A Scalable Two Stage Approach to Computing Optimal Decision Sets

Alexey Ignatiev\textsuperscript{1}, Edward Lam\textsuperscript{1,2}, Peter J. Stuckey\textsuperscript{1}, Joao Marques-Silva\textsuperscript{3}

February 4–7, 2021 | AAAI

\textsuperscript{1}Monash University, Melbourne, Australia
\textsuperscript{2}CSIRO Data61, Melbourne, Australia
\textsuperscript{3}ANITI, IRIT, CNRS, Toulouse, France
Problem and state of the art
Problem example

(classification scenario)

<table>
<thead>
<tr>
<th>Date</th>
<th>Weekday</th>
<th>Dinner</th>
<th>Weather</th>
<th>TV Show</th>
<th>IF</th>
<th>THEN</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>e</td>
<td>e</td>
<td>Warm</td>
<td>Bad</td>
<td>IF TV Show</td>
<td>THEN Date = No</td>
</tr>
<tr>
<td>2</td>
<td>e</td>
<td>e</td>
<td>Warm</td>
<td>Bad</td>
<td>IF Day = Weekday</td>
<td>THEN Date = No</td>
</tr>
<tr>
<td>3</td>
<td>e</td>
<td>e</td>
<td>Warm</td>
<td>Bad</td>
<td>IF TV Show = Bad ∧ Day = Weekend</td>
<td>THEN Date = Yes</td>
</tr>
</tbody>
</table>

unordered set of if-then rules must respect training data & generalize well...

highly interpretable!
**Problem example**

*(classification scenario)*

<table>
<thead>
<tr>
<th>#</th>
<th>Day</th>
<th>Venue</th>
<th>Weather</th>
<th>TV Show</th>
<th>Date?</th>
</tr>
</thead>
<tbody>
<tr>
<td>$e_1$</td>
<td>Weekday</td>
<td>Dinner</td>
<td>Warm</td>
<td>Bad</td>
<td>No</td>
</tr>
<tr>
<td>$e_2$</td>
<td>Weekend</td>
<td>Club</td>
<td>Warm</td>
<td>Bad</td>
<td>Yes</td>
</tr>
<tr>
<td>$e_3$</td>
<td>Weekend</td>
<td>Club</td>
<td>Warm</td>
<td>Bad</td>
<td>Yes</td>
</tr>
<tr>
<td>$e_4$</td>
<td>Weekend</td>
<td>Club</td>
<td>Cold</td>
<td>Good</td>
<td>No</td>
</tr>
</tbody>
</table>

The smaller — the better!

Highly interpretable!
Problem example

(classification scenario)

<table>
<thead>
<tr>
<th>#</th>
<th>Day</th>
<th>Venue</th>
<th>Weather</th>
<th>TV Show</th>
<th>Date?</th>
</tr>
</thead>
<tbody>
<tr>
<td>$e_1$</td>
<td>Weekday</td>
<td>Dinner</td>
<td>Warm</td>
<td>Bad</td>
<td>No</td>
</tr>
<tr>
<td>$e_2$</td>
<td>Weekend</td>
<td>Club</td>
<td>Warm</td>
<td>Bad</td>
<td>Yes</td>
</tr>
<tr>
<td>$e_3$</td>
<td>Weekend</td>
<td>Club</td>
<td>Warm</td>
<td>Bad</td>
<td>Yes</td>
</tr>
<tr>
<td>$e_4$</td>
<td>Weekend</td>
<td>Club</td>
<td>Cold</td>
<td>Good</td>
<td>No</td>
</tr>
</tbody>
</table>

- IF TV Show = Good THEN Date = No
- IF Day = Weekday THEN Date = No
- IF TV Show = Bad $\land$ Day = Weekend THEN Date = Yes
### Problem example

**Classification scenario**

<table>
<thead>
<tr>
<th>#</th>
<th>Day</th>
<th>Venue</th>
<th>Weather</th>
<th>TV Show</th>
<th>Date?</th>
</tr>
</thead>
<tbody>
<tr>
<td>$e_1$</td>
<td>Weekday</td>
<td>Dinner</td>
<td>Warm</td>
<td>Bad</td>
<td>No</td>
</tr>
<tr>
<td>$e_2$</td>
<td>Weekend</td>
<td>Club</td>
<td>Warm</td>
<td>Bad</td>
<td>Yes</td>
</tr>
<tr>
<td>$e_3$</td>
<td>Weekend</td>
<td>Club</td>
<td>Warm</td>
<td>Bad</td>
<td>Yes</td>
</tr>
<tr>
<td>$e_4$</td>
<td>Weekend</td>
<td>Club</td>
<td>Cold</td>
<td>Good</td>
<td>No</td>
</tr>
</tbody>
</table>

**If-then rules**

- IF TV Show = Good THEN Date = No
- IF Day = Weekday THEN Date = No
- IF TV Show = Bad ∧ Day = Weekend THEN Date = Yes

**unordered set of if-then rules**
Problem example

(classification scenario)

