A Scalable Two Stage Approach to Computing Optimal Decision Sets

Alexey Ignatiev[1], Edward Lam[1,2], Peter J. Stuckey[1], Joao Marques-Silva[3]

February 4–7, 2021 | **AAAI**

[1] Monash University, Melbourne, Australia
[2] CSIRO Data61, Melbourne, Australia
[3] ANITI, IRIT, CNRS, Toulouse, France

Problem and state of the art

Problem example

(classication scenario)

Problem example

(classication scenario)

#	Day	Venue	Weather	TV Show	*Date?*
e_1	Weekday	Dinner	Warm	Bad	No
e_2	Weekend	Club	Warm	Bad	Yes
e_3	Weekend	Club	Warm	Bad	Yes
e_4	Weekend	Club	Cold	Good	No

Problem example

(classication scenario)

#	Day	Venue	Weather	TV Show	*Date?*
e_1	Weekday	Dinner	Warm	Bad	No
e_2	Weekend	Club	Warm	Bad	Yes
e_3	Weekend	Club	Warm	Bad	Yes
e_4	Weekend	Club	Cold	Good	No

IF TV Show $=$ Good **THEN** Date $=$ No
IF Day $=$ Weekday **THEN** Date $=$ No
IF TV Show $=$ Bad $\wedge$ Day $=$ Weekend **THEN** Date $=$ Yes

Problem example

(classication scenario)

#	Day	Venue	Weather	TV Show	***Date?***
e_1	Weekday	Dinner	Warm	Bad	No
e_2	Weekend	Club	Warm	Bad	Yes
e_3	Weekend	Club	Warm	Bad	Yes
e_4	Weekend	Club	Cold	Good	No

IF TV Show = Good **THEN** Date = No
IF Day = Weekday **THEN** Date = No
IF TV Show = Bad $\wedge$ Day = Weekend **THEN** Date = Yes

unordered set of *if-then* rules

Problem example

(classication scenario)

#	Day	Venue	Weather	TV Show	*Date?*
e_1	Weekday	Dinner	Warm	Bad	No
e_2	Weekend	Club	Warm	Bad	Yes
e_3	Weekend	Club	Warm	Bad	Yes
e_4	Weekend	Club	Cold	Good	No

IF TV Show = Good **THEN** Date = No
IF Day = Weekday **THEN** Date = No
IF TV Show = Bad $\wedge$ Day = Weekend **THEN** Date = Yes

unordered set of *if-then* rules

must respect training data & *generalize well...*

Problem example

(classication scenario)

#	Day	Venue	Weather	TV Show	*Date?*
e_1	Weekday	Dinner	Warm	Bad	No
e_2	Weekend	Club	Warm	Bad	Yes
e_3	Weekend	Club	Warm	Bad	Yes
e_4	Weekend	Club	Cold	Good	No

IF TV Show $=$ Good **THEN** Date $=$ No
IF Day $=$ Weekday **THEN** Date $=$ No
IF TV Show $=$ Bad $\wedge$ Day $=$ Weekend **THEN** Date $=$ Yes

unordered set of *if-then* rules

must respect training data & *generalize well...*

the smaller — the better!

Problem example

(classication scenario)

#	Day	Venue	Weather	TV Show	*Date?*
e_1	Weekday	Dinner	Warm	Bad	No
e_2	Weekend	Club	Warm	Bad	Yes
e_3	Weekend	Club	Warm	Bad	Yes
e_4	Weekend	Club	Cold	Good	No

IF TV Show $=$ Good **THEN** Date $=$ No
IF Day $=$ Weekday **THEN** Date $=$ No
IF TV Show $=$ Bad $\wedge$ Day $=$ Weekend **THEN** Date $=$ Yes

unordered set of *if-then* rules

must respect training data & *generalize well...*

the smaller — the better!

highly interpretable!

rule-based models

rule-based models

⬇

"transparent" **and easy to interpret**

rule-based models

⬇

“*transparent*” **and easy to interpret**

⬇

come in handy in XAI

State of the art — a typical approach

input : training data E
output: *smallest*[a] decision set ϕ

```
N ← LB                              # N equals a lower bound on |φ|, which is often set to 1
while True:
    F ← Encode(E, N)                # encode problem "is there a decision set φ of size N for data E?"
    (st, μ) ← Oracle(F)             # call a reasoning oracle to answer the question
    if st is True:
        break
    N ← N + 1
φ ← ExtractRules(μ)                 # extract decision set φ from satisfying assignment μ
return φ
```

