$: formula with fresh set of variables Y_i, associated with each x_i
- Introduce new variable $z_i \leftrightarrow (\mathcal{F}[X/Y_i] \wedge (y_i \leftrightarrow \neg\nu(x_i)))$
  - $z_i = 1$ iff $\mathcal{F}[X/Y_i]$ satisfied with $y_i = \neg\nu(x_i)$
    - i.e. $z_i = 1$ iff x_i is not a backbone variable
- Construct formula:

$$\bigwedge_{i=1}^{\mathrm{var}(\mathcal{F})} (z_i \leftrightarrow (\mathcal{F}[X/Y_i] \wedge (y_i \leftrightarrow \neg\nu(x_i))))$$

Backbones — proof sketch

- $\nu(x_i)$: truth assignment given to x_i in given reference model (optional, but simpler)
- $\mathcal{F}[X/Y_i]$: formula with fresh set of variables Y_i, associated with each x_i
- Introduce new variable $z_i \leftrightarrow (\mathcal{F}[X/Y_i] \wedge (y_i \leftrightarrow \neg\nu(x_i)))$
  - $z_i = 1$ iff $\mathcal{F}[X/Y_i]$ satisfied with $y_i = \neg\nu(x_i)$
    - i.e. $z_i = 1$ iff x_i is not a backbone variable
- Construct formula:

$$\bigwedge_{i=1}^{\mathrm{var}(\mathcal{F})} (z_i \leftrightarrow (\mathcal{F}[X/Y_i] \wedge (y_i \leftrightarrow \neg\nu(x_i))))$$

- Any z_i that can take value 1 represents a non-backbone variable

Backbones — proof sketch

- $\nu(x_i)$: truth assignment given to x_i in given reference model (optional, but simpler)
- $\mathcal{F}[X/Y_i]$: formula with fresh set of variables Y_i, associated with each x_i
- Introduce new variable $z_i \leftrightarrow (\mathcal{F}[X/Y_i] \wedge (y_i \leftrightarrow \neg\nu(x_i)))$
  - $z_i = 1$ iff $\mathcal{F}[X/Y_i]$ satisfied with $y_i = \neg\nu(x_i)$
    - i.e. $z_i = 1$ iff x_i is not a backbone variable
- Construct formula:

$$\bigwedge_{i=1}^{\text{var}(\mathcal{F})} (z_i \leftrightarrow (\mathcal{F}[X/Y_i] \wedge (y_i \leftrightarrow \neg\nu(x_i))))$$

- Any z_i that can take value 1 represents a non-backbone variable
  - Goal is to maximize the number of z_i variables with value 1

Backbones — proof sketch

- $\nu(x_i)$: truth assignment given to x_i in given reference model (optional, but simpler)
- $\mathcal{F}[X/Y_i]$: formula with fresh set of variables Y_i, associated with each x_i
- Introduce new variable $z_i \leftrightarrow (\mathcal{F}[X/Y_i] \wedge (y_i \leftrightarrow \neg\nu(x_i)))$
  - $z_i = 1$ iff $\mathcal{F}[X/Y_i]$ satisfied with $y_i = \neg\nu(x_i)$
    - i.e. $z_i = 1$ iff x_i is not a backbone variable
- Construct formula:

$$\bigwedge_{i=1}^{\mathrm{var}(\mathcal{F})} (z_i \leftrightarrow (\mathcal{F}[X/Y_i] \wedge (y_i \leftrightarrow \neg\nu(x_i))))$$

- Any z_i that can take value 1 represents a non-backbone variable
  - Goal is to maximize the number of z_i variables with value 1
  - Can be modeled with soft clauses: (z_i)

Backbones — proof sketch

- $\nu(x_i)$: truth assignment given to x_i in given reference model (optional, but simpler)
- $\mathcal{F}[X/Y_i]$: formula with fresh set of variables Y_i, associated with each x_i
- Introduce new variable $z_i \leftrightarrow (\mathcal{F}[X/Y_i] \wedge (y_i \leftrightarrow \neg\nu(x_i)))$
  - $z_i = 1$ iff $\mathcal{F}[X/Y_i]$ satisfied with $y_i = \neg\nu(x_i)$
    - i.e. $z_i = 1$ iff x_i is not a backbone variable
- Construct formula:

$$\bigwedge_{i=1}^{\mathrm{var}(\mathcal{F})} (z_i \leftrightarrow (\mathcal{F}[X/Y_i] \wedge (y_i \leftrightarrow \neg\nu(x_i))))$$

- Any z_i that can take value 1 represents a non-backbone variable
  - Goal is to maximize the number of z_i variables with value 1
  - Can be modeled with soft clauses: (z_i)
- This is a(n unweighted) partial MaxSAT problem

Backbones — proof sketch

- $\nu(x_i)$: truth assignment given to x_i in given reference model (optional, but simpler)
- $\mathcal{F}[X/Y_i]$: formula with fresh set of variables Y_i, associated with each x_i
- Introduce new variable $z_i \leftrightarrow (\mathcal{F}[X/Y_i] \wedge (y_i \leftrightarrow \neg\nu(x_i)))$
  - $z_i = 1$ iff $\mathcal{F}[X/Y_i]$ satisfied with $y_i = \neg\nu(x_i)$
    - i.e. $z_i = 1$ iff x_i is not a backbone variable
- Construct formula:

$$\bigwedge_{i=1}^{\mathrm{var}(\mathcal{F})} (z_i \leftrightarrow (\mathcal{F}[X/Y_i] \wedge (y_i \leftrightarrow \neg\nu(x_i))))$$

- Any z_i that can take value 1 represents a non-backbone variable
  - Goal is to maximize the number of z_i variables with value 1
  - Can be modeled with soft clauses: (z_i)
- This is a(n unweighted) partial MaxSAT problem
  - Can find solution with $\mathcal{O}(\log n)$ calls to a SAT oracle

Backbones — proof sketch

- $\nu(x_i)$: truth assignment given to x_i in given reference model (optional, but simpler)
- $\mathcal{F}[X/Y_i]$: formula with fresh set of variables Y_i, associated with each x_i
- Introduce new variable $z_i \leftrightarrow (\mathcal{F}[X/Y_i] \wedge (y_i \leftrightarrow \neg\nu(x_i)))$
  - $z_i = 1$ iff $\mathcal{F}[X/Y_i]$ satisfied with $y_i = \neg\nu(x_i)$
    - i.e. $z_i = 1$ iff x_i is not a backbone variable
- Construct formula:

$$\bigwedge_{i=1}^{\mathrm{var}(\mathcal{F})} (z_i \leftrightarrow (\mathcal{F}[X/Y_i] \wedge (y_i \leftrightarrow \neg\nu(x_i))))$$

- Any z_i that can take value 1 represents a non-backbone variable
  - Goal is to maximize the number of z_i variables with value 1
  - Can be modeled with soft clauses: (z_i)
- This is a(n unweighted) partial MaxSAT problem
  - Can find solution with $\mathcal{O}(\log n)$ calls to a SAT oracle
- $\therefore$ Backbone is in $\mathrm{FP}^{\mathrm{NP}}[\mathrm{wit}, \log n]$

$\text{MUS}_{\#1}$ — proof sketch

- $\mathcal{F}[X/Y_i]$: formula with fresh set of variables Y_i, associated with each c_i

$\text{MUS}_{\#1}$ — proof sketch

- $\mathcal{F}[X/Y_i]$: formula with fresh set of variables Y_i, associated with each c_i
- Introduce new variable $z_i \leftrightarrow (\mathcal{F} \setminus c_i)[X/Y_i]$

$\text{MUS}_{\#1}$ — proof sketch

- $\mathcal{F}[X/Y_i]$: formula with fresh set of variables Y_i, associated with each c_i
- Introduce new variable $z_i \leftrightarrow (\mathcal{F} \setminus c_i)[X/Y_i]$
  - $z_i = 1$ iff $(\mathcal{F} \setminus \{c_i\})[X/Y_i]$ satisfied

$\text{MUS}_{\#1}$ — proof sketch

- $\mathcal{F}[X/Y_i]$: formula with fresh set of variables Y_i, associated with each c_i
- Introduce new variable $z_i \leftrightarrow (\mathcal{F} \setminus c_i)[X/Y_i]$
  - $z_i = 1$ iff $(\mathcal{F} \setminus \{c_i\})[X/Y_i]$ satisfied
    - i.e. $z_i = 1$ iff c_i is in MUS, since there is exactly one MUS

$\mathrm{MUS}_{\#1}$ — proof sketch

- $\mathcal{F}[X/Y_i]$: formula with fresh set of variables Y_i, associated with each c_i
- Introduce new variable $z_i \leftrightarrow (\mathcal{F} \setminus c_i)[X/Y_i]$
  - $z_i = 1$ iff $(\mathcal{F} \setminus \{c_i\})[X/Y_i]$ satisfied
    - i.e. $z_i = 1$ iff c_i is in MUS, since there is exactly one MUS
- Construct formula:

$$\bigwedge_{i=1}^{|\mathcal{F}|} (z_i \leftrightarrow (\mathcal{F} \setminus c_i)[X/Y_i])$$

$\text{MUS}_{\#1}$ — proof sketch

- $\mathcal{F}[X/Y_i]$: formula with fresh set of variables Y_i, associated with each c_i
- Introduce new variable $z_i \leftrightarrow (\mathcal{F} \setminus c_i)[X/Y_i]$
  - $z_i = 1$ iff $(\mathcal{F} \setminus \{c_i\})[X/Y_i]$ satisfied
    - i.e. $z_i = 1$ iff c_i is in MUS, since there is exactly one MUS
- Construct formula:

$$\bigwedge_{i=1}^{|\mathcal{F}|} (z_i \leftrightarrow (\mathcal{F} \setminus c_i)[X/Y_i])$$

- Any z_i that can take value 1 represents an MUS clause

$\text{MUS}_{\#1}$ — proof sketch

- $\mathcal{F}[X/Y_i]$: formula with fresh set of variables Y_i, associated with each c_i
- Introduce new variable $z_i \leftrightarrow (\mathcal{F} \setminus c_i)[X/Y_i]$
  - $z_i = 1$ iff $(\mathcal{F} \setminus \{c_i\})[X/Y_i]$ satisfied
    - i.e. $z_i = 1$ iff c_i is in MUS, since there is exactly one MUS
- Construct formula:

$$\bigwedge_{i=1}^{|\mathcal{F}|} (z_i \leftrightarrow (\mathcal{F} \setminus c_i)[X/Y_i])$$

- Any z_i that can take value 1 represents an MUS clause
  - Goal is to maximize the number of z_i variables with value 1

$\text{MUS}_{\#1}$ — proof sketch

- $\mathcal{F}[X/Y_i]$: formula with fresh set of variables Y_i, associated with each c_i
- Introduce new variable $z_i \leftrightarrow (\mathcal{F} \setminus c_i)[X/Y_i]$
  - $z_i = 1$ iff $(\mathcal{F} \setminus \{c_i\})[X/Y_i]$ satisfied
    - i.e. $z_i = 1$ iff c_i is in MUS, since there is exactly one MUS
- Construct formula:

$$\bigwedge_{i=1}^{|\mathcal{F}|} (z_i \leftrightarrow (\mathcal{F} \setminus c_i)[X/Y_i])$$

- Any z_i that can take value 1 represents an MUS clause
  - Goal is to maximize the number of z_i variables with value 1
  - Can be modeled with soft clauses: (z_i)

$\text{MUS}_{\#1}$ — proof sketch

- $\mathcal{F}[X/Y_i]$: formula with fresh set of variables Y_i, associated with each c_i
- Introduce new variable $z_i \leftrightarrow (\mathcal{F} \setminus c_i)[X/Y_i]$
  - $z_i = 1$ iff $(\mathcal{F} \setminus \{c_i\})[X/Y_i]$ satisfied
    - i.e. $z_i = 1$ iff c_i is in MUS, since there is exactly one MUS
- Construct formula:

$$\bigwedge_{i=1}^{|\mathcal{F}|} (z_i \leftrightarrow (\mathcal{F} \setminus c_i)[X/Y_i])$$

- Any z_i that can take value 1 represents an MUS clause
  - Goal is to maximize the number of z_i variables with value 1
  - Can be modeled with soft clauses: (z_i)
- This is a(n unweighted) partial MaxSAT problem

$\text{MUS}_{\#1}$ — proof sketch

- $\mathcal{F}[X/Y_i]$: formula with fresh set of variables Y_i, associated with each c_i
- Introduce new variable $z_i \leftrightarrow (\mathcal{F} \setminus c_i)[X/Y_i]$
  - $z_i = 1$ iff $(\mathcal{F} \setminus \{c_i\})[X/Y_i]$ satisfied
    - i.e. $z_i = 1$ iff c_i is in MUS, since there is exactly one MUS
- Construct formula:

$$\bigwedge_{i=1}^{|\mathcal{F}|} (z_i \leftrightarrow (\mathcal{F} \setminus c_i)[X/Y_i])$$

- Any z_i that can take value 1 represents an MUS clause
  - Goal is to maximize the number of z_i variables with value 1
  - Can be modeled with soft clauses: (z_i)
- This is a(n unweighted) partial MaxSAT problem
  - Can find solution with $\mathcal{O}(\log n)$ calls to a SAT oracle

$\mathrm{MUS}_{\#1}$ — proof sketch

- $\mathcal{F}[X/Y_i]$: formula with fresh set of variables Y_i, associated with each c_i
- Introduce new variable $z_i \leftrightarrow (\mathcal{F} \setminus c_i)[X/Y_i]$
  - $z_i = 1$ iff $(\mathcal{F} \setminus \{c_i\})[X/Y_i]$ satisfied
    - i.e. $z_i = 1$ iff c_i is in MUS, since there is exactly one MUS
- Construct formula:

$$\bigwedge_{i=1}^{|\mathcal{F}|} (z_i \leftrightarrow (\mathcal{F} \setminus c_i)[X/Y_i])$$

- Any z_i that can take value 1 represents an MUS clause
  - Goal is to maximize the number of z_i variables with value 1
  - Can be modeled with soft clauses: (z_i)
- This is a(n unweighted) partial MaxSAT problem
  - Can find solution with $\mathcal{O}(\log n)$ calls to a SAT oracle
- $\therefore$ $\mathrm{MUS}_{\#1}$ is in $\mathrm{FP}^{\mathrm{NP}}[\mathrm{wit}, \log n]$

SAT-Based Problem Solving

Lecture #3:
Solving Optimization Problems with SAT Oracles

Joao Marques-Silva[1,2]