<table>
<thead>
<tr>
<th>#</th>
<th>Day</th>
<th>Venue</th>
<th>Weather</th>
<th>TV Show</th>
<th>Date?</th>
</tr>
</thead>
<tbody>
<tr>
<td>$e_1$</td>
<td>Weekday</td>
<td>Dinner</td>
<td>Warm</td>
<td>Bad</td>
<td>No</td>
</tr>
<tr>
<td>$e_2$</td>
<td>Weekend</td>
<td>Club</td>
<td>Warm</td>
<td>Bad</td>
<td>Yes</td>
</tr>
<tr>
<td>$e_3$</td>
<td>Weekend</td>
<td>Club</td>
<td>Warm</td>
<td>Bad</td>
<td>Yes</td>
</tr>
<tr>
<td>$e_4$</td>
<td>Weekend</td>
<td>Club</td>
<td>Cold</td>
<td>Good</td>
<td>No</td>
</tr>
</tbody>
</table>

IF TV Show = Good  THEN  Date = No
IF Day = Weekday   THEN  Date = No
IF TV Show = Bad ∧ Day = Weekend THEN  Date = Yes

unordered set of if-then rules
must respect training data & generalize well...
Problem example

(classification scenario)

<table>
<thead>
<tr>
<th>#</th>
<th>Day</th>
<th>Venue</th>
<th>Weather</th>
<th>TV Show</th>
<th>Date?</th>
</tr>
</thead>
<tbody>
<tr>
<td>$e_1$</td>
<td>Weekday</td>
<td>Dinner</td>
<td>Warm</td>
<td>Bad</td>
<td>No</td>
</tr>
<tr>
<td>$e_2$</td>
<td>Weekend</td>
<td>Club</td>
<td>Warm</td>
<td>Bad</td>
<td>Yes</td>
</tr>
<tr>
<td>$e_3$</td>
<td>Weekend</td>
<td>Club</td>
<td>Warm</td>
<td>Bad</td>
<td>Yes</td>
</tr>
<tr>
<td>$e_4$</td>
<td>Weekend</td>
<td>Club</td>
<td>Cold</td>
<td>Good</td>
<td>No</td>
</tr>
</tbody>
</table>

**unordered set of if-then rules**

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

must respect training data & generalize well...
the smaller — the better!
### Problem example

#### (classification scenario)

<table>
<thead>
<tr>
<th>#</th>
<th>Day</th>
<th>Venue</th>
<th>Weather</th>
<th>TV Show</th>
<th>Date?</th>
</tr>
</thead>
<tbody>
<tr>
<td>$e_1$</td>
<td>Weekday</td>
<td>Dinner</td>
<td>Warm</td>
<td>Bad</td>
<td>No</td>
</tr>
<tr>
<td>$e_2$</td>
<td>Weekend</td>
<td>Club</td>
<td>Warm</td>
<td>Bad</td>
<td>Yes</td>
</tr>
<tr>
<td>$e_3$</td>
<td>Weekend</td>
<td>Club</td>
<td>Warm</td>
<td>Bad</td>
<td>Yes</td>
</tr>
<tr>
<td>$e_4$</td>
<td>Weekend</td>
<td>Club</td>
<td>Cold</td>
<td>Good</td>
<td>No</td>
</tr>
</tbody>
</table>

- **If** TV Show = Good **Then** Date = No
- **If** Day = Weekday **Then** Date = No
- **If** TV Show = Bad $\land$ Day = Weekend **Then** Date = Yes

**unordered set of if-then rules**

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

the smaller — **the better!**

**highly interpretable!**
Motivation for decision sets

rule-based models
Motivation for decision sets

rule-based models

“transparent” and easy to interpret
Motivation for decision sets

rule-based models

“transparent” and easy to interpret

come in handy in XAI
input: training data \( E \)

output: smallest\(^a\) decision set \( \phi \)

1. \( N \leftarrow \text{LB} \)

2. **while** True:
   3. \( F \leftarrow \text{Encode}(E, N) \)
   4. \( (st, \mu) \leftarrow \text{Oracle}(F) \)
   5. **if** \( st \) **is** True:
       6. **break**
   7. \( N \leftarrow N + 1 \)

8. \( \phi \leftarrow \text{ExtractRules}(\mu) \)

9. **return** \( \phi \)

\(^a\)wrt. the number of rules or literals

# \( N \) equals a lower bound on \( |\phi| \), which is often set to 1

# encode problem “is there a decision set \( \phi \) of size \( N \) for data \( E \)?”

# call a reasoning oracle to answer the question

# extract decision set \( \phi \) from satisfying assignment \( \mu \)
State of the art — a typical approach

**input**: training data $E$

**output**: smallest$^a$ decision set $\phi$

1. $N \leftarrow \text{LB}$
   # $N$ equals a lower bound on $|\phi|$, which is often set to 1

2. **while** True:
   
   3. $F \leftarrow \text{Encode}(E, N)$
      # encode problem “is there a decision set $\phi$ of size $N$ for data $E$?”
   
