



# Exploiting Symmetries in MUS Computation

Ignace Bleukx<sup>1</sup>

Hélène Verhaeghe<sup>1,2</sup>

Bart Bogaerts<sup>1,3</sup>

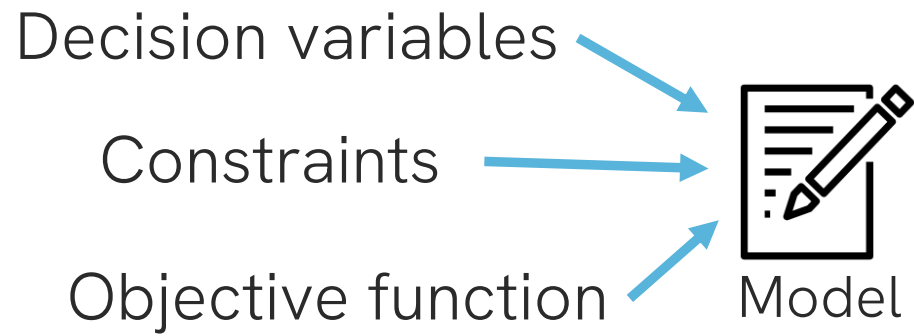
Tias Guns<sup>1</sup>

<sup>1</sup>KU Leuven, Belgium

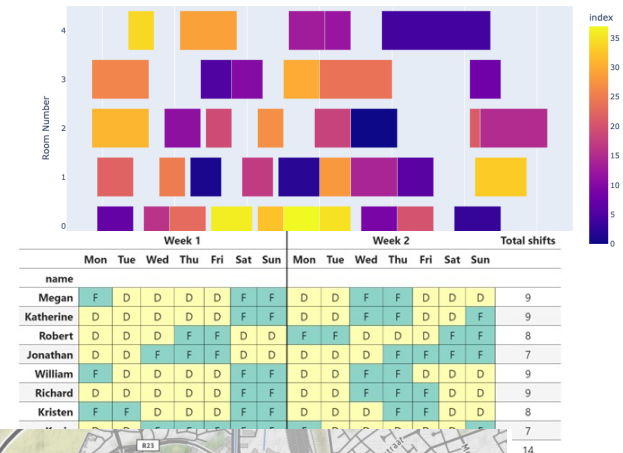
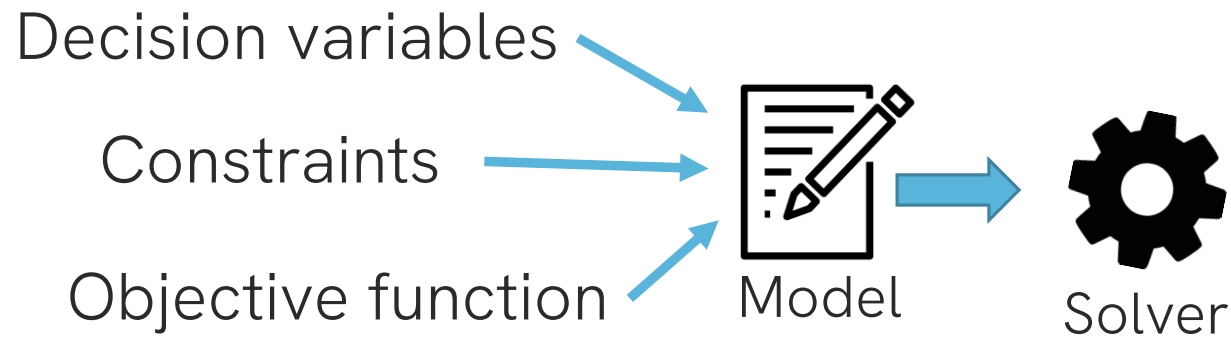
<sup>2</sup>UCLouvain, Belgium

