The Assignment Problem in Constraint Programming

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

based on joint work with:
• Filippo Focacci, ILOG, France
• Michela Milano, University of Bologna, Italy
• Louis-Martin Rousseau, CRT Montréal, Canada
Outline

• The linear Assignment Problem (AP).

• Using the Assignment Problem:
  1. Optimization global constraints.

• Using the Assignment Problem:
  2. Discrepancy-based additive bounding techniques.
The Assignment Problem (1)

- We are given $n$ machines and $n$ jobs, and a square matrix $c$ such that:
  - $c_{ij}$ is the cost associated with job $j$ if performed on machine $i$
- A graph theoretic model involves a bipartite, directed graph $G = (S \cup T; A)$ with costs associated with arcs

machines $\rightarrow$ S

jobs $\rightarrow$ T
The Assignment Problem (2)

- A feasible solution is a perfect matching

- An optimal solution is the least-cost perfect matching
The Assignment Problem (3)

\[ \min Z = \sum_{i \in S}^{\text{2 times}} \sum_{j \in T}^{\text{2 times}} c_{ij} x_{ij} \]

\[ \sum_{i \in S} x_{ij} = 1 \quad \forall j \in \{2 \text{ to } T\} \]

\[ \sum_{j \in T} x_{ij} = 1 \quad \forall i \in \{2 \text{ to } S\} \]

\[ x_{ij} \geq 0; x_{ij} \text{ integer} \quad \forall i \in S; j \in T \]
The Assignment Problem

• The AP is a linear program, thus can be solved with general purpose techniques as the simplex method.

• However, AP has a rather special structure and there are efficient special purpose algorithms to solve it.

• In particular, we will make use of the so-called Hungarian algorithm which is a primal-dual method which can be implemented to run in $O(n^3)$ time.

• We will point out all over the talk why solving the problem through a combinatorial algorithm is so important.
1: Optimization global constraints

• Global Constraints for optimization problems:
  
  • global constraints in CP;
  
  • global constraints with an *optimization component*:
    
    • the *cost-based domain filtering* technique.
  
  • Computational results on *TSP* and *TSPTW*. 
Global Constraints

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

• Global Constraints for optimization problems:
  • we need a pruning based on *optimality reasoning*;
  • we embed an *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 *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

GLOBAL CONSTRAINT

LB, $x^*$ and $\text{grad}(X,v)$

FILTERING ALGORITHM

OPTIMIZATION COMPONENT

COST-BASED FILTERING ALGORITHM

domain reduction
variable instantiation
Optimization Constraints (1)

- **Lower Bound**-based propagation:
  from \( LB \) towards objective function \( Z::[Z_{min}..Z_{max}]: LB < Z_{max} \)

- **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 additional cost to pay if \( X_i = v_j \)

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

  which is the classic OR *variable fixing*. 
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 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 with each node $i$ a variable $X_i$ whose domain contains the next possible nodes in a path, the CP path constraint:

$$X_0::D_0, X_1::D_1, \ldots, X_k::D_k$$

$hathemathrm{path}([X_0,X_1,\ldots,X_k])$

holds if and only if the assignment of variables $X_0,X_1,\ldots,X_k$ defines a simple path involving all nodes $0,\ldots,k$. 
Path constraint: an optimization component (2)

If a \textit{cost} is associated to each arc, and we want to model the \textbf{Asymmetric Traveling 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.

\textbf{AP} can then be used as optimization component for $\text{path}()$. 
Mapping

### CP- Model:
\[
X_i :: [v_1, v_2, \ldots, v_n] \quad i=0..n-1
\]
\[
\text{path}([X_0, X_1, \ldots, X_n])
\]
\[
C_i :: [c_{i1}, c_{i2}, \ldots, c_{in}] \quad i=0..n-1
\]
\[
C_n = 0; \quad X_n = 0;
\]
\[
C_0 + \ldots + C_{n-1} = Z
\]
\[
\text{minimize}(Z)
\]

### IP- Model
\[
\min Z = \sum_{i \in V} \sum_{j \in V} c_{ij} x_{ij}
\]
\[
\sum_{j \in V} x_{ij} = 1 \quad j \in V \quad (A)
\]
\[
\sum_{i \in V} x_{ij} = 1 \quad i \in V \quad (B)
\]
\[
\sum \sum_{j \in V \setminus S} x_{ij} \geq 1 \quad S \subset V \quad S \neq \emptyset
\]
\[
x_{ij} \geq 0 \quad \text{and integer}
\]
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.
Lagrangian relaxation of cuts

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

• Algorithm:
  • optimally solve the original structured relaxation \( \rightarrow LB_{AP} \);
  • 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) \( \rightarrow LB_r \);
  • 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 \( \rightarrow LB_{APm} \).