   4. $(st, \mu) \leftarrow \text{Oracle}(F)$
      # call a reasoning oracle to answer the question

   5. **if** $st$ is True:
      
      6. **break**

   7. $N \leftarrow N + 1$

8. $\phi \leftarrow \text{ExtractRules}(\mu)$
   # extract decision set $\phi$ from satisfying assignment $\mu$

9. **return** $\phi$

---

$^a$ wrt. the number of rules or literals

**encoding is too large!**
(does not scale)
Our approach
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
Our take on the problem

divide the process into two stages:

1. enumerate individual rules
Our take on the problem

divide the process into two stages:

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

Each class is computed independently, the idea is to scale better.
Our take on the problem

divide the process into two stages:

1. enumerate individual rules
   - MaxSAT-based
   - incremental!
   - breaking symmetric rules
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
   - each class is computed independently
   - the idea is to scale better
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
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
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
Stage 1 — learning rules

each rule is a solution to MaxSAT formula

$$\psi \triangleq H \land S$$
Stage 1 — learning rules

each rule is a solution to MaxSAT formula

\[ \psi \triangleq H \land S \]

\( H \) — hard clauses
Stage 1 — learning rules

each rule is a **solution to MaxSAT formula**

\[ \psi \triangleq H \land S \]

---

**H — hard clauses**

1. coverage constraints:
   - rule *must cover* \( \geq 1 \) right instances
Stage 1 — learning rules

**each rule is a solution to MaxSAT formula**

$$\psi \triangleq H \land S$$

---

**H — hard clauses**

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

2. **discrimination constraints:**
   * rule must not cover any wrong instances
Stage 1 — learning rules

each rule is a solution to MaxSAT formula

\[ \psi \triangleq H \land S \]

\[ H \quad \text{— hard clauses} \]

1. coverage constraints:
   • rule must cover \( \geq 1 \) right instances

2. discrimination constraints:
   • rule must not cover any wrong instances

\[ S \quad \text{— soft clauses} \]

• minimize the number of used literals
Stage 1 — learning rules

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
  2. discrimination constraints:
     - rule must not cover any wrong instances

- **S** — soft clauses
  - minimize the number of used literals

\( \Theta(K + M) \) variables and \( \Theta(K \times M) \) clauses

\( K \) — number of features, \( M \) — number of training instances
Stage 2 — computing rule cover

<table>
<thead>
<tr>
<th>#</th>
<th>Day</th>
<th>Venue</th>
<th>Weather</th>
<th>TV Show</th>
<th>Date?</th>
</tr>
</thead>
<tbody>
<tr>
<td>e₁</td>
<td>Weekday</td>
<td>Dinner</td>
<td>Warm</td>
<td>Bad</td>
<td>No</td>
</tr>
<tr>
<td>e₂</td>
<td>Weekend</td>
<td>Club</td>
<td>Warm</td>
<td>Bad</td>
<td>Yes</td>
</tr>
<tr>
<td>e₃</td>
<td>Weekend</td>
<td>Club</td>
<td>Warm</td>
<td>Bad</td>
<td>Yes</td>
</tr>
<tr>
<td>e₄</td>
<td>Weekend</td>
<td>Club</td>
<td>Cold</td>
<td>Good</td>
<td>No</td>
</tr>
</tbody>
</table>
### Stage 2 — computing rule cover

<table>
<thead>
<tr>
<th>#</th>
<th>Day</th>
<th>Venue</th>
<th>Weather</th>
<th>TV Show</th>
<th>Date?</th>
</tr>
</thead>
<tbody>
<tr>
<td>$e_1$</td>
<td>Weekday</td>
<td>Dinner</td>
<td>Warm</td>
<td>Bad</td>
<td>No</td>
</tr>
<tr>
<td>$e_2$</td>
<td>Weekend</td>
<td>Club</td>
<td>Warm</td>
<td>Bad</td>
<td>Yes</td>
</tr>
<tr>
<td>$e_3$</td>
<td>Weekend</td>
<td>Club</td>
<td>Warm</td>
<td>Bad</td>
<td>Yes</td>
</tr>
<tr>
<td>$e_4$</td>
<td>Weekend</td>
<td>Club</td>
<td>Cold</td>
<td>Good</td>
<td>No</td>
</tr>
</tbody>
</table>

\[ \pi_1 = \left[ \text{IF Day = Weekday THEN Date = No} \right] \]
## Stage 2 — computing rule cover

<table>
<thead>
<tr>
<th>#</th>
<th>Day</th>
<th>Venue</th>
<th>Weather</th>
<th>TV Show</th>
<th>Date?</th>
</tr>
</thead>
<tbody>
<tr>
<td>e₁</td>
<td>Weekday</td>
<td>Dinner</td>
<td>Warm</td>
<td>Bad</td>
<td>No</td>
</tr>
<tr>
<td>e₂</td>
<td>Weekend</td>
<td>Club</td>
<td>Warm</td>
<td>Bad</td>
<td>Yes</td>
</tr>
<tr>
<td>e₃</td>
<td>Weekend</td>
<td>Club</td>
<td>Warm</td>
<td>Bad</td>
<td>Yes</td>
</tr>
<tr>
<td>e₄</td>
<td>Weekend</td>
<td>Club</td>
<td>Cold</td>
<td>Good</td>
<td>No</td>
</tr>
</tbody>
</table>

\[ π₁ = \begin{cases} \text{IF Day = Weekday} & \text{THEN Date = No} \\ \end{cases} \]

\[ π₂ = \begin{cases} \text{IF Venue = Dinner} & \text{THEN Date = No} \\ \end{cases} \]
### Stage 2 — computing rule cover