<sup>3</sup>Vrije Universiteit Brussel, Belgium

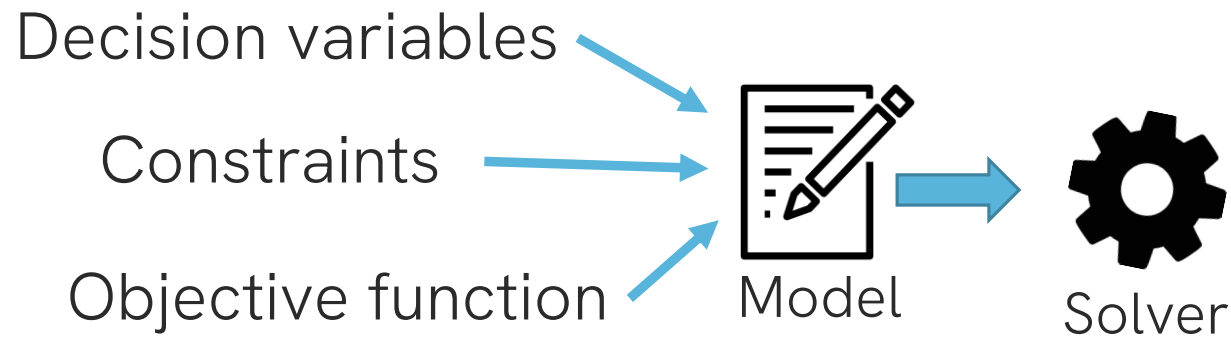
# Constraint Solving



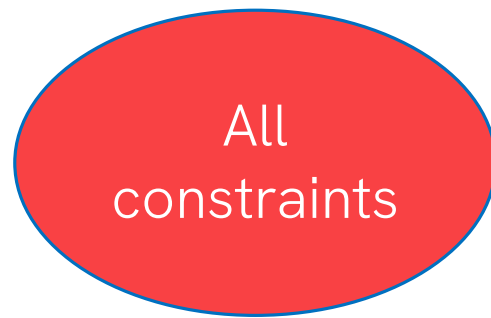
# Constraint Solving



# Explainable Constraint Solving

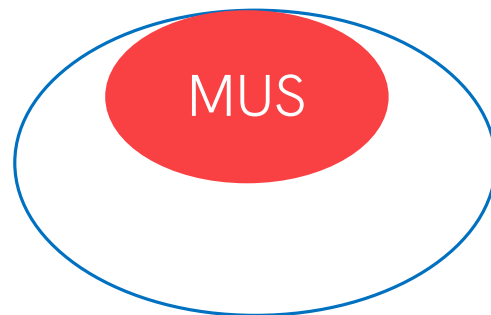


# Minimal Unsatisfiable Subsets (MUSes)



Subset of constraints causing a conflict

Reduced cognitive effort for user



Several MUSes for 1 problem may exist

# Minimal Unsatisfiable Subsets (MUSes)

Week 1								Week 2							Total shifts
	Mon	Tue	Wed	Thu	Fri	Sat	Sun	Mon	Tue	Wed	Thu	Fri	Sat	Sun	
name															
Megan															14
Katherine															14
Robert															14
Jonathan															14
William															14
Richard															14
Kristen															14
Kevin															14
Cover D	0/5	0/7	0/6	0/4	0/5	0/5	0/5	0/6	0/7	0/4	0/2	0/5	0/6	0/4	14

Robert has a day off on Tuesday

Richard has a day off on Tuesday

On Tuesday, 7 out of 8 nurses should work

# How to compute a MUS?

Deletion-based MUS-computation  
Simplest algorithm

Implicit-hitting-set algorithms  
Finding small/optimal MUSes

Seed-and shrink methods  
Enumerating MUSes

# Deletion-based MUS-computation

---

**Algorithm 1: SHRINK( $\phi$ )**

---

```
1:  $U \leftarrow \phi$ ;  
2: while there are unmarked constraints in  $U$  do  
3:    $c \leftarrow$  next unmarked constraint in  $U$   
4:    $(\text{is\_sat}, \alpha, U') \leftarrow \text{SAT}(U \setminus \{c\})$   
5:   if  $\text{is\_sat}$  then  
6:     mark  $c$  as required  
12:  else  
13:     $U \leftarrow U'$   
14: return  $U$ 
```

---

Try removing a constraint

If the remainder is SAT, re-add

If the remainder is UNSAT,  
remove permanently



# Computing MUSes can be slow

---

## Algorithm 1: SHRINK( $\phi$ )

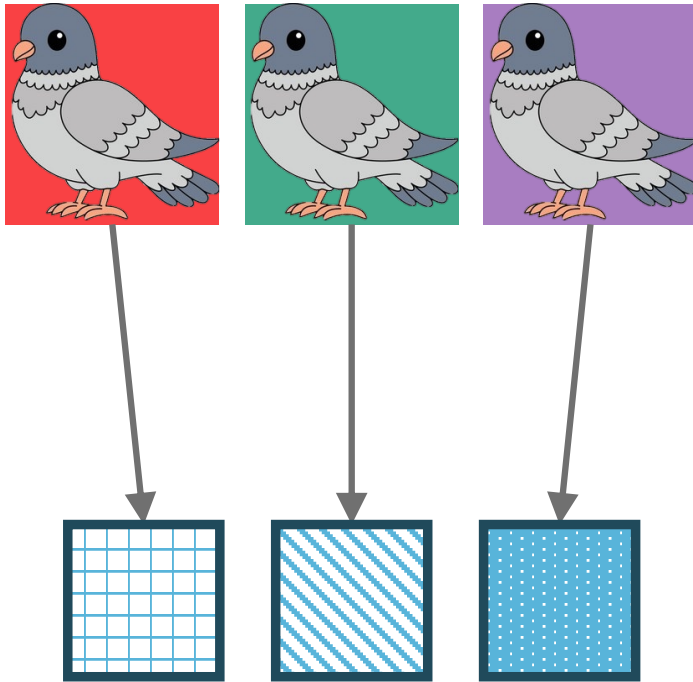
---

```
1:  $U \leftarrow \phi$ ;  
2: while there are unmarked constraints in  $U$  do  
3:    $c \leftarrow$  next unmarked constraint in  $U$   
4:    $(\text{is\_sat}, \alpha, U') \leftarrow \text{SAT}(U \setminus \{c\})$   
5:   if  $\text{is\_sat}$  then  
6:     mark  $c$  as required  
12:  else  
13:     $U \leftarrow U'$   
14: return  $U$ 
```

