SAT vs. CP

Mohamed Siala
https://siala.github.io

INSA-Toulouse & LAAS-CNRS

January 18, 2022

Back to Constraint Programming

Back to Constraint Programming

- A constraint is a finite relation (i.e., subset of a Cartesian product)

Back to Constraint Programming

- A constraint is a finite relation (i.e., subset of a Cartesian product)
- A constraint can be expressed in extension (table constraint) or intention (expression)

Back to Constraint Programming

- A constraint is a finite relation (i.e., subset of a Cartesian product)
- A constraint can be expressed in extension (table constraint) or intention (expression)
- A constraint network is defined by a triplet $P = (X, D, C)$ where
    - X is a set of variables
    - D is a set of domains for the variables in X
    - C is a set of constraints

Back to Constraint Programming

- A constraint is a finite relation (i.e., subset of a Cartesian product)
- A constraint can be expressed in extension (table constraint) or intention (expression)
- A constraint network is defined by a triplet $P = (X, D, C)$ where
  - X is a set of variables
  - D is a set of domains for the variables in X
  - C is a set of constraints
- The constraint satisfaction problem (CSP) is the problem of deciding if a constraint network has a solution

Back to Constraint Programming

- A constraint is a finite relation (i.e., subset of a Cartesian product)
- A constraint can be expressed in extension (table constraint) or intention (expression)
- A constraint network is defined by a triplet $P = (X, D, C)$ where
  - X is a set of variables
  - D is a set of domains for the variables in X
  - C is a set of constraints
- The constraint satisfaction problem (CSP) is the problem of deciding if a constraint network has a solution
- Mostly solvable by backtracking algorithms (Search and Filtering)

Search

Search

Search

Search

- Search: decisions to explore the search tree

Search

Search

- Search: decisions to explore the search tree
- Search in CP= variable ordering + value ordering

Search

Search

- Search: decisions to explore the search tree
- Search in CP= variable ordering + value ordering
- Standard or customized

Search

Search

- Search: decisions to explore the search tree
- Search in CP= variable ordering + value ordering
- Standard or customized

Variable Ordering

'Fail-first' principle [Haralick and Elliott, 1980]:
"To succeed, try first where you are most likely to fail"

Search

Search

- Search: decisions to explore the search tree
- Search in CP= variable ordering + value ordering
- Standard or customized

Variable Ordering

'Fail-first' principle [Haralick and Elliott, 1980]:
"To succeed, try first where you are most likely to fail"

Value Ordering

'Succeed-first' [Geelen, 1992]:
"Follow the best chances leading to a solution"

Filtering

- Filtering (propagation/pruning): inferences based on the current state

Filtering

- Filtering (propagation/pruning): inferences based on the current state
- Constraint $\leftrightarrow$ a propagator

Filtering

- Filtering (propagation/pruning): inferences based on the current state
- Constraint $\leftrightarrow$ a propagator
- Propagators are executed sequentially before taking any decision

Filtering

- Filtering (propagation/pruning): inferences based on the current state
- Constraint $\leftrightarrow$ a propagator
- Propagators are executed sequentially before taking any decision
- The level of pruning $\leftrightarrow$ local consistency (for instance, bound consistency, arc consistency, etc)

Filtering

- Filtering (propagation/pruning): inferences based on the current state
- Constraint $\leftrightarrow$ a propagator
- Propagators are executed sequentially before taking any decision
- The level of pruning $\leftrightarrow$ local consistency (for instance, bound consistency, arc consistency, etc)

Arc Consistency

Let C be a constraint and D be a list of domains for the variables in the scope of C.

Filtering

- Filtering (propagation/pruning): inferences based on the current state
- Constraint $\leftrightarrow$ a propagator
- Propagators are executed sequentially before taking any decision
- The level of pruning $\leftrightarrow$ local consistency (for instance, bound consistency, arc consistency, etc)

Arc Consistency

Let C be a constraint and D be a list of domains for the variables in the scope of C.
C is Arc Consistent (AC) iff for every variable x in the scope of C, for every value $v \in D(x)$, there exists an assignment w in D satisfying C in which v is assigned to x

Filtering algorithm

Filtering algorithm

- A Filtering algorithm associated to a constraint C takes as input a list of domains (for the variables in the scope of C) and returns a list of domains that are smaller or identical to the original domains.

Filtering algorithm

- A Filtering algorithm associated to a constraint C takes as input a list of domains (for the variables in the scope of C) and returns a list of domains that are smaller or identical to the original domains.
- For a filtering algorithm to be correct: no consistent value should be removed (by consistent we mean belongs to a satisfying assignment).

