DM826 – Spring 2012

Modeling and Solving Constrained Optimization Problems

Lecture 12
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
  Scheduling
- Search
- Set variables
- Symmetries

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. Set 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 coverered 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:

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

$\leadsto$ 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, $v \in s(x)$.

Social Golfers Problem

Find a schedule for a golf tournament:

- $g \cdot s$ golfers
- who want to play a tournament in g groups of s golfers each over w weeks
- such that no two golfers play against each other more than once during the tournament.

A solution for the instance $w = 4, g = 3, s = 3$
(players are numbered from 0 to 8)

	Group 0			*Group 1*			*Group 2*		
Week 0	0	1	2	3	4	5	6	7	8
Week 1	0	3	6	1	4	7	2	5	8
Week 2	0	4	8	1	5	6	2	3	7
Week 3	0	5	7	1	3	8	2	4	6

Model

See script

In Gecode

```
Space.setvar(int glbMin, int glbMax, int lubMin, int lubMax, int cardMin=
     MIN, int cardMax=MAX)
```

```
A = m.setvar(0, 1, 0, 5, 3, 3)
```

```
m.glbValues(A): [0, 1]  # lists of ints representing the greatest lower set bound
m.glbSize(A): 2  # num. of elements in the greatest lower bound
m.glbMin(A): 0  # minimum element of greatest lower bound
m.glbMax(A): 1  # maximum of greatest lower bound
m.glbRanges(A): [(0, 1)]  # lists of pairs of ints representing the gl set bound

m.lubValues(A): [0, 1, 2, 3, 4, 5]
m.lubSize(A): 6  # num. of elements in the least upper bound
m.lubMin(A): 0  # minimum element of least upper bound
m.lubMax(A): 5  # maximum element of least upper bound
m.lubRanges(A): [(0, 5)]

m.unknownValues(A): [2, 3, 4, 5]
m.unknownSize(A): 4  # num. of unknown elements (elements in lub but not in glb)
m.unknownRanges(A): [(2, 5)]

m.cardMin(A): 3  # cardinality minimum
m.cardMax(A): 3  # cardinality maximum
```

In Gecode


```
Space.setvar(IntSet glb, int lubMin, int lubMax, int cardMin=MIN, int
     cardMax=MAX)
```

```
A = m.setvar(intset(), 0, 5, 0, 4)
```

```
m.glbValues(A): []  # lists of ints representing the greatest lower set bound
m.glbSize(A): 0 # num. of elements in the greatest lower bound
m.glbMin(A): 1073741823 # minimum element of greatest lower bound
m.glbMax(A): -1073741823 # maximum of greatest lower bound
m.glbRanges(A): []  # lists of pairs of ints representing the corresponding set bounds

m.lubValues(A): [0, 1, 2, 3, 4, 5]
m.lubSize(A): 6 # num. of elements in the least upper bound
m.lubMin(A): 0 # minimum element of least upper bound
m.lubMax(A): 5 # maximum element of least upper bound
m.lubRanges(A): [(0, 5)]

m.unknownValues(A): [0, 1, 2, 3, 4, 5]
m.unknownSize)(A): 6 # num. of unknown elements (elements in lub but not in glb)
m.unknownRanges(A): [(0, 5)]

m.cardMin(A): 0 # cardinality minimum
m.cardMax(A): 4 # cardinality maximum
```

In Gecode


```
Space.setvar(int glbMin, int glbMax, IntSet lub, int cardMin=MIN, int
     cardMax=MAX)
```

```
A = m.setvar(1, 3, intset([(1,4),(8,12)]), 2, 4)
```

```
m.glbValues(A): [1, 2, 3]  # lists of ints representing the greatest lower set bound
m.glbSize(A): 3  # num. of elements in the greatest lower bound
m.glbMin(A): 1  # minimum element of greatest lower bound
m.glbMax(A): 3  # maximum of greatest lower bound
m.glbRanges(A): [(1, 3)]  # lists of pairs of ints representing the corresponding set
      bounds

m.lubValues(A): [1, 2, 3, 4, 8, 9, 10, 11, 12]
m.lubSize(A): 9  # num. of elements in the least upper bound
m.lubMin(A): 1  # minimum element of least upper bound
m.lubMax(A): 12  # maximum element of least upper bound
m.lubRanges(A): [(1, 4), (8, 12)]

m.unknownValues(A): [4, 8, 9, 10, 11, 12]
m.unknownSize)(A): 6  # num. of unknown elements (elements in lub but not in glb)
m.unknownRanges(A): [(4, 4), (8, 12)]

m.cardMin(A): 3  # cardinality minimum
m.cardMax(A): 4  # cardinality maximum
```

In Gecode

Array of set variables:

```
Space.setvars(int N, ...)
groups = m.setvars(g*w, intset(), 0, g*s-1, s, s)
```

size $g \cdot w$, where each group can contain the players $0 \ldots g \cdot s - 1$ and has cardinality s

```
w = 4;
g = 3;
s = 3;

golfers = g * s;
Golfer = range(golfers)

m=space()

groups = m.setvars(g*w, intset(), 0, g*s-1, s, s)
```