---

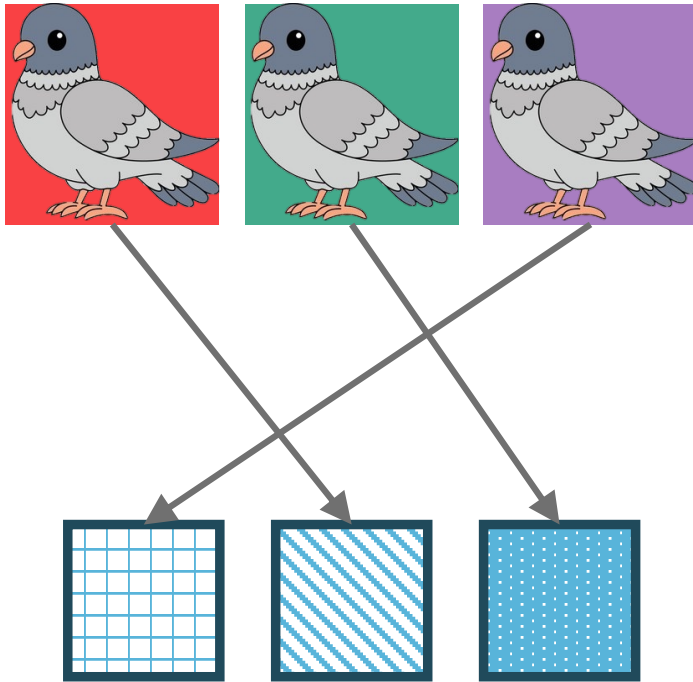
Requires call to  
NP-complete SAT-oracle

# Symmetries



Constraint problems can exhibit symmetries  
Swap assignments, without changing outcome

# Symmetries



Constraint problems can exhibit symmetries  
Swap assignments, without changing outcome

# Symmetries for the MUS-problem

## SAT-problem

Reason over variable **assignments**  
→ symmetries of **assignments**

Assignment satisfies constraints iff  
symmetric image satisfies constraints

Detect automatically (e.g., BreakID)

## MUS-problem







Reason over subsets of **constraints**  
→ symmetries of **constraints**

Set of constraints is SAT iff  
symmetric image is

Derive from syntactic symmetries

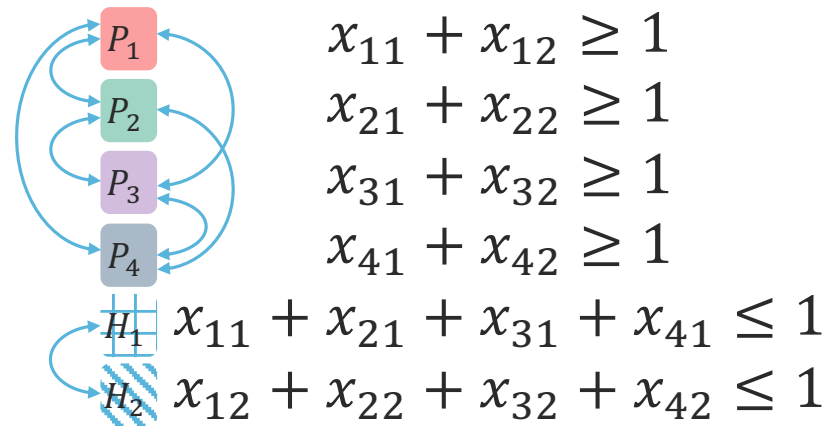
# Symmetries for the MUS problem

Pigeon-hole problem (4 pigeons, 2 holes)

 $P_1$	$x_{11} + x_{12} \geq 1$
 $P_2$	$x_{21} + x_{22} \geq 1$
 $P_3$	$x_{31} + x_{32} \geq 1$
 $P_4$	$x_{41} + x_{42} \geq 1$
 $H_1$	$x_{11} + x_{21} + x_{31} + x_{41} \leq 1$
 $H_2$	$x_{12} + x_{22} + x_{32} + x_{42} \leq 1$