Filtering algorithm

- A Filtering algorithm associated to a constraint C takes as input a list of domains (for the variables in the scope of C) and returns a list of domains that are smaller or identical to the original domains.
- For a filtering algorithm to be correct: no consistent value should be removed (by consistent we mean belongs to a satisfying assignment).
- If all the domains are singleton, the propagator must be able to check if the assignment corresponds to a solution or not.

CP vs. SAT

CP vs. SAT

- CP: rich modelling language, powerful filtering, dedicated search strategies

CP vs. SAT

- CP: rich modelling language, powerful filtering, dedicated search strategies
- SAT: simple input format, clause learning and backjumping, autonomous search

CP vs. SAT

- CP: rich modelling language, powerful filtering, dedicated search strategies
- SAT: simple input format, clause learning and backjumping, autonomous search

- Every CSP can be encoded into SAT

CP vs. SAT

- CP: rich modelling language, powerful filtering, dedicated search strategies
- SAT: simple input format, clause learning and backjumping, autonomous search

- Every CSP can be encoded into SAT
- When should we encode to SAT, when shouldn't we?

CP vs. SAT

- CP: rich modelling language, powerful filtering, dedicated search strategies
- SAT: simple input format, clause learning and backjumping, autonomous search

- Every CSP can be encoded into SAT
- When should we encode to SAT, when shouldn't we?
- CP vs. SAT: a fundamental difference is the presence of global reasoning win CP.

CP vs. SAT : To decompose or nor to decompose?

CP vs. SAT : To decompose or nor to decompose?

- Decomposition is the task of reformulating a (global) constraint into smaller and simpler constraints.

CP vs. SAT : To decompose or nor to decompose?

- Decomposition is the task of reformulating a (global) constraint into smaller and simpler constraints.
- Take the example of AllDifferent: it can be decomposed into simple binary inequalities. **Remember the tutorial!**.

CP vs. SAT : To decompose or nor to decompose?

- Decomposition is the task of reformulating a (global) constraint into smaller and simpler constraints.
- Take the example of AllDifferent: it can be decomposed into simple binary inequalities. **Remember the tutorial!**.
- **In general, decomposition makes the filtering weaker. We loose all the powerful filtering from the global constraints by decomposing.**
- On the one hand, by decomposing into clauses, we loose the powerful filtering from CP

CP vs. SAT : To decompose or nor to decompose?

- Decomposition is the task of reformulating a (global) constraint into smaller and simpler constraints.
- Take the example of AllDifferent: it can be decomposed into simple binary inequalities. **Remember the tutorial!**.
- **In general, decomposition makes the filtering weaker. We loose all the powerful filtering from the global constraints by decomposing.**
- On the one hand, by decomposing into clauses, we loose the powerful filtering from CP
- Also the size of the encoding matters. An exponential encoding is better avoided!

CP vs. SAT : To decompose or nor to decompose?

- Decomposition is the task of reformulating a (global) constraint into smaller and simpler constraints.
- Take the example of AllDifferent: it can be decomposed into simple binary inequalities. **Remember the tutorial!**.
- **In general, decomposition makes the filtering weaker. We loose all the powerful filtering from the global constraints by decomposing.**
- On the one hand, by decomposing into clauses, we loose the powerful filtering from CP
- Also the size of the encoding matters. An exponential encoding is better avoided!
- On the other hand, clause learning in SAT is quite powerful to learn new clauses and to backjump in the search tree

CP vs. SAT : To decompose or nor to decompose?

- Decomposition is the task of reformulating a (global) constraint into smaller and simpler constraints.
- Take the example of AllDifferent: it can be decomposed into simple binary inequalities. **Remember the tutorial!**.
- **In general, decomposition makes the filtering weaker. We loose all the powerful filtering from the global constraints by decomposing.**
- On the one hand, by decomposing into clauses, we loose the powerful filtering from CP
- Also the size of the encoding matters. An exponential encoding is better avoided!
- On the other hand, clause learning in SAT is quite powerful to learn new clauses and to backjump in the search tree
- **Can we find something that takes advantages from both worlds?**

CP vs. SAT : To decompose or nor to decompose?