[1]University College Dublin, Ireland

[2]IST/INESC-ID, Lisbon, Portugal

University of Calabria, Italy

February 2015

Part IV

Computing Minimal Cardinality Sets

Maximum satisfiability

$x_6 \vee x_2$	$\neg x_6 \vee x_2$	$\neg x_2 \vee x_1$	$\neg x_1$
$\neg x_6 \vee x_8$	$x_6 \vee \neg x_8$	$x_2 \vee x_4$	$\neg x_4 \vee x_5$
$x_7 \vee x_5$	$\neg x_7 \vee x_5$	$\neg x_5 \vee x_3$	$\neg x_3$

- Unsatisfiable formula

Maximum satisfiability

$x_6 \vee x_2$	$\neg x_6 \vee x_2$	$\neg x_2 \vee x_1$	$\neg x_1$
$\neg x_6 \vee x_8$	$x_6 \vee \neg x_8$	$x_2 \vee x_4$	$\neg x_4 \vee x_5$
$x_7 \vee x_5$	$\neg x_7 \vee x_5$	$\neg x_5 \vee x_3$	$\neg x_3$

- Unsatisfiable formula
- Find largest subset of clauses that is satisfiable

Maximum satisfiability

$\boxed{x_6 \vee x_2}$	$\neg x_6 \vee x_2$	$\neg x_2 \vee x_1$	$\neg x_1$
$\neg x_6 \vee x_8$	$x_6 \vee \neg x_8$	$x_2 \vee x_4$	$\boxed{\neg x_4 \vee x_5}$
$\boxed{x_7 \vee x_5}$	$\neg x_7 \vee x_5$	$\neg x_5 \vee x_3$	$\neg x_3$

- Unsatisfiable formula
- Find largest subset of clauses that is satisfiable
- Recap:
  A Minimal Correction Subset (**MCS**) is an irreducible relaxation of the formula

Maximum satisfiability

$x_6 \vee x_2$	$\neg x_6 \vee x_2$	$\neg x_2 \vee x_1$	$\boxed{\neg x_1}$
$\neg x_6 \vee x_8$	$x_6 \vee \neg x_8$	$x_2 \vee x_4$	$\neg x_4 \vee x_5$
$\boxed{x_7 \vee x_5}$	$\neg x_7 \vee x_5$	$\neg x_5 \vee x_3$	$\neg x_3$

- Unsatisfiable formula
- Find largest subset of clauses that is satisfiable
- Recap:
  A Minimal Correction Subset (**MCS**) is an irreducible relaxation of the formula
- The MaxSAT solution is one of the smallest MCSes

MaxSAT problem(s)

		Hard Clauses?	
		No	Yes
Weights?	No		
	Yes		

MaxSAT problem(s)

		Hard Clauses?	
		No	Yes
Weights?	No	Plain	Partial
	Yes	Weighted	Weighted Partial

MaxSAT problem(s)

		Hard Clauses?	
		No	Yes
Weights?	No	Plain	Partial
	Yes	Weighted	Weighted Partial

- Must satisfy hard clauses, if any
- Compute set of satisfied soft clauses with maximum cost
  - Without weights, cost of each falsified soft clause is 1
- **Or**, compute set of falsified soft clauses with minimum cost (s.t. hard & remaining soft clauses are satisfied)

MaxSAT problem(s)

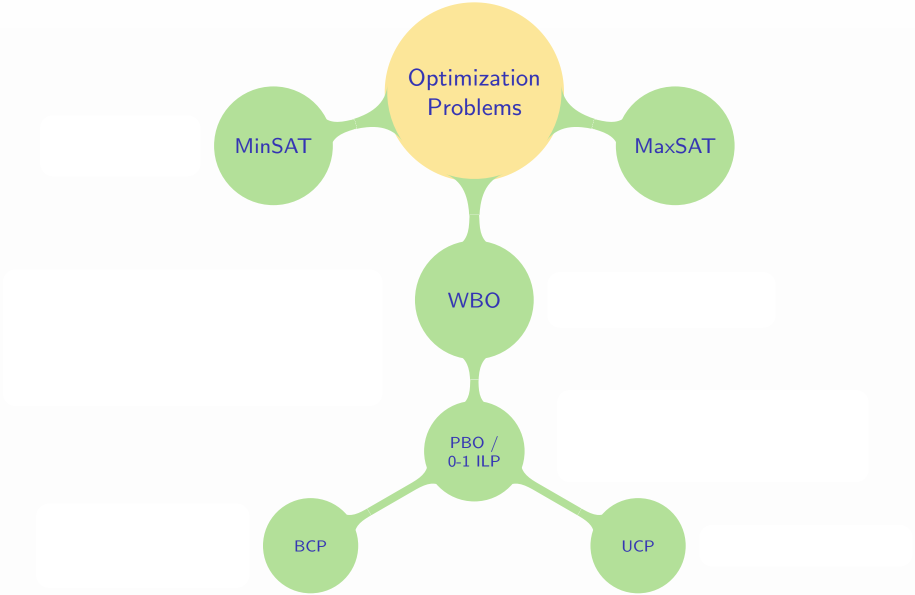
		Hard Clauses?	
		No	Yes
Weights?	No	Plain	Partial
	Yes	Weighted	Weighted Partial

- Must satisfy hard clauses, if any
- Compute set of satisfied soft clauses with maximum cost
  - Without weights, cost of each falsified soft clause is 1
- **Or**, compute set of falsified soft clauses with minimum cost (s.t. hard & remaining soft clauses are satisfied)
- **Note**: goal is to compute **set** of satisfied (or falsified) clauses; **not** just the cost **!**

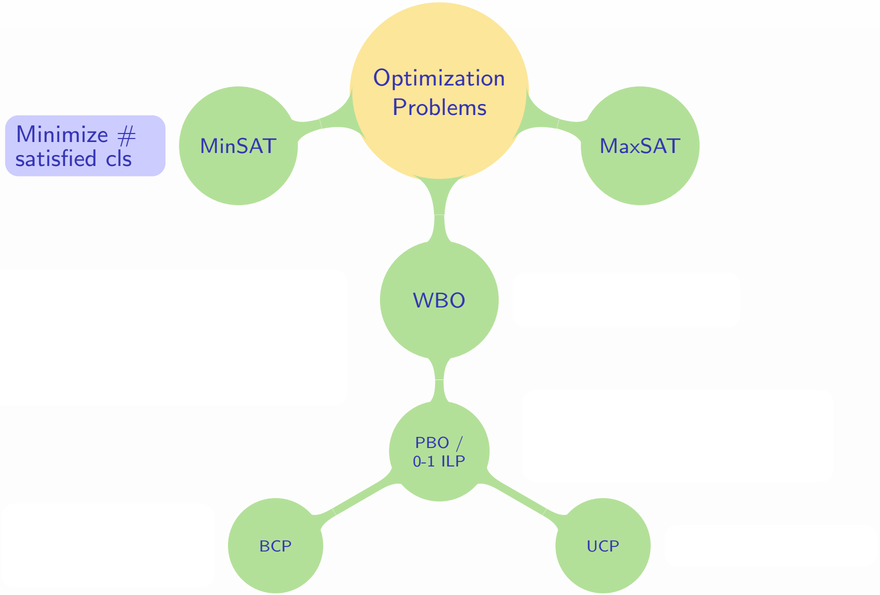
Practical relevancy of MaxSAT?

- Package management; Timetabling; Haplotyping; Configuration; Fault localization; Design debugging; Model based diagnosis; Telecom feature subscription; Resource constrained scheduling; Planning; Pseudo Boolean formulas; Binate covering; Filter design; FPGA routing; Power estimation; Technology mapping; etc.

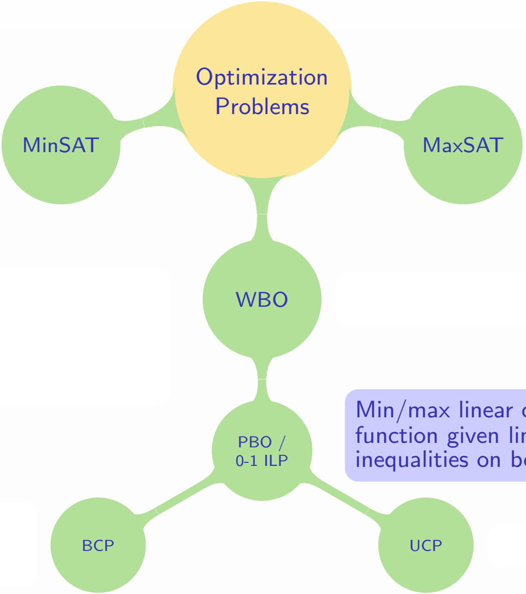
Other optimization problems (on Boolean variables)



Other optimization problems (on Boolean variables)

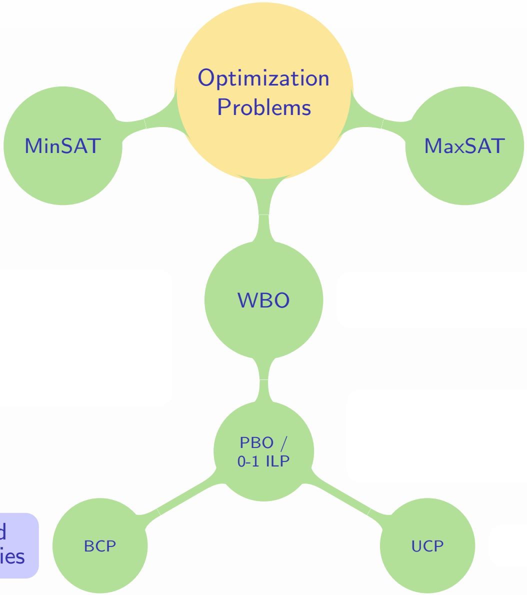


Other optimization problems (on Boolean variables)



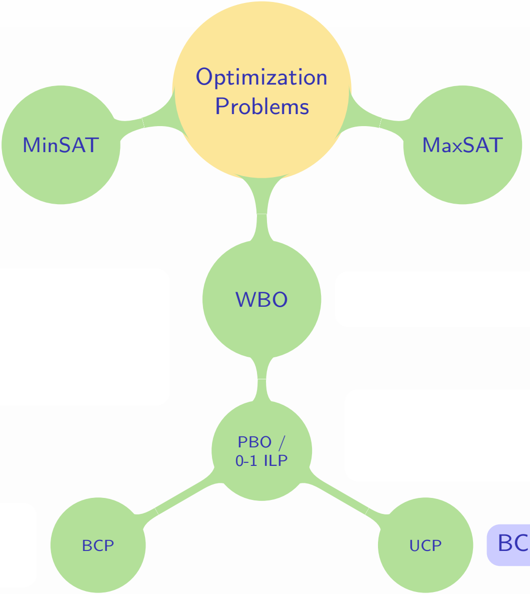
Min/max linear cost function given linear inequalities on boolean vars

Other optimization problems (on Boolean variables)



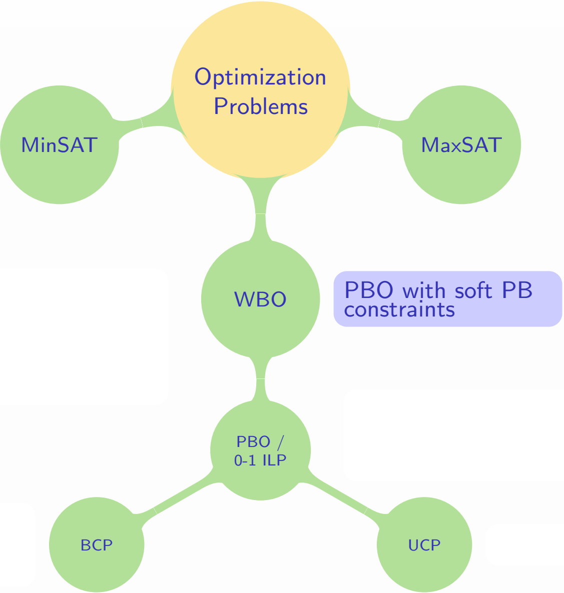
PBO if cls instead of linear inequalities

Other optimization problems (on Boolean variables)

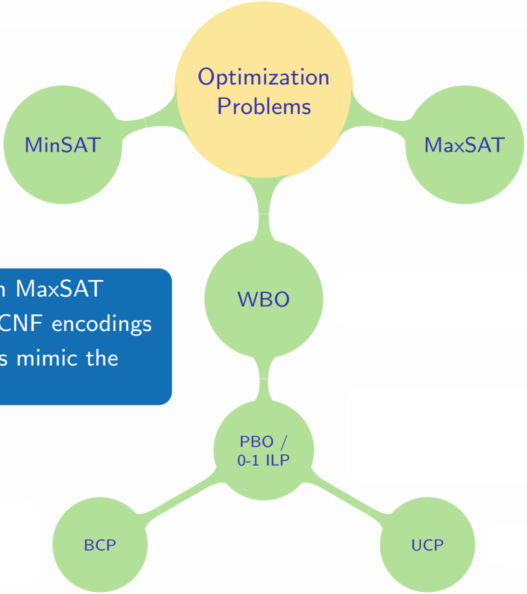


BCP if positive cls

Other optimization problems (on Boolean variables)



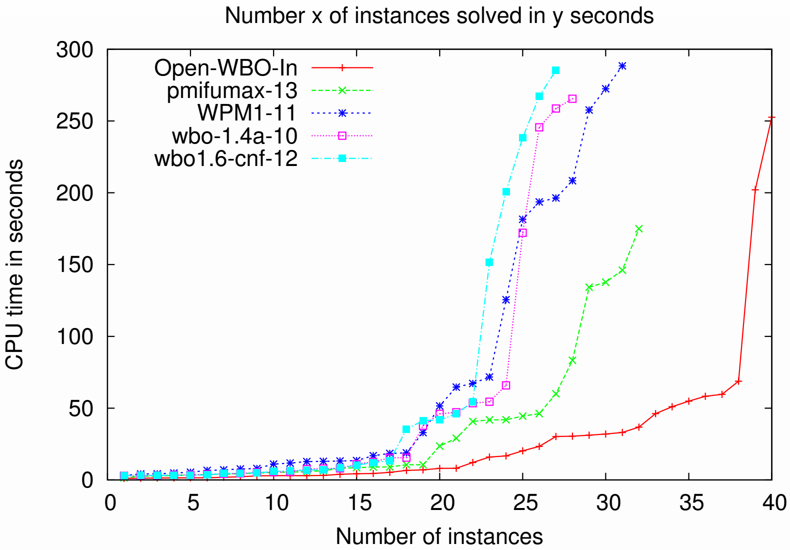
Other optimization problems (on Boolean variables)



