DM826 (5 ECTS - 3rd Quarter) Modeling and Solving Constrained Optimization Problems

Modeller og løsningsmetoder for optimeringsproblemer med sidebetingelser

(Constraint Programming)

Marco Chiarandini
lektor, IMADA
www.imada.sdu.dk/~marco/

Constraint Programming

Programming with Constraints =

representation (language) +

reasoning (search + propagation)

Background:

- Programming languages (DM509)
  - logic based representation
- Search Algorithms (several IMADA courses)

Problems with Constraints

Social Golfer Problem

9 golfers: {1, 2, 3, 4, 5, 6, 7, 8, 9}

wish to play in groups of 3 players for 4 days

such that no golfer plays in the same group with any other golfer more than just once.

Is it possible?

	Group 1	Group 2	Group 3
Day 0	? ? ?	? ? ?	? ? ?
Day 1	? ? ?	? ? ?	? ? ?
Day 2	? ? ?	? ? ?	? ? ?
Day 3	? ? ?	? ? ?	? ? ?

Constraint Programming

```
players = 9;
groupSize = 3;
weeks = 4;
groups = players/groupSize;

range Players = 1..players;
range Days = 1..days;
range Group = 1..groups;

Solver<CP> m();
var<CP>{int} assign[Players, Days](m,  Group);

explore<m> {
 // C1: Each group has exactly groupSize players
 forall (gr in Group, d in Days)
    m.post(sum (p in Players) (assign[g,d] == gr) == groupSize);
 // C2: Each pair of players only meets once
 forall (p1 in Players, p2 in Players: p1 != p2,
      d1 in Days, d2 in Days: d1!=d2)
    m.post((assign[p1,d1] == assign[p2,d1])
           + (assign[p1,d2] == assign[p2,d2]) == 1);
} using {
  label(m);
}
```

Constraint Programming

```
from gecode import *
m = space()

p = 9 # number of players
s = 3 # size of groups
g = 9/3 # number of groups
w = 4 # number of weeks

groups = m.setvars(g*w, intset(), 0, p-1, s, s)
schedule = Matrix(g, w, groups)
allPlayers = m.setvar(0, p-1, 0, p)

# In each weeks, groups must be disjoint and contain all players
for i in range(g):
     z1 = m.setvars(g, intset(), 0, p-1, 0, p)
     m.rel(SOT_DUNION, schedule[i].row(i), z1[i])
     m.rel(z1[i], SRT_EQ, allPlayers)

# at most one player overlaps between groups
for i,j in itertools.combinations(range(g*w), 2):
    z2 = m.setvar(intset(), 0, p-1, 0, p))
    m.rel(groups[i], SOT_INTER, groups[j], SRT_EQ, z2)
    m.cardinality(z2, 0, 1)

m.branch(groups, SET_VAR_MIN_MIN, SET_VAL_MIN_INC);
```

Solution: Assign and Propagate

Golfers

	Group 1	Group 2	Group 3
Day 0			
Day 1			
Day 2			
Day 3			

Groups

	Day 0	Day 1	Day 2	Day 3
Golfer 0	{1,2,3}	{1,2,3}	{1,2,3}	{1,2,3}
Golfer 1	{1,2,3}	{1,2,3}	{1,2,3}	{1,2,3}
Golfer 2	{1,2,3}	{1,2,3}	{1,2,3}	{1,2,3}
Golfer 3	{1,2,3}	{1,2,3}	{1,2,3}	{1,2,3}
Golfer 4	{1,2,3}	{1,2,3}	{1,2,3}	{1,2,3}
Golfer 5	{1,2,3}	{1,2,3}	{1,2,3}	{1,2,3}
Golfer 6	{1,2,3}	{1,2,3}	{1,2,3}	{1,2,3}
Golfer 7	{1,2,3}	{1,2,3}	{1,2,3}	{1,2,3}
Golfer 8	{1,2,3}	{1,2,3}	{1,2,3}	{1,2,3}