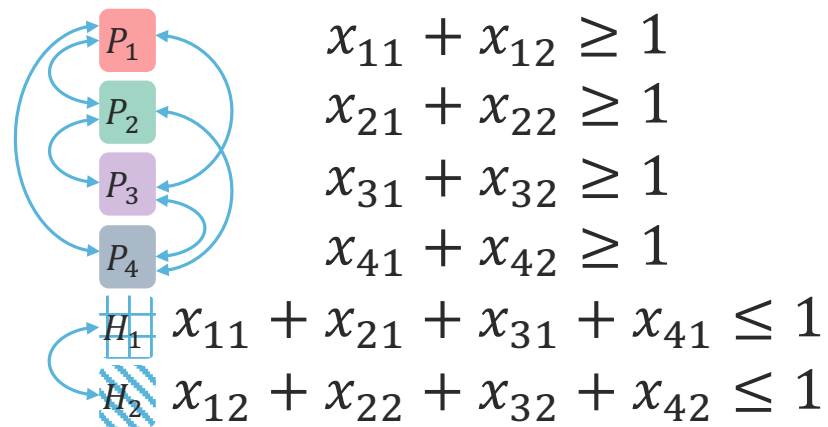
# Symmetries for the MUS problem

Pigeon-hole problem (4 pigeons, 2 holes)

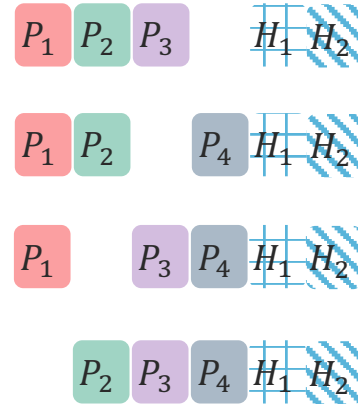


# Symmetries for the MUS problem

Pigeon-hole problem (4 pigeons, 2 holes)



Four **symmetric** MUSes:



# Exploiting symmetries in the MUS-problem

Deletion-based MUS

Case study

Implicit-hitting-set based MUS

In paper

MUS-enumeration

In paper



# Deletion-based MUS-computation

---

**Algorithm 1: SHRINK( $\phi$ )**

---

```
1:  $U \leftarrow \phi$ ;  
2: while there are unmarked constraints in  $U$  do  
3:    $c \leftarrow$  next unmarked constraint in  $U$   
4:    $(\text{is\_sat}, \alpha, U') \leftarrow \text{SAT}(U \setminus \{c\})$   
5:   if  $\text{is\_sat}$  then  
6:     mark  $c$  as required  
12:  else  
13:     $U \leftarrow U'$   
14: return  $U$ 
```

---

Try removing a constraint

If the remainder is SAT, re-add

If the remainder is UNSAT,  
remove permanently

# Deletion-based MUS-computation

---

**Algorithm 1: SYMM-SHRINK( $\phi$ )**

---

```
1:  $U \leftarrow \phi$ ;  
2: while there are unmarked constraints in  $U$  do  
3:    $c \leftarrow$  next unmarked constraint in  $U$   
4:    $(\text{is\_sat}, \alpha, U') \leftarrow \text{SAT}(U \setminus \{c\})$   
5:   if  $\text{is\_sat}$  then  
6:     mark  $c$  as required  
9:     for each  $\pi \in \mathcal{G}$  do  
10:      if  $\pi(U) = U$  then  
11:        mark  $\pi(c)$  as required  
12:   else  
13:      $U \leftarrow U'$   
14: return  $U$ 
```

---

Try removing a constraint

If the remainder is SAT, re-add  
And mark all symmetric images

If the remainder is UNSAT,  
remove permanently

# Exploiting symmetries in the MUS-problem

---

## Algorithm 1: SHRINK( $\phi$ )

---

```
1:  $U \leftarrow \phi$ ;  $\mathcal{G} \leftarrow \text{CONSTRAINTSYMMETRIES}(\phi)$ 
2: while there are unmarked constraints in  $U$  do
3:    $c \leftarrow$  next unmarked constraint in  $U$ 
4:    $(\text{is\_sat}, \alpha, U') \leftarrow \text{SAT}(U \setminus \{c\})$ 
5:   if  $\text{is\_sat}$  then
6:     mark  $c$  as required
7:      $\alpha \leftarrow \alpha \cup \{c\}$ 
9:     for each  $\pi \in \mathcal{G}$  do
10:      if  $\pi(U) = U$  then
11:        mark  $\pi(c)$  as required
12:   else
13:      $U \leftarrow U'$ 
14: return  $U$ 
```