- Decomposition is the task of reformulating a (global) constraint into smaller and simpler constraints.
- Take the example of AllDifferent: it can be decomposed into simple binary inequalities. **Remember the tutorial!**.
- **In general, decomposition makes the filtering weaker. We loose all the powerful filtering from the global constraints by decomposing.**
- On the one hand, by decomposing into clauses, we loose the powerful filtering from CP
- Also the size of the encoding matters. An exponential encoding is better avoided!
- On the other hand, clause learning in SAT is quite powerful to learn new clauses and to backjump in the search tree
- **Can we find something that takes advantages from both worlds?** $\rightarrow$ **Clause learning in CP**

Modern Constraint Solvers: Hybrid CP/SAT

Modern Constraint Solvers: Hybrid CP/SAT

- Learning from conflict

Modern Constraint Solvers: Hybrid CP/SAT

- Learning from conflict
- Based on the notion of explanation

Modern Constraint Solvers: Hybrid CP/SAT

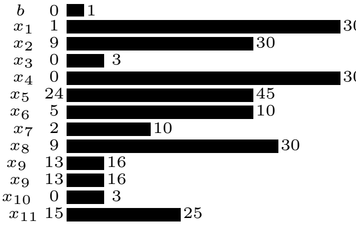
- Learning from conflict
- Based on the notion of explanation
- Generalized Nogoods[Katsirelos and Bacchus, 2005], Lazy Clause generation [Ohrimenko et al., 2009], Clause Learning in sequencing and scheduling problems [Siala, 2015], ...

Modern Constraint Solvers: Hybrid CP/SAT

- Learning from conflict
- Based on the notion of explanation
- Generalized Nogoods[Katsirelos and Bacchus, 2005], Lazy Clause generation [Ohrimenko et al., 2009], Clause Learning in sequencing and scheduling problems [Siala, 2015], ...

Learning in CP

$x_1 + x_7 \geq 4 \wedge$
$x_2 + x_{10} \geq 11 \wedge$
$x_3 + x_9 = 16 \wedge$
$x_5 \geq x_8 + x_9 \wedge$
$b \leftrightarrow (x_9 - x_4 = 14) \wedge$
$b \rightarrow (x_6 \geq 7) \wedge$
$b \rightarrow (x_6 + x_7 \leq 9) \wedge$
$x_{11} \geq x_9 + x_{10}$



Learning in CP

$$
\begin{aligned}
&x_1 + x_7 \geq 4 \wedge \\
&x_2 + x_{10} \geq 11 \wedge \\
&x_3 + x_9 = 16 \wedge \\
&x_5 \geq x_8 + x_9 \wedge \\
&b \leftrightarrow (x_9 - x_4 = 14) \wedge \\
&b \rightarrow (x_6 \geq 7) \wedge \\
&b \rightarrow (x_6 + x_7 \leq 9) \wedge \\
&x_{11} \geq x_9 + x_{10}
\end{aligned}
$$

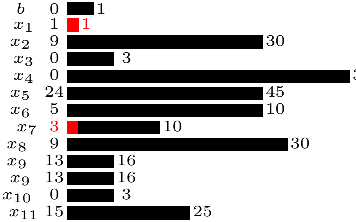
$[\![x_1 = 1]\!]$

b	0	1
x_1	1	1
x_2	9	30
x_3	0	3
x_4	0	[illegible]
x_5	24	45
x_6	5	10
x_7	2	10
x_8	9	30
x_9	13	16
x_9	13	16
x_{10}	0	3
x_{11}	15	25

Learning in CP

$[\![x_1 = 1]\!] \rightarrow [\![x_7 \geq 3]\!]$

$x_1 + x_7 \geq 4 \wedge$
$x_2 + x_{10} \geq 11 \wedge$
$x_3 + x_9 = 16 \wedge$
$x_5 \geq x_8 + x_9 \wedge$
$b \leftrightarrow (x_9 - x_4 = 14) \wedge$
$b \rightarrow (x_6 \geq 7) \wedge$
$b \rightarrow (x_6 + x_7 \leq 9) \wedge$
$x_{11} \geq x_9 + x_{10}$

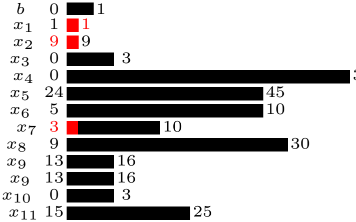


Learning in CP

$$
\begin{aligned}
&x_1 + x_7 \geq 4 \wedge \\
&x_2 + x_{10} \geq 11 \wedge \\
&x_3 + x_9 = 16 \wedge \\
&x_5 \geq x_8 + x_9 \wedge \\
&b \leftrightarrow (x_9 - x_4 = 14) \wedge \\
&b \rightarrow (x_6 \geq 7) \wedge \\
&b \rightarrow (x_6 + x_7 \leq 9) \wedge \\
&x_{11} \geq x_9 + x_{10}
\end{aligned}
$$