<table>
<thead>
<tr>
<th>#</th>
<th>Day</th>
<th>Venue</th>
<th>Weather</th>
<th>TV Show</th>
<th>Date?</th>
</tr>
</thead>
<tbody>
<tr>
<td>e₁</td>
<td>Weekday</td>
<td>Dinner</td>
<td>Warm</td>
<td>Bad</td>
<td>No</td>
</tr>
<tr>
<td>e₂</td>
<td>Weekend</td>
<td>Club</td>
<td>Warm</td>
<td>Bad</td>
<td>Yes</td>
</tr>
<tr>
<td>e₃</td>
<td>Weekend</td>
<td>Club</td>
<td>Warm</td>
<td>Bad</td>
<td>Yes</td>
</tr>
<tr>
<td>e₄</td>
<td>Weekend</td>
<td>Club</td>
<td>Cold</td>
<td>Good</td>
<td>No</td>
</tr>
</tbody>
</table>

\[
\pi_1 = \left[ \text{IF Day} = \text{Weekday} \ \text{THEN Date} = \text{No} \right]
\]

\[
\pi_2 = \left[ \text{IF Venue} = \text{Dinner} \ \text{THEN Date} = \text{No} \right]
\]

\[
\pi_3 = \left[ \text{IF Weather} = \text{Cold} \ \text{THEN Date} = \text{No} \right]
\]

\[
b_j \in \{0, 1\} \quad \text{and} \quad s_j = |\pi_j| \quad \text{for each} \quad \pi_j
\]

\[
A = (a_{ij}) \quad a_{ij} = 1 \iff \pi_j \text{ covers } e_i
\]

\[
\minimize \sum \sum \sum s_j \cdot b_j
\]

\[
\sum \sum \sum a_{ij} \cdot b_j \geq 1, \quad \forall i
\]
Stage 2 — computing rule cover

<table>
<thead>
<tr>
<th>#</th>
<th>Day</th>
<th>Venue</th>
<th>Weather</th>
<th>TV Show</th>
<th>Date?</th>
</tr>
</thead>
<tbody>
<tr>
<td>$e_1$</td>
<td>Weekday</td>
<td>Dinner</td>
<td>Warm</td>
<td>Bad</td>
<td>No</td>
</tr>
<tr>
<td>$e_2$</td>
<td>Weekend</td>
<td>Club</td>
<td>Warm</td>
<td>Bad</td>
<td>Yes</td>
</tr>
<tr>
<td>$e_3$</td>
<td>Weekend</td>
<td>Club</td>
<td>Warm</td>
<td>Bad</td>
<td>Yes</td>
</tr>
<tr>
<td>$e_4$</td>
<td>Weekend</td>
<td>Club</td>
<td>Cold</td>
<td>Good</td>
<td>No</td>
</tr>
</tbody>
</table>

$\pi_1 = \left[ \text{IF Day} = \text{Weekday} \quad \text{THEN Date} = \text{No} \right]$  
$\pi_2 = \left[ \text{IF Venue} = \text{Dinner} \quad \text{THEN Date} = \text{No} \right]$  
$\pi_3 = \left[ \text{IF Weather} = \text{Cold} \quad \text{THEN Date} = \text{No} \right]$  
$\pi_4 = \left[ \text{IF TV Show} = \text{Good} \quad \text{THEN Date} = \text{No} \right]$
### Stage 2 — computing rule cover

<table>
<thead>
<tr>
<th>#</th>
<th>Day</th>
<th>Venue</th>
<th>Weather</th>
<th>TV Show</th>
<th>Date?</th>
</tr>
</thead>
<tbody>
<tr>
<td>$e_1$</td>
<td>Weekday</td>
<td>Dinner</td>
<td>Warm</td>
<td>Bad</td>
<td>No</td>
</tr>
<tr>
<td>$e_2$</td>
<td>Weekend</td>
<td>Club</td>
<td>Warm</td>
<td>Bad</td>
<td>Yes</td>
</tr>
<tr>
<td>$e_3$</td>
<td>Weekend</td>
<td>Club</td>
<td>Warm</td>
<td>Bad</td>
<td>Yes</td>
</tr>
<tr>
<td>$e_4$</td>
<td>Weekend</td>
<td>Club</td>
<td>Cold</td>
<td>Good</td>
<td>No</td>
</tr>
</tbody>
</table>

\[
\pi_1 = \begin{cases} 
\text{IF Day = Weekday} & \text{THEN Date = No} \\
\end{cases}
\]