[a] wrt. the number of rules or literals

State of the art — a typical approach

input : training data E

output: *smallest*[a] decision set ϕ

```
N ← LB                              # N equals a lower bound on |φ|, which is often set to 1
while True:
    F ← Encode(E, N)                # encode problem "is there a decision set φ of size N for data E?"
    (st, μ) ← Oracle(F)             # call a reasoning oracle to answer the question
    if st is True:
        break
    N ← N + 1
φ ← ExtractRules(μ)                 # extract decision set φ from satisfying assignment μ
return φ
```

[a] wrt. the number of rules or literals

encoding is too large!

(does not scale)

Our approach

divide the process into two stages:

divide the process into two stages:

1. enumerate individual rules

divide the process into two stages:

1. enumerate individual rules
   - MaxSAT-based
   - incremental!

divide the process into two stages:

1. enumerate individual rules
   - MaxSAT-based
   - incremental!
   - breaking symmetric rules

divide the process into two stages:

1. enumerate individual rules
   - MaxSAT-based
   - incremental!
   - breaking symmetric rules

2. compute smallest rule cover

divide the process into two stages:

1. enumerate individual rules

- MaxSAT-based
- incremental!
- breaking symmetric rules

2. compute smallest rule cover

- reduced to set cover
- solved with ILP/MaxSAT

divide the process into two stages:

1. enumerate individual rules
   - MaxSAT-based
   - incremental!
   - breaking symmetric rules

2. compute smallest rule cover
   - reduced to set cover
   - solved with ILP/MaxSAT

+

each class is computed *independently*

Our take on the problem

divide the process into two stages:

1. enumerate individual rules

- MaxSAT-based
- incremental!
- breaking symmetric rules

2. compute smallest rule cover

- reduced to set cover
- solved with ILP/MaxSAT

+

each class is computed *independently*

the idea is to scale better

each rule is a solution to MaxSAT formula

$$\psi \triangleq H \wedge S$$

each rule is a solution to MaxSAT formula

$$\psi \triangleq H \wedge S$$

H — hard clauses

each rule is a solution to MaxSAT formula

$$\psi \triangleq H \wedge S$$

H — hard clauses

1. coverage constraints:
   - rule must cover $\geq$ 1 right instances

each rule is a solution to MaxSAT formula

$$\psi \triangleq \mathcal{H} \wedge \mathcal{S}$$

$\mathcal{H}$ — hard clauses

1. **coverage constraints:**
   - **rule must cover $\geq$ 1 right instances**
2. **discrimination constraints:**
   - **rule must not cover any wrong instances**

each rule is a solution to MaxSAT formula

$$\psi \triangleq \mathcal{H} \wedge \mathcal{S}$$

$\mathcal{H}$ — hard clauses	$\mathcal{S}$ — soft clauses
1. coverage constraints: • rule must cover ≥ 1 right instances 2. discrimination constraints: • rule must not cover any wrong instances	• minimize the number of used literals

each rule is a solution to MaxSAT formula

$$\psi \triangleq \mathcal{H} \wedge \mathcal{S}$$

$\mathcal{H}$ — hard clauses

1. coverage constraints:
   - rule must cover $\geq$ 1 right instances
2. discrimination constraints:
   - rule must not cover any wrong instances

$\mathcal{S}$ — soft clauses

- minimize the number of used literals

$\mathcal{O}(K + M)$ variables and $\mathcal{O}(K \times M)$ clauses

(K — number of features, M — number of training instances)

Stage 2 — computing rule cover

#	Day	Venue	Weather	TV Show	*Date?*
e_1	Weekday	Dinner	Warm	Bad	No
e_2	Weekend	Club	Warm	Bad	Yes
e_3	Weekend	Club	Warm	Bad	Yes
e_4	Weekend	Club	Cold	Good	No

Stage 2 — computing rule cover

#	Day	Venue	Weather	TV Show	*Date?*
e_1	Weekday	Dinner	Warm	Bad	No
e_2	Weekend	Club	Warm	Bad	Yes
e_3	Weekend	Club	Warm	Bad	Yes
e_4	Weekend	Club	Cold	Good	No