Through duality theory: \( LB_{AP} \leq LB_r = LB_{APm} \)
AP+Lagrangian 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

- 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>pure AP</th>
<th></th>
<th></th>
<th>AP + Lagrangean relaxation of cuts</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>
</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>
</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>
</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>
</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>
</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>
</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.
TSPTW: aggregated experimental results

- **symmetric TSPTW:**
  - outperforming state of the art methods Pesant et al. 1998
  - Lagrangean relaxation of cuts is very effective

- **asymmetric TSPTW**
  - competitive results with state of the art branch-and-cut methods Ascheuer, et al. 2001
  - Lagrangean relaxation does not pay off since AP bound already effective
2: Linking search and bound

• Discrepancy-based additive bounding:
  • limited discrepancy search;
  • additive bounding techniques;
  • speeding up the proof of optimality.

• Computational results.
Limited discrepancy search

• Explores the most promising branches of a search tree first
  – At each node a heuristic *recommends a branch*
  – Tolerate a maximum number of *discrepancies* from the heuristic’s recommendation

• Limited Discrepancy Search (LDS) is effective to rapidly identify *good solutions*

• LDS has no real incentive to accelerate the proof of optimality
Discrepancy based search

- Introduced by Milano and van Hoeve at CP 2002
- First split the domain of each variable into:
  - a *Good* set containing the promising values
  - a *Bad* set containing the rest of the values.
- Perform LDS with these sets
  - At each node a discrepancy is counted if a variable takes a value in its *Bad* set.
Discrepancy constraint

- Introduce a *Discrepancy Constraint* in the model of any problem solved via the Discrepancy Based Search (DBS)
  - $X$ is the vector of finite domain variables
  - $\beta_V$ is the bad set of variable $V$
  - $k$ is the current accepted level of discrepancy
  - $\text{Discrepancy}_\text{Cst}(X, \beta, k)$

\[
\sum_{i \in N} (X_i \in \beta_{X_i}) = k
\]
Reduced costs

- The *reduced costs* computed as a result of the solution of a linear program:
  - Associated with a variable $V$
  - Represent the cost to add to the optimal solution if the variable $V$ becomes basic at value 1
  - Denoted here by: $c_V$

- If for a given problem:
  - More than one “bound” is available: $B^1, \ldots, B^{nr}$
  - Each bound takes as input a cost vector: $B^k(c^{k-1})$
  - All bounds return a value $LB^k$
  - All bounds output a reduced cost vector: $c^k$
The Additive Bounding Procedure (ABP) is:
- Compute: $LB^1 = B^1(c^0)$ where $c^0$ is the original cost vector
- for all $k:2,…,nr : LB^k = B^k(c^{k-1})$
- $LB = LB^1 + LB^2 + ,…,+ LB^{nr}$

Additive Bounding Procedure (1)
Additive Bounding Procedure (2)

- This remarkable technique has been introduced by Fischetti & Toth as a general framework and successfully applied in the context of the Traveling Salesman Problem.

- Enhancing LDS proof of optimality by improving the quality of the bounds.

- Additional motivation: use ABP in conjunction with DBS to establish a stronger link between search and bound.
Combinatorial Problem solved via DBS

Relaxation solved via a specialized algorithm

Reduced Cost

Relaxation based on Discrepancy Constraint

Lower Bound

Combined Bound
ABP and DBS: what is to be gained

Discrepancy Bound
Discrepancy-Based Additive Bounding

\[
\text{Minimize} : \sum_{i \in N} C_{iX_i}
\]

\text{Subject To:} \quad \text{AllDifferent} (X) \\
\quad \text{AnySideConstraint} (X) \\
\quad X_{i \in N} \in N \\
\quad \text{DiscrepancyCst}(X, \beta, k)

Can be seen as an Assignment Problem

Side Constraint which makes the problem \(NP\)-hard

Discrepancy constraint
Discrepancy-Based Additive Bounding
A first additive bound