\[
\pi_2 = \begin{cases} 
\text{IF Venue = Dinner} & \text{THEN Date = No} \\
\end{cases}
\]

\[
\pi_3 = \begin{cases} 
\text{IF Weather = Cold} & \text{THEN Date = No} \\
\end{cases}
\]

\[
\pi_4 = \begin{cases} 
\text{IF TV Show = Good} & \text{THEN Date = No} \\
\end{cases}
\]

\[b_j \in \{0, 1\} \text{ and } s_j = |\pi_j| \text{ for each } \pi_j\]

\[
\begin{align*}
\sum_{j} b_j & \geq 1, \\
\sum_{j} a_{ij} & \cdot b_j & \geq 1, & \forall i
\end{align*}
\]
Stage 2 — computing rule cover

<table>
<thead>
<tr>
<th>#</th>
<th>Day</th>
<th>Venue</th>
<th>Weather</th>
<th>TV Show</th>
<th>Date?</th>
</tr>
</thead>
<tbody>
<tr>
<td>e₁</td>
<td>Weekday</td>
<td>Dinner</td>
<td>Warm</td>
<td>Bad</td>
<td>No</td>
</tr>
<tr>
<td>e₂</td>
<td>Weekend</td>
<td>Club</td>
<td>Warm</td>
<td>Bad</td>
<td>Yes</td>
</tr>
<tr>
<td>e₃</td>
<td>Weekend</td>
<td>Club</td>
<td>Warm</td>
<td>Bad</td>
<td>Yes</td>
</tr>
<tr>
<td>e₄</td>
<td>Weekend</td>
<td>Club</td>
<td>Cold</td>
<td>Good</td>
<td>No</td>
</tr>
</tbody>
</table>

\[ \begin{align*}
\pi_1 &= [ \text{IF Day = Weekday THEN Date = No} ] \\
\pi_2 &= [ \text{IF Venue = Dinner THEN Date = No} ] \\
\pi_3 &= [ \text{IF Weather = Cold THEN Date = No} ] \\
\pi_4 &= [ \text{IF TV Show = Good THEN Date = No} ]
\end{align*} \]

\[ b_j \in \{0, 1\} \text{ and } s_j = |\pi_j| \text{ for each } \pi_j \]

\[ A = (a_{ij}), a_{ij} = 1 \text{ iff } \pi_j \text{ covers } e_i \]
### Stage 2 — computing rule cover

<table>
<thead>
<tr>
<th>#</th>
<th>Day</th>
<th>Venue</th>
<th>Weather</th>
<th>TV Show</th>
<th>Date?</th>
</tr>
</thead>
<tbody>
<tr>
<td>e₁</td>
<td>Weekday</td>
<td>Dinner</td>
<td>Warm</td>
<td>Bad</td>
<td>No</td>
</tr>
<tr>
<td>e₂</td>
<td>Weekend</td>
<td>Club</td>
<td>Warm</td>
<td>Bad</td>
<td>Yes</td>
</tr>
<tr>
<td>e₃</td>
<td>Weekend</td>
<td>Club</td>
<td>Warm</td>
<td>Bad</td>
<td>Yes</td>
</tr>
<tr>
<td>e₄</td>
<td>Weekend</td>
<td>Club</td>
<td>Cold</td>
<td>Good</td>
<td>No</td>
</tr>
</tbody>
</table>

\[
\begin{align*}
\pi_1 &= \left[ \text{IF } \text{Day} = \text{Weekday} \quad \text{THEN} \quad \text{Date} = \text{No} \right] \\
\pi_2 &= \left[ \text{IF } \text{Venue} = \text{Dinner} \quad \text{THEN} \quad \text{Date} = \text{No} \right] \\
\pi_3 &= \left[ \text{IF } \text{Weather} = \text{Cold} \quad \text{THEN} \quad \text{Date} = \text{No} \right] \\
\pi_4 &= \left[ \text{IF } \text{TV Show} = \text{Good} \quad \text{THEN} \quad \text{Date} = \text{No} \right]
\end{align*}
\]

\[b_j \in \{0, 1\} \quad \text{and} \quad s_j = |\pi_j| \quad \text{for each} \quad \pi_j\]

\[A = (a_{ij}), \quad a_{ij} = 1 \text{ iff } \pi_j \text{ covers } e_i\]

<table>
<thead>
<tr>
<th></th>
<th>(\pi_1)</th>
<th>(\pi_2)</th>
<th>(\pi_3)</th>
<th>(\pi_4)</th>
</tr>
</thead>
<tbody>
<tr>
<td>(a_{ij})</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>(b_j)</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

| \(s_j\) | 1 | 1 | 1 | 1 |
Stage 2 — computing rule cover

<table>
<thead>
<tr>
<th>#</th>
<th>Day</th>
<th>Venue</th>
<th>Weather</th>
<th>TV Show</th>
<th>Date?</th>
</tr>
</thead>
<tbody>
<tr>
<td>$e_1$</td>
<td>Weekday</td>
<td>Dinner</td>
<td>Warm</td>
<td>Bad</td>
<td>No</td>
</tr>
<tr>
<td>$e_2$</td>
<td>Weekend</td>
<td>Club</td>
<td>Warm</td>
<td>Bad</td>
<td>Yes</td>
</tr>
<tr>
<td>$e_3$</td>
<td>Weekend</td>
<td>Club</td>
<td>Warm</td>
<td>Bad</td>
<td>Yes</td>
</tr>
<tr>
<td>$e_4$</td>
<td>Weekend</td>
<td>Club</td>
<td>Cold</td>
<td>Good</td>
<td>No</td>
</tr>
</tbody>
</table>