$[\![x_1 = 1]\!] \rightarrow [\![x_7 \geq 3]\!]$

$[\![x_2 = 9]\!]$

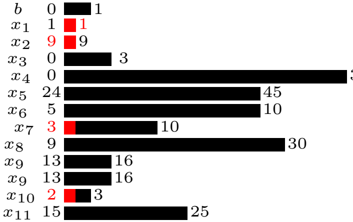


Learning in CP

$x_1 + x_7 \geq 4 \wedge$
$x_2 + x_{10} \geq 11 \wedge$
$x_3 + x_9 = 16 \wedge$
$x_5 \geq x_8 + x_9 \wedge$
$b \leftrightarrow (x_9 - x_4 = 14) \wedge$
$b \rightarrow (x_6 \geq 7) \wedge$
$b \rightarrow (x_6 + x_7 \leq 9) \wedge$
$x_{11} \geq x_9 + x_{10}$

$[\![x_1 = 1]\!] \rightarrow [\![x_7 \geq 3]\!]$

$[\![x_2 = 9]\!] \rightarrow [\![x_{10} \geq 2]\!]$



Learning in CP

$[\![x_1 = 1]\!] \rightarrow [\![x_7 \geq 3]\!]$

$[\![x_2 = 9]\!] \rightarrow [\![x_{10} \geq 2]\!]$

$[\![x_3 = 2]\!]$

$$
\begin{aligned}
& x_1 + x_7 \geq 4 \wedge \\
& x_2 + x_{10} \geq 11 \wedge \\
& x_3 + x_9 = 16 \wedge \\
& x_5 \geq x_8 + x_9 \wedge \\
& b \leftrightarrow (x_9 - x_4 = 14) \wedge \\
& b \rightarrow (x_6 \geq 7) \wedge \\
& b \rightarrow (x_6 + x_7 \leq 9) \wedge \\
& x_{11} \geq x_9 + x_{10}
\end{aligned}
$$

b	0	1
x_1	1	1
x_2	9	9
x_3	2	2
x_4	0	3
x_5	24	45
x_6	5	10
x_7	3	10
x_8	9	30
x_9	13	16
x_9	13	16
x_{10}	2	3
x_{11}	15	25

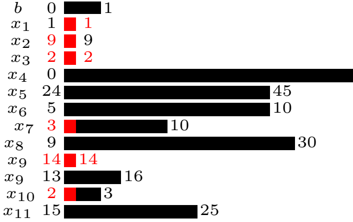
Learning in CP

$$\begin{aligned}
&x_1 + x_7 \geq 4 \wedge \\
&x_2 + x_{10} \geq 11 \wedge \\
&x_3 + x_9 = 16 \wedge \\
&x_5 \geq x_8 + x_9 \wedge \\
&b \leftrightarrow (x_9 - x_4 = 14) \wedge \\
&b \rightarrow (x_6 \geq 7) \wedge \\
&b \rightarrow (x_6 + x_7 \leq 9) \wedge \\
&x_{11} \geq x_9 + x_{10}
\end{aligned}$$

$[\![x_1 = 1]\!] \rightarrow [\![x_7 \geq 3]\!]$

$[\![x_2 = 9]\!] \rightarrow [\![x_{10} \geq 2]\!]$

$[\![x_3 = 2]\!] \rightarrow [\![x_9 = 14]\!]$



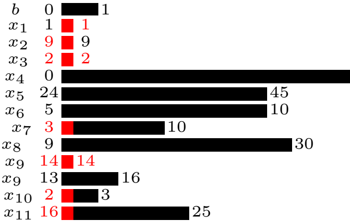
Learning in CP

$\llbracket x_1 = 1 \rrbracket \rightarrow \llbracket x_7 \geq 3 \rrbracket$

$\llbracket x_2 = 9 \rrbracket \rightarrow \llbracket x_{10} \geq 2 \rrbracket$

$\llbracket x_3 = 2 \rrbracket \rightarrow \llbracket x_9 = 14 \rrbracket \rightarrow \llbracket x_{11} \geq 16 \rrbracket$

$$
\begin{aligned}
& x_1 + x_7 \geq 4 \wedge \\
& x_2 + x_{10} \geq 11 \wedge \\
& x_3 + x_9 = 16 \wedge \\
& x_5 \geq x_8 + x_9 \wedge \\
& b \leftrightarrow (x_9 - x_4 = 14) \wedge \\
& b \rightarrow (x_6 \geq 7) \wedge \\
& b \rightarrow (x_6 + x_7 \leq 9) \wedge \\
& x_{11} \geq x_9 + x_{10}
\end{aligned}
$$



Learning in CP

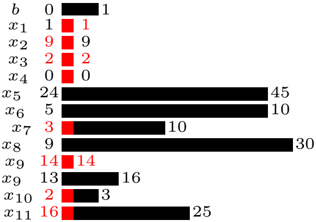
$[\![x_1 = 1]\!] \rightarrow [\![x_7 \geq 3]\!]$