$\pi_1 = [\ \textbf{IF}\ \text{Day} = \text{Weekday} \quad \textbf{THEN}\ \text{Date} = \text{No}\ ]$

Stage 2 — computing rule cover

#	Day	Venue	Weather	TV Show	*Date?*
e_1	Weekday	Dinner	Warm	Bad	No
e_2	Weekend	Club	Warm	Bad	Yes
e_3	Weekend	Club	Warm	Bad	Yes
e_4	Weekend	Club	Cold	Good	No

$\pi_1 = [$ **IF** Day $=$ Weekday **THEN** Date $=$ No $]$

$\pi_2 = [$ **IF** Venue $=$ Dinner **THEN** Date $=$ No $]$

Stage 2 — computing rule cover

#	Day	Venue	Weather	TV Show	*Date?*
e_1	Weekday	Dinner	Warm	Bad	No
e_2	Weekend	Club	Warm	Bad	Yes
e_3	Weekend	Club	Warm	Bad	Yes
e_4	Weekend	Club	Cold	Good	No

$\pi_1 = [$ **IF** Day $=$ Weekday **THEN** Date $=$ No $]$

$\pi_2 = [$ **IF** Venue $=$ Dinner **THEN** Date $=$ No $]$

$\pi_3 = [$ **IF** Weather $=$ Cold **THEN** Date $=$ No $]$

Stage 2 — computing rule cover

#	Day	Venue	Weather	TV Show	*Date?*
e_1	Weekday	Dinner	Warm	Bad	No
e_2	Weekend	Club	Warm	Bad	Yes
e_3	Weekend	Club	Warm	Bad	Yes
e_4	Weekend	Club	Cold	Good	No

$\pi_1 = [$ **IF** Day = Weekday **THEN** Date = No $]$

$\pi_2 = [$ **IF** Venue = Dinner **THEN** Date = No $]$

$\pi_3 = [$ **IF** Weather = Cold **THEN** Date = No $]$

$\pi_4 = [$ **IF** TV Show = Good **THEN** Date = No $]$

Stage 2 — computing rule cover

#	Day	Venue	Weather	TV Show	*Date?*
e_1	Weekday	Dinner	Warm	Bad	No
e_2	Weekend	Club	Warm	Bad	Yes
e_3	Weekend	Club	Warm	Bad	Yes
e_4	Weekend	Club	Cold	Good	No

$\pi_1 = [$ **IF** Day $=$ Weekday **THEN** Date $=$ No $]$

$\pi_2 = [$ **IF** Venue $=$ Dinner **THEN** Date $=$ No $]$

$\pi_3 = [$ **IF** Weather $=$ Cold **THEN** Date $=$ No $]$

$\pi_4 = [$ **IF** TV Show $=$ Good **THEN** Date $=$ No $]$

$b_j \in \{0, 1\}$ **and** $s_j = |\pi_j|$ **for each** π_j

Stage 2 — computing rule cover

#	Day	Venue	Weather	TV Show	***Date?***
e_1	Weekday	Dinner	Warm	Bad	No
e_2	Weekend	Club	Warm	Bad	Yes
e_3	Weekend	Club	Warm	Bad	Yes
e_4	Weekend	Club	Cold	Good	No

$\pi_1 = [$ **IF** Day = Weekday **THEN** Date = No $]$

$\pi_2 = [$ **IF** Venue = Dinner **THEN** Date = No $]$

$\pi_3 = [$ **IF** Weather = Cold **THEN** Date = No $]$

$\pi_4 = [$ **IF** TV Show = Good **THEN** Date = No $]$

$b_j \in \{0, 1\}$ **and** $s_j = |\pi_j|$ **for each** π_j

$A = (a_{ij})$, $a_{ij} = 1$ **iff** π_j **covers** e_i

Stage 2 — computing rule cover

#	Day	Venue	Weather	TV Show	*Date?*
e_1	Weekday	Dinner	Warm	Bad	No
e_2	Weekend	Club	Warm	Bad	Yes
e_3	Weekend	Club	Warm	Bad	Yes
e_4	Weekend	Club	Cold	Good	No