$\pi_1 = \left[ \text{IF Day = Weekday } \Rightarrow \text{ Date = No } \right]$  
$\pi_2 = \left[ \text{IF Venue = Dinner } \Rightarrow \text{ Date = No } \right]$  
$\pi_3 = \left[ \text{IF Weather = Cold } \Rightarrow \text{ Date = No } \right]$  
$\pi_4 = \left[ \text{IF TV Show = Good } \Rightarrow \text{ Date = No } \right]$  

$\mathbf{b}_j \in \{0, 1\}$ and $s_j = |\pi_j|$ for each $\pi_j$

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

\[
\begin{array}{cccc} 
\pi_1 & \pi_2 & \pi_3 & \pi_4 \\
\hline 
a_{ij} & 1 & 1 & 0 & 0 \\
0 & 0 & 1 & 1 \\
\hline 
s_j & 1 & 1 & 1 & 1 \\
\end{array}
\]

minimize $\sum_j s_j \cdot b_j$

subject to $\sum_j a_{ij} \cdot b_j \geq 1, \forall i$
## Breaking symmetric rules

<table>
<thead>
<tr>
<th>#</th>
<th>Day</th>
<th>Venue</th>
<th>Weather</th>
<th>TV Show</th>
<th>Date?</th>
</tr>
</thead>
<tbody>
<tr>
<td>$e_1$</td>
<td>Weekday</td>
<td>Dinner</td>
<td>Warm</td>
<td>Bad</td>
<td>No</td>
</tr>
<tr>
<td>$e_2$</td>
<td>Weekend</td>
<td>Club</td>
<td>Warm</td>
<td>Bad</td>
<td>Yes</td>
</tr>
<tr>
<td>$e_3$</td>
<td>Weekend</td>
<td>Club</td>
<td>Warm</td>
<td>Bad</td>
<td>Yes</td>
</tr>
<tr>
<td>$e_4$</td>
<td>Weekend</td>
<td>Club</td>
<td>Cold</td>
<td>Good</td>
<td>No</td>
</tr>
</tbody>
</table>

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
Breaking symmetric rules

<table>
<thead>
<tr>
<th>#</th>
<th>Day</th>
<th>Venue</th>
<th>Weather</th>
<th>TV Show</th>
<th>Date?</th>
</tr>
</thead>
<tbody>
<tr>
<td>$e_1$</td>
<td>Weekday</td>
<td>Dinner</td>
<td>Warm</td>
<td>Bad</td>
<td>No</td>
</tr>
<tr>
<td>$e_2$</td>
<td>Weekend</td>
<td>Club</td>
<td>Warm</td>
<td>Bad</td>
<td>Yes</td>
</tr>
<tr>
<td>$e_3$</td>
<td>Weekend</td>
<td>Club</td>
<td>Warm</td>
<td>Bad</td>
<td>Yes</td>
</tr>
<tr>
<td>$e_4$</td>
<td>Weekend</td>
<td>Club</td>
<td>Cold</td>
<td>Good</td>
<td>No</td>
</tr>
</tbody>
</table>
## Breaking symmetric rules

<table>
<thead>
<tr>
<th>#</th>
<th>Day</th>
<th>Venue</th>
<th>Weather</th>
<th>TV Show</th>
<th>Date?</th>
</tr>
</thead>
<tbody>
<tr>
<td>$e_1$</td>
<td>Weekday</td>
<td>Dinner</td>
<td>Warm</td>
<td>Bad</td>
<td>No</td>
</tr>
<tr>
<td>$e_2$</td>
<td>Weekend</td>
<td>Club</td>
<td>Warm</td>
<td>Bad</td>
<td>Yes</td>
</tr>
<tr>
<td>$e_3$</td>
<td>Weekend</td>
<td>Club</td>
<td>Warm</td>
<td>Bad</td>
<td>Yes</td>
</tr>
<tr>
<td>$e_4$</td>
<td>Weekend</td>
<td>Club</td>
<td>Cold</td>
<td>Good</td>
<td>No</td>
</tr>
</tbody>
</table>