Can reduce to/from MaxSAT
Extensive work on CNF encodings
Solution approaches mimic the ones for MaxSAT

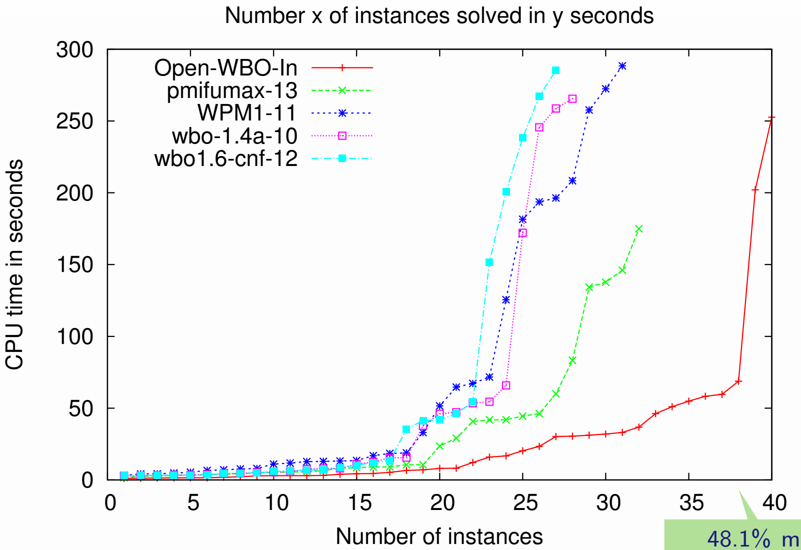
The MaxSAT (r)evolution

The MaxSAT (r)evolution – plain industrial instances



Source: [MaxSAT 2014 organizers]

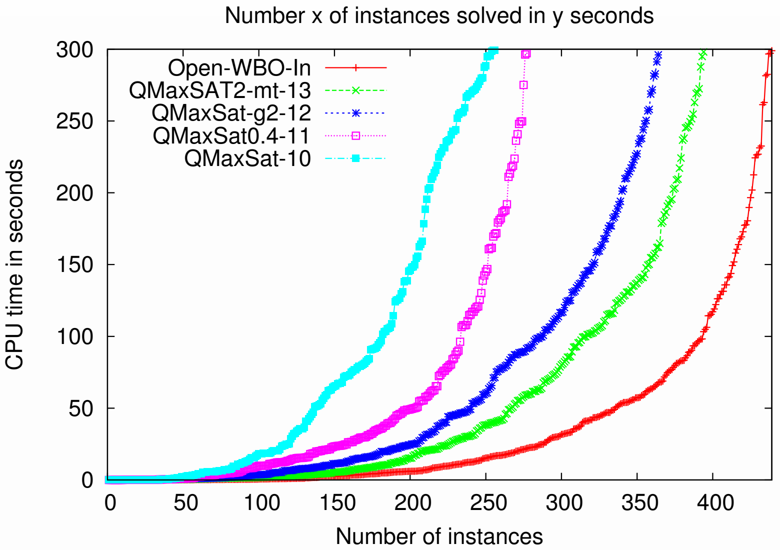
The MaxSAT (r)evolution – plain industrial instances



48.1% more instances solved!

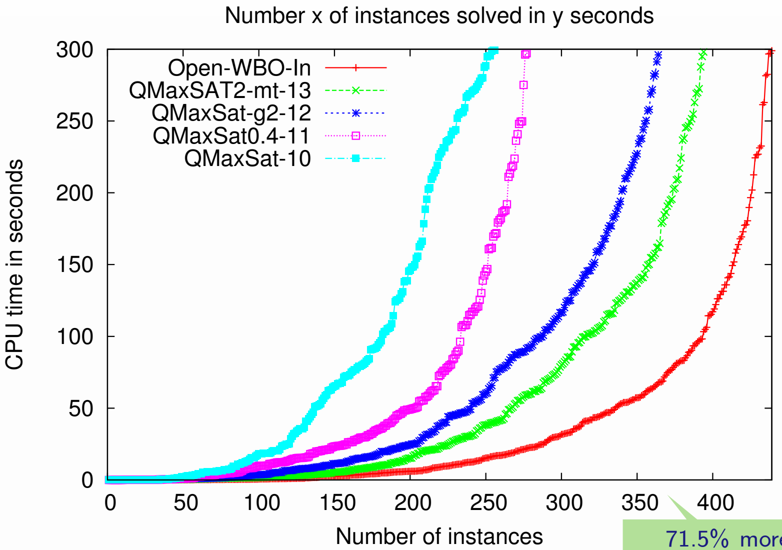
Source: [MaxSAT 2014 organizers]

The MaxSAT (r)evolution – partial



Source: [MaxSAT 2014 organizers]

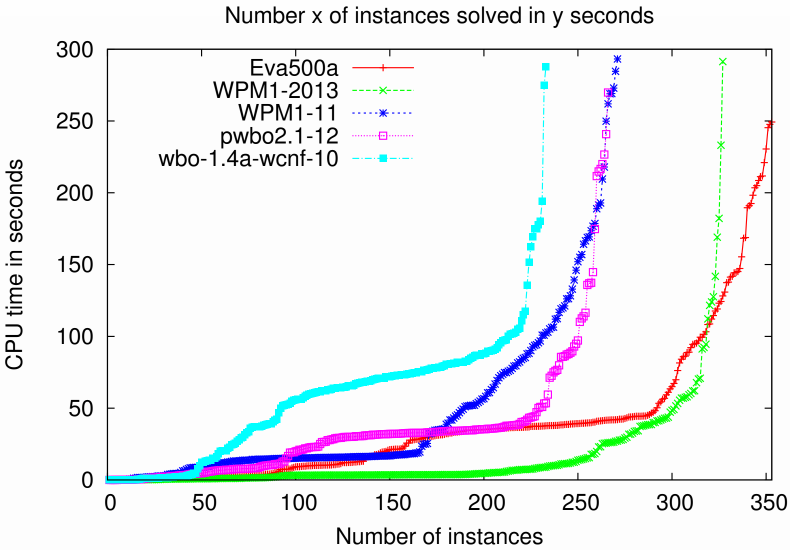
The MaxSAT (r)evolution – partial



Source: [MaxSAT 2014 organizers]

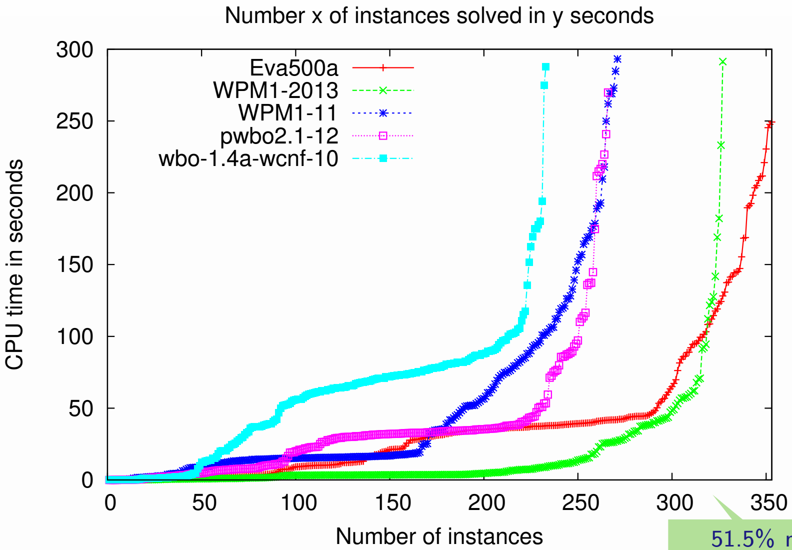
71.5% more instances solved!

The MaxSAT (r)evolution – weighted partial



Source: [MaxSAT 2014 organizers]

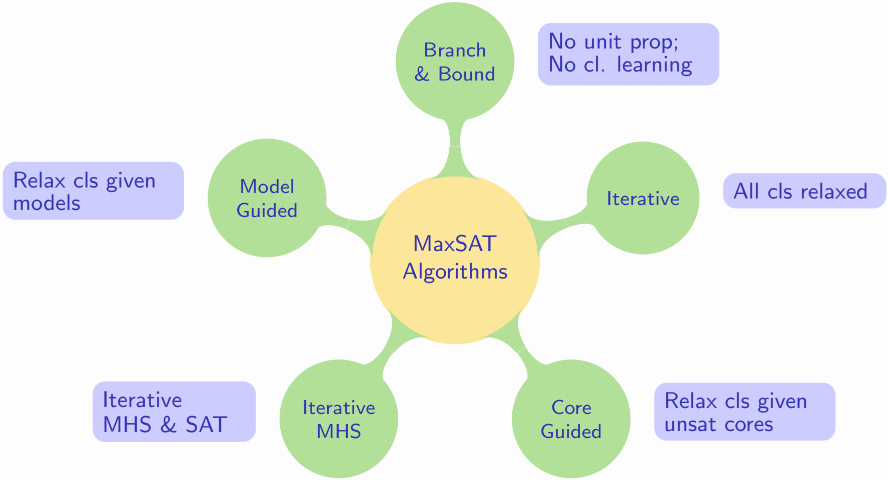
The MaxSAT (r)evolution – weighted partial



Source: [MaxSAT 2014 organizers]

51.5% more instances solved!

Many MaxSAT algorithms



More on MaxSAT algorithms

- Iterative: [MHLPMS13]
  - Linear search SAT/UNSAT (refine UB) [e.g. LBP10]
  - Linear search UNSAT/SAT (refine LB)
  - Binary search [e.g. FM06]
  - Bit-based
  - Mixed linear/binary search [e.g. KZFH12]
- Core-guided: [MHLPMS13,ABL13]
  - FM/(W)MSU1.X/WPM1 [FM06,MSM08,MMSP09,ABL09a,ABGL12]
  - (W)MSU3 [MSP07]
  - (W)MSU4 [MSP08]
  - (W)PM2 [ABL09a,ABL09b,ABL10,ABGL13]
  - Core-guided binary search (w/ disjoint cores) [HMMS11,MHMS12]
    - Bin-Core, Bin-Core-Dis, Bin-Core-Dis2
- Iterative minimal hitting set (MHS) computation [DB11,DB13a,DB13b]
- Model guided approaches [HMPMS12]
- Branch & bound search [HJ90,LM09]

More on MaxSAT algorithms – somewhat out of date

- Iterative: [MHLPMS13]
  - Linear search SAT/UNSAT (refine UB) [e.g. LBP10]
  - Linear search UNSAT/SAT (refine LB)
  - Binary search [e.g. FM06]
  - Bit-based
  - Mixed linear/binary search [e.g. KZFH12]
- Core-guided: [MHLPMS13,ABL13]
  - FM/(W)MSU1.X/WPM1 [FM06,MSM08,MMSP09,ABL09a,ABGL12]
  - (W)MSU3 [MSP07]
  - (W)MSU4 [MSP08]
  - (W)PM2 [ABL09a,ABL09b,ABL10,ABGL13]
  - Core-guided binary search (w/ disjoint cores) [HMMS11,MHMS12]
    - Bin-Core, Bin-Core-Dis, Bin-Core-Dis2
- Iterative minimal hitting set (MHS) computation [DB11,DB13a,DB13b]
- Model guided approaches [HMPMS12]
- Branch & bound search [HJ90,LM09]

Recap last lectures

- CNF encodings of cardinality and PB constraints
  - AtMost1, AtMost$k$, etc.

- SAT oracle: black-box use of SAT solver
  - Witness for **Y** outcomes
  - **And** unsatisfiable core of **N** outcomes
    - **Note**: can be the complete set of soft clauses

- But also, binary search, progression, etc.

Outline

Example Applications

Iterative MaxSAT

Core-Guided MaxSAT

Our Recent Work

Some Results

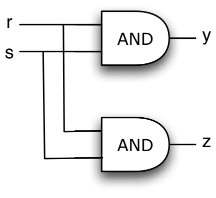
Conclusions

Outline

Design debugging

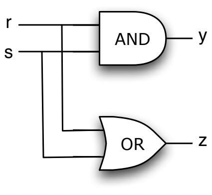
[SMVLS'07]

Correct circuit



Input stimuli: $\langle r, s \rangle = \langle 0, 1 \rangle$
Valid output: $\langle y, z \rangle = \langle 0, 0 \rangle$

Faulty circuit



Input stimuli: $\langle r, s \rangle = \langle 0, 1 \rangle$
Invalid output: $\langle y, z \rangle = \langle 0, 0 \rangle$

- The model:
  - Hard clauses: Input and output values
  - Soft clauses: CNF representation of circuit
- The problem:
  - Maximize number of satisfied clauses (i.e. circuit gates)

Software package upgrades with MaxSAT

[MBCV'06,TSJL'07,AL'08,ALMS'09,ALBL'10]

- Universe of software packages: $\{p_1, \ldots, p_n\}$
- Associate x_i with p_i: $x_i = 1$ iff p_i is installed
- Constraints associated with package p_i: (p_i, D_i, C_i)
  - D_i: dependencies (required packages) for installing p_i
  - C_i: conflicts (disallowed packages) for installing p_i
- Example problem: Maximum Installability
  - Maximum number of packages that can be installed
  - Package constraints represent hard clauses
  - Soft clauses: (x_i)

Package constraints:

$(p_1, \{p_2 \vee p_3\}, \{p_4\})$
$(p_2, \{p_3\}, \{p_4\})$
$(p_3, \{p_2\}, \emptyset)$
$(p_4, \{p_2, p_3\}, \emptyset)$

Software package upgrades with MaxSAT

[MBCV'06,TSJL'07,AL'08,ALMS'09,ALBL'10]

- Universe of software packages: $\{p_1, \ldots, p_n\}$
- Associate x_i with p_i: $x_i = 1$ iff p_i is installed
- Constraints associated with package p_i: (p_i, D_i, C_i)
  - D_i: dependencies (required packages) for installing p_i
  - C_i: conflicts (disallowed packages) for installing p_i
- Example problem: Maximum Installability
  - Maximum number of packages that can be installed
  - Package constraints represent hard clauses
  - Soft clauses: (x_i)

Package constraints:

$$
\begin{array}{l}
(p_1, \{p_2 \vee p_3\}, \{p_4\}) \\
(p_2, \{p_3\}, \{p_4\}) \\
(p_3, \{p_2\}, \emptyset) \\
(p_4, \{p_2, p_3\}, \emptyset)
\end{array}
$$

