Combining forces to solve Combinatorial Problems, a preliminary approach

Mohamed Siala, Emmanuel Hebrard, and Christian Artigues

Tarbes, France
Outline

Context

Background

SAT-Solving with Global Constraints

The AtMostSeqCard Constraint

Experiments

Conclusion & Future work
Combinatorial Problems

Context

- Finite domain variables
- a fixed number of constraints over these variables
Combinatorial Problems

Context

- Finite domain variables
- A fixed number of constraints over these variables
- Is there a solution satisfying these constraints?
Combinatorial Problems

Context
- Finite domain variables
- A fixed number of constraints over these variables
- Is there a solution satisfying these constraints?

Combinatorial Problems
- The size of the search tree is exponential!
- There is no known algorithm for solving them in polynomial time
- NP-Complete/NP-Hard Problems
A constraint satisfaction problem (CSP) is a triplet $P = (\mathcal{X}, \mathcal{D}, \mathcal{C})$ where

- $\mathcal{X}$ is a set of variables.
- $\mathcal{D}$ is the related sets of values.
- $\mathcal{C}$ is a set of constraints.

A solution of a CSP is an assignment $w$ satisfying all the constraints.
A constraint satisfaction problem (CSP) is a triplet $P = (\mathcal{X}, \mathcal{D}, \mathcal{C})$ where

- $\mathcal{X}$ is a set of variables.
- $\mathcal{D}$ is the related sets of values.
- $\mathcal{C}$ is a set of constraints.

A solution of a CSP is an assignment $w$ satisfying all the constraints.

**Example**

- $X = \langle x, y \rangle$
- $D = \langle \{1, 2, 3\}, \{4, 5\} \rangle$
- $C_1 = \{x \text{ is even}\}$
- $C_2 = \{x + y = 6\}$
A propagator (or filtering algorithm) aims to remove some values that are inconsistent.

**Correctness & Checking**

**Figure:** Propagation impact
Global constraints

- A global constraint is constraint over $n$ variables.
- A global constraint captures a sub-problem.
- A global constraint can be used to solve different problems.
- A global constraint $\leftrightarrow$ specific propagator.

Propagation & Global Constraints?

<table>
<thead>
<tr>
<th>Decomposition</th>
<th>Global Constraint</th>
</tr>
</thead>
<tbody>
<tr>
<td>$C_1 : X \neq Y; C_2 : Y \neq Z; C_3 : Z \neq X;$</td>
<td>AllDifferent($X, Y, Z$)</td>
</tr>
</tbody>
</table>

**AllDifferent($X, Y, Z$)**

$X, Y, Z, D_X = D_Y = D_Z = \{1, 2\}$

<table>
<thead>
<tr>
<th>Decomposition</th>
<th>Global Constraint</th>
</tr>
</thead>
</table>
| Propagate($C_1$) :  
  $C_1 : X \neq Y$  
  $D_X = D_Y \{1, 2\}$  
  $\rightarrow$ No propagation!  
Propagate($C_2$), Propagate($C_3$) : No propagation | $D_X = D_Y = D_Z = \{1, 2\}$  
$\rightarrow$ Failure! |
Learning in CP

- a, b, c, d integer variables pairwise different.
- $D(a) = \{1, 2, 3, 4\}$, $D(b) = \{1, 2, 3\}$, $D(c) = \{1, 2, 3\}$, $D(d) = \{1, 2, 3\}$
- $x_1, \ldots, x_n$ n variables and $C_1, \ldots, C_m$ m Constraints over these variables
- suppose that we branch on $a, x_1 \ldots x_n, b, c, d$
Learning in CP

- \( a, b, c, d \) integer variables pairwise different.
- \( D(a) = \{1, 2, 3, 4\}, \ D(b) = \{1, 2, 3\}, \ D(c) = \{1, 2, 3\}, \ D(d) = \{1, 2, 3\} \)
- \( x_1, \ldots x_n \) n variables and \( C_1, \ldots C_m \) m Constraints over these variables
- suppose that we branch on \( a, x_1 \ldots x_n, b, c, d \)

With a standard CP-Solver
Learning in CP

With learning:

- Conflict analyse $\rightarrow [a \leftarrow 3]$ is a no good!
- Backjump to the latest assignment in $[a \leftarrow 3]$
- Learn $[\neg (a=3)]$
Boolean Satisfiability (SAT)

A Sat-Problem
- Boolean variables
- CNF: a set of clauses (i.e. a set of disjunctions over these variables and their negations).
- For instance: \( C \equiv (a \lor b) \land (\neg c \lor d \lor \neg e) \)

Why SAT?
1. There is a community working on SAT-Problems!
2. Modern SAT-Solvers are able to deal with millions of variables and clauses
Suppose now that we want to solve:

\[ \phi \equiv ((x + y) = 32) \lor (a > 17) \land ((w^3 + y = 0.53) \lor p_1 \lor \neg p_2) \]
Suppose now that we want to solve:

\[ \phi \equiv ((x + y) = 32) \lor (a > 17) \land ((w^3 + y = 0.53) \lor p_1 \lor \neg p_2) \]

⇒ It looks like a CNF but . . .
Suppose now that we want to solve:

\[ \phi \equiv ((x + y) = 32) \lor (a > 17)) \land ((w^3 + y = 0.53) \lor p_1 \lor \neg p_2) \]

\[ \Rightarrow \text{It looks like a CNF but . . .} \]

\[ \Rightarrow \text{Satisfiability} \]
Suppose now that we want to solve:
\[ \phi \equiv ((x + y) = 32) \lor (a > 17)) \land ((w^3 + y = 0.53) \lor p_1 \lor \neg p_2) \]
⇒ It looks like a CNF but . . .
⇒ Satisfiability Modulo Theories
Suppose now that we want to solve:
\[ \phi \equiv ((x + y) = 32) \lor (a > 17) \land ((w^3 + y = 0.53) \lor p_1 \lor \neg p_2) \]
⇒ It looks like a CNF but . . .
⇒ Satisfiability Modulo Theories
⇒ First order formulas w.r.t some theories
Suppose now that we want to solve:
\[ \phi \equiv ((x + y) = 32) \lor (a > 17) \land ((w^3 + y = 0.53) \lor p_1 \lor \neg p_2) \]
⇒ It looks like a CNF but . . .
⇒ Satisfiability Modulo Theories
⇒ First order formulas w.r.t some theories

**Lazy SMT**

1. Exploiting SAT by abstracting the formula
2. Theory Propagation
3. Theory explanations for conflicts and propagation
Towards a hybrid solver

SAT-Solver → Search → Propagation → Explanations → Conflict Analyse → SAT-Solver
Definition

\[ \text{AtMostSeqCard}(u, q, d, [x_1, \ldots, x_n]) \iff \]

\[ \bigwedge_{i=0}^{n-q} \left( \sum_{l=1}^{q} x_{i+l} \leq u \right) \wedge \left( \sum_{i=1}^{n} x_i = d \right) \]
**Definition**

\[ \text{AtMostSeqCard}(u, q, d, [x_1, \ldots, x_n]) \iff \]

\[ \bigwedge_{i=0}^{n-q} (\sum_{l=1}^{q} x_{i+l} \leq u) \land \left( \sum_{i=1}^{n} x_i = d \right) \]

**Example** \text{AtMostSeqCard}(2, 4, 4, [x_1, \ldots, x_7])

<table>
<thead>
<tr>
<th>0</th>
<th>1</th>
<th>1</th>
<th>0</th>
<th>1</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>1</th>
<th>1</th>
<th>0</th>
<th>0</th>
<th>1</th>
<th>0</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Mohamed SIALA  
April 2013  
EDSYS Congress
Filtering the Domains

- **Leftmost count from left to right:**
  - $L[i]$
  - $L[n] < ub$
  - $L[n] > ub$
  - $L[n] = ub$

- **Leftmost count from right to left:**
  - $R[i]$
  - $L[i] + R[n - i + 1] \leq ub$
  - $L[i - 1] + R[n - i] < ub$

- **Fail**
- **Nothing to do**

**Equations:**

- $L[i]$
- $R[i]$
- $D(x_i) = \{0\}$
- $D(x_i) = \{1\}$
Explaining the \texttt{AtMostSeqCard} constraint

