Efficiently Explaining CSPs with Unsatisfiable Subset Optimization

Emilio Gamba[1] Bart Bogaerts[1] Tias Guns[1,2]

[1]Vrije Universiteit Brussel, Belgium
[2]KULeuven, Belgium
emilio.gamba@vub.be, bart.bogaerts@vub.be, tias.guns@kuleuven.be

ModRef 2022

Examples of Constraint Satisfaction Problems

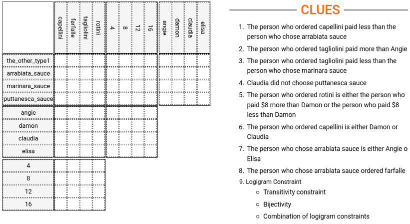


Figure: Logic Grid Puzzle

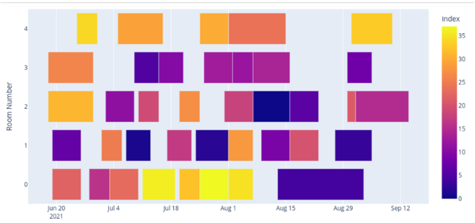


Figure: (Room) Scheduling Problem

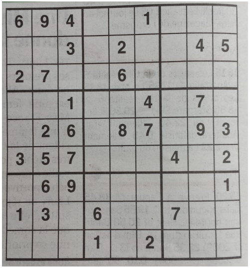


Figure: Sudoku

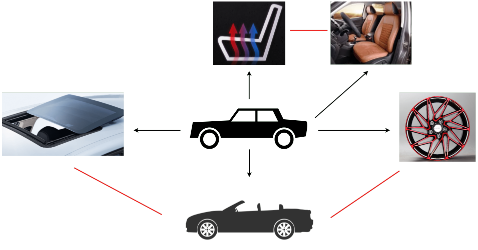


Figure: (Car) configuration problems

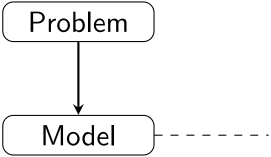
Constraint Satisfaction Problems

Solving

Problem

Constraint Satisfaction Problems

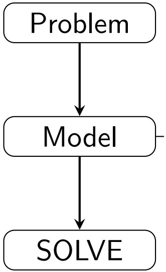
Solving



CPMpy (Guns [2019])
(CP modelling Library)
`pip install cpmpy`

Constraint Satisfaction Problems

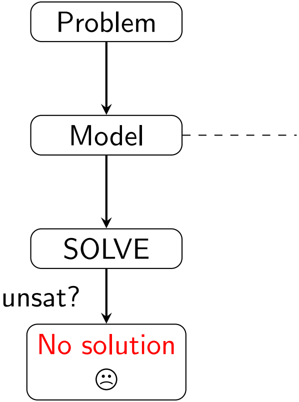
Solving



CPMpy (Guns [2019])
(CP modelling Library)
`pip install cpmpy`

Constraint Satisfaction Problems

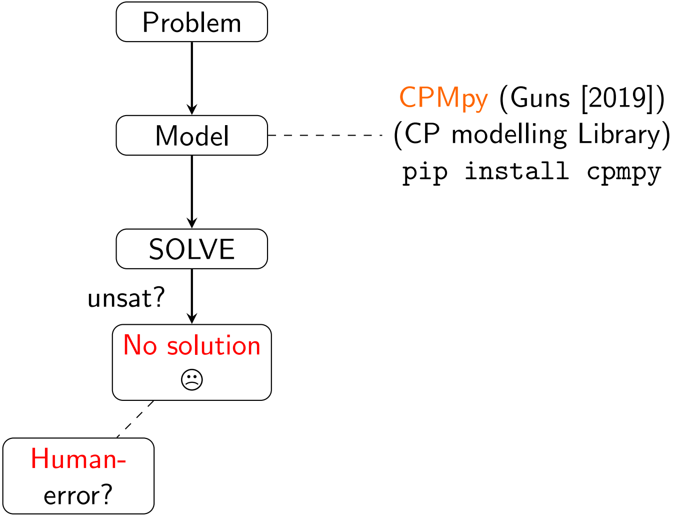
Solving



CPMpy (Guns [2019])
(CP modelling Library)
`pip install cpmpy`

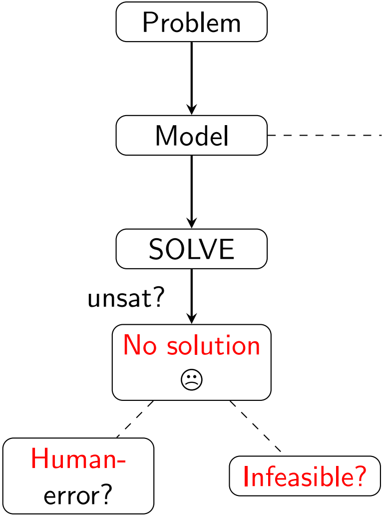
Constraint Satisfaction Problems

Solving



Constraint Satisfaction Problems

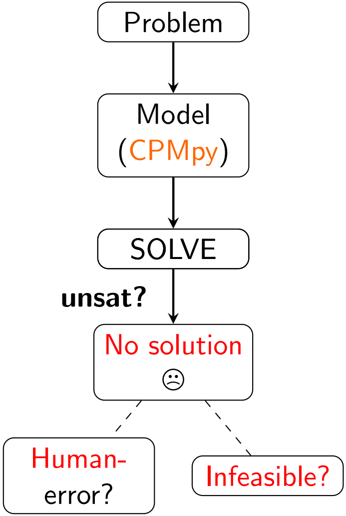
Solving



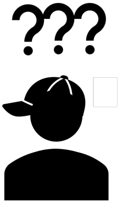
CPMpy (Guns [2019])
(CP modelling Library)
`pip install cpmpy`

Constraint Satisfaction Problems

Solving

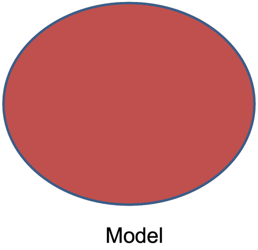


How do you explain UNSAT ?



How do I debug my model

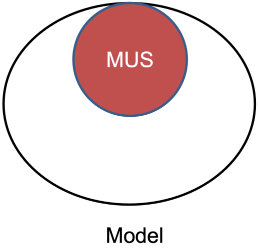
Explanations of unsatisfiability



How do I debug my model

Explanations of unsatisfiability

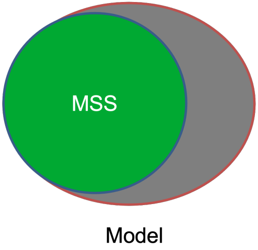
1. Identify conflicting constraints as an explanation (Liffiton and Sakallah [2008]; Ignatiev *et al.* [2015]...)
   → Extract a **Minimum Unsatisfiable Subset** (**MUS**)
   = Irreducible Inconsistent Subsystem (ISS)



How do I debug my model

Explanations of unsatisfiability

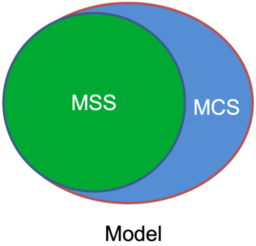
1. Identify conflicting constraints as an explanation (Liffiton and Sakallah [2008]; Ignatiev *et al.* [2015]...)
   → Extract a **Minimum Unsatisfiable Subset** (**MUS**)
   = Irreducible Inconsistent Subsystem (ISS)
2. Identify a **Maximal Satisfiable Subset** (**MSS**) (Ignatiev *et al.* [2019]; Davies and Bacchus [2013]; Hansen and Jaumard [1990]...)



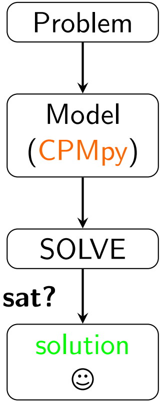
How do I debug my model

Explanations of unsatisfiability

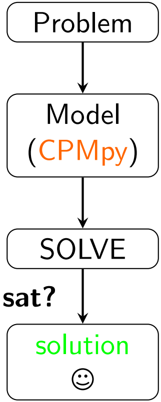
1. Identify conflicting constraints as an explanation (Liffiton and Sakallah [2008]; Ignatiev *et al.* [2015]...)
   - → Extract a **Minimum Unsatisfiable Subset** (**MUS**)
   - = Irreducible Inconsistent Subsystem (ISS)
2. Identify a **Maximal Satisfiable Subset** (**MSS**) (Ignatiev *et al.* [2019]; Davies and Bacchus [2013]; Hansen and Jaumard [1990]...)
3. "Correct" the infeasibility in the constraints (Liffiton and Malik [2013]...)
   - → Extract a **Minimum Correction Subset** (**MCS**)
   - = Complement of some **MSS**, removal/correction leads to a satisfiable subset



Motivation - Explaining SAT problems

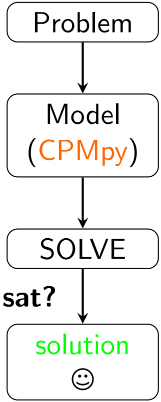


Motivation - Explaining SAT problems



How do I explain SAT ?