MaxSAT formulation:

$$
\begin{array}{rcl}
\varphi_H & = & \{(\neg x_1 \vee x_2 \vee x_3), (\neg x_1 \vee \neg x_4), \\
& & (\neg x_2 \vee x_3), (\neg x_2 \vee \neg x_4), (\neg x_3 \vee x_2), \\
& & (\neg x_4 \vee x_2), (\neg x_4 \vee x_3)\} \\
\varphi_S & = & \{(x_1), (x_2), (x_3), (x_4)\}
\end{array}
$$

Key engine for MUS enumeration

- MUS: irreducible unsatisfiable set of clauses
  - MCS: irreducible set of clauses such that complement is satisfiable
  - MSS: subset maximal satisfiable set of clauses

Key engine for MUS enumeration

- MUS: irreducible unsatisfiable set of clauses
  - MCS: irreducible set of clauses such that complement is satisfiable
  - MSS: subset maximal satisfiable set of clauses

- Enumeration of MUSes finds many applications:
  - Model checking with CEGAR, type inference & checking, etc. [ALS'08,BSW'03]

Key engine for MUS enumeration

- MUS: irreducible unsatisfiable set of clauses
  - MCS: irreducible set of clauses such that complement is satisfiable
  - MSS: subset maximal satisfiable set of clauses

- Enumeration of MUSes finds many applications:
  - Model checking with CEGAR, type inference & checking, etc. [ALS'08,BSW'03]

- How to enumerate MUSes? [E.g. LS'08]
  - Use hitting set duality between MUSes and MCSes [E.g. R'87,BL'03]
    - An MUS is an irreducible hitting set of a formula's MCSes
    - An MCS is an irreducible hitting set of a formula's MUSes
  - Can enumerate MCSes and then use them to compute MUSes

Key engine for MUS enumeration

- MUS: irreducible unsatisfiable set of clauses
  - MCS: irreducible set of clauses such that complement is satisfiable
  - MSS: subset maximal satisfiable set of clauses

- Enumeration of MUSes finds many applications:
  - Model checking with CEGAR, type inference & checking, etc. [ALS'08,BSW'03]

- How to enumerate MUSes? [E.g. LS'08]
  - Use hitting set duality between MUSes and MCSes [E.g. R'87,BL'03]
    - An MUS is an irreducible hitting set of a formula's MCSes
    - An MCS is an irreducible hitting set of a formula's MUSes
  - Can enumerate MCSes and then use them to compute MUSes
  - Use MaxSAT enumeration for computing **all** MSSes

Many other applications – recap

- Error localization in C code [JM'11]
- Haplotyping with pedigrees [GLMSO'10]
- Course timetabling [AN'10]
- Combinatorial auctions [HLGS'08]
- Minimizing Disclosure of Private Information in Credential-Based Interactions [AVFPS'10]
- Reasoning over Biological Networks [GL'12]
- Binate/unate covering
  - Haplotype inference [GMSLO'11]
  - Digital filter design [ACFM'08]
  - FSM synthesis [e.g. HS'96]
  - Logic minimization [e.g. HS'96]
  - ...
- ...

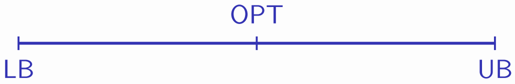
Outline

MaxSAT with iterative SAT solving – definitions

- Cost of assignment:
  - Sum of weights of falsified clauses

MaxSAT with iterative SAT solving – definitions

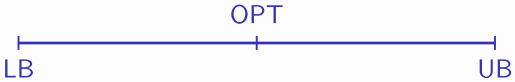
- Cost of assignment:
  - Sum of weights of falsified clauses



- Optimum solution (OPT):
  - Assignment with minimum cost

MaxSAT with iterative SAT solving – definitions

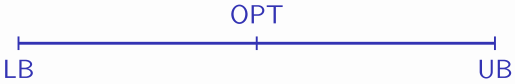
- Cost of assignment:
  - Sum of weights of falsified clauses



- Optimum solution (OPT):
  - Assignment with minimum cost
- Upper Bound (UB):
  - Assignment with cost $\geq$ OPT
  - E.g. $\sum_{c_j \in \varphi} w_j + 1$; hard clauses may be inconsistent
- Lower Bound (LB):
  - No assignment with cost $\leq$ LB
  - E.g. -1; it may be possible to satisfy all soft clauses

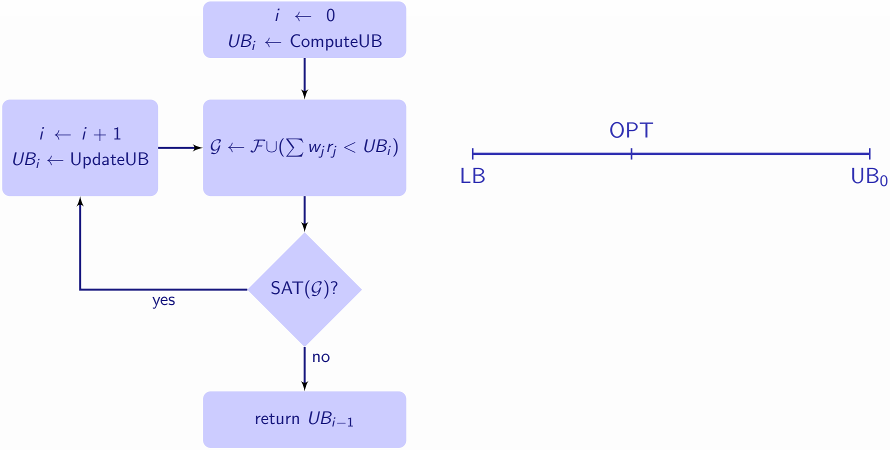
MaxSAT with iterative SAT solving – definitions

- Cost of assignment:
  - Sum of weights of falsified clauses

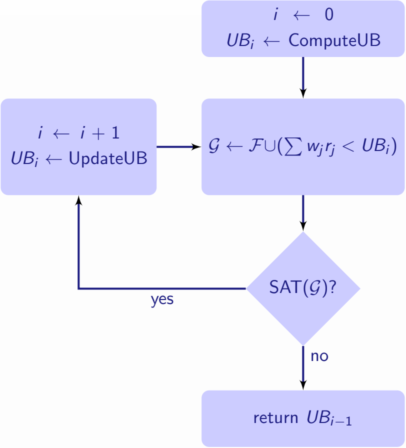


- Optimum solution (OPT):
  - Assignment with minimum cost
- Upper Bound (UB):
  - Assignment with cost $\geq$ OPT
  - E.g. $\sum_{c_j \in \varphi} w_j + 1$; hard clauses may be inconsistent
- Lower Bound (LB):
  - No assignment with cost $\leq$ LB
  - E.g. -1; it may be possible to satisfy all soft clauses
- Relax each soft clause c_j: $(c_j \vee r_j)$ (on-demand in core-guided)

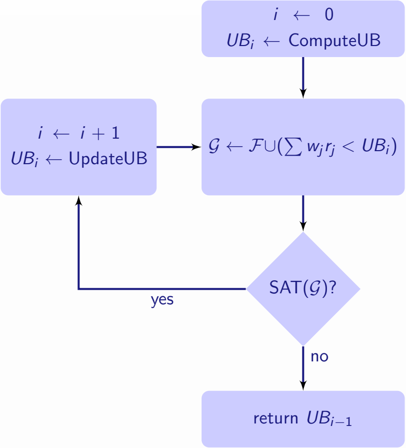
MaxSAT with iterative SAT solving – refine UB



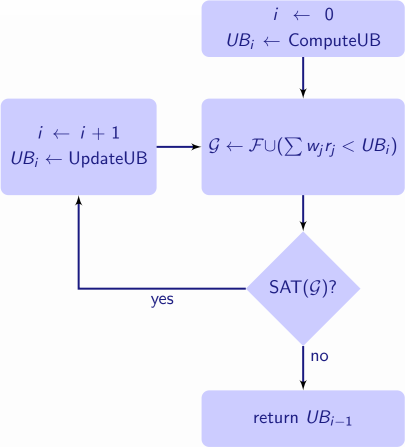
MaxSAT with iterative SAT solving – refine UB

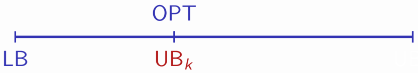


MaxSAT with iterative SAT solving – refine UB

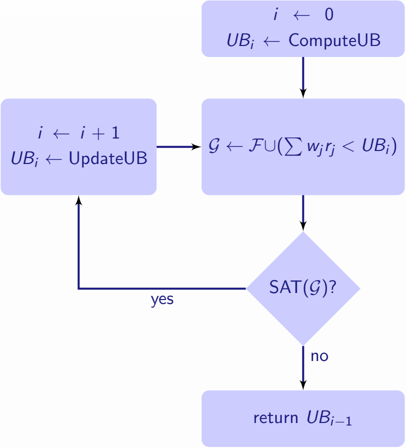


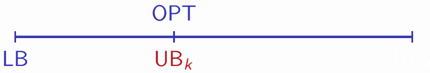
MaxSAT with iterative SAT solving – refine UB





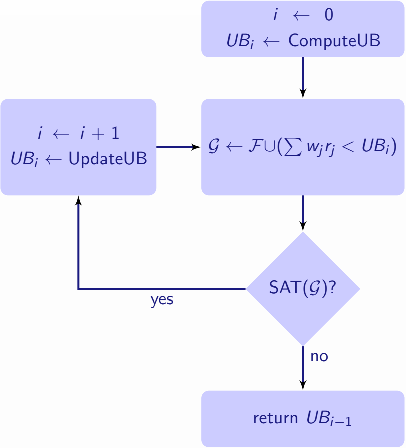
MaxSAT with iterative SAT solving – refine UB

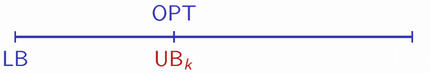




- Worst-case # of iterations **exponential** on instance size (# bits)
  - Improvement: use binary search instead

MaxSAT with iterative SAT solving – refine UB





- Worst-case # of iterations **exponential** on instance size (# bits)
  - Improvement: use binary search instead
- Many example solvers: Minisat+, SAT4J, QMaxSat [ES06,LBP10,KZFH12]

MaxSAT with iterative SAT solving – example

$x_6 \vee x_2$	$\neg x_6 \vee x_2$	$\neg x_2 \vee x_1$	$\neg x_1$
$\neg x_6 \vee x_8$	$x_6 \vee \neg x_8$	$x_2 \vee x_4$	$\neg x_4 \vee x_5$
$x_7 \vee x_5$	$\neg x_7 \vee x_5$	$\neg x_5 \vee x_3$	$\neg x_3$

Example CNF formula

MaxSAT with iterative SAT solving – example

$x_6 \vee x_2 \vee r_1$ $\quad \neg x_6 \vee x_2 \vee r_2$ $\quad \neg x_2 \vee x_1 \vee r_3$ $\quad \neg x_1 \vee r_4$

$\neg x_6 \vee x_8 \vee r_5$ $\quad x_6 \vee \neg x_8 \vee r_6$ $\quad x_2 \vee x_4 \vee r_7$ $\quad \neg x_4 \vee x_5 \vee r_8$

$x_7 \vee x_5 \vee r_9$ $\quad \neg x_7 \vee x_5 \vee r_{10}$ $\quad \neg x_5 \vee x_3 \vee r_{11}$ $\quad \neg x_3 \vee r_{12}$

$\sum_{i=1}^{12} r_i \leq 12$

Relax all clauses; Set $UB = 12 + 1$

MaxSAT with iterative SAT solving – example

$x_6 \vee x_2 \vee r_1$	$\neg x_6 \vee x_2 \vee r_2$	$\neg x_2 \vee x_1 \vee r_3$	$\neg x_1 \vee r_4$
$\neg x_6 \vee x_8 \vee r_5$	$x_6 \vee \neg x_8 \vee r_6$	$x_2 \vee x_4 \vee r_7$	$\neg x_4 \vee x_5 \vee r_8$
$x_7 \vee x_5 \vee r_9$	$\neg x_7 \vee x_5 \vee r_{10}$	$\neg x_5 \vee x_3 \vee r_{11}$	$\neg x_3 \vee r_{12}$

$\sum_{i=1}^{12} r_i \leq 12$

Formula is SAT; E.g. all $x_i = 0$ and $r_1 = r_7 = r_9 = 1$ (i.e. cost $= 3$)

MaxSAT with iterative SAT solving – example

$x_6 \vee x_2 \vee r_1$ $\neg x_6 \vee x_2 \vee r_2$ $\neg x_2 \vee x_1 \vee r_3$ $\neg x_1 \vee r_4$

$\neg x_6 \vee x_8 \vee r_5$ $x_6 \vee \neg x_8 \vee r_6$ $x_2 \vee x_4 \vee r_7$ $\neg x_4 \vee x_5 \vee r_8$

$x_7 \vee x_5 \vee r_9$ $\neg x_7 \vee x_5 \vee r_{10}$ $\neg x_5 \vee x_3 \vee r_{11}$ $\neg x_3 \vee r_{12}$

$\sum_{i=1}^{12} r_i \leq 2$

Refine $UB = 3$

MaxSAT with iterative SAT solving – example

$x_6 \vee x_2 \vee r_1$	$\neg x_6 \vee x_2 \vee r_2$	$\neg x_2 \vee x_1 \vee r_3$	$\neg x_1 \vee r_4$
$\neg x_6 \vee x_8 \vee r_5$	$x_6 \vee \neg x_8 \vee r_6$	$x_2 \vee x_4 \vee r_7$	$\neg x_4 \vee x_5 \vee r_8$
$x_7 \vee x_5 \vee r_9$	$\neg x_7 \vee x_5 \vee r_{10}$	$\neg x_5 \vee x_3 \vee r_{11}$	$\neg x_3 \vee r_{12}$

$\sum_{i=1}^{12} r_i \leq 2$

Formula is SAT; E.g. $x_1 = x_2 = 1$; $x_3 = \ldots = x_8 = 0$ and $r_4 = r_9 = 1$ (i.e. cost $= 2$)

MaxSAT with iterative SAT solving – example

$x_6 \vee x_2 \vee r_1$ $\quad \neg x_6 \vee x_2 \vee r_2$ $\quad \neg x_2 \vee x_1 \vee r_3$ $\quad \neg x_1 \vee r_4$

$\neg x_6 \vee x_8 \vee r_5$ $\quad x_6 \vee \neg x_8 \vee r_6$ $\quad x_2 \vee x_4 \vee r_7$ $\quad \neg x_4 \vee x_5 \vee r_8$

$x_7 \vee x_5 \vee r_9$ $\quad \neg x_7 \vee x_5 \vee r_{10}$ $\quad \neg x_5 \vee x_3 \vee r_{11}$ $\quad \neg x_3 \vee r_{12}$