**Rule:**

- **IF** TV Show $=$ Good $\quad$ **THEN** Date $=$ No

**Note:** 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.
Breaking symmetric rules

<table>
<thead>
<tr>
<th>#</th>
<th>Day</th>
<th>Venue</th>
<th>Weather</th>
<th>TV Show</th>
<th>Date?</th>
</tr>
</thead>
<tbody>
<tr>
<td>$e_1$</td>
<td>Weekday</td>
<td>Dinner</td>
<td>Warm</td>
<td>Bad</td>
<td>No</td>
</tr>
<tr>
<td>$e_2$</td>
<td>Weekend</td>
<td>Club</td>
<td>Warm</td>
<td>Bad</td>
<td>Yes</td>
</tr>
<tr>
<td>$e_3$</td>
<td>Weekend</td>
<td>Club</td>
<td>Warm</td>
<td>Bad</td>
<td>Yes</td>
</tr>
<tr>
<td>$e_4$</td>
<td>Weekend</td>
<td>Club</td>
<td>Cold</td>
<td>Good</td>
<td>No</td>
</tr>
</tbody>
</table>

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

vs.

IF Weather $=$ Cold THEN Date $=$ No
### Breaking symmetric rules

<table>
<thead>
<tr>
<th>#</th>
<th>Day</th>
<th>Venue</th>
<th>Weather</th>
<th>TV Show</th>
<th>Date?</th>
</tr>
</thead>
<tbody>
<tr>
<td>$e_1$</td>
<td>Weekday</td>
<td>Dinner</td>
<td>Warm</td>
<td>Bad</td>
<td>No</td>
</tr>
<tr>
<td>$e_2$</td>
<td>Weekend</td>
<td>Club</td>
<td>Warm</td>
<td>Bad</td>
<td>Yes</td>
</tr>
<tr>
<td>$e_3$</td>
<td>Weekend</td>
<td>Club</td>
<td>Warm</td>
<td>Bad</td>
<td>Yes</td>
</tr>
<tr>
<td>$e_4$</td>
<td>Weekend</td>
<td>Club</td>
<td>Cold</td>
<td>Good</td>
<td>No</td>
</tr>
</tbody>
</table>

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

**vs.**  

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

rules covering same instances are **symmetric**
### Breaking symmetric rules

<table>
<thead>
<tr>
<th>#</th>
<th>Day</th>
<th>Venue</th>
<th>Weather</th>
<th>TV Show</th>
<th>Date?</th>
</tr>
</thead>
<tbody>
<tr>
<td>$e_1$</td>
<td>Weekday</td>
<td>Dinner</td>
<td>Warm</td>
<td>Bad</td>
<td>No</td>
</tr>
<tr>
<td>$e_2$</td>
<td>Weekend</td>
<td>Club</td>
<td>Warm</td>
<td>Bad</td>
<td>Yes</td>
</tr>
<tr>
<td>$e_3$</td>
<td>Weekend</td>
<td>Club</td>
<td>Warm</td>
<td>Bad</td>
<td>Yes</td>
</tr>
<tr>
<td>$e_4$</td>
<td>Weekend</td>
<td>Club</td>
<td>Cold</td>
<td>Good</td>
<td>No</td>
</tr>
</tbody>
</table>

- **IF** TV Show = Good **THEN** Date = No
- **IF** Weather = Cold **THEN** Date = No

**rules covering same instances are symmetric**

no point in computing both!
Breaking symmetric rules

<table>
<thead>
<tr>
<th>#</th>
<th>Day</th>
<th>Venue</th>
<th>Weather</th>
<th>TV Show</th>
<th>Date?</th>
</tr>
</thead>
<tbody>
<tr>
<td>$e_1$</td>
<td>Weekday</td>
<td>Dinner</td>
<td>Warm</td>
<td>Bad</td>
<td>No</td>
</tr>
<tr>
<td>$e_2$</td>
<td>Weekend</td>
<td>Club</td>
<td>Warm</td>
<td>Bad</td>
<td>Yes</td>
</tr>
<tr>
<td>$e_3$</td>
<td>Weekend</td>
<td>Club</td>
<td>Warm</td>
<td>Bad</td>
<td>Yes</td>
</tr>
<tr>
<td>$e_4$</td>
<td>Weekend</td>
<td>Club</td>
<td>Cold</td>
<td>Good</td>
<td>No</td>
</tr>
</tbody>
</table>

IF TV Show $= \text{Good}$ THEN Date $= \text{No}$ vs. IF Weather $= \text{Cold}$ THEN Date $= \text{No}$

rules covering same instances are symmetric
no point in computing both!

for each rule, add one clause enforcing
Breaking symmetric rules

<table>
<thead>
<tr>
<th>#</th>
<th>Day</th>
<th>Venue</th>
<th>Weather</th>
<th>TV Show</th>
<th>Date?</th>
</tr>
</thead>
<tbody>
<tr>
<td>$e_1$</td>
<td>Weekday</td>
<td>Dinner</td>
<td>Warm</td>
<td>Bad</td>
<td>No</td>
</tr>
<tr>
<td>$e_2$</td>
<td>Weekend</td>
<td>Club</td>
<td>Warm</td>
<td>Bad</td>
<td>Yes</td>
</tr>
<tr>
<td>$e_3$</td>
<td>Weekend</td>
<td>Club</td>
<td>Warm</td>
<td>Bad</td>
<td>Yes</td>
</tr>
<tr>
<td>$e_4$</td>
<td>Weekend</td>
<td>Club</td>
<td>Cold</td>
<td>Good</td>
<td>No</td>
</tr>
</tbody>
</table>