$[\![x_2 = 9]\!] \rightarrow [\![x_{10} \geq 2]\!]$

$[\![x_3 = 2]\!] \rightarrow [\![x_9 = 14]\!] \rightarrow [\![x_{11} \geq 16]\!]$

$[\![x_4 = 0]\!]$

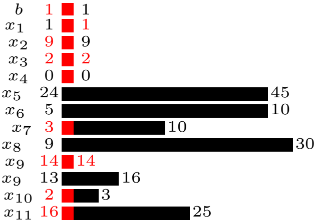
$x_1 + x_7 \geq 4 \wedge$
$x_2 + x_{10} \geq 11 \wedge$
$x_3 + x_9 = 16 \wedge$
$x_5 \geq x_8 + x_9 \wedge$
$b \leftrightarrow (x_9 - x_4 = 14) \wedge$
$b \rightarrow (x_6 \geq 7) \wedge$
$b \rightarrow (x_6 + x_7 \leq 9) \wedge$
$x_{11} \geq x_9 + x_{10}$



Learning in CP

$$\begin{tikzcd}
\llbracket x_1 = 1 \rrbracket \arrow[r] & \llbracket x_7 \geq 3 \rrbracket & \\
\llbracket x_2 = 9 \rrbracket \arrow[r] & \llbracket x_{10} \geq 2 \rrbracket \arrow[dr] & \\
\llbracket x_3 = 2 \rrbracket \arrow[r] & \llbracket x_9 = 14 \rrbracket \arrow[r] \arrow[d] & \llbracket x_{11} \geq 16 \rrbracket \\
\llbracket x_4 = 0 \rrbracket \arrow[r] & \llbracket b = 1 \rrbracket &
\end{tikzcd}$$

$$\begin{aligned}
&x_1 + x_7 \geq 4 \wedge \\
&x_2 + x_{10} \geq 11 \wedge \\
&x_3 + x_9 = 16 \wedge \\
&x_5 \geq x_8 + x_9 \wedge \\
&b \leftrightarrow (x_9 - x_4 = 14) \wedge \\
&b \rightarrow (x_6 \geq 7) \wedge \\
&b \rightarrow (x_6 + x_7 \leq 9) \wedge \\
&x_{11} \geq x_9 + x_{10}
\end{aligned}$$



Learning in CP

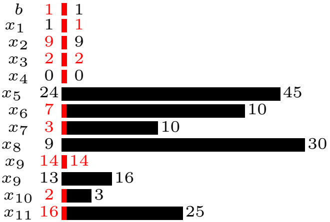
$[\![x_1 = 1]\!] \rightarrow [\![x_7 \geq 3]\!]$

$[\![x_2 = 9]\!] \rightarrow [\![x_{10} \geq 2]\!]$

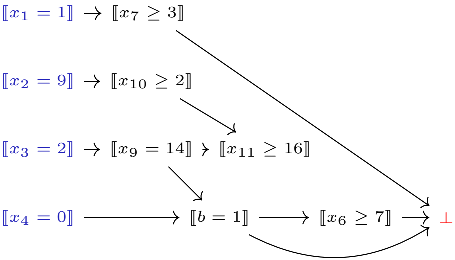
$[\![x_3 = 2]\!] \rightarrow [\![x_9 = 14]\!] \rightarrow [\![x_{11} \geq 16]\!]$

$[\![x_4 = 0]\!] \longrightarrow [\![b = 1]\!] \longrightarrow [\![x_6 \geq 7]\!]$

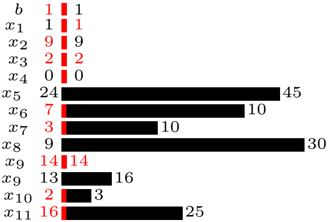
$x_1 + x_7 \geq 4 \wedge$
$x_2 + x_{10} \geq 11 \wedge$
$x_3 + x_9 = 16 \wedge$
$x_5 \geq x_8 + x_9 \wedge$
$b \leftrightarrow (x_9 - x_4 = 14) \wedge$
$b \rightarrow (x_6 \geq 7) \wedge$
$b \rightarrow (x_6 + x_7 \leq 9) \wedge$
$x_{11} \geq x_9 + x_{10}$



