Global Constraints

Willem-Jan van Hoeve
Tepper School of Business, Carnegie Mellon University
vanhoeve@andrew.cmu.edu

ACP Summer School on Practical Constraint Programming
June 16-20, 2014, Bologna, Italy
Outline

• Introduction
• Alldifferent constraint
• Knapsack constraint
• Regular constraint
• Research directions
Example: Graph coloring

Assign a color to each country

Adjacent countries must have different colors

Can this be done with $k$ colors? (minimize $k$)
Smaller (8 variable) instance
Solution with four colors
CP Model for $k=3$ colors

Variables and domains:  $x_i$ in \{r, g, b\}  for all $i$

Constraints:  $x_i \neq x_j$  for all edges $(i,j)$
Search choice: $x_2 = r$
(by symmetry, no need to consider $x_2 = g, b$)
Search & propagate

Search choice: $x_5 = g$
(be prepared to backtrack)
Search & propagate

... and propagate

$x_7$ has an empty domain: we need to backtrack
Search & propagate

Propagate...
Search & propagate

\[
x_2 = r \\
x_5 = g \\
\]

\[
x_5 \neq g
\]
...and propagate

\( x_7 \) has an empty domain: we are done
Can we do more propagation?
After $x_2 = r$ we are done.
Introduce global constraints

• We can increase the inference by adding more knowledge to the solver
  – in this case, group not-equal constraints that form a clique
  – use *alldifferent* constraints

\[
alldifferent(x_1,x_2,...,x_n) := \bigwedge_{i<j} x_i \neq x_j
\]

Model 1: \(x_1 \in \{g,b\}, x_4 \in \{g,b\}, x_8 \in \{r,g,b\}\)

\(x_1 \neq x_4, x_1 \neq x_8, x_4 \neq x_8\) \hspace{1cm} \text{no propagation}

Model 2: \(x_1 \in \{g,b\}, x_4 \in \{g,b\}, x_8 \in \{r,g,b\}\)

\(alldifferent(x_1,x_4,x_8)\) \hspace{1cm} x_8 = r
Impact of global constraint propagation

• Graph coloring problem; random instances

• Can set \textit{alldifferent} propagation level from ‘low’ to ‘extended’
  – ‘low’: pairwise not-equal constraints
  – ‘extended’: best possible propagation
  – notice the difference in search tree size (search choices or failures) and solving time
Global constraints overview

- **Examples**
  - *Alldifferent, Cardinality, Circuit, BinPacking, ...*

- **Global constraints represent combinatorial structure**
  - can be viewed as the combination of elementary constraints
  - expressive building blocks for modeling applications
  - embed powerful algorithms from OR, Graph Theory, AI, CS, ...

- **Essential for the successful application of CP**
  - User can identify global constraints to be used in model
  - Automated detection for certain constraints (ILOG CPO)
## Embedded Algorithms

<table>
<thead>
<tr>
<th>Constraint</th>
<th>Structure/technique</th>
</tr>
</thead>
<tbody>
<tr>
<td>alldifferent</td>
<td>bipartite matching [Régin, 1994]</td>
</tr>
<tr>
<td>cardinality</td>
<td>network flow [Régin, 1996]</td>
</tr>
<tr>
<td>knapsack</td>
<td>dynamic programming [Trick, 2003]</td>
</tr>
<tr>
<td>regular</td>
<td>directed acyclic graph [Pesant, 2004]</td>
</tr>
<tr>
<td>sequence</td>
<td>various [vH et al., 2006,09] [Brand et al., 2007] [Maher et al., 2008]</td>
</tr>
<tr>
<td>BinPacking</td>
<td>various [Shaw, 2004] [Cambazard et al., 2010] [Schaus et al., 2010-13]</td>
</tr>
<tr>
<td>N-value</td>
<td>various [Beldiceanu et al., 2001] [Bessiere et al., 2005, 10]</td>
</tr>
<tr>
<td>circuit</td>
<td>network flow [Genc Kaya &amp; Hooker, 2006]</td>
</tr>
<tr>
<td>weighted circuit</td>
<td>AP [Focacci et al., 1999], 1-Tree [Benchimol et al., 2012]</td>
</tr>
<tr>
<td>disjunctive/cumulative</td>
<td>dedicated algorithm [Nuijten 1994, Carlier et al., 1994] [Vilim, 2009]</td>
</tr>
</tbody>
</table>

...          