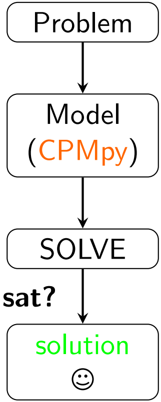
Motivation - Explaining SAT problems



How do I explain SAT ?

- What is an **explanation**?

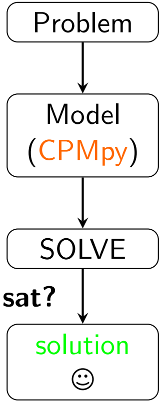
Motivation - Explaining SAT problems



How do I explain SAT ?

- What is an **explanation**?
- What is a **good** explanation ?

Motivation - Explaining SAT problems



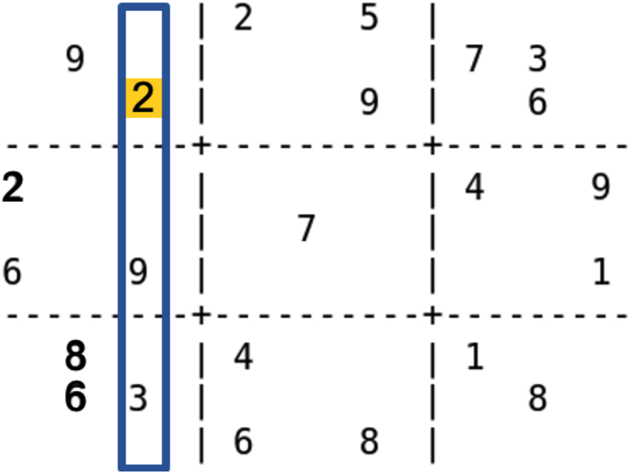
How do I explain SAT ?

- What is an **explanation**?
- What is a **good** explanation ?
- What is a **sequence of explanations** ?

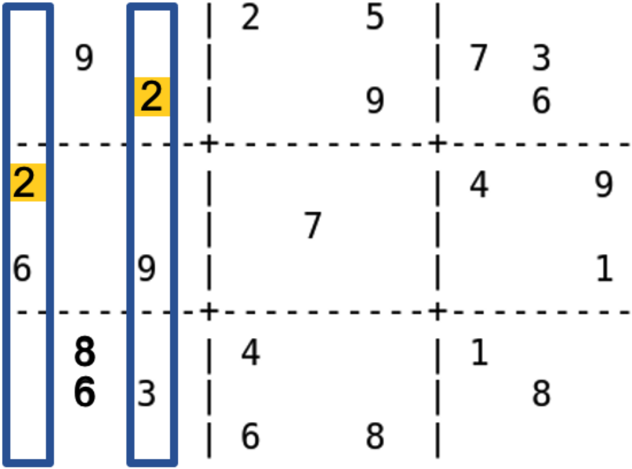
What is an explanation ?

```
      |  2     5  |
   9  |           |  7  3
     2|        9  |     6
------+-----------+---------
2     |           |  4     9
      |     7     |
6    9|           |        1
------+-----------+---------
   8  |  4        |  1
   6 3|           |     8
      |  6     8  |
```

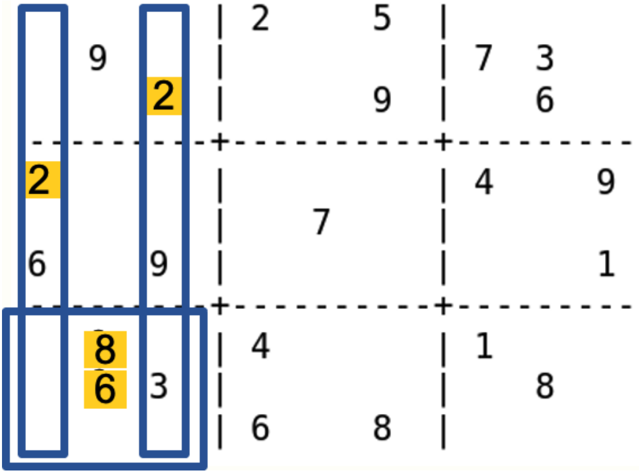
What is an explanation ?



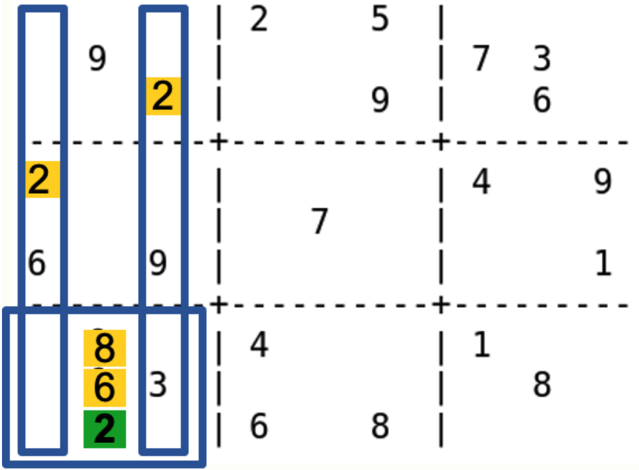
What is an explanation ?



What is an explanation ?



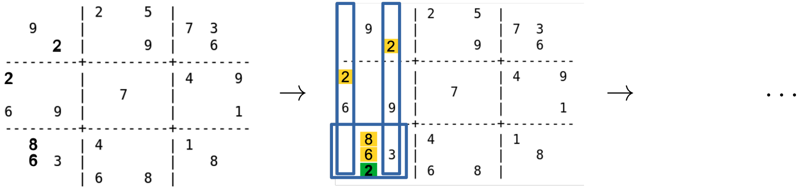
What is an explanation ?



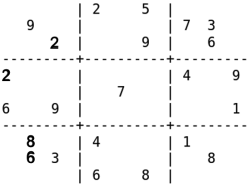
What is a sequence of explanations ?

```
      |  2     5  |
   9  |           |  7  3
     2|        9  |     6
------+-----------+---------
2     |           |  4     9
      |     7     |
6    9|           |        1
------+-----------+---------
   8  |  4        |  1
   6 3|           |     8
      |  6     8  |
```

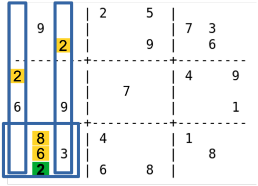
What is a sequence of explanations ?



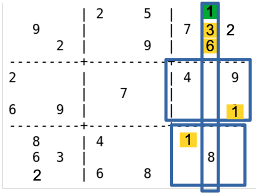
What is a sequence of explanations ?



→

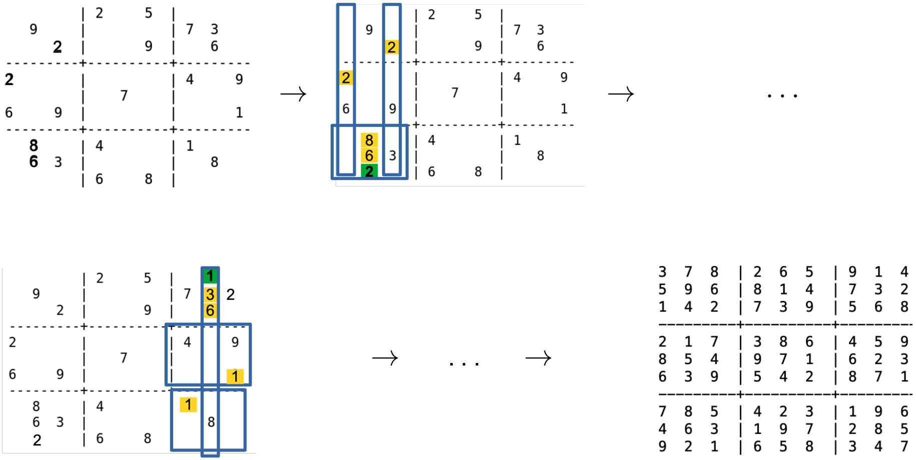


→ . . .



→ . . . →

What is a sequence of explanations ?



