Enumerating Infeasibility: Finding Multiple MUSes Quickly

Mark Liffiton and Ammar Malik

Department of Computer Science
Illinois Wesleyan University

{mliffito,amalik}@iwu.edu

http://www.iwu.edu/~mliffito/

CPAIOR — May 21, 2013
Yorktown Heights, NY

Overview

Problem

Analyzing infeasible constraint sets

"Constraint"
= SAT, SMT, CP, LP, IP, MIP, . . .
(Implemented/tested w/ SAT & SMT.)

"Analyzing"
= Enumerating MUSes/IISes
("Explanations" of infeasibility.)

Overview

Problem

Analyzing infeasible constraint sets

"Constraint"
= SAT, SMT, CP, LP, IP, MIP, ...
(Implemented/tested w/ SAT & SMT.)

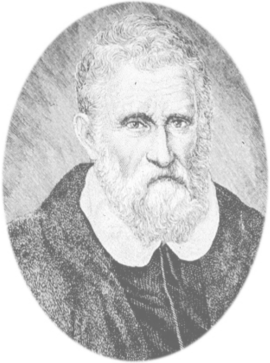
"Analyzing"
= Enumerating MUSes/IISes
("Explanations" of infeasibility.)

Contributions

1. MARCO: Novel, efficient algorithm for MUS/IIS enumeration.
2. POLO: Framework for analyzing/solving infeasibility analysis problems.

(Both constraint-agnostic.)

Outline



Definitions

"Characteristic Subsets" of an infeasible constraint set C

MUS Minimal Unsatisfiable Subset
aka Irreducible Inconsistent Subsystem (IIS).
$M \subseteq C$ s.t. M is UNSAT and $\forall c \in M : M \setminus \{c\}$ is SAT

MSS Maximal Satisfiable Subset
a generalization of MaxSAT / MaxFS.
$M \subseteq C$ s.t. M is SAT and $\forall c \in C \setminus M : M \cup \{c\}$ is UNSAT

MCS Minimal Correction Set
the complement of some MSS;
removal yields a satisfiable MSS ("corrects" the infeasibility).
$M \subseteq C$ s.t. $C \setminus M$ is SAT and $\forall c \in M : (C \setminus M) \cup \{c\}$ is UNSAT

Definitions / Example

"Characteristic Subsets"

- **MUS** Minimal Unsatisfiable Subset
- **MSS** Maximal Satisfiable Subset
- **MCS** Minimal Correction Set

Example (Constraint set C, Boolean SAT)

$$C = \{ \underset{1}{(a)} , \underset{2}{(\neg a \vee b)} , \underset{3}{(\neg b)} , \underset{4}{(\neg a)} \}$$

MUSes
$\{1,2,3\}$
$\{1,4\}$

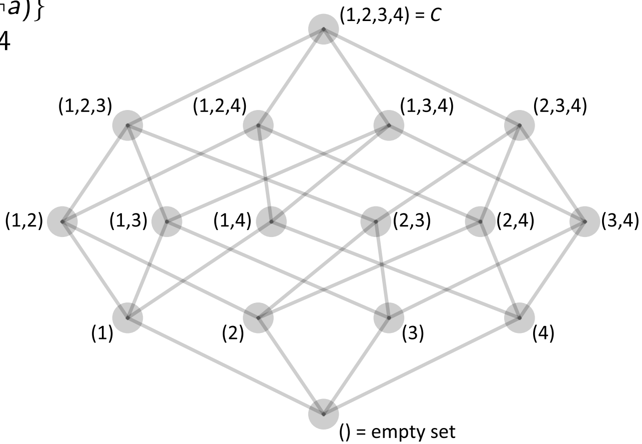
MSSes	MCSes
$\{2,3,4\}$	$\{1\}$
$\{1,3\}$	$\{2,4\}$
$\{1,2\}$	$\{3,4\}$

