

SAT vs. CP

Mohamed Siala
<https://siala.github.io>

INSA-Toulouse & LAAS-CNRS

January 16, 2022

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: 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
- 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

- 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: 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 in CP.

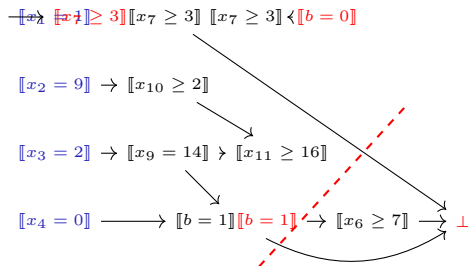
CP vs. SAT : To decompose or not to decompose?

- 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? → Clause learning in CP**

Modern Constraint Solvers: Hybrid CP/SAT

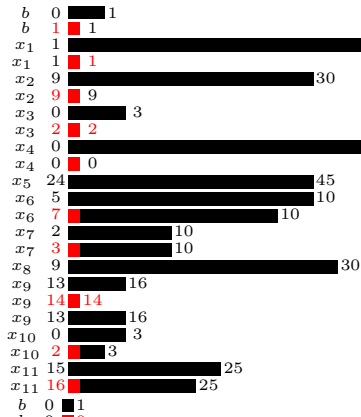
- Learning from conflict
- Based on the notion of explanation
- Generalized Nogoods[Katsirelos and Bacchus, 2005], Lazy Clause generation [Ohrimenko et al., 2009], Clause Learning in sequencing and scheduling problems [Siala, 2015], ...

Learning in CP



- Conflict analysis: $\llbracket b = 1 \rrbracket \wedge \llbracket x_7 \geq 3 \rrbracket \Rightarrow \perp$
- New clause: $\llbracket b \neq 1 \rrbracket \vee \llbracket x_7 \leq 2 \rrbracket$
- Backtrack to level 1
- Propagate the learnt clause

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



Conflict analysis

Algorithm 1: 1-UIP-with-Propagators

```

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

```

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, \dots, x_n) be a sequence of Boolean variables, and let d be a positive integer.
- The $\text{CARDINALITY}(x_1, \dots, x_n, d)$ constraint holds iff exactly d variables from the sequence (x_1, \dots, 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($[x_1, \dots, x_n], d$)

```

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

```

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

- 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 séquençement et d'ordonnancement).
PhD thesis, INSA Toulouse, France.