Optimization Constraints

Andrea Lodi
DEIS, University of Bologna
alodi@deis.unibo.it

Filippo Focacci   Michela Milano
ILOG France       DEIS Bologna
Outline

• Global Constraints for optimization problems:
  
  • preliminaries of Constraint Programming (CP);
  
  • global constraints in CP;
  
  • global constraints with an optimization component:
    
    • the cost-based domain filtering technique.
  
• The path constraint example.

• Computational results on TSP and TSPTW.
CP Preliminaries

• We consider CP on Finite Domains CP(FD) : a programming paradigm exploiting Constraint Satisfaction techniques

• A Constraint Satisfaction Problem (CSP) consists of:
  – a set of variables ($V_1, V_2, \ldots, V_n$)
  – a discrete domain ($D_1, D_2, \ldots, D_n$) for each variable
  – a set of constraints on those variables:
    “relations among variables which represent a subset of the Cartesian product of the domains $D_1 \times D_2 \times \ldots \times D_n$”

Solution of a CSP:
an assignment of values to variables consistent with the constraints

CP Preliminaries

Constraint Programming provides:

• A **modeling methodology** for stating decision variables, constraints, and **objective functions**

• A **programming language** for stating a **search** algorithm for finding values of the variables that satisfy the constraints and optimize the objective

• A **programming system** that includes:
  • Predefined constraints with powerful **filtering** algorithms for reducing the size of the search space
  • Functionality to allow definitions of new constraints and filtering algorithms
CP Preliminaries - example: Map Coloring

Aim: find an assignments of colors to zones s.t. no two adjacent zones are colored with the same color

- variables $V_1, V_2, V_3, V_4, V_5$: zones
- domains $D_1, D_2, D_3, D_4, D_5$: [red, blue, green, yellow, pink]
- constraints: $\text{near}(V_i, V_j) \Rightarrow V_i \neq V_j$
**CP Preliminaries – constraint graph**

- A CSP can be represented by a constraint graph:
  - variables ↔ nodes
  - constraints ↔ (hyper)-arcs

**Feasible Solution:**
- \( V_1 = \text{red} \)
- \( V_2 = \text{green} \)
- \( V_3 = \text{blue} \)
- \( V_4 = \text{yellow} \)
- \( V_5 = \text{pink} \)
enum Country {zone1, zone2, zone3, zone4, zone5};
enum Colors {red, blue, green, yellow, pink};
var Colors color[Country];
solve {
    color[zone1] <> color[zone2];
    color[zone1] <> color[zone3];
    color[zone1] <> color[zone4];
    color[zone1] <> color[zone5];
    color[zone2] <> color[zone3];
    color[zone2] <> color[zone4];
    color[zone2] <> color[zone5];
    color[zone3] <> color[zone4];
    color[zone4] <> color[zone5];
}
CP Preliminaries - consistency

• NODE CONSISTENCY
  – a network is node consistent if in each node each domain value is consistent with unary constraints

• ARC CONSISTENCY
  – a network is arc consistent if for each arc connecting variables $V_i$ and $V_j$ for each value in the domain of $V_i$ there exists a value in the domain of $V_j$ consistent with binary constraints

Not Node consistent
Not Arc consistent

Node consistent
Arc consistent
**CP Preliminaries – constraint propagation**

– Symbolic Constraint: example 1
  
  • `alldifferent([X_1,...X_n])`
    
    all variables must have different values

  Declaratively equivalent to a set of binary constraints
  
  `alldifferent([X_1,...X_n]) ↔ X_1 ≠ X_2, X_1 ≠ X_3,..., X_{n-1} ≠ X_n`

  Operationally more powerful constraints

– `x_1::[1,2,3], x_2::[1,2,3], x_3::[1,2,3], x_4::[1,2,3,4]`

– Arc consistency: **NO PROPAGATION**

– Filtering algorithm [Regin AAAI94]: values 1 2 3 removed from `x_4`
CP Preliminaries – constraint propagation

- Symbolic Constraint: example 1 (continues)
- \( x_1 : [1, 2, 3], x_2 : [1, 2, 3], x_3 : [1, 2, 3], x_4 : [1, 2, 3, 4], \)
- Filtering algorithm: values 1 2 3 removed from \( x_4 \)

Set of variables whose cardinality is 3 ranging on the same set of values whose cardinality is 3