Example, Powerset Visualization

Hasse diagram of powerset for:

$$C = \{(a), (\neg a \vee b), (\neg b), (\neg a)\}$$

1 2 3 4

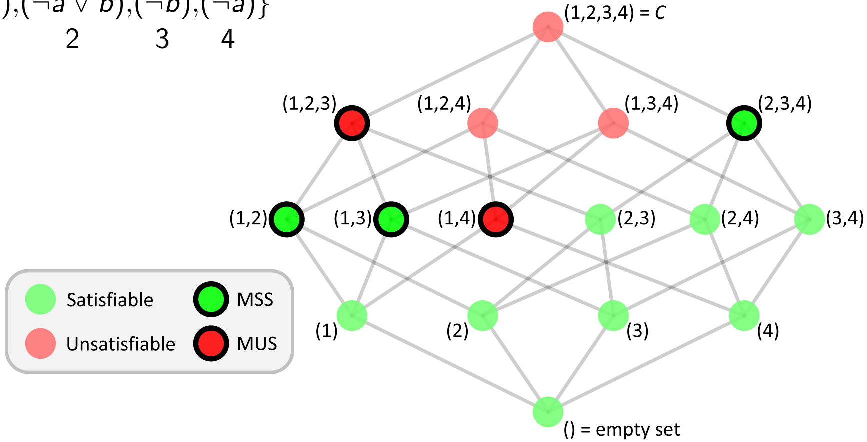


Example, Powerset Visualization

Hasse diagram of powerset for:

$$C = \{(a),(\neg a \vee b),(\neg b),(\neg a)\}$$

1 2 3 4



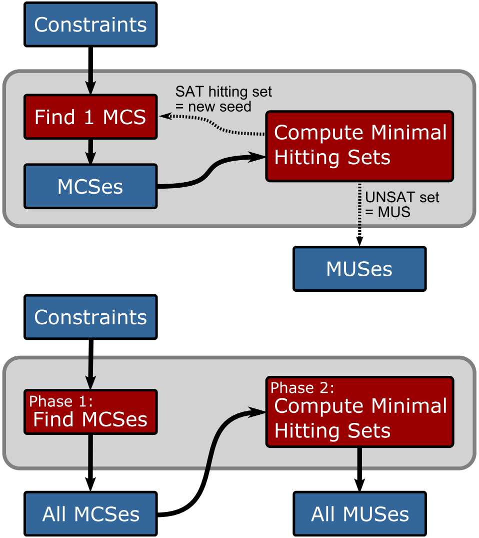
Earlier Work on MUS Enumeration

Dualize and Advance (DAA)

[Bailey & Stuckey, *PADL* 2005]

CAMUS

[Liffiton & Sakallah, *SAT* 2005]



Goals of This Work

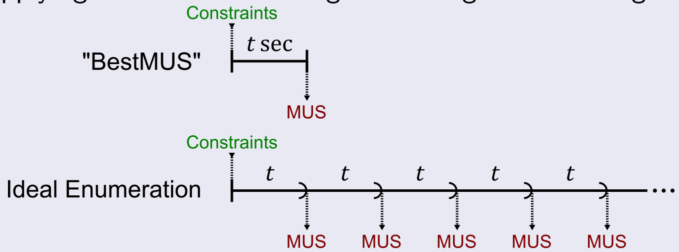
1. Constraint-agnostic
2. None of the scaling / intractability issues of DAA or CAMUS
3. As efficient as the best single-MUS algorithm
4. Good anytime behavior

Goals of This Work

1. Constraint-agnostic
2. None of the scaling / intractability issues of DAA or CAMUS
3. As efficient as the best single-MUS algorithm
4. Good anytime behavior

Ideal

… like repeatedly applying the best-known single-MUS algorithm for a given constraint type…