$\sum_{i=1}^{12} r_i \leq 1$

Refine $UB = 2$

MaxSAT with iterative SAT solving – example

$x_6 \vee x_2 \vee r_1$	$\neg x_6 \vee x_2 \vee r_2$	$\neg x_2 \vee x_1 \vee r_3$	$\neg x_1 \vee r_4$
$\neg x_6 \vee x_8 \vee r_5$	$x_6 \vee \neg x_8 \vee r_6$	$x_2 \vee x_4 \vee r_7$	$\neg x_4 \vee x_5 \vee r_8$
$x_7 \vee x_5 \vee r_9$	$\neg x_7 \vee x_5 \vee r_{10}$	$\neg x_5 \vee x_3 \vee r_{11}$	$\neg x_3 \vee r_{12}$

$\sum_{i=1}^{12} r_i \leq 1$

Formula is UNSAT; terminate

MaxSAT with iterative SAT solving – example

$x_6 \vee x_2 \vee r_1$ $\qquad$ $\neg x_6 \vee x_2 \vee r_2$ $\qquad$ $\neg x_2 \vee x_1 \vee r_3$ $\qquad$ $\neg x_1 \vee r_4$

$\neg x_6 \vee x_8 \vee r_5$ $\qquad$ $x_6 \vee \neg x_8 \vee r_6$ $\qquad$ $x_2 \vee x_4 \vee r_7$ $\qquad$ $\neg x_4 \vee x_5 \vee r_8$

$x_7 \vee x_5 \vee r_9$ $\qquad$ $\neg x_7 \vee x_5 \vee r_{10}$ $\qquad$ $\neg x_5 \vee x_3 \vee r_{11}$ $\qquad$ $\neg x_3 \vee r_{12}$

$\sum_{i=1}^{12} r_i \leq 1$

MaxSAT solution is last satisfied UB: $UB = 2$

MaxSAT with iterative SAT solving – example

$x_6 \vee x_2 \vee r_1$ $\quad$ $\neg x_6 \vee x_2 \vee r_2$ $\quad$ $\neg x_2 \vee x_1 \vee r_3$ $\quad$ $\neg x_1 \vee r_4$

$\neg x_6 \vee x_8 \vee r_5$ $\quad$ $x_6 \vee \neg x_8 \vee r_6$ $\quad$ $x_2 \vee x_4 \vee r_7$ $\quad$ $\neg x_4 \vee x_5 \vee r_8$

$x_7 \vee x_5 \vee r_9$ $\quad$ $\neg x_7 \vee x_5 \vee r_{10}$ $\quad$ $\neg x_5 \vee x_3 \vee r_{11}$ $\quad$ $\neg x_3 \vee r_{12}$

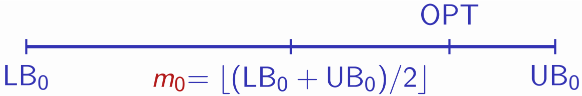
$\sum_{i=1}^{12} r_i \leq 1$

MaxSAT solution is last satisfied UB: $UB = 2$

AtMost$k$/PB constraints
over **all** relaxation variables

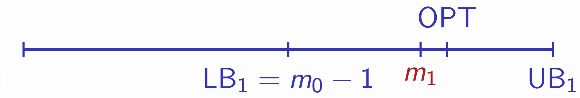
All (possibly many)
soft clauses relaxed

MaxSAT with iterative SAT solving – binary search



- Invariant: $LB_k \leq UB_k - 1$
- Require $\sum w_i \, r_i \leq m_0$

MaxSAT with iterative SAT solving – binary search



- Invariant: $\mathrm{LB}_k \leq \mathrm{UB}_k - 1$
- Require $\sum w_i\, r_i \leq m_0$
- Repeat
  - If UNSAT, refine $\mathrm{LB}_1 = m_0, \ldots$
  - Compute new mid value $m_1, \ldots$

MaxSAT with iterative SAT solving – binary search

- Invariant: $\mathrm{LB}_k \leq \mathrm{UB}_k - 1$
- Require $\sum w_i\, r_i \leq m_0$
- Repeat
  - If UNSAT, refine $\mathrm{LB}_1 = m_0, \ldots$
  - Compute new mid value $m_1, \ldots$
  - If SAT, refine $\mathrm{UB}_3 = m_2, \ldots$

MaxSAT with iterative SAT solving – binary search

- Invariant: $\mathrm{LB}_k \leq \mathrm{UB}_k - 1$
- Require $\sum w_i \, r_i \leq m_0$
- Repeat
  - If UNSAT, refine $\mathrm{LB}_1 = m_0, \ldots$
  - Compute new mid value $m_1, \ldots$
  - If SAT, refine $\mathrm{UB}_3 = m_2, \ldots$

MaxSAT with iterative SAT solving – binary search

- Invariant: $\mathrm{LB}_k \leq \mathrm{UB}_k - 1$
- Require $\sum w_i\, r_i \leq m_0$
- Repeat
  - If UNSAT, refine $\mathrm{LB}_1 = m_0, \ldots$
  - Compute new mid value $m_1, \ldots$
  - If SAT, refine $\mathrm{UB}_3 = m_2, \ldots$
- Until $\mathrm{LB}_k = \mathrm{UB}_k - 1$

MaxSAT with iterative SAT solving – binary search

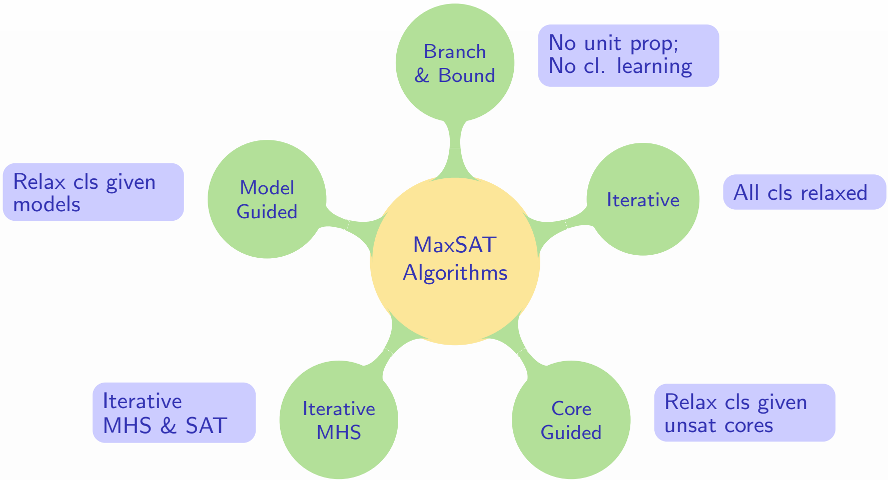
- Invariant: $\mathrm{LB}_k \leq \mathrm{UB}_k - 1$
- Require $\sum w_i \, r_i \leq m_0$
- Repeat
  - If UNSAT, refine $\mathrm{LB}_1 = m_0, \ldots$
  - Compute new mid value $m_1, \ldots$
  - If SAT, refine $\mathrm{UB}_3 = m_2, \ldots$
- Until $\mathrm{LB}_k = \mathrm{UB}_k - 1$
- Worst-case $\#$ of iterations **linear** on instance size

MaxSAT with iterative SAT solving – binary search

- Invariant: $\mathrm{LB}_k \leq \mathrm{UB}_k - 1$
- Require $\sum w_i \, r_i \leq m_0$
- Repeat
  - If UNSAT, refine $\mathrm{LB}_1 = m_0, \ldots$
  - Compute new mid value $m_1, \ldots$
  - If SAT, refine $\mathrm{UB}_3 = m_2, \ldots$
- Until $\mathrm{LB}_k = \mathrm{UB}_k - 1$
- Worst-case # of iterations **linear** on instance size
- Example tools:
  - Counter-based MaxSAT solver [FM'06]
  - MathSAT [CFGSS'10]
  - MSUnCore [HMMS'11]

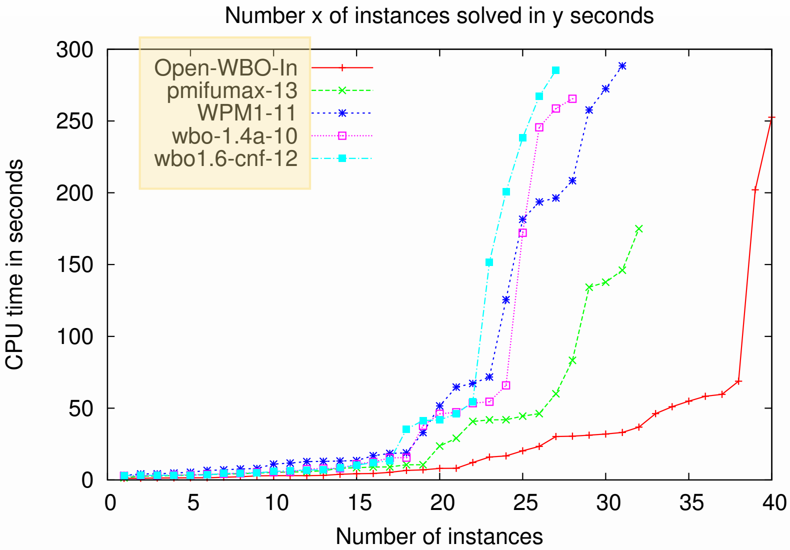
Outline

Many MaxSAT approaches



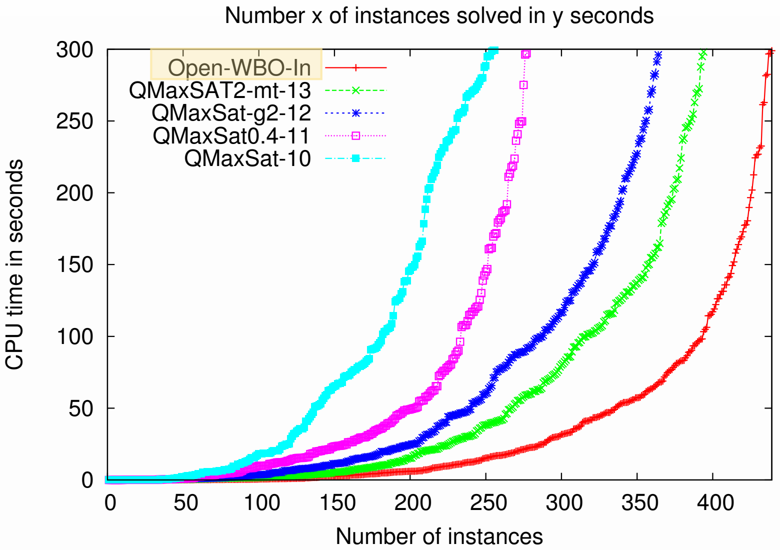
- For practical (**industrial**) instances: **core-guided** approaches are the most effective [MaxSAT14]

Core-guided solver performance – plain



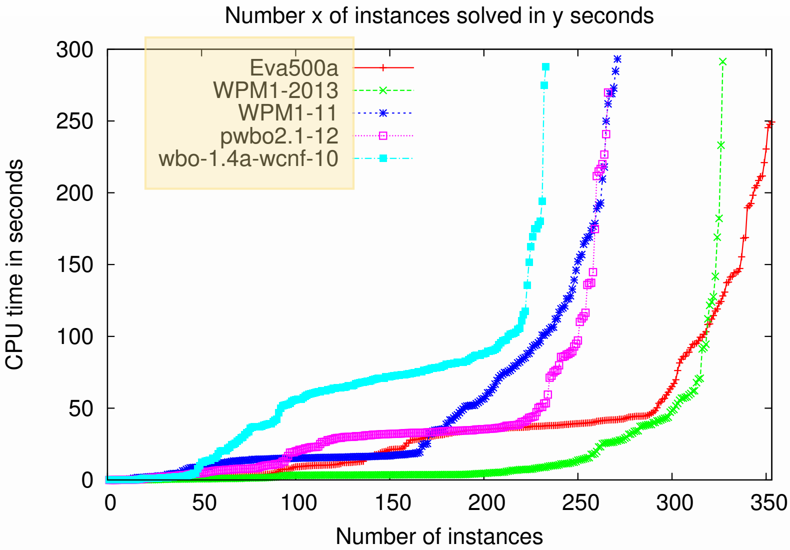
Source: [MaxSAT 2014 organizers]

Core-guided solver performance – partial



Source: [MaxSAT 2014 organizers]

Core-guided solver performance – weighted partial



Source: [MaxSAT 2014 organizers]

Core-guided MaxSAT

$x_6 \vee x_2$	$\neg x_6 \vee x_2$	$\neg x_2 \vee x_1$	$\neg x_1$
$\neg x_6 \vee x_8$	$x_6 \vee \neg x_8$	$x_2 \vee x_4$	$\neg x_4 \vee x_5$
$x_7 \vee x_5$	$\neg x_7 \vee x_5$	$\neg x_5 \vee x_3$	$\neg x_3$

- **Goal**: Do **not** relax all clauses

Core-guided MaxSAT

$x_6 \vee x_2$	$\neg x_6 \vee x_2$	$\neg x_2 \vee x_1$	$\neg x_1$
$\neg x_6 \vee x_8$	$x_6 \vee \neg x_8$	$x_2 \vee x_4$	$\neg x_4 \vee x_5$
$x_7 \vee x_5$	$\neg x_7 \vee x_5$	$\neg x_5 \vee x_3$	$\neg x_3$

- **Goal**: Do **not** relax all clauses
  - **Why?**
    - Some clauses never relevant for computing MaxSAT solution
    - Simplify cardinality/PB constraints

Core-guided MaxSAT

$x_6 \vee x_2$	$\neg x_6 \vee x_2$	$\neg x_2 \vee x_1$	$\neg x_1$
$\neg x_6 \vee x_8$	$x_6 \vee \neg x_8$	$x_2 \vee x_4$	$\neg x_4 \vee x_5$
$x_7 \vee x_5$	$\neg x_7 \vee x_5$	$\neg x_5 \vee x_3$	$\neg x_3$

- **Goal**: Do **not** relax all clauses
  - **Why?**
    - Some clauses never relevant for computing MaxSAT solution
    - Simplify cardinality/PB constraints
- How to relax clauses **on demand**?
  - Relax clauses given computed unsatisfiable cores
    - Many alternative ways to instrument code-guided algorithms

Outline

Fu&Malik's (FM) core-guided algorithm

$x_6 \vee x_2$	$\neg x_6 \vee x_2$	$\neg x_2 \vee x_1$	$\neg x_1$
$\neg x_6 \vee x_8$	$x_6 \vee \neg x_8$	$x_2 \vee x_4$	$\neg x_4 \vee x_5$
$x_7 \vee x_5$	$\neg x_7 \vee x_5$	$\neg x_5 \vee x_3$	$\neg x_3$