Stepwise explanations for CSPs

Goal

Given

C Constraints

E Facts, i.e. given sudoku numbers

Stepwise explanations for CSPs

Goal

Given

C Constraints

E Facts, i.e. given sudoku numbers

Goal:

- Generate a *sequence* of *simple* explanations

Stepwise explanations for CSPs

Goal

Given

C Constraints

E Facts, i.e. given sudoku numbers

Goal:

- Generate a *sequence* of *simple* explanations
- Explain step-by-step the solution of a Constraint Satisfaction Problem

Stepwise explanations for CSPs

Goal

Given

C Constraints

E Facts, i.e. given sudoku numbers

Goal:

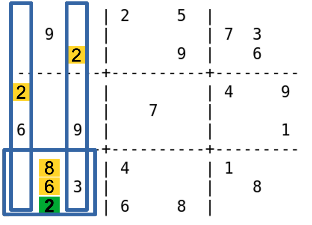
- Generate a *sequence* of *simple* explanations
- Explain step-by-step the solution of a Constraint Satisfaction Problem
- Explain 1 fact at a time

Stepwise explanations for CSPs

Explanation step - Formal definition

Let an EXPLANATION STEP (Bogaerts *et al.* [2020]) be:

$$E' \;\&\; C' \implies n$$



Stepwise explanations for CSPs

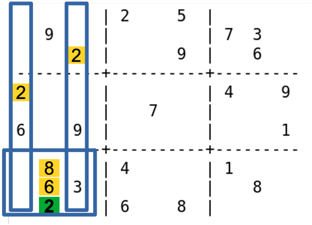
Explanation step - Formal definition

Let an EXPLANATION STEP (Bogaerts *et al.* [2020]) be:

$$E' \,\&\, C' \implies n$$

E' A subset of previously derived facts E

(Sudoku) Given and derived digits in the grid



Stepwise explanations for CSPs

Explanation step - Formal definition

Let an EXPLANATION STEP (Bogaerts *et al.* [2020]) be:

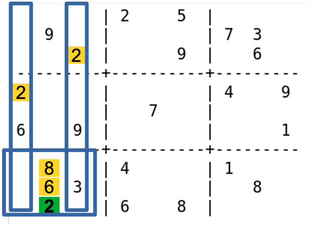
E' & C' $\implies$ ***n***

E' A subset of previously derived facts **E**

(Sudoku) Given and derived digits in the grid

C' A minimal subset of model constraints **C**

(Sudoku) Alldifferent column, row, box constraints



Stepwise explanations for CSPs

Explanation step - Formal definition

Let an EXPLANATION STEP (Bogaerts *et al.* [2020]) be:

E' & C' $\implies$ ***n***

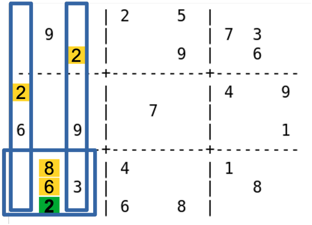
E' A subset of previously derived facts **E**

(Sudoku) Given and derived digits in the grid

C' A minimal subset of model constraints **C**

(Sudoku) Alldifferent column, row, box constraints

n A newly derived fact s.t. E' & C' $\implies$ ***n***



Stepwise explanations for CSPs

Explanation step - Formal definition

Let an EXPLANATION STEP (Bogaerts *et al.* [2020]) be:

$$E' \,\&\, C' \implies n$$

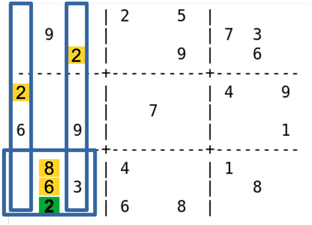
E' A subset of previously derived facts E

(Sudoku) Given and derived digits in the grid

C' A minimal subset of model constraints C

(Sudoku) Alldifferent column, row, box constraints

n A newly derived fact s.t. $E' \,\&\, C' \implies n$



How ? MUS(E & C & ¬ n) is a valid explanation step

Stepwise explanations for CSPs

EXPLANATION STEP

Example

(Sudoku) Let E contain the assigned variables at the current state of the grid (e.g. $I = \{V_{(3,3)} = 2, \ldots\}$).

MUS(C & E &¬ ***n***)

$\{\texttt{alldifferent}(\{V_{(r,1)} | r \in 1..9\}), V_{(4,1)} = 2,$
$\texttt{alldifferent}(\{V_{(r,3)} | r \in 1..9\}), V_{(3,3)} = 2,$
$\texttt{alldifferent}(\{V_{(r_i,c_j)} | r_i \in 7..9, c_j \in 1..3\}),$
$V_{(7,2)} = 2, V_{(8,2)} = 2, V_{(9,2)} \neq 2\}$

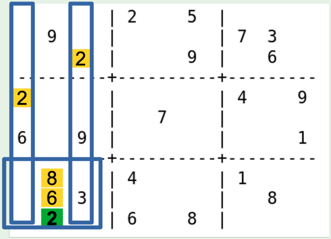
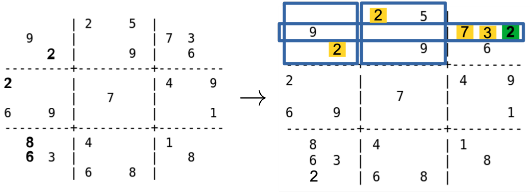


Figure: Example of a non-redundant explanation for $V_{(9,2)} = 2$

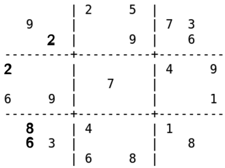
What is a good explanation ?

```
      |  2     5  |
   9  |           |  7  3
      2  |     9  |     6
---------+--------+---------
2        |        |  4     9
         |    7   |
6     9  |        |        1
---------+--------+---------
   8     |  4     |  1
   6  3  |        |     8
         |  6     8  |
```

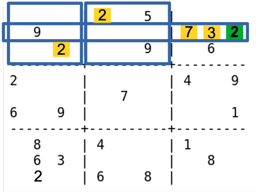
What is a good explanation ?



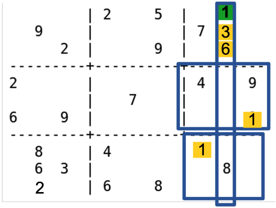
What is a good explanation ?



→



or



The best/easiest explanation step...

What is a good explanation ?

The best/easiest explanation step...

What is a good explanation ?

Let **f**(E', C', ***n***) be a cost-function that quantifies how good (e.g. easy to understand) an explanation step is.

The best/easiest explanation step...

What is a good explanation ?

Let **f**(E', C', ***n***) be a cost-function that quantifies how good (e.g. easy to understand) an explanation step is.

What is the best/easiest explanation step?

The best/easiest explanation step...

What is a good explanation ?
Let **f**(E', C', ***n***) be a cost-function that quantifies how good (e.g. easy to understand) an explanation step is.

What is the best/easiest explanation step?

$X_{best} \leftarrow nil$;

The best/easiest explanation step...

What is a good explanation ?

Let **f**(E', C', ***n***) be a cost-function that quantifies how good (e.g. easy to understand) an explanation step is.

What is the best/easiest explanation step?

$X_{best} \leftarrow nil$;

for ***n*** $\in$ `propagate(`C`)` **do**

  $X \leftarrow$ `MUS(` E & C &$\neg$ ***n*** `);`

The best/easiest explanation step...

What is a good explanation ?

Let **f**(E', C', ***n***) be a cost-function that quantifies how good (e.g. easy to understand) an explanation step is.

What is the best/easiest explanation step?

$X_{best} \leftarrow nil$;
for ***n*** $\in$ `propagate(`C`)` **do**
    $X \leftarrow$ `MUS(` E & C &$\neg$ ***n*** `)`;
    if $f(X) < f(X_{best})$ **then**
        $X_{best} \leftarrow X$;
end
return X_{best}

Motivation

Challenges and open questions from Bogaerts *et al.* [2020]

Q1 **Optimality** w.r.t **f**

- ▷ MUS guarantees non-redundancy but ... not optimality.
- ▷ Alternative: SMUS #-minimal (Ignatiev *et al.* [2015])

$X_{best} \leftarrow nil$;
for n $\in$ `propagate(`C`)` **do**
  $X \leftarrow \mathrm{MUS}(C \wedge E \wedge \neg n)$;
  if $f(X) < f(X_{best})$ **then**
    $X_{best} \leftarrow X$;
end
return X_{best}

Motivation

Challenges and open questions from Bogaerts *et al.* [2020]

Q1 **Optimality** w.r.t **f**

- MUS guarantees non-redundancy but ... not optimality.
- Alternative: SMUS #-minimal (Ignatiev *et al.* [2015])

Explanation generation takes a lot of time:

$X_{best} \leftarrow nil$;
for $n \in \texttt{propagate}(C)$ **do**
  $X \leftarrow \mathrm{MUS}(C \wedge E \wedge \neg n)$;
  if $f(X) < f(X_{best})$ **then**
    $X_{best} \leftarrow X$;
end
return X_{best}

Motivation

Challenges and open questions from Bogaerts *et al.* [2020]

Q1 **Optimality** w.r.t **f**

- ▷ MUS guarantees non-redundancy but ... not optimality.
- ▷ Alternative: SMUS #-minimal (Ignatiev *et al.* [2015])

Explanation generation takes a lot of time:

Q2 Can we avoid looping over the literals when searching for the next best explanation ?

$X_{best} \leftarrow nil$;
for n $\in$ `propagate(C)` **do**
  $X \leftarrow \mathrm{MUS}(C \wedge E \wedge \neg n)$;
  if $f(X) < f(X_{best})$ **then**
    $X_{best} \leftarrow X$;
end
return X_{best}

Motivation

Challenges and open questions from Bogaerts *et al.* [2020]

Q1 **Optimality** w.r.t **f**

- ▷ MUS guarantees non-redundancy but ... not optimality.
- ▷ Alternative: SMUS #-minimal (Ignatiev *et al.* [2015])

Explanation generation takes a lot of time:

Q2 Can we avoid looping over the literals when searching for the next best explanation ?