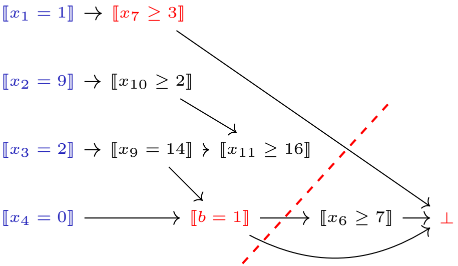
Learning in CP



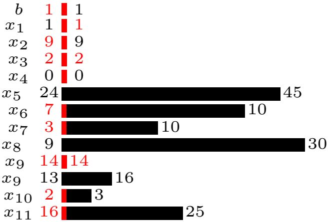
$x_1 + x_7 \geq 4 \wedge$
$x_2 + x_{10} \geq 11 \wedge$
$x_3 + x_9 = 16 \wedge$
$x_5 \geq x_8 + x_9 \wedge$
$b \leftrightarrow (x_9 - x_4 = 14) \wedge$
$b \to (x_6 \geq 7) \wedge$
$b \to (x_6 + x_7 \leq 9) \wedge$
$x_{11} \geq x_9 + x_{10}$



Learning in CP

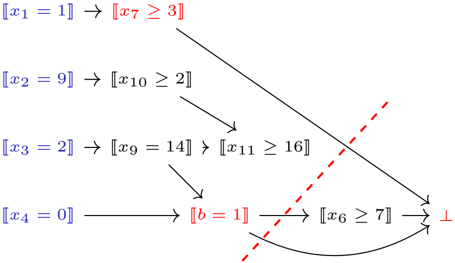


$x_1 + x_7 \geq 4 \wedge$
$x_2 + x_{10} \geq 11 \wedge$
$x_3 + x_9 = 16 \wedge$
$x_5 \geq x_8 + x_9 \wedge$
$b \leftrightarrow (x_9 - x_4 = 14) \wedge$
$b \to (x_6 \geq 7) \wedge$
$b \to (x_6 + x_7 \leq 9) \wedge$
$x_{11} \geq x_9 + x_{10}$

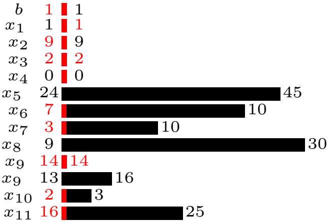


- Conflict analysis: $[\![b = 1]\!] \wedge [\![x_7 \geq 3]\!] \Rightarrow \bot$

Learning in CP



$x_1 + x_7 \geq 4 \wedge$
$x_2 + x_{10} \geq 11 \wedge$
$x_3 + x_9 = 16 \wedge$
$x_5 \geq x_8 + x_9 \wedge$
$b \leftrightarrow (x_9 - x_4 = 14) \wedge$
$b \rightarrow (x_6 \geq 7) \wedge$
$b \rightarrow (x_6 + x_7 \leq 9) \wedge$
$x_{11} \geq x_9 + x_{10}$



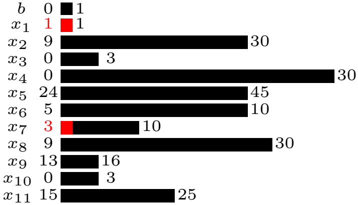
- Conflict analysis: $[\![b = 1]\!] \wedge [\![x_7 \geq 3]\!] \Rightarrow \perp$
- New clause: $[\![b \neq 1]\!] \vee [\![x_7 \leq 2]\!]$

Learning in CP

$x_1 + x_7 \geq 4 \wedge$
$x_2 + x_{10} \geq 11 \wedge$
$x_3 + x_9 = 16 \wedge$
$x_5 \geq x_8 + x_9 \wedge$
$b \leftrightarrow (x_9 - x_4 = 14) \wedge$
$b \rightarrow (x_6 \geq 7) \wedge$
$b \rightarrow (x_6 + x_7 \leq 9) \wedge$
$x_{11} \geq x_9 + x_{10}$

$[\![x_1 = 1]\!] \rightarrow [\![x_7 \geq 3]\!]$

- Conflict analysis: $[\![b = 1]\!] \wedge [\![x_7 \geq 3]\!] \Rightarrow \perp$
- New clause: $[\![b \neq 1]\!] \vee [\![x_7 \leq 2]\!]$
- Backtrack to level 1

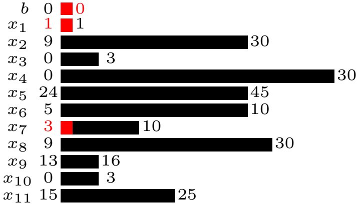


Learning in CP

$[\![x_1 = 1]\!] \rightarrow [\![x_7 \geq 3]\!] \longrightarrow [\![b = 0]\!]$

- Conflict analysis: $[\![b = 1]\!] \wedge [\![x_7 \geq 3]\!] \Rightarrow \bot$
- New clause: $[\![b \neq 1]\!] \vee [\![x_7 \leq 2]\!]$
- Backtrack to level 1
- Propagate the learnt clause

$$
\begin{aligned}
& x_1 + x_7 \geq 4 \wedge \\
& x_2 + x_{10} \geq 11 \wedge \\
& x_3 + x_9 = 16 \wedge \\
& x_5 \geq x_8 + x_9 \wedge \\
& b \leftrightarrow (x_9 - x_4 = 14) \wedge \\
& b \rightarrow (x_6 \geq 7) \wedge \\
& b \rightarrow (x_6 + x_7 \leq 9) \wedge \\
& x_{11} \geq x_9 + x_{10}
\end{aligned}
$$