Example CNF formula

Fu&Malik's (FM) core-guided algorithm

$x_6 \vee x_2$	$\neg x_6 \vee x_2$	$\neg x_2 \vee x_1$	$\neg x_1$
$\neg x_6 \vee x_8$	$x_6 \vee \neg x_8$	$x_2 \vee x_4$	$\neg x_4 \vee x_5$
$x_7 \vee x_5$	$\neg x_7 \vee x_5$	$\neg x_5 \vee x_3$	$\neg x_3$

Formula is UNSAT; $\text{OPT} \leq |\varphi| - 1$; Get unsat core

Fu&Malik's (FM) core-guided algorithm

$x_6 \vee x_2$	$\neg x_6 \vee x_2$	$\neg x_2 \vee x_1 \vee r_1$	$\neg x_1 \vee r_2$
$\neg x_6 \vee x_8$	$x_6 \vee \neg x_8$	$x_2 \vee x_4 \vee r_3$	$\neg x_4 \vee x_5 \vee r_4$
$x_7 \vee x_5$	$\neg x_7 \vee x_5$	$\neg x_5 \vee x_3 \vee r_5$	$\neg x_3 \vee r_6$

$\sum_{i=1}^{6} r_i \leq 1$

Add relaxation variables and AtMost1 constraint

Fu&Malik's (FM) core-guided algorithm

$x_6 \vee x_2$ $\quad \neg x_6 \vee x_2$ $\quad \neg x_2 \vee x_1 \vee r_1$ $\quad \neg x_1 \vee r_2$

$\neg x_6 \vee x_8$ $\quad x_6 \vee \neg x_8$ $\quad x_2 \vee x_4 \vee r_3$ $\quad \neg x_4 \vee x_5 \vee r_4$

$x_7 \vee x_5$ $\quad \neg x_7 \vee x_5$ $\quad \neg x_5 \vee x_3 \vee r_5$ $\quad \neg x_3 \vee r_6$

$\sum_{i=1}^{6} r_i \leq 1$

Formula is (again) UNSAT; $\text{OPT} \leq |\varphi| - 2$; Get unsat core

Fu&Malik's (FM) core-guided algorithm

$x_6 \vee x_2 \vee r_7$	$\neg x_6 \vee x_2 \vee r_8$	$\neg x_2 \vee x_1 \vee r_1 \vee r_9$	$\neg x_1 \vee r_2 \vee r_{10}$
$\neg x_6 \vee x_8$	$x_6 \vee \neg x_8$	$x_2 \vee x_4 \vee r_3$	$\neg x_4 \vee x_5 \vee r_4$
$x_7 \vee x_5 \vee r_{11}$	$\neg x_7 \vee x_5 \vee r_{12}$	$\neg x_5 \vee x_3 \vee r_5 \vee r_{13}$	$\neg x_3 \vee r_6 \vee r_{14}$
$\sum_{i=1}^{6} r_i \leq 1$	$\sum_{i=7}^{14} r_i \leq 1$		

Add new relaxation variables and AtMost1 constraint

Fu&Malik's (FM) core-guided algorithm

$x_6 \vee x_2 \vee r_7$	$\neg x_6 \vee x_2 \vee r_8$	$\neg x_2 \vee x_1 \vee r_1 \vee r_9$	$\neg x_1 \vee r_2 \vee r_{10}$
$\neg x_6 \vee x_8$	$x_6 \vee \neg x_8$	$x_2 \vee x_4 \vee r_3$	$\neg x_4 \vee x_5 \vee r_4$
$x_7 \vee x_5 \vee r_{11}$	$\neg x_7 \vee x_5 \vee r_{12}$	$\neg x_5 \vee x_3 \vee r_5 \vee r_{13}$	$\neg x_3 \vee r_6 \vee r_{14}$
$\sum_{i=1}^{6} r_i \leq 1$	$\sum_{i=7}^{14} r_i \leq 1$		

Instance is now SAT

Fu&Malik's (FM) core-guided algorithm

$x_6 \vee x_2 \vee r_7$	$\neg x_6 \vee x_2 \vee r_8$	$\neg x_2 \vee x_1 \vee r_1 \vee r_9$	$\neg x_1 \vee r_2 \vee r_{10}$
$\neg x_6 \vee x_8$	$x_6 \vee \neg x_8$	$x_2 \vee x_4 \vee r_3$	$\neg x_4 \vee x_5 \vee r_4$
$x_7 \vee x_5 \vee r_{11}$	$\neg x_7 \vee x_5 \vee r_{12}$	$\neg x_5 \vee x_3 \vee r_5 \vee r_{13}$	$\neg x_3 \vee r_6 \vee r_{14}$
$\sum_{i=1}^{6} r_i \leq 1$	$\sum_{i=7}^{14} r_i \leq 1$		

MaxSAT solution is $|\varphi| - \mathcal{I} = 12 - 2 = 10$

Fu&Malik's (FM) core-guided algorithm

$x_6 \vee x_2 \vee r_7$	$\neg x_6 \vee x_2 \vee r_8$	$\neg x_2 \vee x_1 \vee r_1 \vee r_9$	$\neg x_1 \vee r_2 \vee r_{10}$
$\neg x_6 \vee x_8$	$x_6 \vee \neg x_8$	$x_2 \vee x_4 \vee r_3$	$\neg x_4 \vee x_5 \vee r_4$
$x_7 \vee x_5 \vee r_{11}$	$\neg x_7 \vee x_5 \vee r_{12}$	$\neg x_5 \vee x_3 \vee r_5 \vee r_{13}$	$\neg x_3 \vee r_6 \vee r_{14}$
$\sum_{i=1}^{6} r_i \leq 1$	$\sum_{i=7}^{14} r_i \leq 1$		

MaxSAT solution is $|\varphi| - \mathcal{I} = 12 - 2 = 10$

Only AtMost1 constraints used

Relaxed soft clauses remain **soft**

Fu&Malik's (FM) core-guided algorithm

$x_6 \vee x_2 \vee r_7$	$\neg x_6 \vee x_2 \vee r_8$	$\neg x_2 \vee x_1 \vee r_1 \vee r_9$	$\neg x_1 \vee r_2 \vee r_{10}$
$\neg x_6 \vee x_8$	$x_6 \vee \neg x_8$	$x_2 \vee x_4 \vee r_3$	$\neg x_4 \vee x_5 \vee r_4$
$x_7 \vee x_5 \vee r_{11}$	$\neg x_7 \vee x_5 \vee r_{12}$	$\neg x_5 \vee x_3 \vee r_5 \vee r_{13}$	$\neg x_3 \vee r_6 \vee r_{14}$
$\sum_{i=1}^{6} r_i \leq 1$	$\sum_{i=7}^{14} r_i \leq 1$		

MaxSAT solution is $|\varphi| - \mathcal{I} = 12 - 2 = 10$

Only AtMost1 constraints used

Some clauses not relaxed

Relaxed soft clauses remain **soft**

Outline

MSU3 core-guided algorithm

$x_6 \vee x_2$	$\neg x_6 \vee x_2$	$\neg x_2 \vee x_1$	$\neg x_1$
$\neg x_6 \vee x_8$	$x_6 \vee \neg x_8$	$x_2 \vee x_4$	$\neg x_4 \vee x_5$
$x_7 \vee x_5$	$\neg x_7 \vee x_5$	$\neg x_5 \vee x_3$	$\neg x_3$

Example CNF formula

MSU3 core-guided algorithm

$x_6 \vee x_2$	$\neg x_6 \vee x_2$	$\neg x_2 \vee x_1$	$\neg x_1$
$\neg x_6 \vee x_8$	$x_6 \vee \neg x_8$	$x_2 \vee x_4$	$\neg x_4 \vee x_5$
$x_7 \vee x_5$	$\neg x_7 \vee x_5$	$\neg x_5 \vee x_3$	$\neg x_3$

Formula is UNSAT; $\text{OPT} \leq |\varphi| - 1$; Get unsat core

MSU3 core-guided algorithm

$x_6 \vee x_2$	$\neg x_6 \vee x_2$	$\neg x_2 \vee x_1 \vee r_1$	$\neg x_1 \vee r_2$
$\neg x_6 \vee x_8$	$x_6 \vee \neg x_8$	$x_2 \vee x_4 \vee r_3$	$\neg x_4 \vee x_5 \vee r_4$
$x_7 \vee x_5$	$\neg x_7 \vee x_5$	$\neg x_5 \vee x_3 \vee r_5$	$\neg x_3 \vee r_6$

$\sum_{i=1}^{6} r_i \leq 1$

Add relaxation variables and AtMost1 constraint

MSU3 core-guided algorithm

$x_6 \vee x_2$	$\neg x_6 \vee x_2$	$\neg x_2 \vee x_1 \vee r_1$	$\neg x_1 \vee r_2$
$\neg x_6 \vee x_8$	$x_6 \vee \neg x_8$	$x_2 \vee x_4 \vee r_3$	$\neg x_4 \vee x_5 \vee r_4$
$x_7 \vee x_5$	$\neg x_7 \vee x_5$	$\neg x_5 \vee x_3 \vee r_5$	$\neg x_3 \vee r_6$

$\sum_{i=1}^{6} r_i \leq 1$

Formula is (again) UNSAT; OPT $\leq |\varphi| - 2$; Get unsat core

MSU3 core-guided algorithm

$x_6 \vee x_2 \vee r_7$	$\neg x_6 \vee x_2 \vee r_8$	$\neg x_2 \vee x_1 \vee r_1$	$\neg x_1 \vee r_2$
$\neg x_6 \vee x_8$	$x_6 \vee \neg x_8$	$x_2 \vee x_4 \vee r_3$	$\neg x_4 \vee x_5 \vee r_4$
$x_7 \vee x_5 \vee r_9$	$\neg x_7 \vee x_5 \vee r_{10}$	$\neg x_5 \vee x_3 \vee r_5$	$\neg x_3 \vee r_6$

$\sum_{i=1}^{10} r_i \leq 2$

Add new relaxation variables and AtMost1 constraint

MSU3 core-guided algorithm

$x_6 \vee x_2 \vee r_7$	$\neg x_6 \vee x_2 \vee r_8$	$\neg x_2 \vee x_1 \vee r_1$	$\neg x_1 \vee r_2$
$\neg x_6 \vee x_8$	$x_6 \vee \neg x_8$	$x_2 \vee x_4 \vee r_3$	$\neg x_4 \vee x_5 \vee r_4$
$x_7 \vee x_5 \vee r_9$	$\neg x_7 \vee x_5 \vee r_{10}$	$\neg x_5 \vee x_3 \vee r_5$	$\neg x_3 \vee r_6$

$\sum_{i=1}^{10} r_i \leq 2$

Instance is now SAT

MSU3 core-guided algorithm

$x_6 \vee x_2 \vee r_7$ $\quad$ $\neg x_6 \vee x_2 \vee r_8$ $\quad$ $\neg x_2 \vee x_1 \vee r_1$ $\quad$ $\neg x_1 \vee r_2$

$\neg x_6 \vee x_8$ $\quad$ $x_6 \vee \neg x_8$ $\quad$ $x_2 \vee x_4 \vee r_3$ $\quad$ $\neg x_4 \vee x_5 \vee r_4$

$x_7 \vee x_5 \vee r_9$ $\quad$ $\neg x_7 \vee x_5 \vee r_{10}$ $\quad$ $\neg x_5 \vee x_3 \vee r_5$ $\quad$ $\neg x_3 \vee r_6$

$\sum_{i=1}^{10} r_i \leq 2$

MaxSAT solution is $|\varphi| - \mathcal{I} = 12 - 2 = 10$

MSU3 core-guided algorithm

$x_6 \vee x_2 \vee r_7$ $\qquad \neg x_6 \vee x_2 \vee r_8$ $\qquad \neg x_2 \vee x_1 \vee r_1$ $\qquad \neg x_1 \vee r_2$

$\neg x_6 \vee x_8$ $\qquad x_6 \vee \neg x_8$ $\qquad x_2 \vee x_4 \vee r_3$ $\qquad \neg x_4 \vee x_5 \vee r_4$

$x_7 \vee x_5 \vee r_9$ $\qquad \neg x_7 \vee x_5 \vee r_{10}$ $\qquad \neg x_5 \vee x_3 \vee r_5$ $\qquad \neg x_3 \vee r_6$

$\sum_{i=1}^{10} r_i \leq 2$

MaxSAT solution is $|\varphi| - \mathcal{I} = 12 - 2 = 10$

AtMost$k$/PB constraints used

Relaxed soft clauses become **hard**

MSU3 core-guided algorithm

$x_6 \vee x_2 \vee r_7$ $\quad \neg x_6 \vee x_2 \vee r_8$ $\quad \neg x_2 \vee x_1 \vee r_1$ $\quad \neg x_1 \vee r_2$

$\boxed{\neg x_6 \vee x_8 \quad x_6 \vee \neg x_8}$ $\quad x_2 \vee x_4 \vee r_3$ $\quad \neg x_4 \vee x_5 \vee r_4$

$x_7 \vee x_5 \vee r_9$ $\quad \neg x_7 \vee x_5 \vee r_{10}$ $\quad \neg x_5 \vee x_3 \vee r_5$ $\quad \neg x_3 \vee r_6$

$\sum_{i=1}^{10} r_i \leq 2$

MaxSAT solution is $|\varphi| - \mathcal{I} = 12 - 2 = 10$

AtMost$k$/PB constraints used

Some clauses not relaxed

Relaxed soft clauses become **hard**

Outline

Outline

Recap binary search for MaxSAT (Bin)

[e.g. FM'06]

$(R, \varphi_W) \leftarrow \mathrm{Relax}(\emptyset, \varphi, \mathrm{Soft}(\varphi))$
$(\lambda, \mu, \mathcal{A}_M) \leftarrow (-1, \sum_{i=1}^{m} w_i + 1, \emptyset)$
while $\lambda < \mu - 1$ **do**
  $\nu \leftarrow \lfloor(\lambda + \mu)/2\rfloor$
  $\varphi_E \leftarrow \mathrm{CNF}(\sum_{r_i \in R} w_i\, r_i \leq \nu)$
  $(\mathrm{st}, \mathcal{A}) \leftarrow \mathrm{SAT}(\varphi_W \cup \varphi_E)$
  if st $=$ *true* **then**
    $(\mathcal{A}_M, \mu) \leftarrow (\mathcal{A}, \sum_{i=1}^{m} w_i\, \mathcal{A}\langle r_i\rangle)$
  else
    $\lambda \leftarrow \nu$
return $\mathrm{Init}(\mathcal{A}_M)$

Towards core-guided MaxSAT

- MaxSAT by iterative SAT solving: **all** clauses relaxed

- How to relax clauses **on demand**, given binary search?

