DM826 – Spring 2011

Modeling and Solving Constrained Optimization Problems

Lecture 11
Global Variables

Marco Chiarandini

Department of Mathematics & Computer Science
University of Southern Denmark

Resume

- Modelling in IP and CP
- Global constraints
- Local consistency notions
- Filtering algorithms for global constraints
- Search
- Symmetries
- Set variables
- Integrated/Advanced Approaches:
  - Branch and price
  - Logic-based Benders decomposition
- Scheduling

Global Variables

Global variables: complex variable types representing combinatorial structures in which problems find their most natural formulation

Eg:
sets, multisets, strings, functions, graphs
bin packing, set partitioning, mapping problems

We will see:

- Set variables

- Graph variables

Outline

1. Global Variables

2. Graph Variables

Finite-Set Variables

- A finite-domain integer variable takes values from a finite set of integers.
- A finite-domain set variable takes values from the power set of a finite set of integers.
  Eg.:
  domain of x is the set of subsets of $\{1, 2, 3\}$:

$$\{\{\}, \{1\}, \{2\}, \{3\}, \{1, 2\}, \{1, 3\}, \{2, 3\}, \{1, 2, 3\}\}$$

Finite-Set Variables

Recall the shift-assignment problem

We have a lower and an upper bound on the number of shifts that each worker is to staff (symmetric cardinality constraint)

- one variable for each worker that takes as value the set of shifts coverd by the worker. $\rightsquigarrow$ exponential number of values

- set variables with domain $D(x) = [lb(x), ub(x)]$
  $D(x)$ consists of only two sets:
  - $lb(x)$ mandatory elements
  - $ub(x) \setminus lb(x)$ of possible elements

  The value assigned to x should be a set $s(x)$ such that
  $lb \subseteq s(x) \subseteq ub(x)$

In practice good to keep dual views with channelling

Finite-Set Variables

Example:

domain of x is the set of subsets of $\{1, 2, 3\}$:

$$\{\{\}, \{1\}, \{2\}, \{3\}, \{1, 2\}, \{1, 3\}, \{2, 3\}, \{1, 2, 3\}\}$$

can be represented in space-efficient way by:

$$[\{\}..\{1, 2, 3\}]$$

The representation is however an approximation!

Example:

domain of x is the set of subsets of $\{1, 2, 3\}$:

$$\{\{1\}, \{2\}, \{3\}, \{1, 2\}, \{1, 3\}, \{2, 3\}\}$$

cannot be captured exactly by an interval. The closest interval would be still:

$$[\{\}..\{1, 2, 3\}]$$

$\rightsquigarrow$ we store additionally cardinality bounds: $\#[i..j]$

Set Variables

Definition

set variable is a variable with domain $D(x) = [lb(x), ub(x)]$
$D(x)$ consists of only two sets:

- $lb(x)$ mandatory elements (intersection of all subsets)
- $ub(x) \setminus lb(x)$ of possible elements (union of all subsets)

The value assigned to x must be a set $s(x)$ such that $lb \subseteq s(x) \subseteq ub(x)$

We are not interested in domain consistency but in bound consistency:

Enforcing bound consistency

A bound consistency for a constraint C defined on a set variable x requires that we:

- Remove a value v from $ub(x)$ if there is no solution to C in which $v \in s(x)$.
- Include a value $v \in ub(x)$ in $lb(x)$ if in all solutions to C, $v \in s(x)$.

In Comet

```
import cotfd;
Solver cp();
var<CP>{set{int}} S(cp,1..5,2..4);
var<CP>{set{int}} S1(cp,{3,5,7,8,9},2..4);
```

$lb(x)$ cannot be specified. It can be stated in the CP model.

```
boolean bound();
set{int} getValue();
set{int} evalIntSet();

var<CP>{int} getCardinalityVariable();
set{int} getRequiredSet(); //lb(x)
set{int} getPossibleSet(); //ub(x)
boolean isRequired(int v);
boolean isExcluded(int v);
```

In Comet


```
import cotfd;
Solver<CP> cp();
var<CP>{set{int}} S(cp,1..5,2..4);
cp.post(S.getCardinalityVariable()!=3);
cp.post(requiresValue(S,3));
cp.post(excludesValue(S,2));
cout << S.getRequiredSet() << endl;
cout << S.getPossibleSet() << endl;
cout << S.isRequired(4) << endl;
cout << S.isExcluded(2) << endl;
```

What are the possible values of the variable S?
And the state of the variable S?

```
((1){3},(1){2},(4){1,3,4,5} | (DOM:2)[2,4])
```