---

$P_1$

$P_2$

$P_3$

$P_4$

$H_1$

$H_2$

# Exploiting symmetries in the MUS-problem

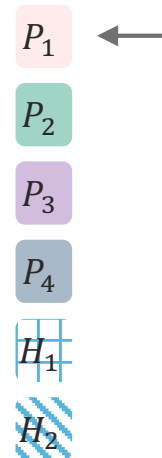
---

## Algorithm 1: SHRINK( $\phi$ )

---

```
1:  $U \leftarrow \phi$ ;  $\mathcal{G} \leftarrow \text{CONSTRAINTSYMMETRIES}(\phi)$ 
2: while there are unmarked constraints in  $U$  do
3:    $c \leftarrow$  next unmarked constraint in  $U$ 
4:    $(\text{is\_sat}, \alpha, U') \leftarrow \text{SAT}(U \setminus \{c\})$ 
5:   if  $\text{is\_sat}$  then
6:     mark  $c$  as required
7:   for each  $\pi \in \mathcal{G}$  do
8:     if  $\pi(U) = U$  then
9:       mark  $\pi(c)$  as required
10:  else
11:     $U \leftarrow U'$ 
12: return  $U$ 
```

---



# Exploiting symmetries in the MUS-problem

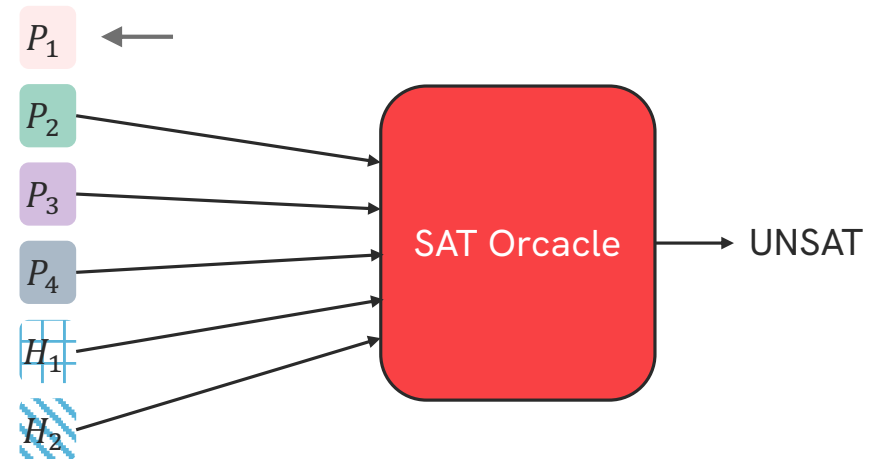
---

## Algorithm 1: SHRINK( $\phi$ )

---

```
1:  $U \leftarrow \phi$ ;  $\mathcal{G} \leftarrow \text{CONSTRAINTSYMMETRIES}(\phi)$ 
2: while there are unmarked constraints in  $U$  do
3:    $c \leftarrow$  next unmarked constraint in  $U$ 
4:    $(\text{is\_sat}, \alpha, U') \leftarrow \text{SAT}(U \setminus \{c\})$ 
5:   if  $\text{is\_sat}$  then
6:     mark  $c$  as required
7:   for each  $\pi \in \mathcal{G}$  do
8:     if  $\pi(U) = U$  then
9:       mark  $\pi(c)$  as required
10:  else
11:     $U \leftarrow U'$ 
12:  return  $U$ 
```

---



# Exploiting symmetries in the MUS-problem

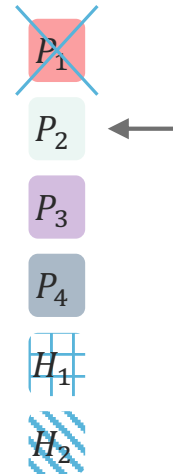
---

## Algorithm 1: SHRINK( $\phi$ )

---

```
1:  $U \leftarrow \phi$ ;  $\mathcal{G} \leftarrow \text{CONSTRAINTSYMMETRIES}(\phi)$ 
2: while there are unmarked constraints in  $U$  do
3:    $c \leftarrow$  next unmarked constraint in  $U$ 
4:    $(\text{is\_sat}, \alpha, U') \leftarrow \text{SAT}(U \setminus \{c\})$ 
5:   if  $\text{is\_sat}$  then
6:     mark  $c$  as required
7:   else
8:     for each  $\pi \in \mathcal{G}$  do
9:       if  $\pi(U) = U$  then
10:        mark  $\pi(c)$  as required
11:   else
12:      $U \leftarrow U'$ 
13: return  $U$ 
```