\( x_4 : \{\alpha, \beta, \gamma, 4\} \)
Symbolic Constraint: example 2

- another propagation example used in the resource constraint is that based on the edge finding [Baptiste, Le Pape, Nuijten, IJCAI95]

Consider a unary resource and three activities:

<table>
<thead>
<tr>
<th>Activity</th>
<th>Start</th>
<th>Duration</th>
</tr>
</thead>
<tbody>
<tr>
<td>S₁</td>
<td>0</td>
<td>6</td>
</tr>
<tr>
<td>S₂</td>
<td>1</td>
<td>4</td>
</tr>
<tr>
<td>S₃</td>
<td>1</td>
<td>3</td>
</tr>
</tbody>
</table>
We can deduce that the earliest start time of $S_1$ is 8.
This is based on the fact that $S_1$ must be scheduled after $S_2$ and $S_3$.

**Global reasoning:**

suppose either $S_2$ or $S_3$ is scheduled after $S_1$. Then, the maximum of the completion times of $S_2$ and $S_3$ is at least 13 (out of the domain of $S_2$ and $S_3$).
In general each variable is involved in many constraints. Consequently, each change in variable domains as a result of propagation may result in further propagations on other variables.

Constraints agents view: during their lifetime constraints alternate their status between suspended and waking states (triggered by events).

Example:
**CP Preliminaries –**

*interaction among constraints*

- First propagation of \( X = Y + 1 \) yields to

\[
\begin{align*}
X & : [2..5], \\
Y & : [1..4], \\
Z & : [1..5]
\end{align*}
\]

\( X = Y + 1 \) is then suspended
CP Preliminaries – interaction among constraints

- Second propagation of $Y = Z + 1$ yields to

\[
\begin{align*}
X &:: [2..5], \\
Y &:: [2..4], \\
Z &:: [1..3]
\end{align*}
\]

$Y = Z + 1$ is then suspended

- The domain of $Y$ has changed, $X = Y + 1$ is then awakened

\[
\begin{align*}
X &:: [3..5], \\
Y &:: [2..4], \\
Z &:: [1..3]
\end{align*}
\]

$X = Y + 1$ is then suspended
Third propagation of $x = z - 1$ yields to

\[\begin{array}{c}
x::[] \\ y::[2..4] \\ z::[1..3]
\end{array}\]  

FAILURE detected

The order in which constraints are considered (delayed and awakened) does not affect the propagation results, BUT can affect the performance of the propagation algorithm.
Branching strategies define the way of partitioning the problem $P$ into easier subproblems $P_1, P_2, \ldots, P_n$.

To each subproblem: apply again propagation.

New branches can be pruned because of the new information derived from the branching.

Pruning all the infeasible values from variable domains has the same complexity of solving the original problem.

Propagation algorithms are then incomplete, i.e., when propagation is stopped at a fix point still some values left in the variable domains can be inconsistent: a search step is executed.

---

**CP Preliminaries – search**

- Pruning all the infeasible values from variable domains has the same complexity of solving the original problem.

- Propagation algorithms are then **incomplete**, i.e., when propagation is stopped at a fix point still some values left in the variable domains can be inconsistent: a **search** step is executed.
CP Preliminaries – search

variable selection

value selection

constraint propagation

const 1

const 2

const 3

continue on success

backtrack on failure
what about the objective function?

As soon as a feasible solution of value $Z^*$ is found, a new constraint (called *bounding constraint*) is added imposing that further solutions must have a better value: $Z < Z^*$;

The propagation of the bounding constraint is in general very weak: $Z$ is in general the sum of the value assumed by many variables.

Hence, CP solves a sequence of *feasibility* problems, constrained to improve the objective function value.
Global Constraints

• Global Constraints:
  • capture sub-problems that frequently constitute a sub-structure of more general problems;
  • include \textit{propagation} algorithms which perform pruning on domain variables on the basis of \textit{feasibility reasoning}.

• Global Constraints for optimization problems:
  • we need a pruning based on \textit{optimality reasoning};
  • we embed an \textit{optimization component}, i.e., a software component which solves to optimality a relaxation of the problem represented by the global constraint;
  • the relaxation depends on the \textit{objective function}. 
Global Constraints for Optimization Problems

• The optimization component is typically based on effective OR algorithms, thus a mapping between CP variables and the OR model is needed.