$\pi_1 = [$ **IF** Day $=$ Weekday **THEN** Date $=$ No $]$

$\pi_2 = [$ **IF** Venue $=$ Dinner **THEN** Date $=$ No $]$

$\pi_3 = [$ **IF** Weather $=$ Cold **THEN** Date $=$ No $]$

$\pi_4 = [$ **IF** TV Show $=$ Good **THEN** Date $=$ No $]$

$b_j \in \{0, 1\}$ **and** $s_j = |\pi_j|$ **for each** π_j

$A = (a_{ij})$, $a_{ij} = 1$ **iff** π_j **covers** e_i

	π_1	π_2	π_3	π_4
a_{ij}	1	1	0	0
	0	0	1	1
s_j	1	1	1	1

Stage 2 — computing rule cover

#	Day	Venue	Weather	TV Show	*Date?*
e_1	Weekday	Dinner	Warm	Bad	No
e_2	Weekend	Club	Warm	Bad	Yes
e_3	Weekend	Club	Warm	Bad	Yes
e_4	Weekend	Club	Cold	Good	No

$\pi_1 = [$ **IF** Day = Weekday **THEN** Date = No $]$

$\pi_2 = [$ **IF** Venue = Dinner **THEN** Date = No $]$

$\pi_3 = [$ **IF** Weather = Cold **THEN** Date = No $]$

$\pi_4 = [$ **IF** TV Show = Good **THEN** Date = No $]$

$b_j \in \{0, 1\}$ **and** $s_j = |\pi_j|$ **for each** π_j

$A = (a_{ij}),\ a_{ij} = 1$ **iff** π_j **covers** e_i

	π_1	π_2	π_3	π_4
a_{ij}	1	1	0	0
	0	0	1	1
s_j	1	1	1	1

$$\textbf{minimize} \sum_j s_j \cdot b_j$$

$$\textbf{subject to} \sum_j a_{ij} \cdot b_j \geq 1, \forall i$$

Breaking symmetric rules

#	Day	Venue	Weather	TV Show	*Date?*
e_1	Weekday	Dinner	Warm	Bad	No
e_2	Weekend	Club	Warm	Bad	Yes
e_3	Weekend	Club	Warm	Bad	Yes
e_4	Weekend	Club	Cold	Good	No

Breaking symmetric rules

#	Day	Venue	Weather	TV Show	*Date?*
e_1	Weekday	Dinner	Warm	Bad	No
e_2	Weekend	Club	Warm	Bad	Yes
e_3	Weekend	Club	Warm	Bad	Yes
e_4	Weekend	Club	Cold	Good	No

Breaking symmetric rules

#	Day	Venue	Weather	TV Show	*Date?*
e_1	Weekday	Dinner	Warm	Bad	No
e_2	Weekend	Club	Warm	Bad	Yes
e_3	Weekend	Club	Warm	Bad	Yes
e_4	Weekend	Club	Cold	Good	No

IF TV Show = Good **THEN** Date = No

Breaking symmetric rules

#	Day	Venue	Weather	TV Show	*Date?*
e_1	Weekday	Dinner	Warm	Bad	No
e_2	Weekend	Club	Warm	Bad	Yes
e_3	Weekend	Club	Warm	Bad	Yes
e_4	Weekend	Club	Cold	Good	No

IF TV Show = Good **THEN** Date = No

vs.

IF Weather = Cold **THEN** Date = No

Breaking symmetric rules

#	Day	Venue	Weather	TV Show	*Date?*
e_1	Weekday	Dinner	Warm	Bad	No
e_2	Weekend	Club	Warm	Bad	Yes
e_3	Weekend	Club	Warm	Bad	Yes
e_4	Weekend	Club	Cold	Good	No

IF TV Show = Good **THEN** Date = No

vs.

IF Weather = Cold **THEN** Date = No

rules covering same instances are symmetric

Breaking symmetric rules

#	Day	Venue	Weather	TV Show	*Date?*
e_1	Weekday	Dinner	Warm	Bad	No
e_2	Weekend	Club	Warm	Bad	Yes
e_3	Weekend	Club	Warm	Bad	Yes
e_4	Weekend	Club	Cold	Good	No

IF TV Show = Good **THEN** Date = No

vs.

IF Weather = Cold **THEN** Date = No

rules covering same instances are symmetric

no point in computing both!

Breaking symmetric rules

#	Day	Venue	Weather	TV Show	*Date?*
e_1	Weekday	Dinner	Warm	Bad	No
e_2	Weekend	Club	Warm	Bad	Yes
e_3	Weekend	Club	Warm	Bad	Yes
e_4	Weekend	Club	Cold	Good	No

IF TV Show $=$ Good **THEN** Date $=$ No

vs.