The ‘global constraint catalog’ currently contains 364 constraints
http://sofdem.github.io/gccat/
Role of global constraints

Global constraints can typically play three roles

1. Convenient modeling
   - Global constraints are the building blocks of a complex problem

2. More effective constraint propagation
   - Identify more inconsistent domain values; reduce the search space

3. Help guide the search
   - Provide variable and value ordering heuristics
Consistency notions for global constraints

• Hyperarc consistency
  – (a global constraint defines a hyperarc in the constraint network)
  – ensure that all domain values are consistent w.r.t. the constraint
  – a.k.a. generalized arc consistency or domain consistency

• Bounds consistency
  – treat the domains as intervals, and ensure that all domain bounds are consistent

• Ad-hoc consistencies
  – constraint dependent; can be based on relaxations of the constraint
Global constraints during propagation

- Algorithms that enforce a local consistency are referred to as *domain filtering* algorithms, or *propagation* algorithms.
- General tasks for a propagation algorithm:
  1. Determine whether the constraint is satisfiable (consistency check)
  2. Remove some or all inconsistent domain values (the actual domain filtering)
- The consistency check and the filtering are typically done separately for efficiency reasons.
Propagation algorithm for \textit{alldifferent}

Alldifferent Propagation

- Goal: establish domain consistency on \textit{alldifferent} 
  - Guarantee that each remaining domain value participates in at least one solution
  - Can we do this in polynomial time?

- We already saw that the decomposition is not sufficient to establish domain consistency

\[ x_1 \in \{a, b\}, \ x_2 \in \{a, b\}, \ x_3 \in \{a, b, c\} \]

\[ x_1 \neq x_2, \ x_1 \neq x_3, \ x_2 \neq x_3 \quad \text{versus} \quad \text{alldifferent}(x_1, x_2, x_3) \]
First attempt: Hall’s Theorem

Hall’s Marriage Theorem [1935]:

If a group of men and women marry only if they have been introduced to each other previously, then a complete set of marriages is possible if and only if every subset of men has collectively been introduced to at least as many women, and vice versa.

For \textit{alldifferent}(X) this means that a solution exists iff

\[
|K| \leq \left| \bigcup_{x \in K} D(x) \right| \quad \forall K \subseteq X
\]

Example: \(x_1 \in \{b,c\}, x_2 \in \{b,c\}, x_3 \in \{a,b,c\}, x_4 \in \{a,b,c\}\)
- solution exists for any subset of 3 variables
- no solution when \(K = \{x_1, x_2, x_3, x_4\}\)
• **Definition**: Let $G = (V,E)$ be a graph with vertex set $V$ and edge set $E$. A *matching* in $G$ is a subset of edges $M$ such that no two edges in $M$ share a vertex.

• A *maximum matching* is a matching of maximum size

• **Definition**: An *$M$-augmenting path* is a vertex-disjoint path with an odd number of edges whose endpoints are $M$-free

• **Theorem**: Either $M$ is a maximum-size matching, or there exists an $M$-augmenting path  

[ Petersen, 1891]
Finding a maximum matching

- The augmenting path theorem can be used to iteratively find a maximum matching in a graph G:
  - given M, find an M-augmenting path P
  - if P exists, augment M along P and repeat
  - otherwise, M is maximum

- For a bipartite graph $G = (V_1, V_2, E)$, an M-augmenting path can be found in $O(|E|)$ time
  - finding a maximum matching can then be done in $O(|V_1| \cdot |E|)$, as we need to compute at most $|V_1|$ paths (assume $|V_1| \leq |V_2|$)
  - this can be improved to $O(\sqrt{|V_1| \cdot |E|})$ time \[ Hopcroft & Karp, 1973 \]

- For general graphs this is more complex, but still tractable
  - can be done in $O(\sqrt{|V| \cdot |E|})$ time \[ Micali & Vazirani, 1980 \]
**Value Graph Representation**

- **Definition**: The *value graph* of a set of variables $X$ is a bipartite graph $(X, D, E)$ where
  - node set $X$ represents the variables
  - node set $D$ represents the union of the variable domains
  - edge set $E$ is $\{(x, d) \mid x \in X, d \in D(x)\}$

- **Example**:
  
  $x_1 \in \{a, b\}$
  
  $x_2 \in \{a, b\}$
  
  $x_3 \in \{b, c\}$
Lemma [Régin, 1994]:

solution to *alldifferent*(X) ⇔
matching in value graph covering X

Example:

\[ x_1 \in \{a, b\}, \ x_2 \in \{a, b\}, \ x_3 \in \{b, c\} \]

*alldifferent*(x₁,x₂,x₃)

Theorem: Domain consistency for *alldifferent*:

remove all edges (and corresponding domain values) that are not in any maximum matching
Propagation Algorithm

1. Verify consistency of the constraint
   - find maximum matching $M$ in value graph $O(\sqrt{|X| \cdot |E|})$
   - if $M$ does not cover all variables: inconsistent