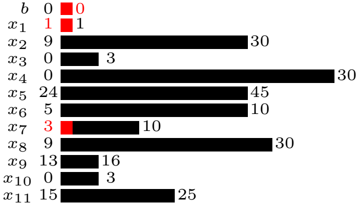


Learning in CP

$[\![x_1 = 1]\!] \rightarrow \; [\![x_7 \geq 3]\!] \longrightarrow [\![b = 0]\!]$

- Conflict analysis: $[\![b = 1]\!] \wedge [\![x_7 \geq 3]\!] \Rightarrow \perp$
- New clause: $[\![b \neq 1]\!] \vee [\![x_7 \leq 2]\!]$
- Backtrack to level 1
- Propagate the learnt clause
- Continue exploration

$$
\begin{aligned}
&x_1 + x_7 \geq 4 \wedge \\
&x_2 + x_{10} \geq 11 \wedge \\
&x_3 + x_9 = 16 \wedge \\
&x_5 \geq x_8 + x_9 \wedge \\
&b \leftrightarrow (x_9 - x_4 = 14) \wedge \\
&b \rightarrow (x_6 \geq 7) \wedge \\
&b \rightarrow (x_6 + x_7 \leq 9) \wedge \\
&x_{11} \geq x_9 + x_{10}
\end{aligned}
$$



Conflict analysis

Algorithm 1: 1-UIP-with-Propagators

$\mathit{\Psi} \leftarrow explain(\perp)$;
while $|\{q \in \mathit{\Psi} \mid level(q) = current\ level\}| > 1$ **do**
  $p \leftarrow \arg\max_q(\{rank(q) \mid level(q) = current\ level\ \wedge\ q \in \mathit{\Psi}\})$;
  $\mathit{\Psi} \leftarrow \mathit{\Psi} \cup \{q \mid q \in explain(p) \wedge level(q) > 0\} \setminus \{p\}$;
return $\mathit{\Psi}$;

Explaining constraints

Explaining constraints

- To enable clause learning in CP, every propagator must be able to explain their filtering in the form of clauses ("Lazy Clause Generation").

Explaining constraints

- To enable clause learning in CP, every propagator must be able to explain their filtering in the form of clauses ("Lazy Clause Generation").
- We distinguish two types of explanations:

Explaining constraints

- To enable clause learning in CP, every propagator must be able to explain their filtering in the form of clauses ("Lazy Clause Generation").
- We distinguish two types of explanations:
  - Explaining Failure
  - Explaining Domain filtering

Explaining constraints

- To enable clause learning in CP, every propagator must be able to explain their filtering in the form of clauses ("Lazy Clause Generation").
- We distinguish two types of explanations:
  - Explaining Failure
  - Explaining Domain filtering
- Example: Explain the constraint $X \leq Y$ with two scenarios (failure and propagation).

Exercise

- Let $(x_1, \ldots, x_n)$ be a sequence of Boolean variables, and let d be a positive integer.
- The CARDINALITY$(x_1, \ldots, x_n, d)$ constraint holds iff exactly d variables from the sequence $(x_1, \ldots, x_n)$ are true.
- Write a filtering algorithm for CARDINALITY.
- What is the time complexity?
- Does it enforce arc consistency?
- Explain the CARDINALITY filtering.

Correction

Algorithm 4: CARDINALITY($[x_1, \ldots, x_n], d$)

if $\left|\{x_j \mid \mathcal{D}(x_j) = \{1\}\}\right| > d$ **then**
    $\mathcal{D} \leftarrow \perp$;
if $\left|\{x_j \mid \mathcal{D}(x_j) = \{0\}\}\right| > n - d$ **then**
    $\mathcal{D} \leftarrow \perp$;
if $\left|\{x_j \mid \mathcal{D}(x_j) = \{1\}\}\right| = d$ **then**
    foreach $i \in \{1..n\}$ **do**
        if $\mathcal{D}(x_i) = \{0, 1\}$ **then**
            $\mathcal{D}(x_i) \leftarrow \{0\}$;
else
    if $\left|\{x_j \mid \mathcal{D}(x_j) = \{0\}\}\right| = n - d$ **then**
        foreach $i \in \{1..n\}$ **do**
            if $\mathcal{D}(x_i) = \{0, 1\}$ **then**
                $\mathcal{D}(x_i) \leftarrow \{1\}$;
return $\mathcal{D}$;