Q3 Can we reuse information from an explanation call to another?

```
X_best ← nil;
for n ∈ propagate(C) do
    X ← MUS(C ∧ E ∧ ¬n);
    if f(X) < f(X_best) then
        X_best ← X;
end
return X_best
```

OCUS-Based explanation generation

The OCUS[1] problem

Definition

Let $\mathcal{F}$ be a formula, $f : 2^{\mathcal{F}} \rightarrow \mathbb{N}$ a cost function and p a predicate $p : 2^{\mathcal{F}} \rightarrow \{\mathbf{t}, \mathbf{f}\}$. We call $\mathcal{S} \subseteq \mathcal{F}$ an OCUS of $\mathcal{F}$ (with respect to f and p) if

- $\mathcal{S}$ is unsatisfiable,
- $p(\mathcal{S})$ is true
- all other unsatisfiable $\mathcal{S}' \subseteq \mathcal{F}$ with $p(\mathcal{S}') = \mathbf{t}$ satisfy $f(\mathcal{S}') \geq f(\mathcal{S})$.

[1] Optimal Constrained Unsatisfiable Subset

OCUS-Based explanation generation

The OCUS[1] problem

Definition

Let $\mathcal{F}$ be a formula, $f : 2^{\mathcal{F}} \rightarrow \mathbb{N}$ a cost function and p a predicate $p : 2^{\mathcal{F}} \rightarrow \{\mathbf{t}, \mathbf{f}\}$. We call $\mathcal{S} \subseteq \mathcal{F}$ an OCUS of $\mathcal{F}$ (with respect to f and p) if

- $\mathcal{S}$ is unsatisfiable,
- $p(\mathcal{S})$ is true
- all other unsatisfiable $\mathcal{S}' \subseteq \mathcal{F}$ with $p(\mathcal{S}') = \mathbf{t}$ satisfy $f(\mathcal{S}') \geq f(\mathcal{S})$.

Applied to explanation generation:

[1] Optimal Constrained Unsatisfiable Subset

OCUS-Based explanation generation

The OCUS[1] problem

Definition

Let $\mathcal{F}$ be a formula, $f : 2^{\mathcal{F}} \rightarrow \mathbb{N}$ a cost function and p a predicate $p : 2^{\mathcal{F}} \rightarrow \{\mathbf{t}, \mathbf{f}\}$. We call $\mathcal{S} \subseteq \mathcal{F}$ an OCUS of $\mathcal{F}$ (with respect to f and p) if

- $\mathcal{S}$ is unsatisfiable,
- $p(\mathcal{S})$ is true
- all other unsatisfiable $\mathcal{S}' \subseteq \mathcal{F}$ with $p(\mathcal{S}') = \mathbf{t}$ satisfy $f(\mathcal{S}') \geq f(\mathcal{S})$.

Applied to explanation generation:

Q1 *(Optimality)* f ensures finding ‘best’ explanation

[1] Optimal Constrained Unsatisfiable Subset

OCUS-Based explanation generation

The OCUS[1] problem

Definition

Let $\mathcal{F}$ be a formula, $f : 2^{\mathcal{F}} \rightarrow \mathbb{N}$ a cost function and p a predicate $p : 2^{\mathcal{F}} \rightarrow \{\mathbf{t}, \mathbf{f}\}$. We call $\mathcal{S} \subseteq \mathcal{F}$ an OCUS of $\mathcal{F}$ (with respect to f and p) if

- $\mathcal{S}$ is unsatisfiable,
- $p(\mathcal{S})$ is true
- all other unsatisfiable $\mathcal{S}' \subseteq \mathcal{F}$ with $p(\mathcal{S}') = \mathbf{t}$ satisfy $f(\mathcal{S}') \geq f(\mathcal{S})$.

Applied to explanation generation:

Q1 *(Optimality)* f ensures finding 'best' explanation

Q2 *(Looping)* p allows formulating explaining 1 literal at a time using an extra constraint

[1] Optimal Constrained Unsatisfiable Subset

OCUS-Based explanation generation

$X_{best} \leftarrow nil$;
for $\mathtt{n} \in \mathtt{propagate}(C)$ **do**
    $X \leftarrow \mathrm{MUS}(C \wedge E \wedge \neg n)$;
    if $f(X) < f(X_{best})$ **then**
        $X_{best} \leftarrow X$;
end
return X_{best}

OCUS-Based explanation generation

$X_{best} \leftarrow nil$;
for n $\in$ `propagate`(C) **do**
  $X \leftarrow \mathrm{MUS}(C \wedge E \wedge \neg n)$;
  if $f(X) < f(X_{best})$ **then**
    $X_{best} \leftarrow X$;
end
return X_{best}

$\downarrow$

$p \leftarrow$ `exactly-one`($\{\neg$ ***n*** $|$ ***n*** $\in DigitToExplain\}$)

`OCUS`(C $\wedge$ E $\wedge \{\neg$ ***n*** $|$ ***n*** $\in DigitToExplain\}$, **f**, **p**)

An OCUS algorithm: A hitting set-based algorithm

Implicit Hitting set duality

- Exploit implicit hitting set–based duality between MCSes and MUSes (Liffiton and Sakallah [2008]; Reiter [1987])

An OCUS algorithm: A hitting set-based algorithm

Implicit Hitting set duality

- ▷ Exploit implicit hitting set–based duality between MCSes and MUSes (Liffiton and Sakallah [2008]; Reiter [1987])

Theorem

A set $\mathcal{S} \subseteq \mathcal{F}$ is a MCS of $\mathcal{F}$ iff it is a minimum hitting set of $\text{MUSs}(\mathcal{F})$.
A set $\mathcal{S} \subseteq \mathcal{F}$ is a MUS of $\mathcal{F}$ iff it is a minimum hitting set of $\text{MCSs}(\mathcal{F})$.

An OCUS algorithm: A hitting set-based algorithm

Implicit Hitting set duality

- Exploit implicit hitting set–based duality between MCSes and MUSes (Liffiton and Sakallah [2008]; Reiter [1987])

Theorem

A set $\mathcal{S} \subseteq \mathcal{F}$ is a MCS of $\mathcal{F}$ iff it is a minimum hitting set of $\textsc{MUSs}(\mathcal{F})$*.*
A set $\mathcal{S} \subseteq \mathcal{F}$ is a MUS of $\mathcal{F}$ iff it is a minimum hitting set of $\textsc{MCSs}(\mathcal{F})$*.*

- Extend this to OCUS:
  - Compute optimal hitting sets

An OCUS algorithm: A hitting set-based algorithm

Implicit Hitting set duality

- Exploit implicit hitting set–based duality between MCSes and MUSes (Liffiton and Sakallah [2008]; Reiter [1987])

Theorem

A set $\mathcal{S} \subseteq \mathcal{F}$ is a MCS of $\mathcal{F}$ iff it is a minimum hitting set of $\textsc{MUS}s(\mathcal{F})$.
A set $\mathcal{S} \subseteq \mathcal{F}$ is a MUS of $\mathcal{F}$ iff it is a minimum hitting set of $\textsc{MCS}s(\mathcal{F})$.

- Extend this to OCUS:
  - Compute optimal hitting sets
  - Impose constraint on the hitting sets

Optimal Hitting set problem

Optimal Hitting set problem

- Set of elements (i.e. constraints) $\{c_1, c_2, ..., c_n\}$
- Collection $\mathcal{H}$ of sets-to-hit (i.e. subsets of the set of elements)

Optimal Hitting set problem

- Set of elements (i.e. constraints) $\{c_1, c_2, ..., c_n\}$
- Collection $\mathcal{H}$ of sets-to-hit (i.e. subsets of the set of elements)
- cost associated for every element:

$$w_1 = 3, w_2 = 5, w_3 = w_4 = ... = w_8 = 1$$

Optimal Hitting set problem

- Set of elements (i.e. constraints) $\{c_1, c_2, ..., c_n\}$
- Collection $\mathcal{H}$ of sets-to-hit (i.e. subsets of the set of elements)
- cost associated for every element:

$$w_1 = 3, w_2 = 5, w_3 = w_4 = ... = w_8 = 1$$

Goal Find an optimal hitting set such that every set-to-hit is hit by at least once an element of the hitting set with minimal total cost.

Optimal Hitting set problem

- Set of elements (i.e. constraints) $\{c_1, c_2, ..., c_n\}$
- Collection $\mathcal{H}$ of sets-to-hit (i.e. subsets of the set of elements)
- cost associated for every element:

$$w_1 = 3, w_2 = 5, w_3 = w_4 = ... = w_8 = 1$$

Goal Find an optimal hitting set such that every set-to-hit is hit by at least once an element of the hitting set with minimal total cost.

For example:

$$H_1 = \{c_3, c_5\}$$
$$H_2 = \{c_2, c_4, c_7\}$$
$$H_3 = \{c_2, c_6\}$$
$$H_4 = \{c_1, c_8\}$$

Optimal hitting sets:

- $\{c_3, \}$

Optimal Hitting set problem

- Set of elements (i.e. constraints) $\{c_1, c_2, ..., c_n\}$
- Collection $\mathcal{H}$ of sets-to-hit (i.e. subsets of the set of elements)
- cost associated for every element:

$$w_1 = 3, w_2 = 5, w_3 = w_4 = ... = w_8 = 1$$

Goal Find an optimal hitting set such that every set-to-hit is hit by at least once an element of the hitting set with minimal total cost.