2. Verify consistency of each edge $O(\sqrt{|X| \cdot |E|^2})$
   - for each edge $e$ in value graph:
     fix $e$ in $M$, and extend $M$ to maximum matching
     if $M$ does not cover all variables: remove $e$ from graph

What is the time complexity?
• Establishes domain consistency in polynomial time
• But not very efficient in practice... can we do better?
A useful theorem

- **Theorem** [Petersen, 1891] [Berge, 1970]: Let G be graph and M a maximum matching in G. An edge e belongs to a maximum-size matching if and only if
  - it either belongs to M
  - or to an even M-alternating path starting at an M-free vertex
  - or to an M-alternating circuit
1. compute a maximum matching $M$: covering all variables $X$?
2. direct edges in $M$ from $X$ to $D$, and edges not in $M$ from $D$ to $X$
3. compute the strongly connected components (SCCs)
4. edges in $M$, edges within SCCs and edges on path starting from M-free vertices are all consistent
5. all other edges are not consistent and can be removed

Note: SCCs correspond to ‘tight’ Hall sets $K$: $|K| = \left| \bigcup_{x \in K} D(x) \right|$
Important aspects

• Separation of consistency check ( \(O(\sqrt{|X| \cdot |E|})\) ) and domain filtering ( \(O(|E|)\) )

• Incremental algorithm
  – Maintain the graph structure during search
  – When \(k\) domain values have been removed, we can repair the matching in \(O(km)\) time
  – Note that these algorithms are typically invoked many times during search / constraint propagation, so being incremental is very important in practice
Propagation algorithm for *knapsack*

Knapsack constraints

- Knapsack constraints restrict a weighted linear sum to be no more than a given maximum:
  - Variables $X = \{x_1,\ldots,x_n\}$ with finite integer domains
  - Integer weights $w_i$ ($i=1..n$)
  - Integer variable $z$ representing the capacity
  - $Knapsack(X, z, w) := \sum_i w_i x_i \leq z$

Questions:
1. Can we determine in polynomial time whether the knapsack constraint is consistent (satisfiable)?  **NP-hard** [Garey&Johnson, 1979]
2. Can we establish domain consistency (remove all inconsistent domain values) in polynomial time?
Example:
- \( x_1 \in \{2, 4\} \), \( x_2 \in \{2, 3, 4\} \), \( x_3 \in \{1, 3\} \), \( z \in \{7, 9, 12\} \)
- Unit weights (\( w_i = 1 \))
- \( \sum_i x_i \leq z \)
Lemma: Any path in the graph from the origin to a goal state corresponds to a feasible solution to the knapsack constraint

Lemma: If a variable $x_i$ has no edge with label $d$ in the graph, then $d$ can be removed from $D(x_i)$ without affecting the set of solutions

Theorem: Domain consistency for knapsack: remove all edges (and corresponding domain values) that are not in any path to a goal state
Filtering the graph and domains

- Example:
  - $x_1 \in \{2, 4\}$, $x_2 \in \{2, 3, 4\}$, $x_3 \in \{1, 3\}$, $z \in \{7, 9, 12\}$
  - unit weights ($w_i = 1$)
  - $\sum_i x_i \leq z$
• Filtering the graph takes linear time
  – one forward and one backward pass suffices to establish domain consistency
  – but size of graph depends on domain size: pseudo-polynomial time
  – no need to re-compute from scratch each time; we can maintain the graph incrementally
Propagation algorithm for regular


A regular language can be represented by a deterministic finite automaton (DFA):

automaton accepts string $\iff$ string belongs to regular language

Example:

start state: $q_0$, end states: $q_3$ and $q_4$
each transition between states has a label
e.g. strings ‘aabbaa’ and ‘ccc’ accepted
string ‘caabbac’ not accepted

Given a DFA, the constraint regular($x_1,x_2,\ldots,x_n,$ DFA) imposes that the ‘string’ $x_1x_2\ldots x_n$ is accepted by DFA (actually; NFA is also fine)
Nurse rostering problem

- each nurse works at most one shift a day
- each shift contains 8 consecutive hours
  - day shift: 8am-4pm
  - evening shift: 4pm-12am
  - night shift: 12am-8am
- after a night shift, nurse needs to take one day rest
- after an evening shift, nurse may not work a day shift

Feasible (7-day) schedule: day - day - evening - night - rest - day - day

- For each nurse, introduce variables $X = \{x_1, x_2, ..., x_7\}$ representing shift on day 1, 2, ..., 7 with domains $D(x) = \{r, d, e, n\}$ for all $x \in X$
- Model the requirements as $\text{regular}(X, \text{DFA})$ for each nurse
Theorem:
solution to regular $\Leftrightarrow$ path from $q_0$ to ‘goal state’ in layered graph

Example:

$x_1 \in \{a, b, c\}$, $x_2 \in \{a, b, c\}$, $x_3 \in \{a, b, c\}$, $x_4 \in \{a, b, c\}$

$\text{regular}(x_1, x_2, x_3, x_4, \text{DFA})$

Domain consistency: remove all arcs whose label is not supported by domain value and vice versa (linear time in size of graph)
Theorem:
solution to regular $\iff$ path from $q_0$ to ‘goal state’ in layered graph

Example:

$x_1 \in \{a,b,c\}$, $x_2 \in \{a,b,c\}$, $x_3 \in \{a,b,c\}$, $x_4 \in \{a,b,c\}$

regular($x_1, x_2, x_3, x_4, DFA$)

Domain consistency: remove all arcs whose label is not supported by domain value and vice versa (linear time in size of graph)
We can ‘decompose’ regular into separate transitions:

1. create a ‘table’ representing all possible transitions (edges)

\[ T: \{ (q_0, a, q_1), (q_2, a, q_3), (q_1, a, q_1), (q_3, a, q_3), (q_1, b, q_2), (q_0, c, q_4), (q_2, b, q_2), (q_4, c, q_4) \} \]

2. define ‘state’ variables \( Q_0, Q_1, Q_2, Q_3, Q_4 \), with

\[ Q_0 \in \{q_0\}, Q_1, Q_2, Q_3 \in \{q_0,q_1,q_2,q_3,q_4\}, Q_4 \in \{q_3, q_4\} \]

3. define transition constraint \( T(Q_i, x_{i+1}, Q_{i+1}) \) for \( i=0,1,2,3 \)
• **Theorem** [Beldiceanu et al. 2004, 2005]:
  
  – establishing domain consistency on the reformulation is equivalent to establishing domain consistency on regular
  
  – the reformulation can be made domain consistent in $O(n|T|)$ time (here $|T|$ is number of transitions), which is the same as regular

*Proof*: dual constraint graph is acyclic

$$
\begin{align*}
T(Q_0,x_1,Q_1) & \quad Q_1 \quad T(Q_1,x_2,Q_2) \quad Q_2 \quad \ldots \quad Q_{n-1} \quad T(Q_{n-1},x_n,Q_n)
\end{align*}
$$

• The reformulation is easier to implement, and can be more efficient than Pesant’s algorithm in practice [Quimper&Walsh, 2006]
Research directions

• Not all 364 constraints in the catalog are equally useful
  – Most solvers only support a handful of constraints: alldifferent, cardinality, table constraints, constraints for scheduling
  – Unsupported global constraints are simply reformulated or decomposed

• Challenge seems not to be in creating new constraints, but into handling/utilizing existing constraints better
Direction 0: optimization

- By design, pure CP solvers are based on feasibility reasoning
  - relatively weak support for optimization (compared to e.g., MIP)

- Adapt global constraints for optimization
- Utilize known relaxations (linear programming, Lagrangian relaxations, ...)
  - progress over last 10~15 years
  - this will be covered in other lectures (incl. Hybrid Methods on Thursday)
Direction 1: learn and automate

• Automate the process of identifying the ‘right’ global constraint to apply
  – ModelSeeker does this by learning constraints from example solutions [Beldiceanu&Simonis, 2012]
  – IBM ILOG CPO does this by grouping together specific constraints

• Learn no-goods during search
  – Record the implications from the propagation process
  – Explain search failure by identifying a minimal conflict set to be added as ‘no-good’ (e.g., Lazy Clause Generation)
  – Need to derive explanations from global constraints [Rochart et al., 2003-2005], [Downing et al., 2012]
Direction 2: guide search

- Use global constraints to dynamically define a good variable and value selection heuristic
  - Counting-based search: for each variable/value pair, count the number of solutions in which it appears [Pesant, Zanarini, et al., 2007-2013]
  - Two strategies: highest solution density first, or lowest solution density first

- Global constraints can also be used to guide local search methods
Direction 3: improve communication

• Current CP solvers are centered around *domain* propagation
  – In effect, very limited information is communicated between (global) constraints

• One approach is to study pairs (or more) of constraints

• Another approach is to propagate more structured information
  – precedence constraints in scheduling applications
  – constraints over structured domains such as set variables
  – for general CP: propagate *approximate decision diagrams* [Andersen et al., 2007], [Hadzic et al., 2007-2009], [Hoda et al., 2010], ...
Summary

• Global constraints provide convenient building blocks for modeling and solving practical applications of optimization

• Constraint propagation is usually divided in two parts
  – consistency check
  – domain filtering
  (in some cases, domain consistency can be established in polynomial time)

• Global constraints embed efficient algorithms
  – some are adapted from known techniques: matchings, networks, dynamic programming, ...
  – others are new, dedicated, algorithms
References