Core-guided binary search (Bin-Core)

[HMMS'11]

$(R, \varphi_W, \varphi_S) \leftarrow (\emptyset, \varphi, \text{Soft}(\varphi))$
$(\lambda, \mu, \mathcal{A}_M) \leftarrow (-1, \sum_{i=1}^{m} w_i + 1, \emptyset)$
while $\lambda < \mu - 1$ **do**
  $\nu \leftarrow \lfloor(\lambda + \mu)/2\rfloor$
  $\varphi_E \leftarrow \text{CNF}(\sum_{r_i \in R} w_i\, r_i \leq \nu)$
  $(\text{st}, \varphi_C, \mathcal{A}) \leftarrow \text{SAT}(\varphi_W \cup \varphi_E)$
  if $\text{st} = \textit{true}$ **then**
    $(\mathcal{A}_M, \mu) \leftarrow (\mathcal{A}, \sum_{i=1}^{m} w_i\, \mathcal{A}\langle r_i\rangle)$
  else
    if $\varphi_C \cap \varphi_S = \emptyset$ **then**
      $\lambda \leftarrow \nu$
    else
      $(R, \varphi_W) \leftarrow \text{Relax}(R, \varphi_W, \varphi_C \cap \varphi_S)$
return $\text{Init}(\mathcal{A}_M)$

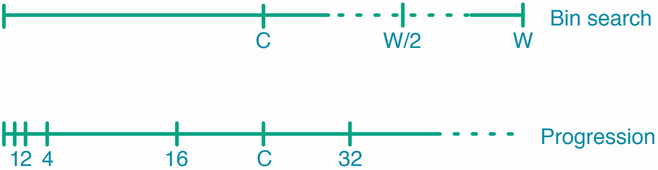
Bin-Core with disjoint cores (Bin-Core-Dis)

- Organization similar to Bin-Core
- Keep set of disjoint unsatisfiable cores [HMMS'11]
  - Need to join unsatisfiable cores
- Integrate lower **&** upper bounds [HMMS'11,MHMS'12]
  - Essential to reduce number of iterations
- Integrate additional pruning techniques [MHMS'12]
  - BMO condition
  - etc.

Outline

Binary search vs. progression

- Motivation:



  - Avoid unnecessary binary search iterations when $\boxed{W \gg C}$

MaxSAT using geometric progressions

```
Progression_Iterative(𝓕)
    Input: 𝓕 = 𝓕_S ∪ 𝓕_H
    (R, 𝓕_W) ← Relax(∅, 𝓕, 𝓕_S)          // Relax & harden soft clause c_i with r_i
    (λ, j) ← (0, 0)                        // LB & progression index
    while true do
        τ ← 2^j − 1                        // Tentative UB w/ geom. prog.
        if τ > Σ_{r_i∈R} w_i then
            return BinSearch(𝓕_W, R, λ, ∅)  // Bin search if UB ≥ W
        (st, 𝒜) ← SAT(𝓕_W ∪ CNF(Σ_{r_i∈R} w_i r_i ≤ τ))
        if st = true then
            return BinSearch(𝓕_W, R, λ, 𝒜)  // Bin search given (actual) UB
        else
            λ ← τ                          // Update LB
            j ← j + 1                      // Increase progression index
```

MaxSAT using geometric progressions

Progression_Iterative($\mathcal{F}$)

Input: $\mathcal{F} = \mathcal{F}_S \cup \mathcal{F}_H$

$(R, \mathcal{F}_W) \leftarrow$ Relax$(\emptyset, \mathcal{F}, \mathcal{F}_S)$ // Relax & harden soft clause c_i with r_i

$(\lambda, j) \leftarrow (0, 0)$ // LB & progression index

while true **do**

$\tau \leftarrow 2^j - 1$ // Tentative UB w/ geom. prog.

if $\tau > \sum_{r_i \in R} w_i$ **then**

return BinSearch$(\mathcal{F}_W, R, \lambda, \emptyset)$ // Bin search if UB $\geq W$

$(\mathrm{st}, \mathcal{A}) \leftarrow$ SAT$(\mathcal{F}_W \cup$ CNF$(\sum_{r_i \in R} w_i r_i \leq \tau))$

if st $=$ true **then**

return BinSearch$(\mathcal{F}_W, R, \lambda, \mathcal{A})$ // Bin search given (actual) UB

else

$\lambda \leftarrow \tau$ // Update LB

$j \leftarrow j + 1$ // Increase progression index

- Worst-case number of oracle calls: $\mathcal{O}(\log C)$

Earlier work using geometric progressions

- Used for improving lower bounds
  - Optimization problems in planning [SS07]
  - Job shop scheduling [MSV13]

- Used in algorithms for computing a **minimal set subject to a monotone predicate** (**MSMP**) [MSBJ13]
  - E.g. MUSes, MCSes, minimal models, etc. [MSJ14]
  - Also used being developed by ILOG [L14]

Progression & core-guided algorithms

- Use geometric progression (instead of binary) search
- Refine computed upper bound with core-guided algorithm:
  - Core-guided binary search (Bin Core, BC) [HMMS11]
  - Bin Core with disjoint cores (BCD/BCD2) [HMMS11,MHMS12]
  - Actually, any core-guided algorithm that refines $(LB, UB]$ can be used
- Worst case number of oracle calls in $\mathcal{O}(m + \log C)$
  - $\mathcal{O}(m + \log C)$: geometric progression step
  - $\mathcal{O}(m + \log C)$: BC/BCD/BCD2 step
  - Where, $\mathcal{O}(m)$ captures the iterative relaxation of soft clauses
  - Compare with $\mathcal{O}(m + \log W)$ for BC, BCD/BCD2 [HMMS11]

Progression with core-guided binary search

```
Progression_BinCore(F)
    Input: F = F_S ∪ F_H
    (R, F_W) ← (∅, F)                          // Initially no clauses relaxed
    (λ, j) ← (0, 0)                            // LB & progression index
    while true do
        τ ← 2^j − 1                            // Tentative UB w/ geom. prog.
        if τ > Σ_{r_i∈R} w_i then
            return BinCore(F_W, R, λ, ∅)       // Bin core if UB ≥ W
        (st, U, A) ← SAT(F_W ∪ CNF(Σ_{r_i∈R} w_i r_i ≤ τ))
        if st = true then
            return BinCore(F_W, R, λ, A)       // Bin core given (actual) UB
        else
            if U ∩ F_S = ∅ then
                λ ← τ                          // Update LB
                j ← j + 1                      // Increase progression index
            else
                (R, F_W) ← Relax(R, F_W, U ∩ F_S)   // Relax & harden soft clauses
```

Outline

Why soft cardinality constraints?

- Like MSU3 (and others):
  - Use a **single** relaxation variable per clause
- Like FM:
  - Create **one** new cardinality constraint per core
- Similarly to FM:
  - **No** need for PB constraints: use **only** AtMost$k$ constraints

Core-guided with soft constraints

$x_6 \vee x_2$	$\neg x_6 \vee x_2$	$\neg x_2 \vee x_1$	$\neg x_1$
$\neg x_6 \vee x_8$	$x_6 \vee \neg x_8$	$x_2 \vee x_4$	$\neg x_4 \vee x_5$
$x_7 \vee x_5$	$\neg x_7 \vee x_5$	$\neg x_5 \vee x_3$	$\neg x_3$

Example CNF formula

Core-guided with soft constraints

$x_6 \vee x_2$	$\neg x_6 \vee x_2$	$\neg x_2 \vee x_1$	$\neg x_1$
$\neg x_6 \vee x_8$	$x_6 \vee \neg x_8$	$x_2 \vee x_4$	$\neg x_4 \vee x_5$
$x_7 \vee x_5$	$\neg x_7 \vee x_5$	$\neg x_5 \vee x_3$	$\neg x_3$

Formula is UNSAT; $\text{OPT} \leq |\varphi| - 1$; Get unsat core

Core-guided with soft constraints

$x_6 \vee x_2$	$\neg x_6 \vee x_2$	$\neg x_2 \vee x_1 \vee r_1$	$\neg x_1 \vee r_2$
$\neg x_6 \vee x_8$	$x_6 \vee \neg x_8$	$x_2 \vee x_4 \vee r_3$	$\neg x_4 \vee x_5 \vee r_4$
$x_7 \vee x_5$	$\neg x_7 \vee x_5$	$\neg x_5 \vee x_3 \vee r_5$	$\neg x_3 \vee r_6$

$S_1 \leq 1$

Aux sums: $S_1 = \sum_{i=1}^{6} r_i$;

Add relaxation variables and AtMost1 constraint

Core-guided with soft constraints

$x_6 \vee x_2$	$\neg x_6 \vee x_2$	$\neg x_2 \vee x_1 \vee r_1$	$\neg x_1 \vee r_2$
$\neg x_6 \vee x_8$	$x_6 \vee \neg x_8$	$x_2 \vee x_4 \vee r_3$	$\neg x_4 \vee x_5 \vee r_4$
$x_7 \vee x_5$	$\neg x_7 \vee x_5$	$\neg x_5 \vee x_3 \vee r_5$	$\neg x_3 \vee r_6$

$S_1 \leq 1$

Aux sums: $S_1 = \sum_{i=1}^{6} r_i$;

Formula is (again) UNSAT; OPT $\leq |\varphi| - 2$; Get unsat core

Core-guided with soft constraints

$x_6 \vee x_2 \vee r_7$	$\neg x_6 \vee x_2 \vee r_8$	$\neg x_2 \vee x_1 \vee r_1$	$\neg x_1 \vee r_2$
$\neg x_6 \vee x_8$	$x_6 \vee \neg x_8$	$x_2 \vee x_4 \vee r_3$	$\neg x_4 \vee x_5 \vee r_4$
$x_7 \vee x_5 \vee r_9$	$\neg x_7 \vee x_5 \vee r_{10}$	$\neg x_5 \vee x_3 \vee r_5$	$\neg x_3 \vee r_6$

$S_1 \leq 2 \qquad S_2' + \neg(S_1 \leq 1) \leq 1$

Aux sums: $S_1 = \sum_{i=1}^{6} r_i$; $S_2' = \sum_{i=7}^{10} r_i$; $S_2 = S_2' + \neg(S_1 \leq 1)$

Add new relaxation variables (S_2'), update AtMost$k$ constraint and add new AtMost1 constraint

Core-guided with soft constraints

$x_6 \vee x_2 \vee r_7$	$\neg x_6 \vee x_2 \vee r_8$	$\neg x_2 \vee x_1 \vee r_1$	$\neg x_1 \vee r_2$
$\neg x_6 \vee x_8$	$x_6 \vee \neg x_8$	$x_2 \vee x_4 \vee r_3$	$\neg x_4 \vee x_5 \vee r_4$
$x_7 \vee x_5 \vee r_9$	$\neg x_7 \vee x_5 \vee r_{10}$	$\neg x_5 \vee x_3 \vee r_5$	$\neg x_3 \vee r_6$

$\boxed{S_1 \leq 2}$ $\boxed{S_2' + \neg(S_1 \leq 1) \leq 1}$

$S_1 \geq 2 \rightarrow S_2' = 0$
$S_1 \leq 1 \rightarrow S_2' \leq 1$

Aux sums: $S_1 = \sum_{i=1}^{6} r_i$; $S_2' = \sum_{i=7}^{10} r_i$; $S_2 = S_2' + \neg(S_1 \leq 1)$

Add new relaxation variables (S_2'), update AtMost$k$ constraint and add new AtMost1 constraint

Core-guided with soft constraints

$x_6 \vee x_2 \vee r_7$	$\neg x_6 \vee x_2 \vee r_8$	$\neg x_2 \vee x_1 \vee r_1$	$\neg x_1 \vee r_2$
$\neg x_6 \vee x_8$	$x_6 \vee \neg x_8$	$x_2 \vee x_4 \vee r_3$	$\neg x_4 \vee x_5 \vee r_4$
$x_7 \vee x_5 \vee r_9$	$\neg x_7 \vee x_5 \vee r_{10}$	$\neg x_5 \vee x_3 \vee r_5$	$\neg x_3 \vee r_6$
$S_1 \leq 2$	$S_2' + \neg(S_1 \leq 1) \leq 1$		

Aux sums: $S_1 = \sum_{i=1}^{6} r_i$; $S_2' = \sum_{i=7}^{10} r_i$; $S_2 = S_2' + \neg(S_1 \leq 1)$

Instance is now SAT

Core-guided with soft constraints

$x_6 \vee x_2 \vee r_7$	$\neg x_6 \vee x_2 \vee r_8$	$\neg x_2 \vee x_1 \vee r_1$	$\neg x_1 \vee r_2$
$\neg x_6 \vee x_8$	$x_6 \vee \neg x_8$	$x_2 \vee x_4 \vee r_3$	$\neg x_4 \vee x_5 \vee r_4$
$x_7 \vee x_5 \vee r_9$	$\neg x_7 \vee x_5 \vee r_{10}$	$\neg x_5 \vee x_3 \vee r_5$	$\neg x_3 \vee r_6$

$S_1 \leq 2$ $\quad S_2' + \neg(S_1 \leq 1) \leq 1$

Aux sums: $S_1 = \sum_{i=1}^{6} r_i$; $S_2' = \sum_{i=7}^{10} r_i$; $S_2 = S_2' + \neg(S_1 \leq 1)$

BinCore solution is $|\varphi| - \mathcal{I} = 12 - 2 = 10$

Core-guided with soft constraints

$x_6 \vee x_2 \vee r_7$ $\neg x_6 \vee x_2 \vee r_8$ $\neg x_2 \vee x_1 \vee r_1$ $\neg x_1 \vee r_2$

$\neg x_6 \vee x_8$ $x_6 \vee \neg x_8$ $x_2 \vee x_4 \vee r_3$ $\neg x_4 \vee x_5 \vee r_4$

$x_7 \vee x_5 \vee r_9$ $\neg x_7 \vee x_5 \vee r_{10}$ $\neg x_5 \vee x_3 \vee r_5$ $\neg x_3 \vee r_6$

$S_1 \leq 2$ $S_2' + \neg(S_1 \leq 1) \leq 1$

Aux sums: $S_1 = \sum_{i=1}^{6} r_i$; $S_2' = \sum_{i=7}^{10} r_i$; $S_2 = S_2' + \neg(S_1 \leq 1)$