---



# Exploiting symmetries in the MUS-problem

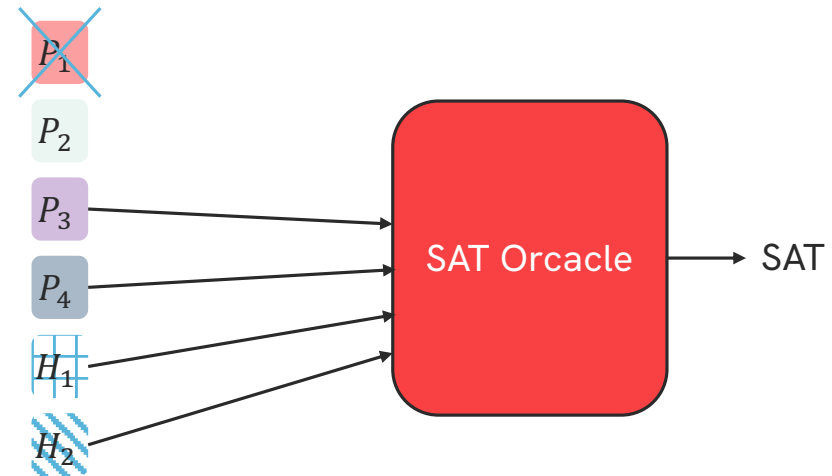
---

## Algorithm 1: SHRINK( $\phi$ )

---

```
1:  $U \leftarrow \phi$ ;  $\mathcal{G} \leftarrow \text{CONSTRAINTSYMMETRIES}(\phi)$ 
2: while there are unmarked constraints in  $U$  do
3:    $c \leftarrow$  next unmarked constraint in  $U$ 
4:    $(\text{is\_sat}, \alpha, U') \leftarrow \text{SAT}(U \setminus \{c\})$ 
5:   if  $\text{is\_sat}$  then
6:     mark  $c$  as required
7:   for each  $\pi \in \mathcal{G}$  do
8:     if  $\pi(U) = U$  then
9:       mark  $\pi(c)$  as required
10:  else
11:     $U \leftarrow U'$ 
12: return  $U$ 
```

---



# Exploiting symmetries in the MUS-problem

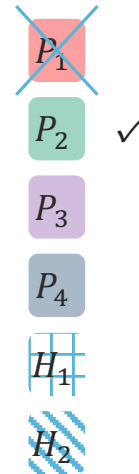
---

## Algorithm 1: SHRINK( $\phi$ )

---

```
1:  $U \leftarrow \phi$ ;  $\mathcal{G} \leftarrow \text{CONSTRAINTSYMMETRIES}(\phi)$ 
2: while there are unmarked constraints in  $U$  do
3:    $c \leftarrow$  next unmarked constraint in  $U$ 
4:    $(\text{is\_sat}, \alpha, U') \leftarrow \text{SAT}(U \setminus \{c\})$ 
5:   if  $\text{is\_sat}$  then
6:     mark  $c$  as required
7:   for each  $\pi \in \mathcal{G}$  do
8:     if  $\pi(U) = U$  then
9:       mark  $\pi(c)$  as required
10:  else
11:     $U \leftarrow U'$ 
12: return  $U$ 
```

---





# Exploiting symmetries in the MUS-problem

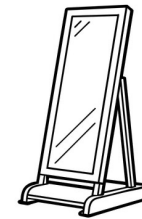
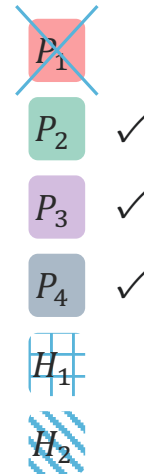
---

## Algorithm 1: SHRINK( $\phi$ )

---

```
1:  $U \leftarrow \phi$ ;  $\mathcal{G} \leftarrow \text{CONSTRAINTSYMMETRIES}(\phi)$ 
2: while there are unmarked constraints in  $U$  do
3:    $c \leftarrow$  next unmarked constraint in  $U$ 
4:    $(\text{is\_sat}, \alpha, U') \leftarrow \text{SAT}(U \setminus \{c\})$ 
5:   if  $\text{is\_sat}$  then
6:     mark  $c$  as required
7:   for each  $\pi \in \mathcal{G}$  do
8:     if  $\pi(U) = U$  then
9:       mark  $\pi(c)$  as required
10:  else
11:     $U \leftarrow U'$ 
12:  return  $U$ 
```

---



# Exploiting symmetries in the MUS-problem

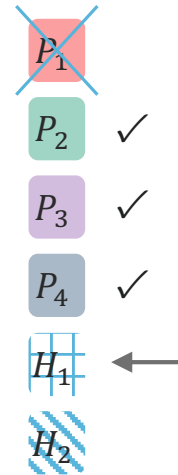
---

## Algorithm 1: SHRINK( $\phi$ )

---

```
1:  $U \leftarrow \phi$ ;  $\mathcal{G} \leftarrow \text{CONSTRAINTSYMMETRIES}(\phi)$ 
2: while there are unmarked constraints in  $U$  do
3:    $c \leftarrow$  next unmarked constraint in  $U$ 
4:    $(\text{is\_sat}, \alpha, U') \leftarrow \text{SAT}(U \setminus \{c\})$ 
5:   if  $\text{is\_sat}$  then
6:     mark  $c$  as required
7:   for each  $\pi \in \mathcal{G}$  do
8:     if  $\pi(U) = U$  then
9:       mark  $\pi(c)$  as required
10:  else
11:     $U \leftarrow U'$ 
12: return  $U$ 
```