For example:

$H_1 = \{c_3, c_5\}$

$H_2 = \{c_2, c_4, c_7\}$

$H_3 = \{c_2, c_6\}$

$H_4 = \{c_1, c_8\}$

Optimal hitting sets:

- $\{c_3, c_4, \}$

Optimal Hitting set problem

- Set of elements (i.e. constraints) $\{c_1, c_2, ..., c_n\}$
- Collection $\mathcal{H}$ of sets-to-hit (i.e. subsets of the set of elements)
- cost associated for every element:

$$w_1 = 3, w_2 = 5, w_3 = w_4 = ... = w_8 = 1$$

Goal Find an optimal hitting set such that every set-to-hit is hit by at least once an element of the hitting set with minimal total cost.

For example:

$H_1 = \{c_3, c_5\}$

$H_2 = \{c_2, c_4, c_7\}$

$H_3 = \{c_2, c_6\}$

$H_4 = \{c_1, c_8\}$

Optimal hitting sets:

- $\{c_3, c_4, c_6, \}$

Optimal Hitting set problem

- Set of elements (i.e. constraints) $\{c_1, c_2, ..., c_n\}$
- Collection $\mathcal{H}$ of sets-to-hit (i.e. subsets of the set of elements)
- cost associated for every element:

$$w_1 = 3, w_2 = 5, w_3 = w_4 = ... = w_8 = 1$$

Goal Find an optimal hitting set such that every set-to-hit is hit by at least once an element of the hitting set with minimal total cost.

For example:

$$H_1 = \{c_3, c_5\}$$
$$H_2 = \{c_2, c_4, c_7\}$$
$$H_3 = \{c_2, c_6\}$$
$$H_4 = \{c_1, c_8\}$$

Optimal hitting sets:

- $\{c_3, c_4, c_6, c_8\}$

Optimal Hitting set problem

- Set of elements (i.e. constraints) $\{c_1, c_2, ..., c_n\}$
- Collection $\mathcal{H}$ of sets-to-hit (i.e. subsets of the set of elements)
- cost associated for every element:

$$w_1 = 3, w_2 = 5, w_3 = w_4 = ... = w_8 = 1$$

Goal Find an optimal hitting set such that every set-to-hit is hit by at least once an element of the hitting set with minimal total cost.

For example:

$$H_1 = \{c_3, c_5\}$$
$$H_2 = \{c_2, c_4, c_7\}$$
$$H_3 = \{c_2, c_6\}$$
$$H_4 = \{c_1, c_8\}$$

Optimal hitting sets:

- $\{c_3, c_4, c_6, c_8\}$ ✓
  - cost: 4

Optimal Hitting set problem

- Set of elements (i.e. constraints) $\{c_1, c_2, ..., c_n\}$
- Collection $\mathcal{H}$ of sets-to-hit (i.e. subsets of the set of elements)
- cost associated for every element:

$$w_1 = 3, w_2 = 5, w_3 = w_4 = ... = w_8 = 1$$

Goal Find an optimal hitting set such that every set-to-hit is hit by at least once an element of the hitting set with minimal total cost.

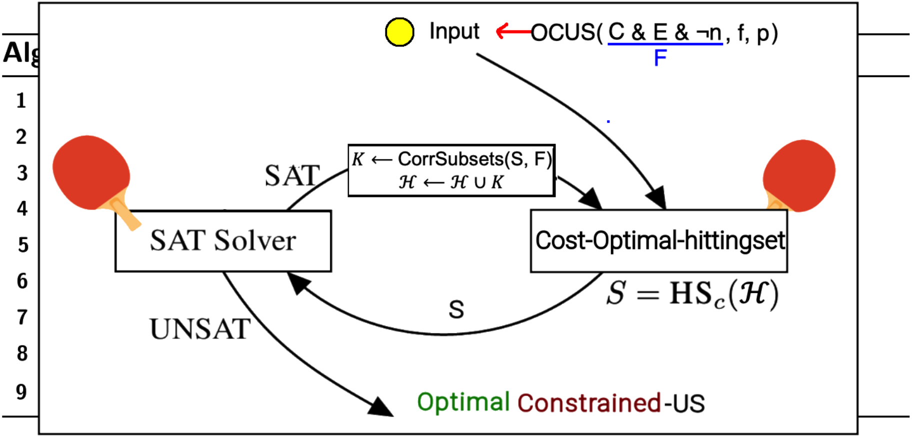
For example:

$$H_1 = \{c_3, c_5\}$$
$$H_2 = \{c_2, c_4, c_7\}$$
$$H_3 = \{c_2, c_6\}$$
$$H_4 = \{c_1, c_8\}$$

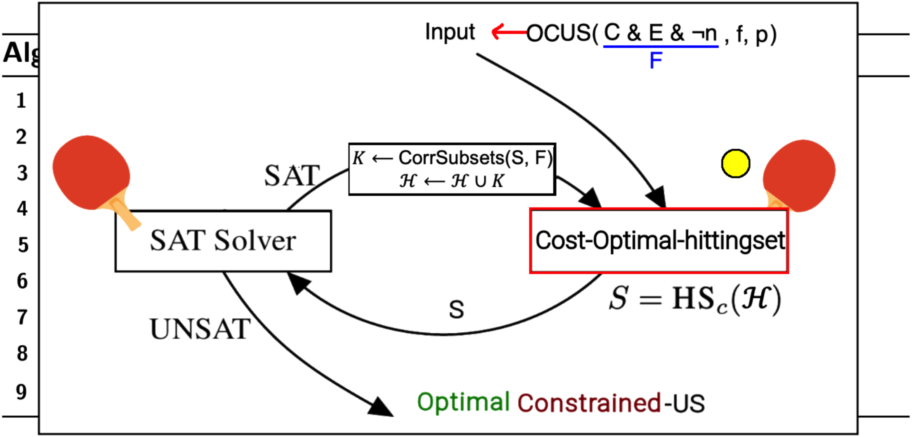
Optimal hitting sets:

- $\{c_3, c_4, c_6, c_8\}$
- $\{c_1, c_2, c_3\}$
  - cost: 7
- ...