• The optimization component must provide:
  • $LB$: the optimal solution value of the relaxation;
  • $x^*$: the optimal solution of the relaxation in the OR model;
  • $\text{grad}(X,v)$: a gradient function estimating the additional cost of variable-value assignments.
Global Constraints for Optimization Problems
Optimization Constraints (1)

• **LowerBound**-based propagation:
  from **LB** towards objective function $Z::[Z_{\text{min}}..Z_{\text{max}}]$: $\text{LB} < Z_{\text{max}}$

• **cost**-based propagation:
  from the gradient function towards decision variables:

  for each $X_i::[v_1,v_2,\ldots,v_m]$ and $v_j$ there is a gradient function $\text{grad}(X_i,v_j)$ measuring the cost to pay if $X_i = v_j$

  \[
  \text{if } \text{LB} + \text{grad}(X_i,v_j) \geq Z_{\text{max}} \text{ then } X_i \neq v_j
  \]

  which is the classic OR **variable fixing**.
Optimization Constraints (2)

• Finally, the optimal solution of the relaxation in the OR model may help, through the mapping, to guide the search.

• The simplest example of gradient function are the linear programming *reduced costs* which can be computed for some special cases by combinatorial algorithms.

• We consider in the following:
  the **Assignment Problem** (AP) as a relaxation of the path constraint, and the **Hungarian Algorithm** as (combinatorial) optimization component.
Path constraint: an optimization component (1)

Given a directed graph $G=(V,A)$ with $|V| = n$, and associated to each node $i$ a variable $X_i$ whose domain contains the next possible nodes in a path, the CP path constraint:

$$X_0::D_0, X_1::D_1,..., X_k::D_k$$

$$\text{path([}X_0,X_1,...,X_k\text{])}$$

holds iff the assignment of variables $X_0,X_1,...,X_k$ defines a simple path involving all nodes $0,...,k$. 


If a cost is associated to each arc, and we want to model the Asymmetric Travelling Salesman Problem (ATSP), we can use the path constraint as follows:

- one of the node, say 0, is duplicated generating node n;
- node n reaches only node 0 with zero cost, while it is reached from each node (but 0) with the same cost paid to reach node 0;
- the constraint \( \text{path([X_0,X_1,\ldots,X_n])} \) is imposed.

AP can then be used as optimization component for \( \text{path()} \).
Path constraint: an optimization component (3)

AP is the following Integer Programming problem:

\[
\begin{align*}
\text{min } Z &= \sum_{i \in V} \sum_{j \in V} c_{ij} x_{ij} \\
\sum_{i \in V} x_{ij} &= 1 \quad \forall j \in V \quad \text{(A)} \\
\sum_{j \in V} x_{ij} &= 1 \quad \forall i \in V \quad \text{(B)} \\
x_{ij} &\geq 0 \text{ and integer } \forall i,j \in V = \{0,\ldots,n-1\}
\end{align*}
\]
CP- Model:
\[ X_i :: [v_1, v_2, ..., v_n] \] \( i=0...n-1 \)
\[ \text{path}([X_0, X_1, ..., X_n]) \]
\[ C_i :: [c_{i1}, c_{i2}, ..., c_{in}] \] \( i=0...n-1 \)
\[ C_n = 0; \ X_n = 0; \]
\[ C_0 + ... + C_{n-1} = Z \]
\[ \text{minimize}(Z) \]

IP- Model
\[
\begin{align*}
\min Z &= \sum_{i \in V} \sum_{j \in V} c_{ij} x_{ij} \\
\sum_{i \in V} x_{ij} &= 1 \quad j \in V \quad \text{(A)} \\
\sum_{j \in V} x_{ij} &= 1 \quad i \in V \quad \text{(B)} \\
\sum_{i \in S} \sum_{j \in V \setminus S} x_{ij} &\geq 1 \quad S \subset V \quad S \neq \emptyset \\
x_{ij} &\geq 0 \quad \text{and integer}
\end{align*}
\]

Mapping

\[ X_i = v_j \]
\[ \text{path}([X_0, X_1, ..., X_n]) \]
\[ C_i = c_{ij} \]

relaxed in

\[ x_{ij} = 1 \quad \text{(A)} + (B) \]
\[ c_{ij} \text{ iff } x_{ij} = 1 \]
Path constraint: an optimization component (4)

• The connectivity constraints are relaxed, and by the Hungarian algorithm we obtain a lower bound value $Z_{AP}$, an integer solution $x^*$, and the reduced costs. In addition the Hungarian algorithm is incremental ($O(n^3)$ first solution, $O(n^2)$ each re-computation).