Minimize: $\sum_{i \in N} C_{ix_i}$

Subject To: AllDifferent ($X$)

$X_{i \in N} \in N$

Discrepancy Cst ($X$, $\beta$, $k$)

AnySideConstraint ($X$)

First bound from the Assignment Problem

Second bound from Discrepancy Constraint
Discrepancy-Based Additive Bounding
A first additive bound

\[
\begin{align*}
\text{Minimize} & : \sum_{i \in N} C_{iX_i} \\
\text{Subject To} & : \text{AllDifferent}(X) \\
& X_{i \in N} \in N
\end{align*}
\]

First bound from a \textit{primal-dual} Algorithm

\[
\begin{align*}
\text{Minimize} & : \sum_{i \in N} C_{iX_i}^{1} \\
\text{Subject To} & : \sum_{i \in N} (X_i \in \beta_{X_i}) = k \\
& X_{i \in N} \in N
\end{align*}
\]

Second bound from counting the \( k \) smallest reduced cost in \( \beta \)
Discrepancy-Based Additive Bounding Linear Model

\[ x_{ij} = 1 \Leftrightarrow X_i = j \quad \forall i, j \in N \]
\[ \sum_{j \in N} x_{ij} = 1 \quad \forall i \in N \]
\[ x \in \{0,1\}^N \]

Minimize: \[ \sum_{i \in N} c_{ij} x_{ij} \]

Subject To: \[ \sum_{i \in N} x_{ij} = 1 \quad \forall j \in N \]
\[ \sum_{j \in N} x_{ij} = 1 \quad \forall i \in N \]
\[ \sum_{i \in N} \sum_{j \in \beta_{x_i}} x_{ij} = k \]

AnyLinearSideConstraint \((X)\)
\[ x \in \{0,1\}^N \]
Discrepancy-Based Additive Bounding
a second (general) bound

Minimize : \( \sum_{i \in N} c_{ij}^1 x_{ij} \)

Subject To : \( \sum_{j \in \beta_{x_i}} x_{ij} \leq 1 \quad \forall i \in N \)
\( \sum_{i \in N} \sum_{j \in \beta_{x_i}} x_{ij} = k \)
\( x \in \{0,1\}^N \)

To solve:
- Identify the minimum reduced cost associated with every CP variable
- Sum the \( k \) minimum reduced costs selected
Discrepancy-Based Additive Bounding
Generality of the method

- The technique has been applied in conjunction with the AP but it is obviously independent on it.

- More precisely, the framework can be applied with any (combinatorial) relaxation providing the reduced cost vector or, even more likely, with a sequence of relaxations.

- “Combinatorial relaxation” means that a special purpose algorithm is used to solve the relaxation, thus the framework does not affect the structure of the relaxation itself.
Discrepancy-Based Additive Bounding
Further exploiting the structure: AllDifferent

\[ \text{Minimize} \ : \ \sum_{i \in N} c_{ij} x_{ij} \]

\[ \text{Subject To} \ : \ \sum_{i \in N} x_{ij} \leq 1 \quad \forall j \in N \]
\[ \sum_{j \in N} x_{ij} \leq 1 \quad \forall i \in N \]
\[ \sum_{i,j \in N} x_{ij} = k \]
\[ x \in \{0,1\}^N \]

it is known as the \( k \)-Assignment problem and can be solved in polynomial time

it is also a relaxation of the previous model
(since \( c^1_{ij} = 0 \))

Incorporate more information than just the discrepancy constraint

use of a second bound
Defining benchmarks (1)

Lower Assignment Problem (LAP)

Minimize: $\sum_{i \in N} C_{iX_i}$

Subject To: AllDifferent($X$)

$L \leq \sum_{i \in N} C_{iX_i} \leq L$

$L \leq X_{i \in N} \in N$

Assignment Structure

$L$ constraint abstracts the side constraints and arbitrarily makes the AP bound poor.

Didactic but NP-hard and very hard to solve
Defining benchmarks (2)

Asymmetric Traveling Salesman Problem (ATSP)

Minimize: \[ \sum_{i \in N} C_{ix_i} \]

Subject To: AllDifferent\( (X) \)

SubTourEli mination\( (X) \)

\[ X_{i \in N} \in N \]
Defining benchmarks (3)

Resource Constraint Assignment Problem (RCAP)

\[ \text{Minimize} : \sum_{i \in N} C_{iX_i} \]

\[ \text{Subject To: AllDifferent}(X) \]

\[ \sum_{i \in N} R_{iX_i} \leq M \]

\[ X_{i \in N} \in N \]

Assignment Structure

classical knapsack constraint modeling a resource.