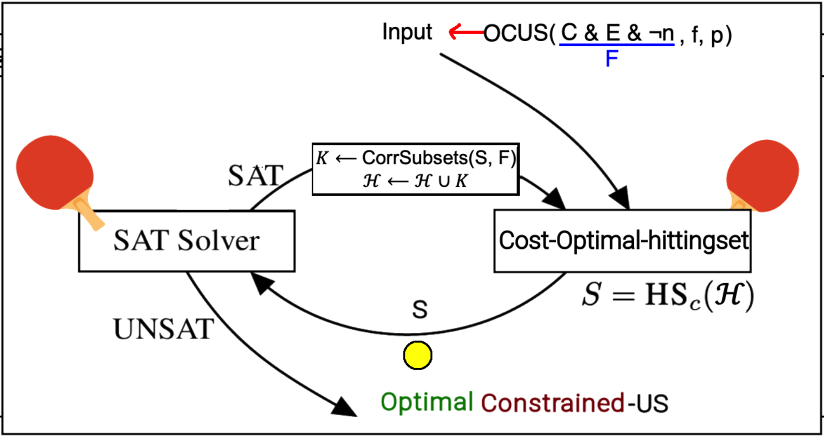
OCUS: An efficient game of Ping-Pong



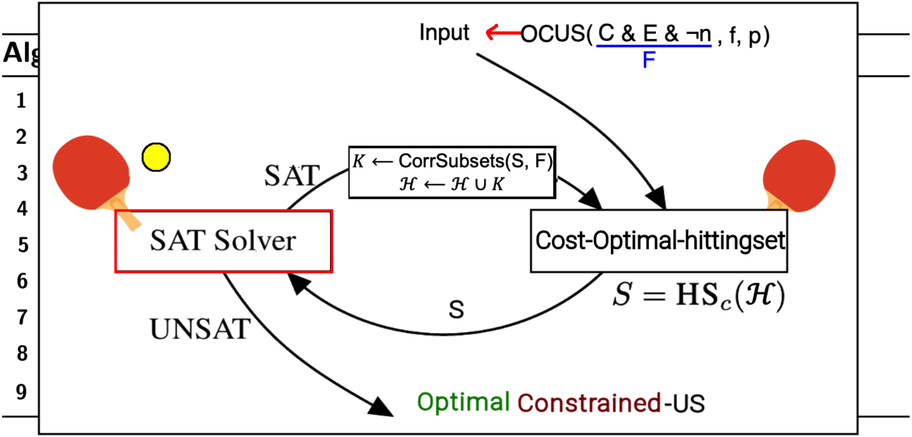
OCUS: An efficient game of Ping-Pong



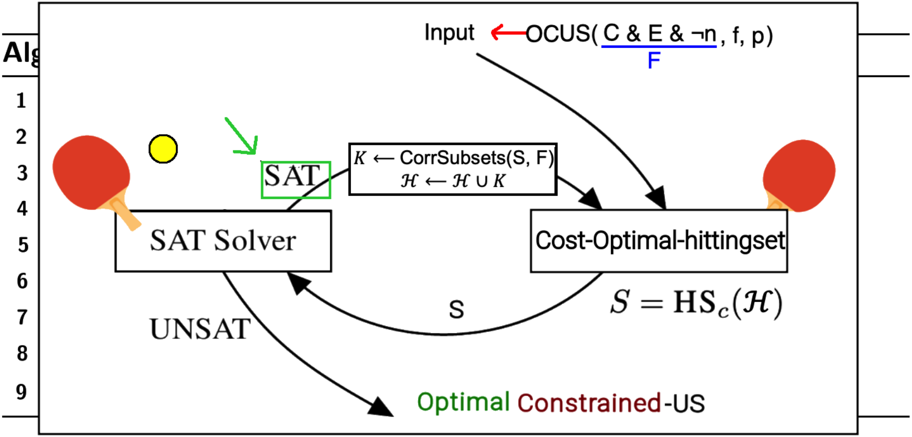
OCUS: An efficient game of Ping-Pong



OCUS: An efficient game of Ping-Pong

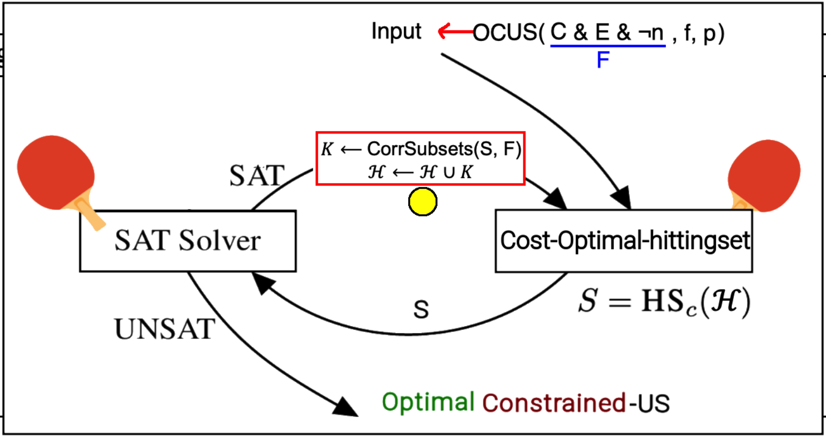


OCUS: An efficient game of Ping-Pong

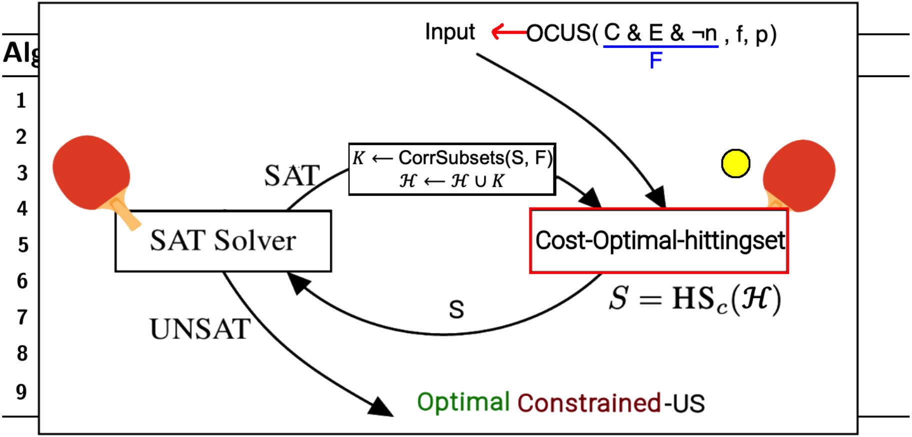


OCUS: An efficient game of Ping-Pong

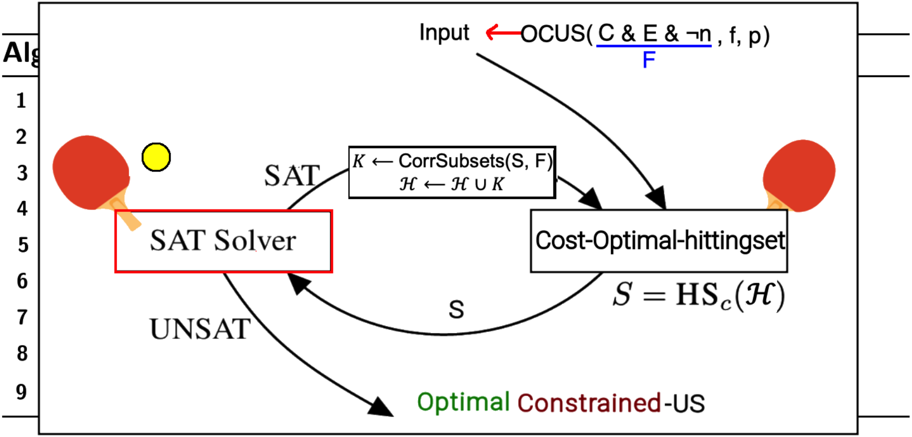
Alg



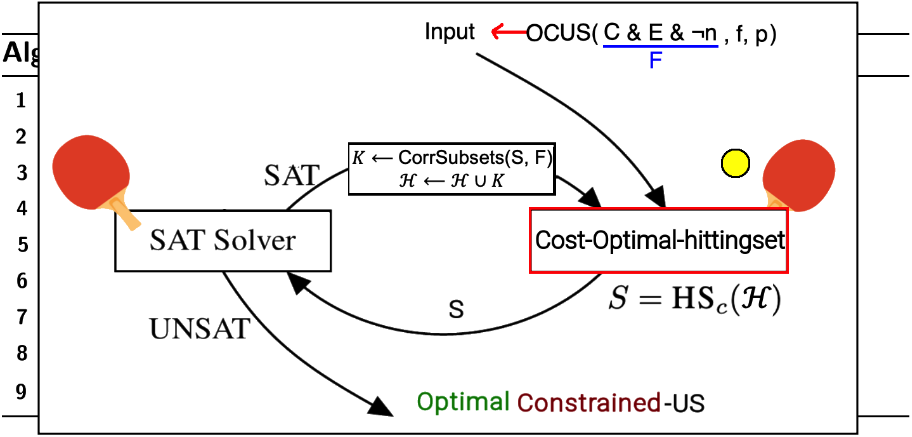
OCUS: An efficient game of Ping-Pong



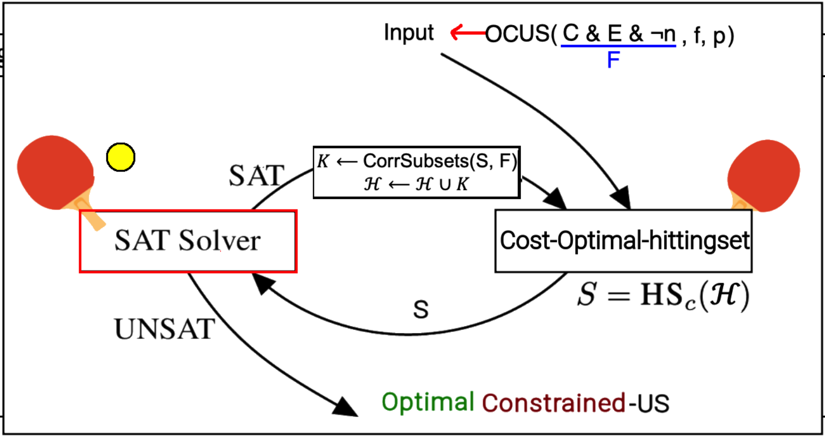
OCUS: An efficient game of Ping-Pong



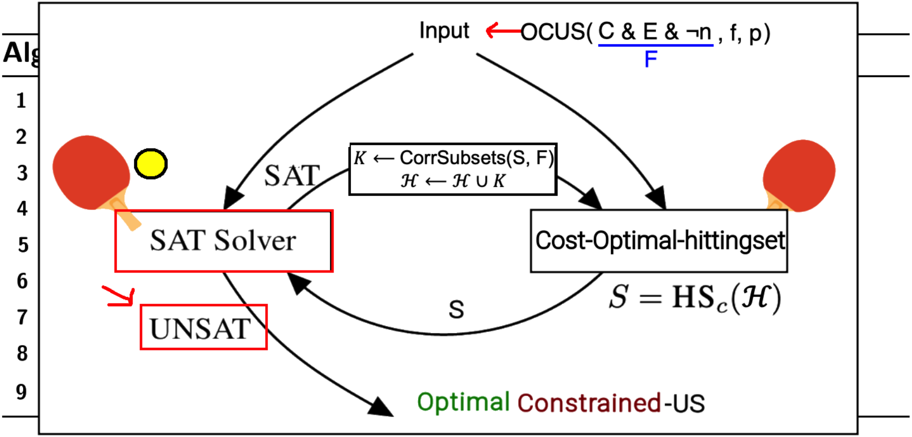
OCUS: An efficient game of Ping-Pong



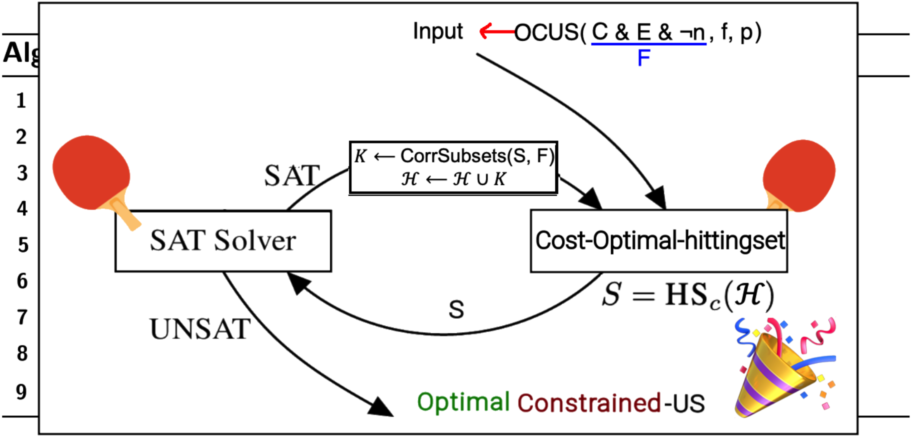
OCUS: An efficient game of Ping-Pong



OCUS: An efficient game of Ping-Pong

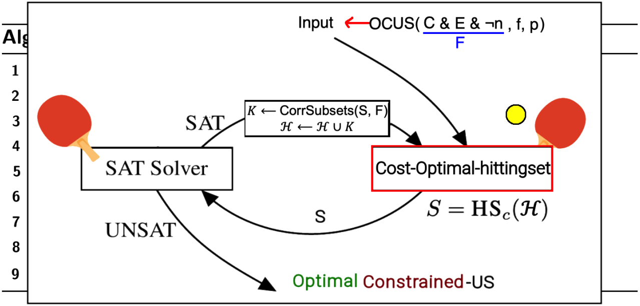


OCUS: An efficient game of Ping-Pong

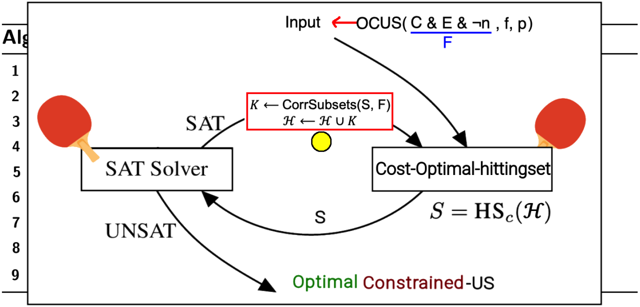


https://flyclipart.com/party-popper-emoji-on-apple-ios-celebration-emoji-png-610281

How do I efficiently compute OCUSs?



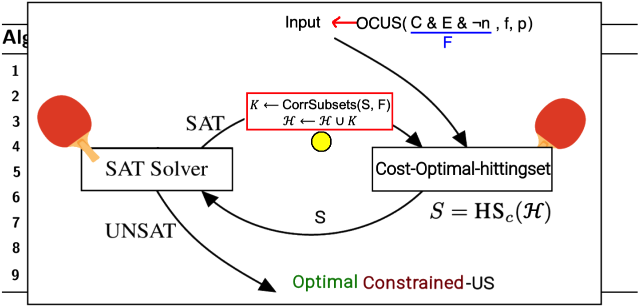
How do I efficiently compute OCUSs?



CORRECTION-SUBSETS$(\mathcal{S}, \mathcal{F})$

- $\{\mathcal{F} \setminus \mathcal{S}\}$
- $\{\mathcal{F} \setminus \text{GROW}(\mathcal{S}, \mathcal{F})\}$

How do I efficiently compute OCUSs?

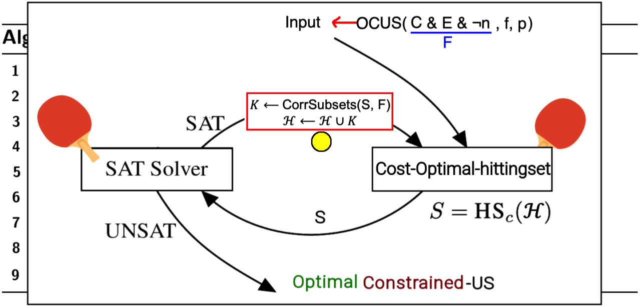


CORRECTION-SUBSETS$(\mathcal{S}, \mathcal{F})$

- $\{\mathcal{F} \setminus \mathcal{S}\}$
- $\{\mathcal{F} \setminus \text{GROW}(\mathcal{S}, \mathcal{F})\}$

  $\rightarrow$ *efficient* vs *qualitative* grow ?

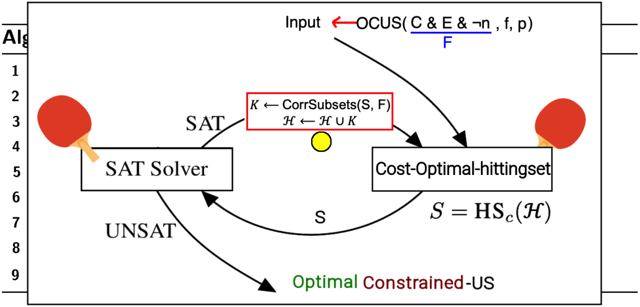
How do I efficiently compute OCUSs?



Correction-Subsets($\mathcal{S}, \mathcal{F}$)

- $\{\mathcal{F} \setminus \mathcal{S}\}$
- $\{\mathcal{F} \setminus \text{Grow}(\mathcal{S}, \mathcal{F})\}$