---



# Exploiting symmetries in the MUS-problem

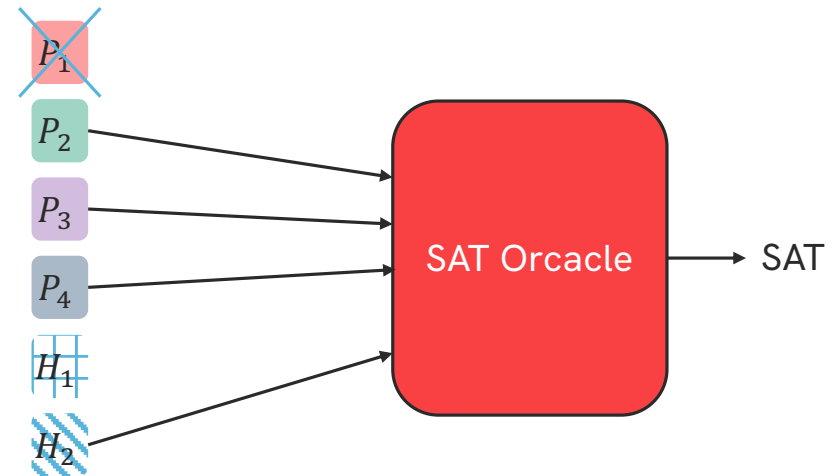
---

## Algorithm 1: SHRINK( $\phi$ )

---

```
1:  $U \leftarrow \phi$ ;  $\mathcal{G} \leftarrow \text{CONSTRAINTSYMMETRIES}(\phi)$ 
2: while there are unmarked constraints in  $U$  do
3:    $c \leftarrow$  next unmarked constraint in  $U$ 
4:    $(\text{is\_sat}, \alpha, U') \leftarrow \text{SAT}(U \setminus \{c\})$ 
5:   if  $\text{is\_sat}$  then
6:     mark  $c$  as required
7:   for each  $\pi \in \mathcal{G}$  do
8:     if  $\pi(U) = U$  then
9:       mark  $\pi(c)$  as required
10:  else
11:     $U \leftarrow U'$ 
12: return  $U$ 
```

---



# Exploiting symmetries in the MUS-problem

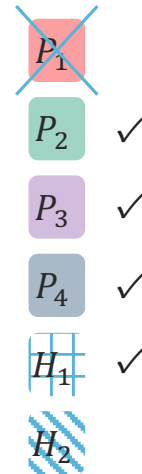
---

## Algorithm 1: SHRINK( $\phi$ )

---

```
1:  $U \leftarrow \phi$ ;  $\mathcal{G} \leftarrow \text{CONSTRAINTSYMMETRIES}(\phi)$ 
2: while there are unmarked constraints in  $U$  do
3:    $c \leftarrow$  next unmarked constraint in  $U$ 
4:    $(\text{is\_sat}, \alpha, U') \leftarrow \text{SAT}(U \setminus \{c\})$ 
5:   if  $\text{is\_sat}$  then
6:     mark  $c$  as required
7:   for each  $\pi \in \mathcal{G}$  do
8:     if  $\pi(U) = U$  then
9:       mark  $\pi(c)$  as required
10:  else
11:     $U \leftarrow U'$ 
12: return  $U$ 
```

---



# Exploiting symmetries in the MUS-problem

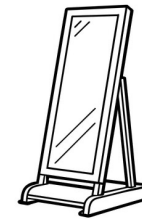
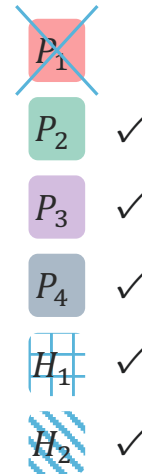
---

## Algorithm 1: SHRINK( $\phi$ )

---

```
1:  $U \leftarrow \phi$ ;  $\mathcal{G} \leftarrow \text{CONSTRAINTSYMMETRIES}(\phi)$ 
2: while there are unmarked constraints in  $U$  do
3:    $c \leftarrow$  next unmarked constraint in  $U$ 
4:    $(\text{is\_sat}, \alpha, U') \leftarrow \text{SAT}(U \setminus \{c\})$ 
5:   if  $\text{is\_sat}$  then
6:     mark  $c$  as required
7:   else
8:     for each  $\pi \in \mathcal{G}$  do
9:       if  $\pi(U) = U$  then
10:        mark  $\pi(c)$  as required
11:   else
12:      $U \leftarrow U'$ 
13: return  $U$ 
```