In Comet

In addition, create variables on the presence of values:

```
var<CP>{boolean}
var<CP>{boolean}
boolean
boolean
getRequired(int v);
getExcluded(int v);
hasRequiredVariable(int v);
hasExcludedVariable(int v);
```

Constraints on FS variables

Basic operations

```
cp.post(S1==S2);
cp.post(subset(S1,S2));
cp.post(setunion(S1,S2,RES));
cp.post(setinter(S1,S2,RES));
cp.post(setdifference(S1,S2,RES));
```

```
RES = setunion(S1,S2);
RES = setinter(S1,S2);
RES = setdifference(S1,S2);
```

Constraints on FS variables

Set cardinality

```
cp.post(cardinality(S1,k));
cp.post(S1.getCardinalityVariable()!=k);
```

```
cp.post(atleastIntersection(S1,S2,k));
cp.post(atmostIntersection(S1,S2,k));
cp.post(exactIntersection(S1,S2,k));
```

```
cp.post(disjoint(S1,S2));
cp.post(allDisjoint(SA));
```

where S1 and S2 are set variables and SA is an array of set variables.

Constraints on FS variables

Requirement and exclusion constraints


```
cp.post(requiresValue(S,k1));
cp.post(excludesValue(S,k2));
```

```
cp.post(requiresVariable(S,x1));
cp.post(excludesVariable(S,x2));
```

Constraints on FS variables

Channeling constraints

SA_1 and SA_2 two arrays of set variables

```
cp.post(channeling(SA1,SA2));
```

$$SA_1[i] = s \iff \forall j \in s : i \in SA_2[j]$$

$$\begin{aligned} SA_1[i] &= \{j \| SA_2[j] \text{contains} i\} \\ SA_2[j] &= \{i \| SA_1[i] \text{contains} j\} \end{aligned}$$

Example:

```
SA1 = [{1,2},{3},{1,2}]
SA2 = [{1,3},{1,3},{2}]
```

Constraints on FS variables
Channeling constraints

SA an array of set variables, X an array of integer variables

```
cp.post(channeling(SA,X));
```

$$SA[i] = s \iff \forall j \in s : X[j] = i$$

```
SA = [{1,2},{3}]
X = [1,1,2]
```

Constraints on FS variables

Set Global Cardinality

bounds the minimum and maximum number of occurrences of an element in an array of set variables:

$$\forall v \in U : l_v \leq |\mathcal{S}_v| \leq u_v$$

where $\mathcal{S}_v$ is the set of set variables that contain the element v, i.e., $\mathcal{S}_v = \{s \in S : v \in s\}$

```
cp.post(setGlobalCardinality(l,SA,u));
```

Constraints on FS variables

Set Global Cardinality

Table 1. Intersection × Cardinality.

	$\forall i < j \ \ldots$			
$\forall k \ \ldots$	$\lvert X_i \cap X_j \rvert = 0$	$\lvert X_i \cap X_j \rvert \leq k$	$\lvert X_i \cap X_j \rvert \geq k$	$\lvert X_i \cap X_j \rvert = k$
-	`Disjoint` polynomial *decomposable*	`Intersect`$_\leq$ polynomial *decomposable*	`Intersect`$_\geq$ polynomial *decomposable*	`Intersect`$_=$ NP-hard *not decomposable*
$\lvert X_k \rvert > 0$	`NEDisjoint` polynomial *not decomposable*	`NEIntersect`$_\leq$ polynomial *decomposable*	`NEIntersect`$_\geq$ polynomial *decomposable*	`FCIntersect`$_=$ NP-hard *not decomposable*
$\lvert X_k \rvert = m_k$	`FCDisjoint` poly on sets, NP-hard on multisets *not decomposable*	`FCIntersect`$_\leq$ NP-hard *not decomposable*	`FCIntersect`$_\geq$ NP-hard *not decomposable*	`NEIntersect`$_=$ NP-hard *not decomposable*

Table 2. Partition + Intersection × Cardinality.

	$\bigcup_i X_i = X \ \wedge \ \forall i < j \ \ldots$			
$\forall k \ \ldots$	$\lvert X_i \cap X_j \rvert = 0$	$\lvert X_i \cap X_j \rvert \leq k$	$\lvert X_i \cap X_j \rvert \geq k$	$\lvert X_i \cap X_j \rvert = k$
-	`Partition`: polynomial *decomposable*	?	?	?
$\lvert X_k \rvert > 0$	`NEPartition`: polynomial *not decomposable*	?	?	?
$\lvert X_k \rvert = m_k$	`FCPartition` polynomial on sets, NP-hard on multisets *not decomposable*	?	?	?