  - → *efficient* vs *qualitative* grow ?
  - → MaxSAT vs SAT-based heuristic

How do I efficiently compute OCUSs?



CORRECTION-SUBSETS($\mathcal{S}, \mathcal{F}$)

- $\{\mathcal{F} \setminus \mathcal{S}\}$
- $\{\mathcal{F} \setminus \text{GROW}(\mathcal{S}, \mathcal{F})\}$
  - $\rightarrow$ *efficient* vs *qualitative* grow ?
  - $\rightarrow$ MaxSAT vs SAT-based heuristic
- Multiple *disjoint* correction subsets ?

Improving efficiency of the OCUS algorithm..

...in the context of explanation sequence generation

Can we improve OCUS-calls in the context of explanation sequence generation?

Improving efficiency of the OCUS algorithm..

...in the context of explanation sequence generation

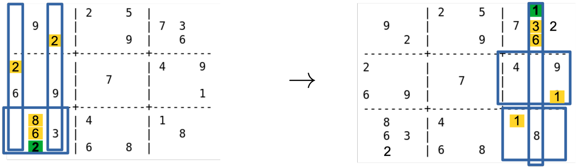
Incrementality

- Can we re-use of information between explanation calls ?

Improving efficiency of our OCUS algorithm..

...in the context of explanation sequence generation

OCUS(C & E &¬ ***n*** , f, p)

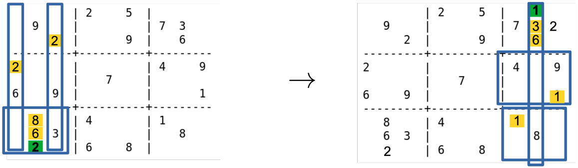


C Model constraints

Improving efficiency of our OCUS algorithm..

...in the context of explanation sequence generation

OCUS(C & E &¬ n , f, p)



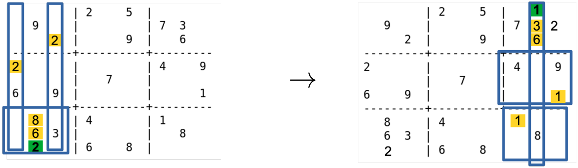
C Model constraints

- Do not change from an explanation step to another

Improving efficiency of our OCUS algorithm..

...in the context of explanation sequence generation

OCUS(C & E &¬ *n* , f, p)

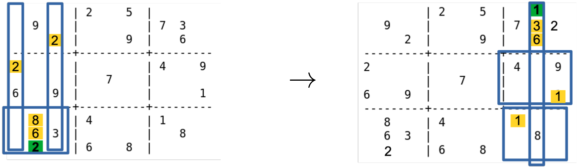


- C Model constraints
  - Do not change from an explanation step to another
- E Derived facts E

Improving efficiency of our OCUS algorithm..

...in the context of explanation sequence generation

$$\textbf{OCUS}(\ \boxed{C}\ \&\ E\ \&\neg\ \boldsymbol{n}\ , f, p)$$



C Model constraints

- Do not change from an explanation step to another

E Derived facts E

- Precision-increasing !