IF Weather $=$ Cold **THEN** Date $=$ No

rules covering same instances are symmetric

no point in computing both!

for each rule, add one clause enforcing

Breaking symmetric rules

#	Day	Venue	Weather	TV Show	*Date?*
e_1	Weekday	Dinner	Warm	Bad	No
e_2	Weekend	Club	Warm	Bad	Yes
e_3	Weekend	Club	Warm	Bad	Yes
e_4	Weekend	Club	Cold	Good	No

IF TV Show $=$ Good **THEN** Date $=$ No

vs.

IF Weather $=$ Cold **THEN** Date $=$ No

rules covering same instances are symmetric

no point in computing both!

for each rule, add one clause enforcing
all following rules to cover $\geq$ 1 other instance

Experimental results

Experimental setup

- **machine configuration:**
  - Intel Xeon Silver-4110 2.10GHz with 64GByte RAM

Experimental setup

- **machine configuration:**
  - Intel Xeon Silver-4110 2.10GHz with 64GByte RAM
  - running Debian Linux

Experimental setup

- **machine configuration:**
  - Intel Xeon Silver-4110 2.10GHz with 64GByte RAM
  - running Debian Linux
  - 1800s timeout + 8GB memout

Experimental setup

- **machine configuration:**
  - Intel Xeon Silver-4110 2.10GHz with 64GByte RAM
  - running Debian Linux
  - 1800s timeout + 8GB memout

- **UCI Machine Learning Repository + Penn Machine Learning Benchmarks**

Experimental setup

- **machine configuration:**
  - Intel Xeon Silver-4110 2.10GHz with 64GByte RAM
  - running Debian Linux
  - 1800s timeout + 8GB memout

- **UCI Machine Learning Repository + Penn Machine Learning Benchmarks**
  - **1065 benchmarks** in total (71 datasets × 5-cross validation × 3 quantized families)

Experimental setup

- **machine configuration:**
  - Intel Xeon Silver-4110 2.10GHz with 64GByte RAM
  - running Debian Linux
  - 1800s timeout + 8GB memout
- **UCI Machine Learning Repository** + **Penn Machine Learning Benchmarks**
  - **1065 benchmarks** in total (71 datasets $\times$ 5-cross validation $\times$ 3 quantized families)
  - **3–384 features** (one-hot encoded)

Experimental setup

- **machine configuration:**
  - Intel Xeon Silver-4110 2.10GHz with 64GByte RAM
  - running Debian Linux
  - 1800s timeout + 8GB memout

- **UCI Machine Learning Repository + Penn Machine Learning Benchmarks**
  - **1065 benchmarks** in total (71 datasets $\times$ 5-cross validation $\times$ 3 quantized families)
  - **3–384 features** (one-hot encoded)
  - **14–67557 training** instances

Experimental setup

- **competition tested:**
  - **mds_2** – minimization of number of rules

[1] `https://github.com/alexeyignatiev/minds`

Experimental setup

- **competition tested:**
  - **mds$_2$** – minimization of number of rules
  - **mds$_2^\star$** – *lexicographic* minimization of number of rules + literals

[1] https://github.com/alexeyignatiev/minds

Experimental setup

- **competition tested:**
  - **$\mathbf{mds_2}$** – minimization of number of rules
  - **$\mathbf{mds_2^\star}$** – *lexicographic* minimization of number of rules + literals
  - **opt** – minimization of number of literals

[1] `https://github.com/alexeyignatiev/minds`

Experimental setup

- **competition tested:**
  - **mds$_2$** – minimization of number of rules
  - **mds$_2^\star$** – *lexicographic* minimization of number of rules + literals
  - **opt** – minimization of number of literals

- **prototype**[1]
  - **ruler$_*^\circ$**
    - **same code base** and SAT solver – Glucose 3

[1] `https://github.com/alexeyignatiev/minds`

Experimental setup

- **competition tested:**
  - **mds$_2$** – minimization of number of rules
  - **mds$_2^\star$** – *lexicographic* minimization of number of rules + literals
  - **opt** – minimization of number of literals
- **prototype**[1]
  - **ruler$_*^\circ$**
    - **same code base** and SAT solver – Glucose 3
    - $\circ \in \{l, r\}$ – optimization criterion
    - **stage 1** – **incremental calls** to RC2 MaxSAT solver
    - **stage 2** – $* \in \{rc2, ilp\}$ – either *RC2 MaxSAT* or *Gurobi ILP*