Sonet **problem**

Optical fiber network design

Sonet problem

Input: weighted undirected demand graph $G = (N, E; d)$, where each node $u \in N$ represents a client and weighted edges $(u, v) \in E$ correspond to traffic demands of a pair of clients.

Two nodes can communicate, only if they join the same ring; nodes may join more than one ring. We must respect:

- maximum number of rings r
- maximum number of clients per ring a
- maximum bandwidth capacity of each ring c

Task: find a topology that minimizes the sum, over all rings, of the number of nodes that join each ring while clients' traffic demands are met.

Sonet **problem**

Sonet problem

A solution of the SONET problem is an assignment of rings to nodes and of capacity to demands such that

- all demands of each client pairs are satisfied;
- the ring traffic does not exceed the bandwidth capacity;
- at most r rings are used;
- at most a ADMs on each ring;
- the total number of ADMs used is minimized.

Sonet: **variables**

- Set variable X_i represents the set of nodes assigned to ring i
- Set variable Y_u represents the set of rings assigned to node u
- Integer variable Z_{ie} represents the amount of bandwidth assigned to demand pair e on ring i.

Sonet: **model**

$$
\begin{aligned}
\min \quad & \sum_{i \in R} |X_i| & \\
\text{s.t.} \quad & |Y_u \cap Y_v| \geq 1, & \forall (u,v) \in E, \\
& Z_{i,(u,v)} > 0 \Rightarrow i \in (Y_u \cap Y_v), & \forall i \in R, (u,v) \in E, \\
& Z_{ie} = d(e), & \forall e \in E, \\
& u \in X_i \Leftrightarrow i \in Y_u, & \forall \in R, u \in N, \\
& |X_i| \leq a, & \forall i \in R \\
& \sum_{e \in E} Z_{ie} \leq c, & \forall i \in R. \\
& X_i \preceq X_j, & \forall i,j \in R : i < j.
\end{aligned}
$$

Outline

1. Global Variables

2. Graph Variables

Graph Variables

Definition

A graph variable is simply two set variables V and E, with an inherent constraint $E \subseteq V \times V$.

Hence, the domain $D(G) = [lb(G), ub(G)]$ of a graph variable G consists of:

- mandatory vertices and edges $lb(G)$ (the lower bound graph) and
- possible vertices and edges $ub(G) \setminus lb(G)$ (the upper bound graph).

The value assigned to the variable G must be a subgraph of $ub(G)$ and a super graph of the $lb(G)$.

Bound consistency on Graph Variables

Graph variables are convinient for possiblity of efficient filtering algorithms

Example:

`Subgraph(G,S)`

specifies that S is a subgraph of G. Computing bound consistency for the `subgraph` constraint means the following:

1. If $lb(S)$ is not a subgraph of $ub(G)$, the constraint has no solution (consistency check).
2. For each $e \in ub(G) \cap lb(S)$, include e in $lb(G)$.
3. For each $e \in ub(S) \setminus ub(G)$, remove e from $ub(S)$.

Constraint on Graph Variables

- Tree constraint: enforces the partitioning of a digraph into a set of vertex-disjoint anti-arborescences. (see, [Beldiceanu2005])
- Weghted Spanning Tree constraint: given a weighted undirected graph $G = (V, E)$ and a weight K, the constraint enforces that T is a spanning tree of cost at most K (see, [Regin2008,2010] and its application to the TSP [Rousseau2010]).
- Shorter Path constraint: given a weighted directed graph $G = (N, A)$ and a weight K, the constraint specifies that P is a subset of G, corresponding to a path of cost at most K. (see, [Sellmann2003, Gellermann2005])
- (Weighted) Clique Constraint, (see, [Regin2003]).

References

Bessiere C., Hebrard E., Hnich B., and Walsh T. (2004). **Disjoint, partition and intersection constraints for set and multiset variables**. In *Principles and Practice of Constraint Programming – CP 2004*, edited by M. Wallace, vol. 3258 of **Lecture Notes in Computer Science**, pp. 138–152. Springer Berlin / Heidelberg.

Gervet C. (2006). **Constraints over structured domains**. In *Handbook of Constraint Programming*, edited by F. Rossi, P. van Beek, and T. Walsh, chap. 10, pp. 329–376. Elsevier.

van Hoeve W. and Katriel I. (2006). **Global constraints**. In *Handbook of Constraint Programming*, chap. 6. Elsevier.