Incrementality

Kick-start OCUS by bootstrapping $\mathcal{H}$

- Keep track of collection of correction sets $\mathcal{H}$ that need to be hit

Incrementality

Kick-start OCUS by bootstrapping $\mathcal{H}$

- Keep track of collection of correction sets $\mathcal{H}$ that need to be hit
- Each set H in $\mathcal{H}$ is the complement (with respect to the formula at hand) of a satisfiable subset

Incrementality

Kick-start OCUS by bootstrapping $\mathcal{H}$

- Keep track of collection of correction sets $\mathcal{H}$ that need to be hit
- Each set H in $\mathcal{H}$ is the complement (with respect to the formula at hand) of a satisfiable subset
  - $\rightarrow$ Keep track of set of Satisfiable Subsets **SSs**

Incrementality

Kick-start OCUS by bootstrapping $\mathcal{H}$

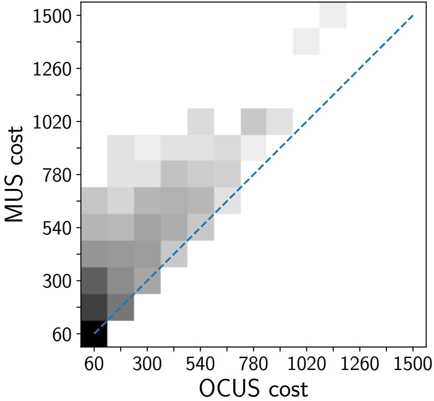
- ▷ Keep track of collection of correction sets $\mathcal{H}$ that need to be hit
- ▷ Each set H in $\mathcal{H}$ is the complement (with respect to the formula at hand) of a satisfiable subset
  - → Keep track of set of Satisfiable Subsets **SSs**
  - → Bootstrap $\mathcal{H} \leftarrow \{\mathcal{F} \setminus \mathcal{S} | \mathcal{S} \in \textbf{SSs}\}$

In practice!

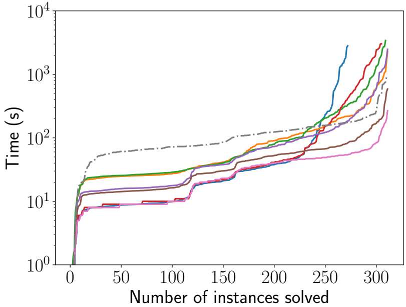
- Incremental OCUS works with the full unsatisfiable formula of step 0
  - $S \wedge E_{end} \wedge \{\neg n | n \in E_{end} \setminus E_0\}$
- Initialize hitting set solver once and modify objective at every explanation step i such that:
  - Underived literal cannot be taken
  - Negated literals already explained cannot selected

Results - Explanation quality

MUS
↓
OCUS



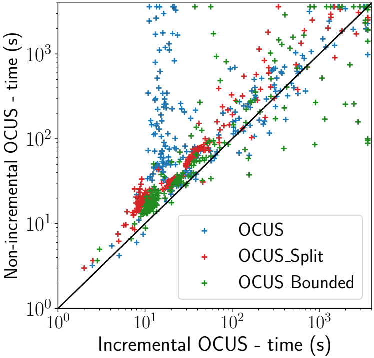
Results - Correction Subset Enumeration



Algorithm: $\textsc{ocus}(\mathcal{F}, f, p)$

$\mathcal{H} \leftarrow \emptyset$
while true **do**
  $\mathcal{S} \leftarrow \textsc{CondOptHittingSet}(\mathcal{H}, f, p)$
  if $\neg\mathrm{SAT}(\mathcal{S})$ **then**
    return $\mathcal{S}$
  end
  $\mathcal{H} \leftarrow$
  $\mathcal{H} \cup \textsc{Correction-Subsets}(\mathcal{S}, \mathcal{F})$
end

Results - Incrementality



Algorithm: $\text{OCUS}(\mathcal{F}, f, p)$

$\mathcal{H} \leftarrow \ldots$

while true **do**

  $\mathcal{S} \leftarrow \text{CondOptHittingSet}(\mathcal{H}, f, p)$

  if $\neg\text{SAT}(\mathcal{S})$ **then**

    return $\mathcal{S}$

  end

  $\mathcal{H} \leftarrow \mathcal{H} \cup \text{Correction-Subsets}(\mathcal{S}, \mathcal{F})$

end

Conclusion

We introduce Optimal Constrained Unsatisfiable Subsets (OCUS) problem and solved it using the implicit hitting set duality between MCSes and MUSes.

Optimality Cost-function quantifies explanation difficulty.

Conclusion

We introduce Optimal Constrained Unsatisfiable Subsets (OCUS) problem and solved it using the implicit hitting set duality between MCSes and MUSes.

Optimality
: Cost-function quantifies explanation difficulty.

Constrainedness
: Impose structural constraints on the form of the explanations.

Conclusion

We introduce Optimal Constrained Unsatisfiable Subsets (OCUS) problem and solved it using the implicit hitting set duality between MCSes and MUSes.

Optimality Cost-function quantifies explanation difficulty.

Constrainedness Impose structural constraints on the form of the explanations.

Incrementality Reuse-information between successive explanation calls.

What can OCUS do for YOU ?

What can OCUS do for YOU ?

- Explaining scheduling, configuration problems .... and puzzles 😎
- Debugging *unsatisfiable models* with preferences on the constraints

What can OCUS do for YOU ?

- Explaining scheduling, configuration problems .... and puzzles 😎
- Debugging *unsatisfiable models* with preferences on the constraints
- Stepwise explaining unsatisfiability

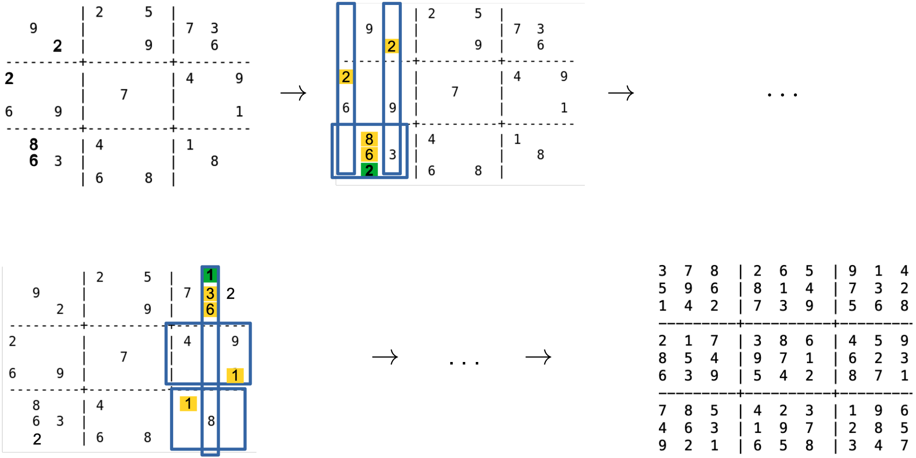
What can OCUS do for YOU ?

- Explaining scheduling, configuration problems .... and puzzles 😎
- Debugging *unsatisfiable models* with preferences on the constraints
- Stepwise explaining unsatisfiability
- Explaining Optimality for Constraint Optimization Problems
  - Why is the objective value not better ?

Future work

- What is a good cost-function to quantify how difficult an explanation is ? (from humans)
- Explaining optimization (different types of “why” queries); close relation to Explainable AI Planning Fox *et al.* [2017]
- Scaling up (approximate algorithms; decomposition of explanation search)

Future work


3 7 8 | 2 6 5 | 9 1 4
5 9 6 | 8 1 4 | 7 3 2
1 4 2 | 7 3 9 | 5 6 8
2 1 7 | 3 8 6 | 4 5 9
8 5 4 | 9 7 1 | 6 2 3
6 3 9 | 5 4 2 | 8 7 1
7 8 5 | 4 2 3 | 1 9 6
4 6 3 | 1 9 7 | 2 8 5
9 2 1 | 6 5 8 | 3 4 7

This is joint work with ...

Emilio GAMBA
emilio.gamba@vub.be

ARTIFICIAL INTELLIGENCE RESEARCH GROUP

References I

Bart Bogaerts, Emilio Gamba, Jens Claes, and Tias Guns. Step-wise explanations of constraint satisfaction problems. In *24th European Conference on Artificial Intelligence (ECAI)*, 2020.

Jessica Davies and Fahiem Bacchus. Exploiting the power of mip solvers in maxsat. In *International conference on theory and applications of satisfiability testing*, pages 166–181. Springer, 2013.

Maria Fox, Derek Long, and Daniele Magazzeni. Explainable planning. *arXiv preprint arXiv:1709.10256*, 2017.

Tias Guns. Increasing modeling language convenience with a universal n-dimensional array, cppy as python-embedded example. In *Proceedings of the 18th workshop on Constraint Modelling and Reformulation at CP (Modref 2019)*, volume 19, 2019.

Pierre Hansen and Brigitte Jaumard. Algorithms for the maximum satisfiability problem. *Computing*, 44(4):279–303, 1990.

References II

Alexey Ignatiev, Alessandro Previti, Mark Liffiton, and Joao Marques Silva. Smallest mus extraction with minimal hitting set dualization. In *International Conference on Principles and Practice of Constraint Programming*, pages 173–182, 2015.

Alexey Ignatiev, Antonio Morgado, and Joao Marques-Silva. Rc2: an efficient maxsat solver. *Journal on Satisfiability, Boolean Modeling and Computation*, 11(1):53–64, 2019.

Mark H Liffiton and Ammar Malik. Enumerating infeasibility: Finding multiple muses quickly. In *International Conference on AI and OR Techniques in Constriant Programming for Combinatorial Optimization Problems*, pages 160–175. Springer, 2013.

Mark H. Liffiton and Karem A. Sakallah. Algorithms for computing minimal unsatisfiable subsets of constraints. *J. Autom. Reasoning*, 40(1):1–33, 2008.

Raymond Reiter. A theory of diagnosis from first principles. *AIJ*, 32(1):57–95, 1987.