[1] `https://github.com/alexeyignatiev/minds`

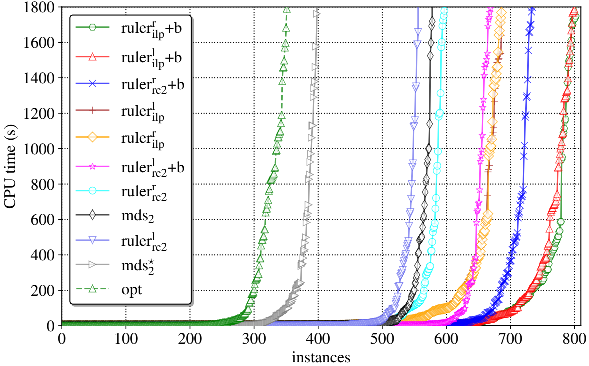
Experimental setup

- **competition tested:**
  - **mds$_2$** – minimization of number of rules
  - **mds$_2^\star$** – *lexicographic* minimization of number of rules + literals
  - **opt** – minimization of number of literals

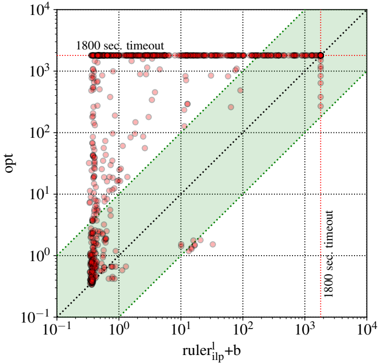
- **prototype**[1]
  - **ruler$_*^\circ$**
    - **same code base** and SAT solver – Glucose 3
    - $\circ \in \{l, r\}$ – optimization criterion
    - **stage 1** – **incremental calls** to RC2 MaxSAT solver
    - **stage 2** – $* \in \{rc2, ilp\}$ – either *RC2 MaxSAT* or *Gurobi ILP*
    - **ruler$_*^\circ$+b** – symmetry breaking *enabled*