NP-hard to solve
Experimental Results

- For all benchmarks we generated 60 test problems using structured cost matrices of the DIMACS ATSP instances.
- Size of the problem was set to $N = 25$.

- For LAP, $L$ is 110% the AP relaxation value.
- For RCAP, first solve AP using $R$ as cost vector, then $M$ is set 4 times this value.

- Ilog Solver 5.2. on a Intel 1.5 GHz Centrino laptop.
- Time limit (TL) for each run is 3,600 CPU seconds.
- Variable selection based on first fail (min. domain size).
- Value selection based on minimum reduced cost.
## Aggregated experimental results (1)

### %reduction

<table>
<thead>
<tr>
<th>Problem</th>
<th>Counting</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Time</td>
</tr>
<tr>
<td>LAP</td>
<td>55%</td>
</tr>
<tr>
<td>ATSP</td>
<td>31%</td>
</tr>
<tr>
<td>RCAP</td>
<td>25%</td>
</tr>
</tbody>
</table>

### average k

<table>
<thead>
<tr>
<th>Problem</th>
<th>normal DBS</th>
<th>counting</th>
<th>K-assignment</th>
</tr>
</thead>
<tbody>
<tr>
<td>LAP</td>
<td>24.75</td>
<td>4.38</td>
<td>4.35</td>
</tr>
<tr>
<td>ATSP</td>
<td>26.00</td>
<td>5.68</td>
<td>5.63</td>
</tr>
<tr>
<td>RCAP</td>
<td>23.91</td>
<td>5.16</td>
<td>5.10</td>
</tr>
</tbody>
</table>
Aggregated experimental results (2)

• Taking search into account in the bounding procedures seems to be particularly effective.

• This is a general (and easy) approach which can be widely used when an efficient algorithm for a relaxation provides reduced costs.

• Further exploiting a problem structure can improve the behavior of discrepancy-based additive bounding.
$k$-discrepancy: Full Integration via LP

- Fully integrate the Discrepancy Constraint with the lower bound
  - Use the linear relaxation as a lower bound
  - Not as efficient since an LP is solved at each node
  - Maximum use of discrepancy information

Minimize: $\sum_{i \in N} c_{ij} x_{ij}$

Subject To:
- $\sum_{i \in N} x_{ij} = 1 \quad \forall j \in N$
- $\sum_{j \in N} x_{ij} = 1 \quad \forall i \in N$
- $\sum_{i \in N} \sum_{j \in \beta_{x_i}} x_{ij} = k$

AnyLinearSideConstraint$(X)$

$x \in (0,1)^N$
**k-discrepancy: Partial Integration via Lagrangean Relaxation**

- Partially integrate the **Discrepancy Constraint** via Lagrangean Relaxation
  - Only some information on discrepancy is used
  - At each node an assignment problem is solved instead of a linear program

**Drawback:**
The Lagrangean multiplier used is not anymore optimal during search

---

**Steps:**

1. **Solve the Linear Relaxation**
   - Once per discrepancy

2. **Lagrangean Relaxation of the Discrepancy Constraint**
   - With optimum multiplier

3. **Solve the Modified Assignment Problem**
   - During Search
Additional experimental results

• More integrated is the *Discrepancy Constraint*, less discrepancy level is needed to prove optimality, i.e. in terms of the value of the bound we have:

  Additive Bounding $<$ Lagr. Relaxation $<$ Linear Relaxation

• LP and Lagrangean relaxations are less effective both in terms of computing time and number of backtracks (they also solve less problems in TL):
  – LPs are more time consuming
  – Lagrangean multiplier deteriorates
  – Cost-based propagation in pure AP case is more effective
Conclusion

• The AP is extensively used as a relaxation for different purposes.

• On the other hand, the techniques shown in this talk do not depend on the AP.

• The key issue is the use of a combinatorial relaxation, i.e., a relaxation which models a linear program but can be solved with a special purpose technique.

• This is often the case with graph theory models!!!
References

• F. Focacci, A. Lodi, M. Milano, 

• F. Focacci, A. Lodi, M. Milano, 

• A. Lodi, M. Milano, L.-M. Rousseau, 