Constraints on FS variables

Domain constraints

```
Space.dom(x, SRT_SUB, 1, 10);
Space.dom(x, SRT_SUP, 1, 3);
Space.dom(y, SRT_DISJ, IntSet(4, 6));
```

```
Space.cardinality(x, 3, 5);
```

Constraints on FS variables

Relation constraints

```
Space.rel(x, SRT_SUB, y)
```

```
Space.rel( x, IRT_GR, y)
```

Constraints on FS variables
Set operations


```
Space.rel(x, SOT_UNION, y, SRT_EQ, z)
```

```
Space.rel(SOT_UNION, x, y)
```

Constraints on FS variables

Element


```
Space.element(x, y, z)
```

for an array of set variables or constants x,
an integer variable y,
and a set variable z.

It constrains z to be the element of array x at index y (where the index starts at 0).

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., $\mathcal{S}_v = \{s \in S : v \in s\}$

(not present in gecode)

Constraints on FS variables
Set Global Cardinality

Bessiere et al. [2004]

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*	$\texttt{Intersect}_{\leq}$ polynomial *decomposable*	$\texttt{Intersect}_{\geq}$ polynomial *decomposable*	$\texttt{Intersect}_{=}$ NP-hard *not decomposable*
$\lvert X_k \rvert > 0$	NEDisjoint polynomial *not decomposable*	$\texttt{NEIntersect}_{\leq}$ polynomial *decomposable*	$\texttt{NEIntersect}_{\geq}$ polynomial *decomposable*	$\texttt{FCIntersect}_{=}$ NP-hard *not decomposable*
$\lvert X_k \rvert = m_k$	FCDisjoint poly on sets, NP-hard on multisets *not decomposable*	$\texttt{FCIntersect}_{\leq}$ NP-hard *not decomposable*	$\texttt{FCIntersect}_{\geq}$ NP-hard *not decomposable*	$\texttt{NEIntersect}_{=}$ 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*	?	?	?

Constraints on FS variables
Constraints connecting set and integer variables

the integer variable y is equal to the cardinality of the set variable x.

```
Space.cardinality(x, y);
```

Minimal and maximal elements of a set:

```
Space.min(x, y);
```

Weighted sets: assigns a weight to each possible element of a set variable x, and then constrains an integer variable y to be the sum of the weights of the elements of x

```
e = [6, 1, 3, 4, 5, 7, 9]
w = [6, -1, 4, 1, 1, 3, 3]
Space.weights(e, w, x, y)
```

enforces that x is a subset of $\{1, 3, 4, 5, 7, 9\}$ (the set of elements), and that y is the sum of the weights of the elements in x, where the weight of the element 1 would be -1, the weight of 3 would be 4 and so on.
Eg. Assigning x to the set $\{3, 7, 9\}$ would therefore result in y be set to $4 + 3 + 3 = 10$

Constraints on FS variables
Channeling constraints

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

```
Space.channel(X, SA)
```

$$X_i = j \iff i \in SA_j \quad 0 \leq i, j < |X|$$

$$SA_i = s \iff \forall j \in s : X_j = i$$

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

Constraints on FS variables
Channeling constraints

set variable S and an array of Boolean variables X

```
Space.channel(X, S)
```

$$X_i = 1 \iff i \in S \quad 0 \leq i < |X|$$

```
S = {1,2}
X = [1,1,0]
```

Constraints on FS variables

Channeling constraints

An array of integer variables x can be channeled to a set variable S using

```
Space.rel(SOT_UNION, x, S)
```

constrains S to be the set $\{x_0, \ldots, x_{|x|-1}\}$

```
Space.channelSorted(x, y);
```

constrains y to be the set $\{x_0, \ldots, x_{|x|-1}\}$, and the integer variables in x are sorted in increasing order ($x_i < x_{i+1}$ for $0 \leq i < |x|$)

Constraints on FS variables

Channeling constraints

SA_1 and SA_2 two arrays of set variables

```
Space.channel(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

Convexity

set variable x:

```
Space.convex(x)
```

The convex hull of a set s is the smallest convex set containing s

```
Space.convex(x, y)
```

enforces that the set variable y is the convex hull of the set variable x.

Constraints on FS variables

Sequence constraints

enforce an order among an array of set variables x

```
Space.sequence(x)
```

sets x being pairwise disjoint, and furthermore $\max(x_i) < \min(x_{i+1})$ for all $0 \leq i < |x| - 1$

```
Space.sequence(x, y)
```

additionally constrains the set variable y to be the union of the x.

Constraints on FS variables
Value precedence constraints

enforce that a value precedes another value in an array of set variables. x is an array of set variables and both s and t are integers,

```
Space.precede(x, s, t)
```

if there exists j $(0 \leq j < |x|)$ such that $s \in x_j$ and $t \in x_j$, then there must exist i with $i < j$ such that $s \in x_i$ and $t \in x_i$

Social golfers
Model with set variables

See script

Outline

1. Set 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. 17, pp. 329–376. Elsevier.

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