[1] `https://github.com/alexeyignatiev/minds`

Results – performance comparison

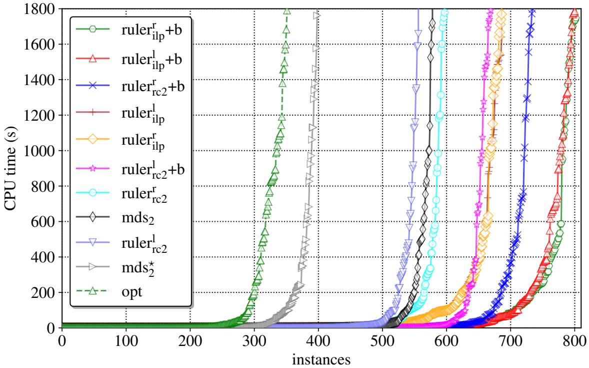


(a) raw performance

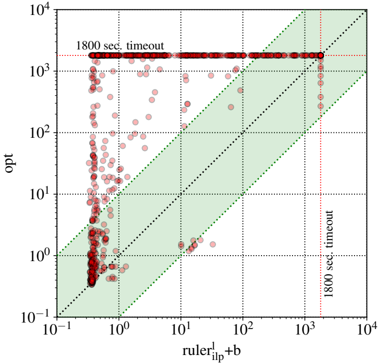


(b) ruler$^{l}_{ilp}$+b vs. *opt* detailed

Results – performance comparison



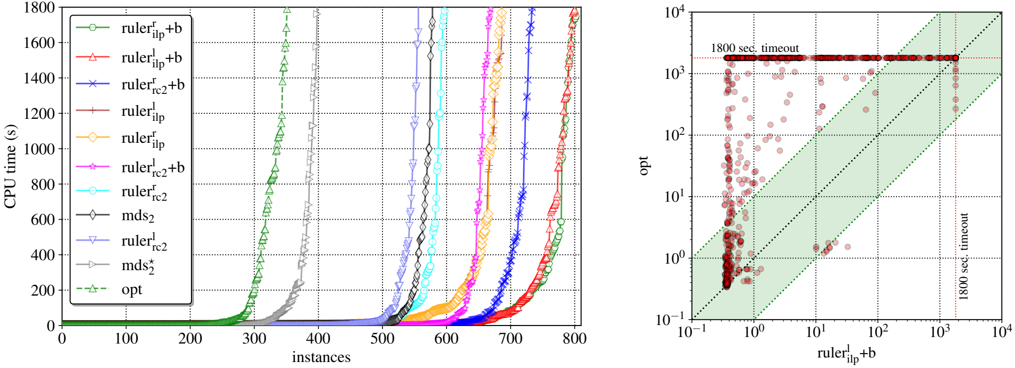
(a) raw performance



(b) $\text{ruler}^{\text{l}}_{\text{ilp}}$+b vs. *opt* detailed

$\text{ruler}^{\text{l}}_{\text{ilp}}$+b vs. **opt** — **up to 4 orders of magnitude** performance improvement

Results – performance comparison



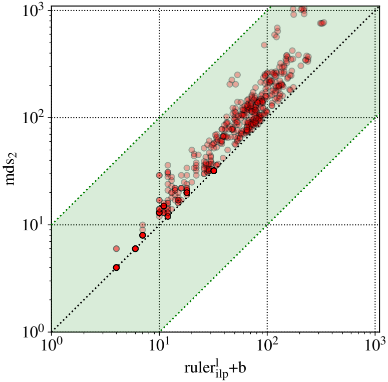
(a) raw performance

(b) ruler$^{l}_{ilp}$+b vs. *opt* detailed

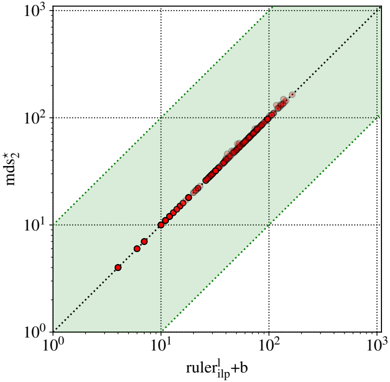
ruler$^{l}_{ilp}$+b vs. **opt** — **up to 4 orders of magnitude** performance improvement

breaking symmetric rules — avg. # of rules goes down **from 19604.4 to 563.7**

Results – model size comparison

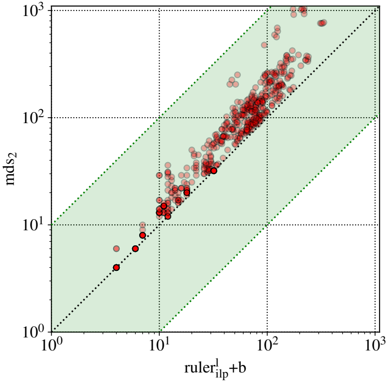


(a) literals or rules: $\mathrm{ruler}^{\mathrm{l}}_{\mathrm{ilp}}$+b vs. mds_2

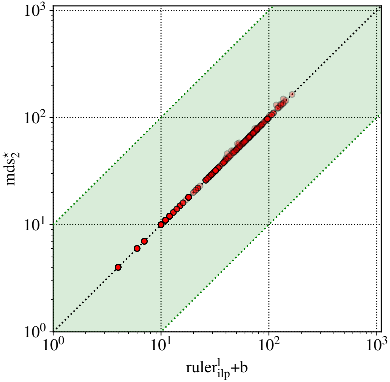


(b) literals or lexicographic: $\mathrm{ruler}^{\mathrm{l}}_{\mathrm{ilp}}$+b vs. $mds^{\star}_2$

Results – model size comparison



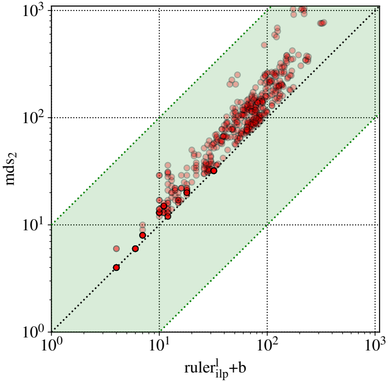
(a) literals or rules: $\mathrm{ruler}^{\mathrm{l}}_{\mathrm{ilp}}$+b vs. mds_2