MARCO POLO

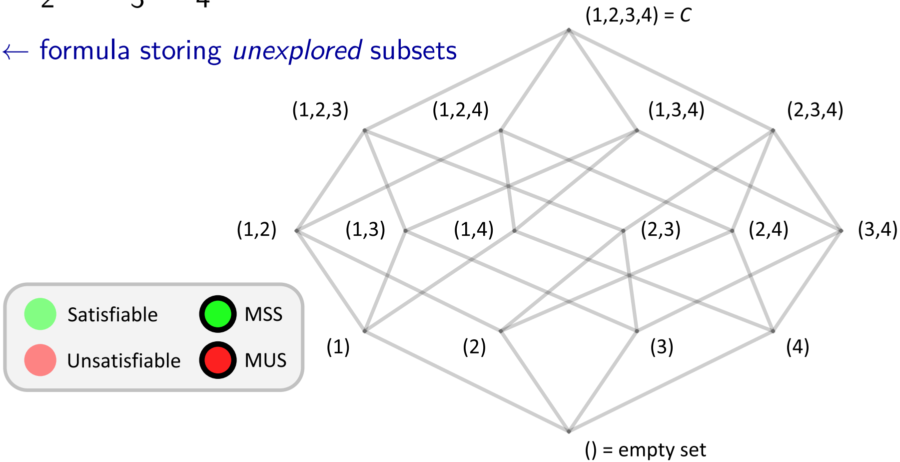
Named after the Venetian explorer Marco Polo.

- MARCO: **Ma**pping **R**egions of **Co**nstraint sets — the MUS enumeration algorithm.
- POLO: **Po**werset **Lo**gic — the general technique of maintaining a "map" of the powerset as a propositional logic formula.

POLO Framework

$C = \{(a),(\neg a \vee b),(\neg b),(\neg a)\}$
1 2 3 4

$Map = \top \leftarrow$ formula storing *unexplored* subsets



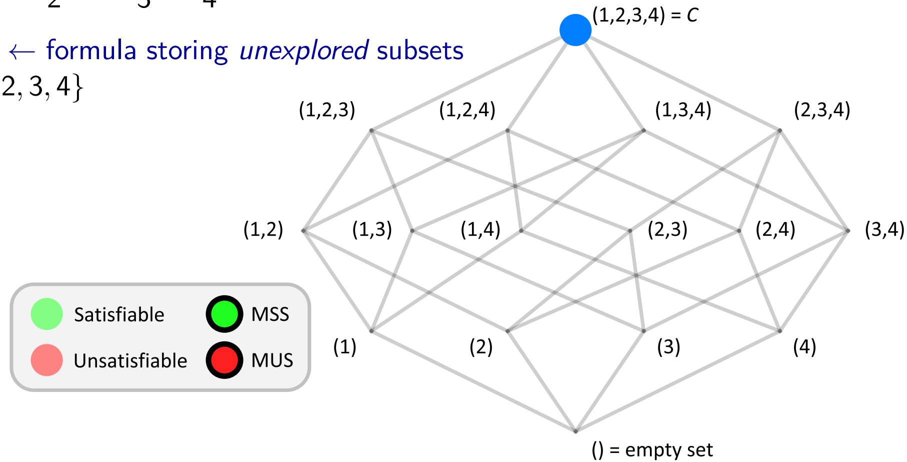
POLO Framework

$C = \{(a),(\neg a \lor b),(\neg b),(\neg a)\}$

1 2 3 4

$Map = \top \leftarrow$ formula storing *unexplored* subsets

Seed: $\{1, 2, 3, 4\}$



POLO Framework

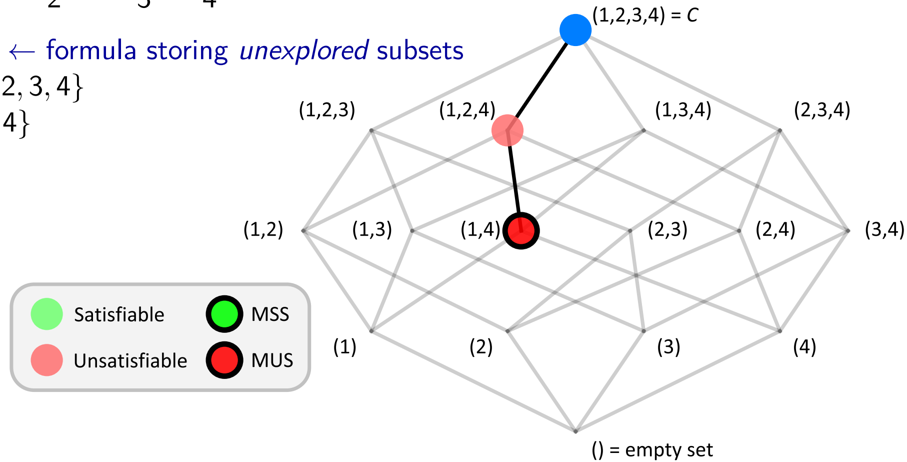
$C = \{(a),(\neg a \vee b),(\neg b),(\neg a)\}$