BinCore solution is $|\varphi| - \mathcal{I} = 12 - 2 = 10$

Only AtMost$k$ constraints used

Sums reused with $\neq$ RHSs

Relaxed soft clauses become **hard**

Handling soft cardinality constraints

- Algorithm for handling soft constraints (iteration j):

Handling soft cardinality constraints

- Algorithm for handling soft constraints (iteration j):

  1. Find original soft (non-relaxed) clauses in core j: S'_j

Handling soft cardinality constraints

- Algorithm for handling soft constraints (iteration j):

  1. Find original soft (non-relaxed) clauses in core j: S'_j
  2. Update RHS of each soft cardinality constraint in core, $i = 1, \ldots, r$:

$$S_{k_1} \leq R_{k_1} + 1, \ldots, S_{k_r} \leq R_{k_r} + 1$$

Handling soft cardinality constraints

- Algorithm for handling soft constraints (iteration j):

1. Find original soft (non-relaxed) clauses in core j: S'_j
2. Update RHS of each soft cardinality constraint in core, $i = 1, \ldots, r$:

$$S_{k_1} \leq R_{k_1} + 1, \ldots, S_{k_r} \leq R_{k_r} + 1$$

3. Create new sum:

$$S_j \triangleq S'_j + \sum_{i=1}^{r} \neg(S_{k_i} \leq R_{k_i})$$

Handling soft cardinality constraints

- Algorithm for handling soft constraints (iteration j):

1. Find original soft (non-relaxed) clauses in core j: S'_j
2. Update RHS of each soft cardinality constraint in core, $i = 1, \ldots, r$:

$$S_{k_1} \leq R_{k_1} + 1, \ldots, S_{k_r} \leq R_{k_r} + 1$$

3. Create new sum:

$$S_j \triangleq S'_j + \sum_{i=1}^{r} \neg(S_{k_i} \leq R_{k_i})$$

4. Create new soft cardinality constraint:

$$S_j \leq 1$$

Additional detail

- Sums represented in **unary**
  - Output bits of each sum can be used in different constraints
    - E.g.: S_1 compared both with 1 and 2
  - Thus, encodings of sums get reused

Additional detail

- Sums represented in **unary**
  - Output bits of each sum can be used in different constraints
    - E.g.: S_1 compared both with 1 and 2
  - Thus, encodings of sums get reused

- Each sum S_j associated with unsatisfiable core j
  - Each sum S_j remains unchanged as the algorithm executes
  - Multiple RHSs can be considered for each sum S_j

Additional detail

- Sums represented in **unary**
  - Output bits of each sum can be used in different constraints
    - E.g.: S_1 compared both with 1 and 2
  - Thus, encodings of sums get reused
- Each sum S_j associated with unsatisfiable core j
  - Each sum S_j remains unchanged as the algorithm executes
  - Multiple RHSs can be considered for each sum S_j
- BMO condition exploited for weighted instances

Besides our work ...

- Improvements to MSU3 [ABL13]
  - Stratification vs. BMO condition
- Partial MaxSAT resolution [NB14]
- Relaxation search [BDTK14]
  - Relate with preferences in SAT [RGM10]
- Incremental cardinality constraints [MSML14]
- Portfolios of solvers [AMS14]

What is BMO? – an example

$$\{\, (x_1, 3), (x_2, 3), (x_3, 1), (x_4, 1),$$
$$(\neg x_1 \vee \neg x_3, \top), (\neg x_2 \vee \neg x_4, \top), (\neg x_1 \vee \neg x_2, \top) \,\}$$

- How to solve this problem?

What is BMO? – an example

$$\{\ (x_1, 3), (x_2, 3), (x_3, 1), (x_4, 1),\\ (\neg x_1 \vee \neg x_3, \top), (\neg x_2 \vee \neg x_4, \top), (\neg x_1 \vee \neg x_2, \top)\ \}$$

- How to solve this problem?
- Formula with special structure

What is BMO? – an example

$$\{\ (x_1, 3), (x_2, 3), (x_3, 1), (x_4, 1),$$
$$(\neg x_1 \vee \neg x_3, \top), (\neg x_2 \vee \neg x_4, \top), (\neg x_1 \vee \neg x_2, \top)\ \}$$

- How to solve this problem?
- Formula with special structure
  - Weighted clauses encode two different criteria
    - Optimum must satisfy largest number of clauses with weight 3

What is BMO? – an example

$$\{\ (x_1, 3), (x_2, 3), (x_3, 1), (x_4, 1),$$
$$(\neg x_1 \lor \neg x_3, \top), (\neg x_2 \lor \neg x_4, \top), (\neg x_1 \lor \neg x_2, \top)\ \}$$

- How to solve this problem?
- Formula with special structure
  - Weighted clauses encode two different criteria
    - Optimum must satisfy largest number of clauses with weight 3
  - Iteratively optimize wrt each clause weight
    - First, target clauses with weight 3
    - Next, target clauses with weight 1, but take into account previous optimum

What is BMO? – an example

$$\{\ (x_1, 3), (x_2, 3), (x_3, 1), (x_4, 1),$$
$$(\neg x_1 \vee \neg x_3, \top), (\neg x_2 \vee \neg x_4, \top), (\neg x_1 \vee \neg x_2, \top)\ \}$$

- How to solve this problem?
- Formula with special structure
  - Weighted clauses encode two different criteria
    - Optimum must satisfy largest number of clauses with weight 3
  - Iteratively optimize wrt each clause weight
    - First, target clauses with weight 3
    - Next, target clauses with weight 1, but take into account previous optimum
- Example of **Boolean Multilevel Optimization (BMO)**

What is BMO? – the condition

- Set of clauses C
  - Partition of C: $\langle C_1, C_2, \ldots, C_m \rangle$
  - Clauses weights: $\langle w_1, w_2, \ldots, w_m = \top \rangle$
- BMO condition requires sufficiently distinct clause weights

$$w_i > \sum_{1 \leq j < i} w_j \cdot |C_j| \qquad i = m-1, \ldots, 2$$

  - Start by optimizing wrt to largest weight
  - Optimize wrt to i^{th} largest weight, but account for previous optima
  - Can harden clauses with already optimized weights

What is BMO? – the condition

- Set of clauses C
  - Partition of C: $\langle C_1, C_2, \ldots, C_m \rangle$
  - Clauses weights: $\langle w_1, w_2, \ldots, w_m = \top \rangle$
- BMO condition requires sufficiently distinct clause weights

$$w_i > \sum_{1 \leq j < i} w_j \cdot |C_j| \qquad i = m-1, \ldots, 2$$

  - Start by optimizing wrt to largest weight
  - Optimize wrt to i^{th} largest weight, but account for previous optima
  - Can harden clauses with already optimized weights

- Example:

$(x_3, 1), (x_4, 1)$
$(x_1, 3), (x_2, 3)$
$(\neg x_1 \vee \neg x_3, \top)$
$(\neg x_2 \vee \neg x_4, \top)$
$(\neg x_1 \vee \neg x_2, \top)$

Clauses: $\{C_1, C_2, C_3\}$, $|C_1| = 2, \ldots$
Weights: $\{w_1 = 1, w_2 = 3, w_3 = \top\}$

BMO condition holds: $3 > 2 \times 1$

MaxSAT solving with SAT oracles

- A sample of recent algorithms:

Algorithm	# Oracle Queries	Reference
Linear search SU	Exponential***	[e.g. LBP10]
Binary search	Linear*	[e.g. FM06]
FM/WMSU1/WPM1	Exponential**	[FM06,MSM08,MMSP09,ABL09a,ABGL12]
WPM2	Exponential**	[ABL10,ABGL13]
Bin-Core-Dis	Linear	[HMMS11,MHMS12]
Iterative MHS	Exponential	[DB11,DB13a,DB13b]

    * $\mathcal{O}(\log m)$ queries with SAT oracle, for (partial) unweighted BinCore
  ** Weighted case; depends on computed cores
*** On # bits of problem instance (due to weights)

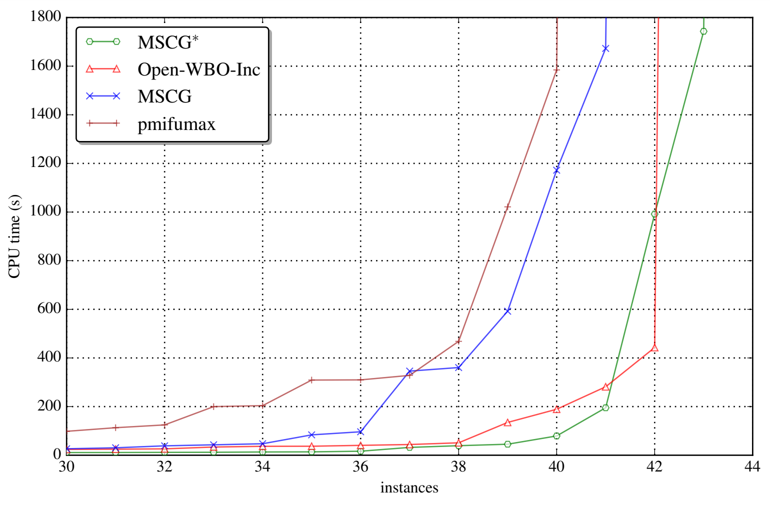
- But also additional recent work:
  - Progression
  - Soft cardinality constraints (OLL)
  - MaxSAT resolution

Outline

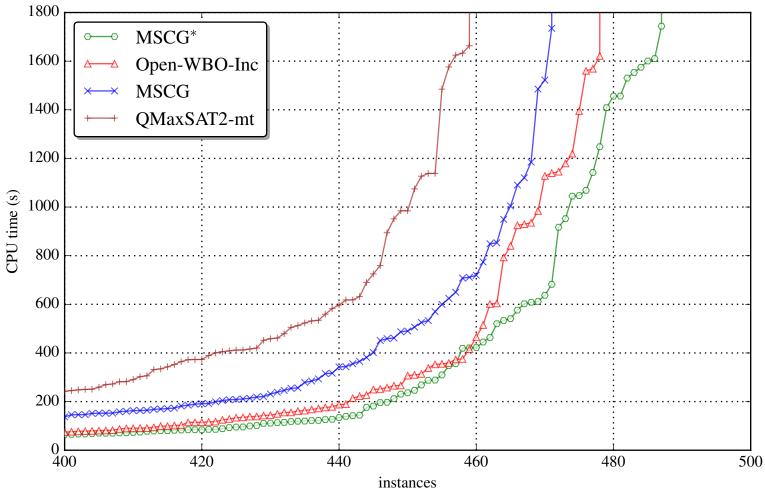
Experimental setup

- All industrial instances from 2014 MaxSAT evaluation
- HPC cluster:
  - Intel Xeon E5-2630-v2 2.60GHz processors with 64GB of RAM
  - Linux OS
- 1800s timeout and 3.5GB of memory limit
- MaxSAT solvers: best from 2013 & 2014 MaxSAT evaluations
  - QMaxSAT2-mt (only partial MaxSAT) [KZFH12]
  - pmifumax [J13]
  - WPM1 2013 [ABL13]
  - Open-WBO-Inc [MSML14]
  - Eva500a [NB14]
  - MSCG [IMMLMS14]
  - MSCG*: current best configuration

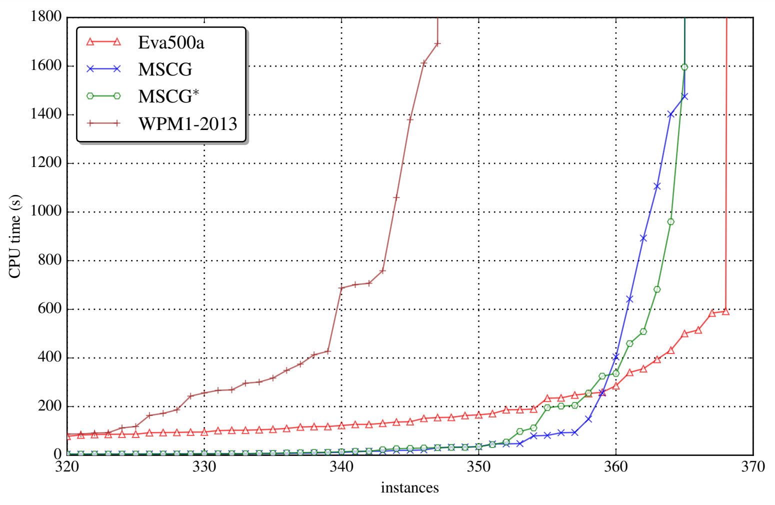
Current results – plain MaxSAT industrial



Current results – partial MaxSAT industrial



Current results – weighted partial MaxSAT industrial



Outline

Conclusions – today

- Remarkable performance improvements in MaxSAT solving
- Fast growing number of practical applications
- MaxSAT also used for solving other optimization problems
- Very active area of research, with many new algorithms
  - Best algorithms in practice are core-guided

Conclusions – today

- Remarkable performance improvements in MaxSAT solving
- Fast growing number of practical applications
- MaxSAT also used for solving other optimization problems
- Very active area of research, with many new algorithms
  - Best algorithms in practice are core-guided
- Where to go from here?
  - Formula preprocessing?
  - Parallelization?
  - More work on portfolios?

Conclusions – today

- Remarkable performance improvements in MaxSAT solving
- Fast growing number of practical applications
- MaxSAT also used for solving other optimization problems
- Very active area of research, with many new algorithms
  - Best algorithms in practice are core-guided
- Where to go from here?
  - Formula preprocessing?
  - Parallelization?
  - More work on portfolios?
  - Better CNF encodings?

Conclusions – today

- Remarkable performance improvements in MaxSAT solving
- Fast growing number of practical applications
- MaxSAT also used for solving other optimization problems
- Very active area of research, with many new algorithms
  - Best algorithms in practice are core-guided
- Where to go from here?
  - Formula preprocessing?
  - Parallelization?
  - More work on portfolios?
  - Better CNF encodings?
  - Better algorithms?

Conclusions – the three lectures

- Reviewed organization of modern CDCL SAT solvers
- Overviewed problem solving with SAT oracles
- Investigated minimal sets computation with SAT oracles
- Discussed minimal cardinality set (i.e. optimization / MaxSAT) computation with SAT oracles
- Identified possible research topics

Thanks!

Thanks to the researchers and visitors at UCD, INESC-ID & UofS

J. Argelich, F. Arif, A. Belov, H. Chen,
C. Dodaro, F. Heras, A. Ignatiev, M. Janota,
I. Lynce, V. Manquinho, C. Mencía,
A. Morgado, J. Planes, A. Previti,
and many others