**Key idea**

Let $S^*$ be a sequence defined as $\forall i \in [1, n]$, the domain of $x_i$ in $S^*$ (denoted by $D^*(x_i)$) is defined as follows:

$$D^*(x_i) = \begin{cases} 
\{0, 1\}, & \text{if } (D(x_i) = \{0\} \text{ and } \max_i = u) \\
\{0, 1\}, & \text{if } (D(x_i) = \{1\} \text{ and } \max_i \neq u) \\
D(x_i) & \text{otherwise}
\end{cases}$$

**Theorem**

*Let $L^*$ the result of leftmost\_max on $S^*$.
$\forall i \in [1, n], L^*[i] = L[i]$.***
Car-sequencing

Constraints

- Each class $c$ is associated with a demand $D_c$.
- For each option $j$, each sub-sequence of size $q_j$ must contain at most $u_j$ cars requiring the option $j$. 
### Easy Sat

<table>
<thead>
<tr>
<th></th>
<th># solved</th>
<th># TIME</th>
</tr>
</thead>
<tbody>
<tr>
<td>mcp</td>
<td>368 / 368 100%</td>
<td>0.17</td>
</tr>
<tr>
<td>hybrid</td>
<td>368 / 368 100%</td>
<td>0.14</td>
</tr>
<tr>
<td>hybridSwitch</td>
<td>368 / 368 100%</td>
<td>0.21</td>
</tr>
<tr>
<td>DefaultHybrid</td>
<td>368 / 368 100%</td>
<td>0.33</td>
</tr>
<tr>
<td>sate2</td>
<td>368 / 368 100%</td>
<td>3.15</td>
</tr>
<tr>
<td>sate3</td>
<td>368 / 368 100%</td>
<td>3.01</td>
</tr>
</tbody>
</table>

### Hard Sat

<table>
<thead>
<tr>
<th></th>
<th># solved</th>
<th># TIME</th>
</tr>
</thead>
<tbody>
<tr>
<td>mcp</td>
<td>35 / 35 100%</td>
<td>16.72</td>
</tr>
<tr>
<td>hybrid</td>
<td>34 / 35 97%</td>
<td>3.05</td>
</tr>
<tr>
<td>hybridSwitch</td>
<td>34 / 35 97%</td>
<td>2.66</td>
</tr>
<tr>
<td>DefaultHybrid</td>
<td>16 / 35 45%</td>
<td>287.84</td>
</tr>
<tr>
<td>sate2</td>
<td>28 / 35 80%</td>
<td>289.32</td>
</tr>
<tr>
<td>sate3</td>
<td>31 / 35 88%</td>
<td>60.99</td>
</tr>
</tbody>
</table>
### Unsat instances

<table>
<thead>
<tr>
<th></th>
<th># solved</th>
<th># TIME</th>
</tr>
</thead>
<tbody>
<tr>
<td>mcp</td>
<td>23 / 136 16%</td>
<td>300.55</td>
</tr>
<tr>
<td>hybrid</td>
<td>23 / 136 16%</td>
<td>300.55</td>
</tr>
<tr>
<td>hybridSwitch</td>
<td>36 / 136 26%</td>
<td>351.86</td>
</tr>
<tr>
<td>DefaultHybrid</td>
<td>35 / 136 25%</td>
<td>225.95</td>
</tr>
<tr>
<td>sate2</td>
<td>85 / 136 62%</td>
<td>92.45</td>
</tr>
<tr>
<td>sate3</td>
<td>66 / 136 48%</td>
<td>186.79</td>
</tr>
</tbody>
</table>
Current Contributions

- A linear time propagator for the \texttt{AtMostSeqCard} constraint
- Explaining the \texttt{AtMostSeqCard} constraint
- Getting started with the Hybrid solver

Future research

- Hybridisation & Hybridisation again . . .
- Treating other problems (scheduling) in a SAT-CP context
- MiniZinc Challenge with a hybrid Solver
- Incremental SAT-Encoding for Finite Domain variables
- . . .
Thank you!

Questions?