1 2 3 4

$Map = \top \leftarrow$ formula storing *unexplored* subsets

Seed: $\{1, 2, 3, 4\}$

MUS: $\{1, 4\}$



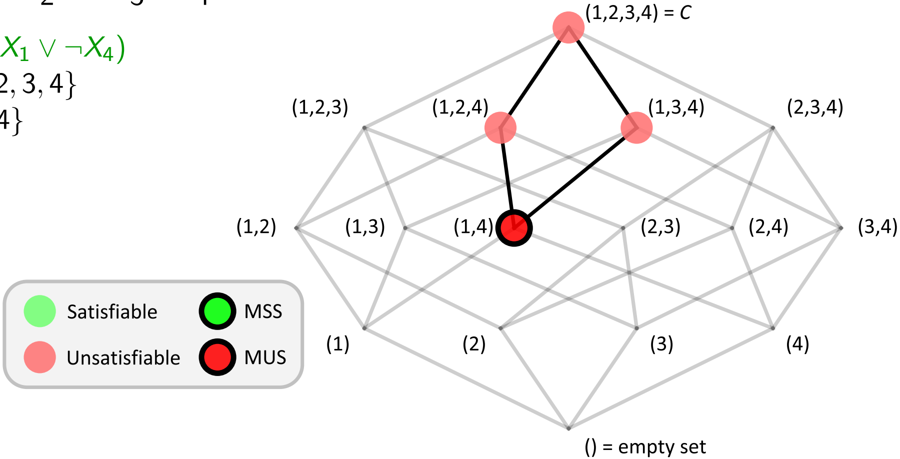
POLO Framework

$C = \{(a), (\neg a \vee b), (\neg b), (\neg a)\}$
1 2 3 4

$Map = (\neg X_1 \vee \neg X_4)$

Seed: $\{1, 2, 3, 4\}$

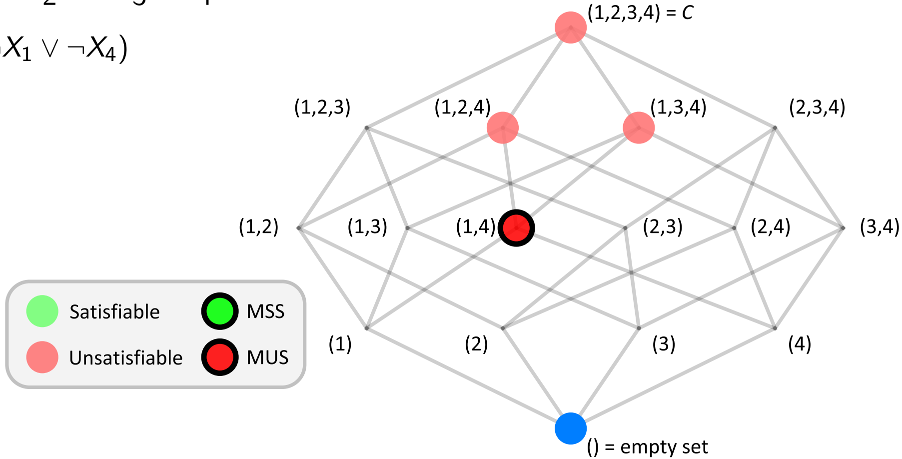
MUS: $\{1, 4\}$



POLO Framework

$C = \{(a),(\neg a \vee b),(\neg b),(\neg a)\}$
1 2 3 4

$Map = (\neg X_1 \vee \neg X_4)$
Seed: {}



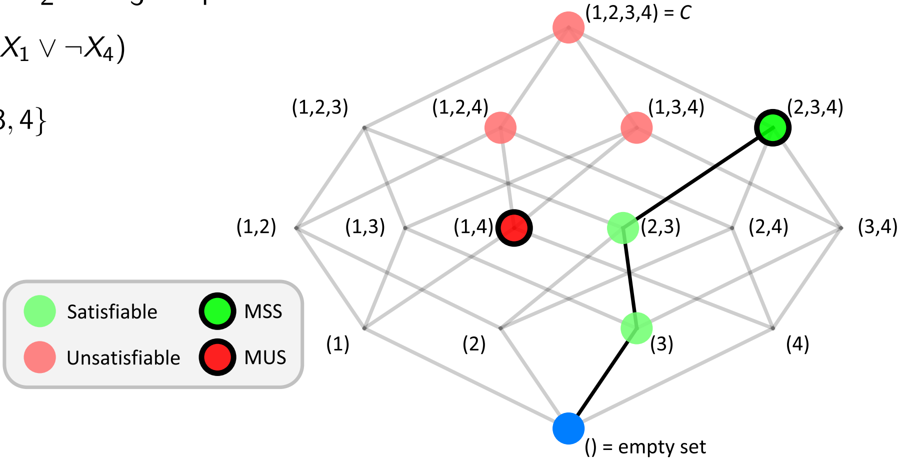
POLO Framework

$C = \{(a),(\neg a \vee b),(\neg b),(\neg a)\}$
1 2 3 4

$Map = (\neg X_1 \vee \neg X_4)$

Seed: $\{\}$

MSS: $\{2, 3, 4\}$



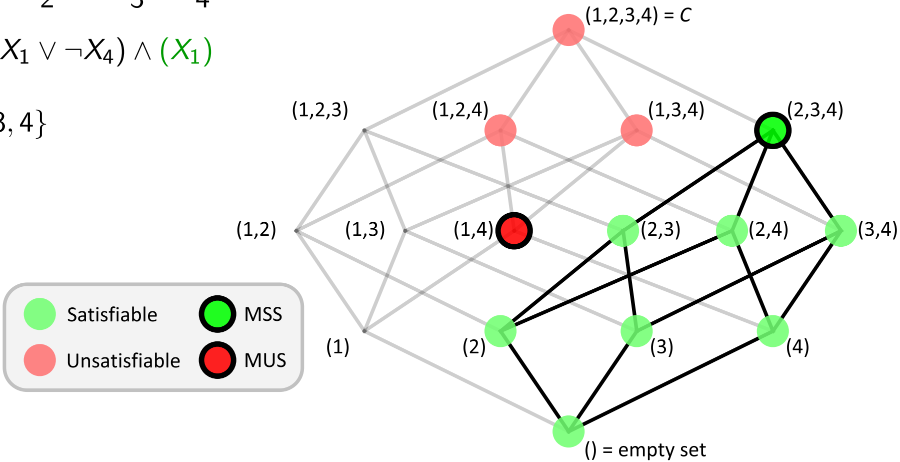
POLO Framework

$$C = \{(a), (\neg a \vee b), (\neg b), (\neg a)\}$$
1 2 3 4

$$Map = (\neg X_1 \vee \neg X_4) \wedge (X_1)$$

Seed: {}

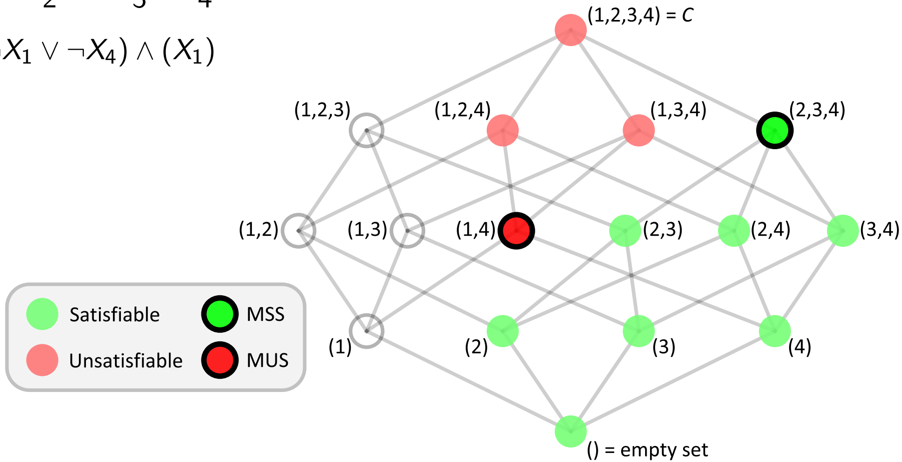