Explaining The Cardinality Constraint

Explaining The Cardinality Constraint

- Failure 1:

$$x^1 \wedge x^2 \wedge x^{d+1} \rightarrow \perp$$

Where $D(x^i) = \{1\}$

Explaining The Cardinality Constraint

- Failure 1:

$$x^1 \wedge x^2 \wedge x^{d+1} \rightarrow \perp$$

  Where $D(x^i) = \{1\}$

- Failure 2:

$$\neg x^1 \wedge \neg x^2 \wedge \neg x^{n-d+1} \rightarrow \perp$$

  Where $D(x^i) = \{0\}$

- Explaining the propagating the value 1: the conjunction of all the assigned variables

Explaining The Cardinality Constraint

- Failure 1:

$$x^1 \wedge x^2 \wedge x^{d+1} \rightarrow \perp$$

Where $D(x^i) = \{1\}$

- Failure 2:

$$\neg x^1 \wedge \neg x^2 \wedge \neg x^{n-d+1} \rightarrow \perp$$

Where $D(x^i) = \{0\}$

- Explaining the propagating the value 1: the conjunction of all the assigned variables
- Explaining the propagating the value 0: the conjunction of all the assigned variables

Take Away Message

Take Away Message

- SAT, CP, MIP, (also, MaxSAT, SMT, QBF, ASP, Pseudo-Boolean) are efficient tools to solve hard combinatorial problems

Take Away Message

- SAT, CP, MIP, (also, MaxSAT, SMT, QBF, ASP, Pseudo-Boolean) are efficient tools to solve hard combinatorial problems
- When you master one or few techniques, it opens the door to work on diverse problems. The more you apply to different problems, the more you learn

Take Away Message

- SAT, CP, MIP, (also, MaxSAT, SMT, QBF, ASP, Pseudo-Boolean) are efficient tools to solve hard combinatorial problems
- When you master one or few techniques, it opens the door to work on diverse problems. The more you apply to different problems, the more you learn
- The choice depends on the problem at hand (is it easy to linearise? what is the size of the SAT encoding? Can we use/invent global constraints?, etc)

Take Away Message

- SAT, CP, MIP, (also, MaxSAT, SMT, QBF, ASP, Pseudo-Boolean) are efficient tools to solve hard combinatorial problems
- When you master one or few techniques, it opens the door to work on diverse problems. The more you apply to different problems, the more you learn
- The choice depends on the problem at hand (is it easy to linearise? what is the size of the SAT encoding? Can we use/invent global constraints?, etc)
- You don't need to implement a solver: use existing ones! check the different solver competitions

Take Away Message

- SAT, CP, MIP, (also, MaxSAT, SMT, QBF, ASP, Pseudo-Boolean) are efficient tools to solve hard combinatorial problems
- When you master one or few techniques, it opens the door to work on diverse problems. The more you apply to different problems, the more you learn
- The choice depends on the problem at hand (is it easy to linearise? what is the size of the SAT encoding? Can we use/invent global constraints?, etc)
- You don't need to implement a solver: use existing ones! check the different solver competitions
- Hybrid approaches are the future: take advantage of diverse methodologies

Take Away Message

- SAT, CP, MIP, (also, MaxSAT, SMT, QBF, ASP, Pseudo-Boolean) are efficient tools to solve hard combinatorial problems
- When you master one or few techniques, it opens the door to work on diverse problems. The more you apply to different problems, the more you learn
- The choice depends on the problem at hand (is it easy to linearise? what is the size of the SAT encoding? Can we use/invent global constraints?, etc)
- You don't need to implement a solver: use existing ones! check the different solver competitions
- Hybrid approaches are the future: take advantage of diverse methodologies

References I

Katsirelos, G. and Bacchus, F. (2005).
Generalized NoGoods in CSPs.
In *Proceedings of the 20th National Conference on Artificial Intelligence, AAAI'05, and the 17th Conference on Innovative Applications of Artificial Intelligence, IAAI'05, Pittsburgh, Pennsylvania, USA*, pages 390–396.

Ohrimenko, O., Stuckey, P. J., and Codish, M. (2009).
Propagation via Lazy Clause Generation.
Constraints, 14(3):357–391.

Siala, M. (2015).
Search, propagation, and learning in sequencing and scheduling problems. (Recherche, propagation et apprentissage dans les problèmes de séquencement et d'ordonnancement).
PhD thesis, INSA Toulouse, France.