• However, the bound could be very poor, mainly for pure problems as TSP, and a classical OR method for improving it is cutting planes generation.

• The simplest cutting planes are the Subtour Elimination Constraints (SECs) whose separation is polynomially solvable.
Cutting planes in global constraints

- The cut generator is again a black-box in the global constraints, but the optimization component is now a general LP solver (the AP structure is lost), whereas the cost-based propagation remains unchanged.
Lagrangean relaxation of cuts

- The drawback of using a general LP solver (not incremental, not integer solution) can be partially overcome by dualizing in Lagrangean fashion the generated cuts.

\[ \min c^T x + \lambda (\alpha x - \alpha_0) \]
\[ x(\delta + (i)) = 1 \]
\[ x(\delta - (i)) = 1 \]
Lagrangian multipliers

• Algorithm:
  • optimally solve the original structured relaxation \( \rightarrow \text{LB}_{\text{AP}} \);
  • repeat
    • generate violated cuts;
    • add cuts to the current formulation;
    • solve the corresponding LP;
  • until a given point (e.g., the end of the root node) \( \rightarrow \text{LB}_r \);
  • extract the dual values associated to tight cuts:
    they are the optimal Lagrangean multipliers of the cuts;
  • dualize tight cuts and update the cost matrix;
  • solve the structured relaxation \( \rightarrow \text{LB}_{\text{APm}} \).
Through duality theory: \( \text{LB}_{\text{AP}} \leq \text{LB}_r = \text{LB}_{\text{APm}} \)
AP+Lagr. vs AP+cuts

• **AP + cuts + Lagrangean Relaxation:**
  + still an AP, i.e. a structured problem;
  + $O(n^2)$ incrementally;
  + $x^*$ is integer;
  - $\lambda$ are optimal only at root node;
  - dynamically purging trivially satisfied cuts.

• **AP + cuts:**
  + LB always accurate;
  - resulting LPs may be huge;
  - only partially incremental.
Results (1)

- Although CP is not competitive to cope with problems like TSP and ATSP, the addition of an optimization component allows the solution of bigger-size instances.

<table>
<thead>
<tr>
<th>Instance</th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Opt</td>
<td>Time</td>
<td>Fails</td>
<td>Opt</td>
<td>Time</td>
<td>Fails</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Gr17</td>
<td>2085</td>
<td>0.39</td>
<td>511</td>
<td>2085</td>
<td>0.49</td>
<td>30</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Fri26</td>
<td>937</td>
<td>0.82</td>
<td>725</td>
<td>937</td>
<td>0.71</td>
<td>80</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Bays29</td>
<td>2020</td>
<td>4.12</td>
<td>4185</td>
<td>2020</td>
<td>1.20</td>
<td>403</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Dantzig42</td>
<td>699</td>
<td>&gt;300</td>
<td>-</td>
<td>699</td>
<td>5.55</td>
<td>1081</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RY48P</td>
<td>14854*</td>
<td>&gt;300</td>
<td>-</td>
<td>14422</td>
<td>130.00</td>
<td>50K</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- Pure CP gets stuck even on problems of this size.
TSP with Time Windows (TSPTW)

• On less pure problems it is possible to exploit the flexibility of CP.

• TSPTW is the TSP variant in which the visit of each city must be done within a prefixed Time Window.

• TSPTW has two main components:
  • a routing component which is basically optimization, i.e. find the tour of minimum cost;
  • a scheduling component which is mainly a feasibility issue.
TSP with Time Windows (TSPTW)

• what does it mean flexibility?

• a CP model for the TSPTW can be immediately obtained from the TSP one by adding global constraints modeling the scheduling part;

• propagation algorithms associated to existing constraints do not have to be changed;

• “old” and “new” constraints naturally interact through shared variables.
TSPTW: computational experiments

- table 1
- table 2
Conclusion

- Constraint Programming vs Mathematical Programming
  - local vs global
  - enumeration vs relaxation
  - feasibility vs optimality
  - logical vs geometrical
Conclusion

• Integration \( \rightarrow \) hybrid algorithms:

  • what works?
    • often simple things, e.g., AP is already very effective

  • where does it work?
    • real-world problems, side constraints, huge problems

  • why does it work?
    • enumeration is the matter, finding solutions