MSS: $\{2, 3, 4\}$



POLO Framework

$$C = \{(a), (\neg a \vee b), (\neg b), (\neg a)\}$$
1 2 3 4

$$Map = (\neg X_1 \vee \neg X_4) \wedge (X_1)$$



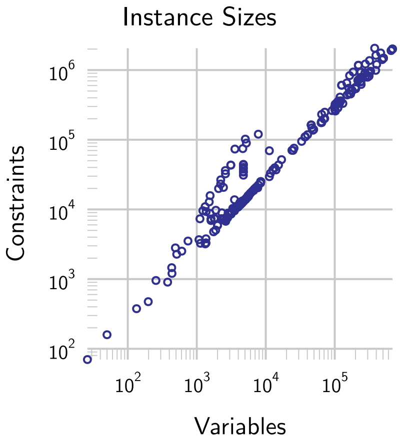
MARCO Algorithm: Pseudocode

input: unsatisfiable constraint set $C = \{C_1, C_2, C_3, \ldots, C_n\}$
output: MSSes and MUSes of C as they are discovered

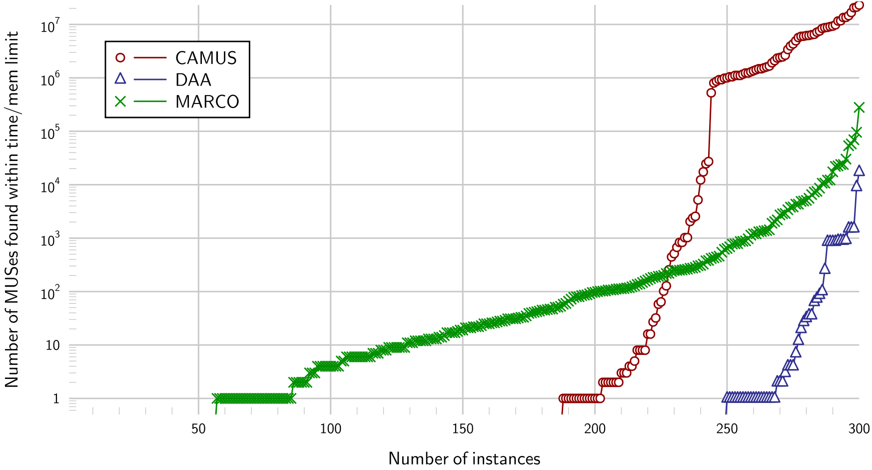
$Map \leftarrow$ BoolFormula($nvars = |C|$) ◁ *Empty formula over $|C|$ Boolean vars*
while Map **is satisfiable**:
    $m \leftarrow$ **getModel**(Map)
    $seed \leftarrow \{C_i \in C : m[x_i] = \text{True}\}$ ◁ *Project the assignment m onto C*
    if $seed$ **is satisfiable**:
        $MSS \leftarrow$ **grow**($seed, C$)
        yield MSS
        $Map \leftarrow Map \wedge$ **blockDown**(MSS)
    else:
        $MUS \leftarrow$ **shrink**($seed, C$) ◁ *Using any single-MUS algorithm*
        yield MUS
        $Map \leftarrow Map \wedge$ **blockUp**(MUS)

Experimental Setup

- Implementation
  - **shrink**: MUSer2 [Belov & Marques-Silva *JSAT* 2012]
  - **grow** & *Map*: MiniSAT v2.2
- Benchmarks
  - 300 Boolean CNF instances from 2011 SAT Competition, "MUS Track"
  - 246 instances for which any algorithm found at least one MUS
  - Vars: min=26, max=686,767
  - Clauses: min=70, max=2,058,906
- Experiments
  - CPU: AMD Phenom II X4 @ 3.4GHz
  - RAM limit: 1800MB
  - Time limit: 1hr / 3600sec



Experimental Results: MUSes Produced Within Resource Limits



Conclusion

Ongoing Work

- Merging/comparing to eMUS [Previti & Marques-Silva, *AAAI* 2013]: same approach with slightly different seed generation.
- Improving MARCO w/ better guidance, increased integration between *Map* and constraint solver.
- Applying POLO to other problems in infeasibility analysis.