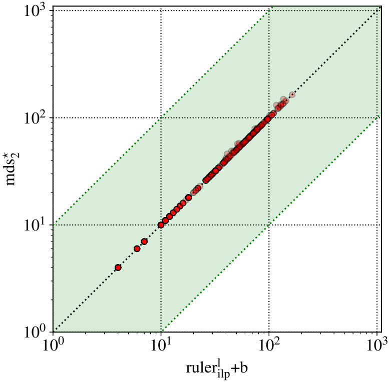
(b) literals or lexicographic: $\mathrm{ruler}^{\mathrm{l}}_{\mathrm{ilp}}$+b vs. $mds^{\star}_2$

$\mathbf{ruler}^{\mathbf{l}}_{\mathbf{ilp}}$**+b** vs. $\mathbf{mds_2}$ — halves avg. size (62.2 vs. 116.2)

Results – model size comparison



(a) literals or rules: ruler$^{\mathsf{l}}_{\mathsf{ilp}}$+b vs. *mds*$_2$



(b) literals or lexicographic: ruler$^{\mathsf{l}}_{\mathsf{ilp}}$+b vs. *mds*$_2^\star$

ruler$^{\mathsf{l}}_{\mathsf{ilp}}$+b vs. **mds$_2$** — halves avg. size (62.2 vs. 116.2)

mds$_2^\star$ vs. **mds$_2$** — *lexicographic optimization* pays off

(but slower!)

Summary and future work

- **novel approach to computing decision sets**
  - *(inspired by two-level logic minimization)*

Summary and future work

- **novel approach to computing decision sets**
  - *(inspired by two-level logic minimization)*
  - **consists of two stages:**
    1. enumeration of individual rules
    2. solving set cover problem

Summary and future work

- **novel approach to computing decision sets**
  - *(inspired by two-level logic minimization)*
  - **consists of two stages:**
    1. enumeration of individual rules
    2. solving set cover problem
  - **effective symmetry breaking**

Summary and future work

- **novel approach to computing decision sets**
  - *(inspired by two-level logic minimization)*
  - **consists of two stages:**
    1. enumeration of individual rules
    2. solving set cover problem
  - **effective symmetry breaking**
  - **smallest size** decision sets wrt.
    - number of rules
    - total number of literals

Summary and future work

- **novel approach to computing decision sets**
  - *(inspired by two-level logic minimization)*
  - **consists of two stages:**
    1. enumeration of individual rules
    2. solving set cover problem
  - **effective symmetry breaking**
  - **smallest size** decision sets wrt.
    - number of rules
    - total number of literals
  - **a few orders of magnitude** performance improvement

Summary and future work

- **novel approach to computing decision sets**
  - *(inspired by two-level logic minimization)*
  - **consists of two stages:**
    1. enumeration of individual rules
    2. solving set cover problem
  - **effective symmetry breaking**
  - **smallest size** decision sets wrt.
    - number of rules
    - total number of literals
  - **a few orders of magnitude** performance improvement

- **future work**
  - further improvements...

Summary and future work

- **novel approach to computing decision sets**
  - *(inspired by two-level logic minimization)*
  - **consists of two stages:**
    1. enumeration of individual rules
    2. solving set cover problem
  - **effective symmetry breaking**
  - **smallest size** decision sets wrt.
    - number of rules
    - total number of literals
  - **a few orders of magnitude** performance improvement

- **future work**
  - further improvements...
  - *sparse* decision sets

Summary and future work

- **novel approach to computing decision sets**
  - *(inspired by two-level logic minimization)*
  - **consists of two stages:**
    1. enumeration of individual rules
    2. solving set cover problem
  - **effective symmetry breaking**
  - **smallest size** decision sets wrt.
    - number of rules
    - total number of literals
  - **a few orders of magnitude** performance improvement

- **future work**
  - further improvements...
  - *sparse* decision sets
  - address rule overlap

Summary and future work

- **novel approach to computing decision sets**
    - *(inspired by two-level logic minimization)*
    - **consists of two stages:**
        1. enumeration of individual rules
        2. solving set cover problem
    - **effective symmetry breaking**
    - **smallest size** decision sets wrt.
        - number of rules
        - total number of literals
    - **a few orders of magnitude** performance improvement
- **future work**
    - further improvements...
    - *sparse* decision sets
    - address rule overlap
    - other rule-based models:
        - decision lists
        - decision trees

Questions?