---



# Benchmarks

Pigeon-hole

Assign pigeons to holes

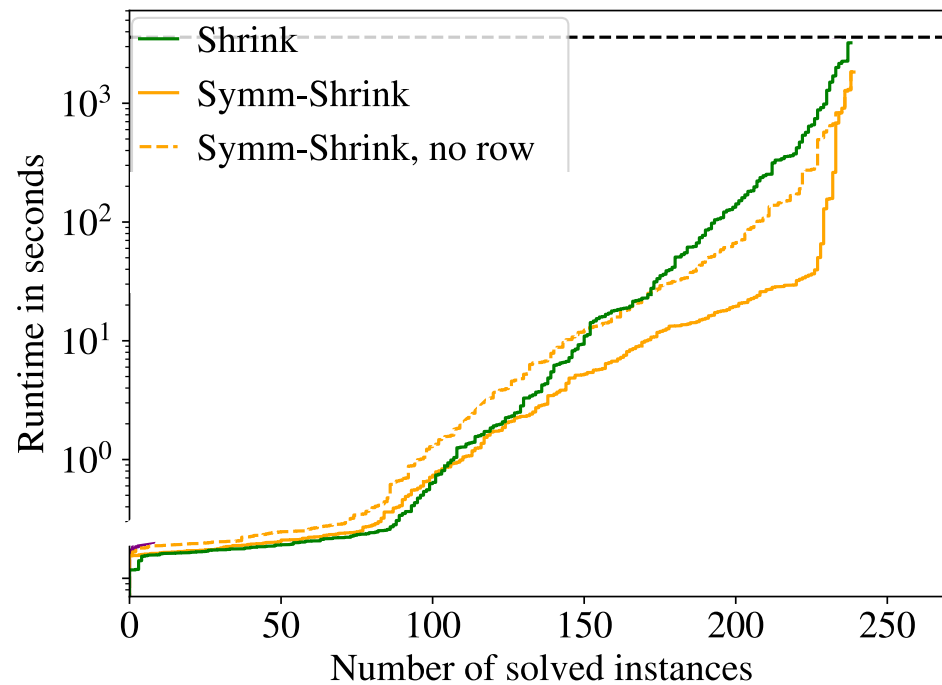
N+k-queens

UNSAT Boolean encoding of N-queens

Bin-packing

Pack items into (not enough)  
equivalent bins

# Results

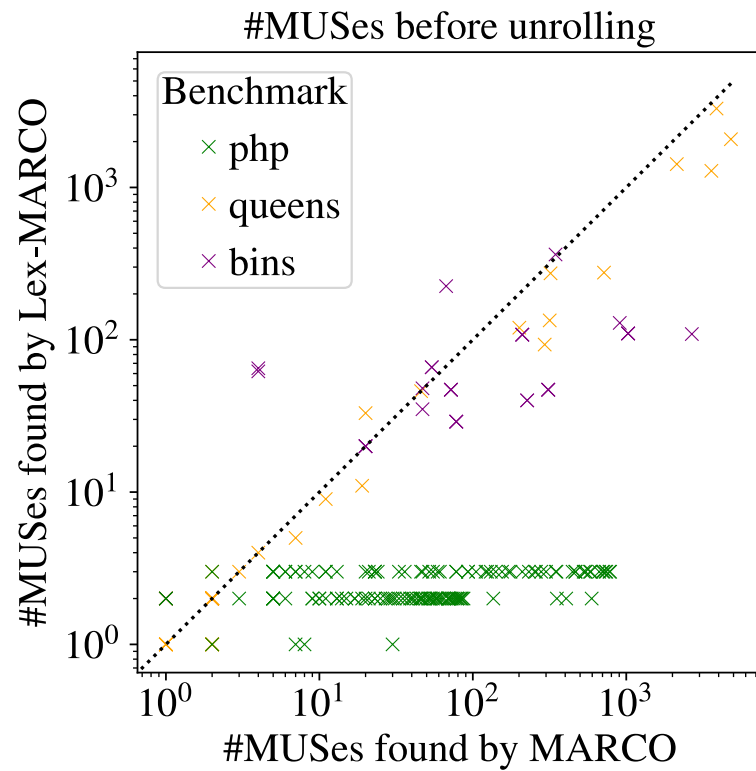


Up to 10x runtime improvement  
Overhead detection + exploiting

More advanced methods and variants possible  
And described in the paper

IHS-algorithms and enumeration results  
Available in the paper

# Results for MUS-enumeration



Enumerate 1 MUS of each “type”  
Similar MUSes irrelevant for user





Symmetries slow down MUS-computation  
And can be exploited to speed-up algorithms

Can symmetries be *used* as a “lifted explanation”?

“No 3 pigeons fit into 2 holes”

Improvements to other algorithms in full paper  
For IHS and MUS enumeration



Tuples  
TRUSTWORTHY AI



Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union or the European Research Council. Neither the European Union nor the granting authority can be held responsible for them.