Conclusion

Contributions

1. MARCO Algorithm:
   - Returns MUSes more quickly than existing enumeration algorithms
   - Avoids scaling issues on previously-intractable instances
   - Can "plug in" any advances in single-MUS algorithms
2. POLO Framework:
   - Constraint-agnostic
   - Intuitive, flexible approach for analyzing infeasibility analysis
   - Requirements: SAT solver + any constraint solver

Conclusion

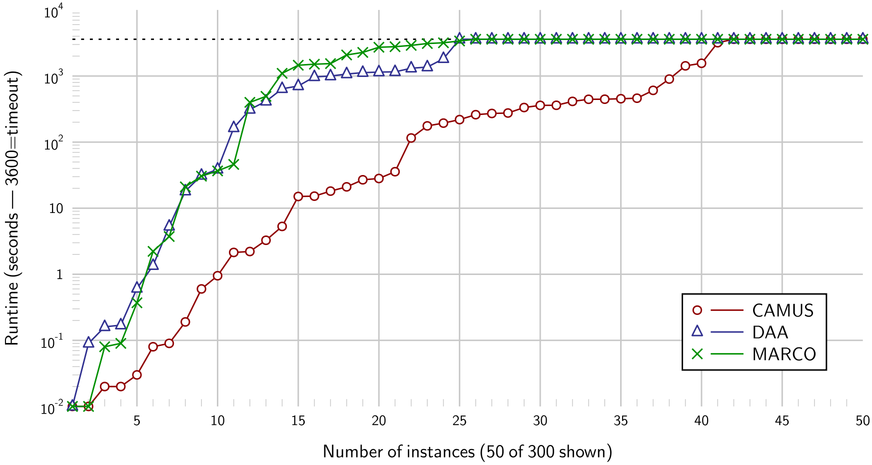
Contributions

1. MARCO Algorithm:
   - Returns MUSes more quickly than existing enumeration algorithms
   - Avoids scaling issues on previously-intractable instances
   - Can "plug in" any advances in single-MUS algorithms
2. POLO Framework:
   - Constraint-agnostic
   - Intuitive, flexible approach for analyzing infeasibility analysis
   - Requirements: SAT solver + any constraint solver

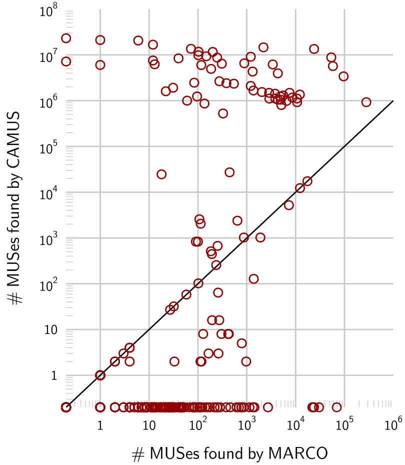
Thank you.

Source code: `http://www.iwu.edu/~mliffito/marco/`

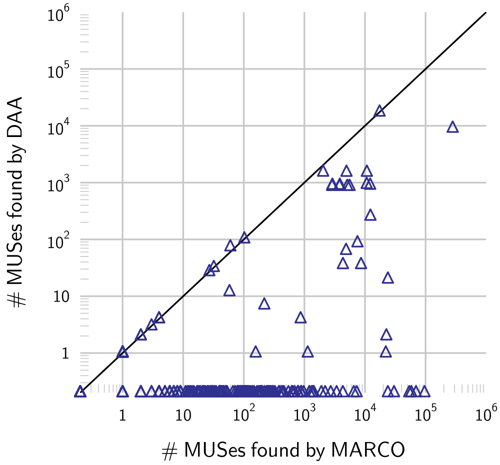
Experimental Results: Runtime for Complete Enumeration



Experimental Results: Pairwise Comparison: MARCO & CAMUS



Experimental Results: Pairwise Comparison: MARCO & DAA



Experimental Results: Anytime Traces, CAMUS & MARCO