**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 64GB 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 × 5-cross validation × 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 × 5-cross validation × 3 quantized families)
  • 3–384 features (one-hot encoded)
  • 14–67557 training instances
Experimental setup

- **competition tested:**
  - $\text{mds}_2$ – minimization of number of rules

\[^1\text{https://github.com/alexeyignatiev/minds}\]
Experimental setup

• competition tested:
  • $\text{mds}_2$ – minimization of number of rules
  • $\text{mds}_2^*$ – lexicographic minimization of number of rules + literals

\[1\text{https://github.com/alexeyignatiev/minds} \]
Experimental setup

• **competition tested:**
  - $\text{mds}_2$ – minimization of number of rules
  - $\text{mds}_2^*$ – lexicographic minimization of number of rules + literals
  - $\text{opt}$ – minimization of number of literals

1https://github.com/alexeyignatiev/minds
Experimental setup

- **competition tested:**
  - \(\text{mds}_2\) – minimization of number of rules
  - \(\text{mds}_2^\star\) – lexicographic minimization of number of rules + literals
  - \(\text{opt}\) – minimization of number of literals

- **prototype\(^1\)**
  - \(\text{ruler}^\circ\)
    - same code base and SAT solver – Glucose 3

\(^1\)https://github.com/alexeyignatiev/minds
Experimental setup

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

- **prototype\(^1\)**
  - $\text{ruler}^o$
    - same code base and SAT solver – Glucose 3
  - $\circ \in \{l, r\}$ – optimization criterion
  - stage 1 – **incremental calls** to RC2 MaxSAT solver
  - stage 2 – $\ast \in \{rc2, ilp\}$ – either RC2 MaxSAT or Gurobi ILP

---
\(^1\)https://github.com/alexeyignatiev/minds
Experimental setup

• **competition tested:**
  • $\text{mds}_2$ – minimization of number of rules
  • $\text{mds}_2^\star$ – *lexicographic* minimization of number of rules + literals
  • $\text{opt}$ – minimization of number of literals

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

¹https://github.com/alexeyignatiev/minds
Figures 4a and 4b show the results of performance comparison.

**Figure 4a: Raw Performance**
- The graph plots CPU time (s) against the number of instances for various algorithms:
  - ruler\(_{\text{lp}}^1 + b\)
  - ruler\(_{\text{lp}}^1\)
  - ruler\(_{\text{fc2}}^1 + b\)
  - ruler\(_{\text{fc2}}^1\)
  - mds\(_2\)
  - mds\(_2^*\)
  - opt

**Figure 4b: ruler\(_{\text{lp}}^1 + b\) vs. opt Detailed**
- The scatter plot illustrates the comparison between ruler\(_{\text{lp}}^1 + b\) and opt for up to 4 orders of magnitude performance improvement.
- It indicates that breaking symmetric rules significantly reduces the average number of rules from 19604.4 to 563.7.

The graph on the right shows a clear distinction between the performance of ruler\(_{\text{lp}}^1 + b\) and opt, with a notable improvement in CPU time for a given number of instances.
Results – performance comparison

(a) raw performance

(b) ruler\textsubscript{lp} + b vs. opt detailed

\textbf{ruler\textsubscript{lp} + b vs. opt} — \textbf{up to 4 orders of magnitude} performance improvement
Results – performance comparison

(a) raw performance

(b) ruler\(_{\text{ip}}\) + b vs. opt detailed

ruler\(_{\text{ip}}\) + 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: $\text{ruler}_{\text{ilp}} + b$ vs. $mds_2$

(b) literals or lexicographic: $\text{ruler}_{\text{ilp}} + b$ vs. $mds_2^*$
Results – model size comparison

(a) literals or rules: \(\text{ruler}_{\text{ilp}+b} \) vs. \( \text{mds}_2 \)

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

\( \text{ruler}_{\text{ilp}+b} \) vs. \( \text{mds}_2 \) — halves avg. size (62.2 vs. 116.2)
Results – model size comparison

(a) literals or rules: \( \text{ruler}_{\text{ilp}+b} \) vs. \( \text{mds}_2 \)

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

\( \text{ruler}_{\text{ilp}+b} \) vs. \( \text{mds}_2 \) — halves avg. size (62.2 vs. 116.2)

\( \text{mds}^*_2 \) vs. \( \text{mds}_2 \) — lexicographic optimization pays off (but slower!)
Summary and future work

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

• effective symmetry breaking
• smallest size decision sets wrt.
• number of rules
• total number of literals

• future work
• further improvements...
• sparse decision sets
• address rule overlap
• other rule-based models:
  • decision lists
  • decision trees
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
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

- future work
  - further improvements...
  - sparse decision sets
  - address rule overlap
  - other rule-based models:
    - decision lists
    - decision trees
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
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?