Solution: Assign and Propagate

Golfers

	Group 1	Group 2	Group 3
Day 0	0 1 2		
Day 1			
Day 2			
Day 3			

Groups

	Day 0	Day 1	Day 2	Day 3
Golfer 0	1	{1,2,3}	{1,2,3}	{1,2,3}
Golfer 1	1	{1,2,3}	{1,2,3}	{1,2,3}
Golfer 2	1	{1,2,3}	{1,2,3}	{1,2,3}
Golfer 3	{2,3}	{1,2,3}	{1,2,3}	{1,2,3}
Golfer 4	{2,3}	{1,2,3}	{1,2,3}	{1,2,3}
Golfer 5	{2,3}	{1,2,3}	{1,2,3}	{1,2,3}
Golfer 6	{2,3}	{1,2,3}	{1,2,3}	{1,2,3}
Golfer 7	{2,3}	{1,2,3}	{1,2,3}	{1,2,3}
Golfer 8	{2,3}	{1,2,3}	{1,2,3}	{1,2,3}

Solution: Assign and Propagate

Golfers

	Group 1	Group 2	Group 3
Day 0	0 1 2	3 4 5	
Day 1			
Day 2			
Day 3			

Groups

	Day 0	Day 1	Day 2	Day 3
Golfer 0	1	{1,2,3}	{1,2,3}	{1,2,3}
Golfer 1	1	{1,2,3}	{1,2,3}	{1,2,3}
Golfer 2	1	{1,2,3}	{1,2,3}	{1,2,3}
Golfer 3	2	{1,2,3}	{1,2,3}	{1,2,3}
Golfer 4	2	{1,2,3}	{1,2,3}	{1,2,3}
Golfer 5	2	{1,2,3}	{1,2,3}	{1,2,3}
Golfer 6	{3}	{1,2,3}	{1,2,3}	{1,2,3}
Golfer 7	{3}	{1,2,3}	{1,2,3}	{1,2,3}
Golfer 8	{3}	{1,2,3}	{1,2,3}	{1,2,3}

Solution: Assign and Propagate

Golfers

	Group 1	Group 2	Group 3
Day 0	0 1 2	3 4 5	6 7 8
Day 1			
Day 2			
Day 3			

Groups

	Day 0	Day 1	Day 2	Day 3
Golfer 0	1	{1,2,3}	{1,2,3}	{1,2,3}
Golfer 1	1	{1,2,3}	{1,2,3}	{1,2,3}
Golfer 2	1	{1,2,3}	{1,2,3}	{1,2,3}
Golfer 3	2	{1,2,3}	{1,2,3}	{1,2,3}
Golfer 4	2	{1,2,3}	{1,2,3}	{1,2,3}
Golfer 5	2	{1,2,3}	{1,2,3}	{1,2,3}
Golfer 6	3	{1,2,3}	{1,2,3}	{1,2,3}
Golfer 7	3	{1,2,3}	{1,2,3}	{1,2,3}
Golfer 8	3	{1,2,3}	{1,2,3}	{1,2,3}

Solution: Assign and Propagate

Golfers

	Group 1	Group 2	Group 3
Day 0	0 1 2	3 4 5	6 7 8
Day 1	0		
Day 2			
Day 3			

Groups

	Day 0	Day 1	Day 2	Day 3
Golfer 0	1	1	{1,2,3}	{1,2,3}
Golfer 1	1	{2,3}	{1,2,3}	{1,2,3}
Golfer 2	1	{2,3}	{1,2,3}	{1,2,3}
Golfer 3	2	{1,2,3}	{1,2,3}	{1,2,3}
Golfer 4	2	{1,2,3}	{1,2,3}	{1,2,3}
Golfer 5	2	{1,2,3}	{1,2,3}	{1,2,3}
Golfer 6	3	{1,2,3}	{1,2,3}	{1,2,3}
Golfer 7	3	{1,2,3}	{1,2,3}	{1,2,3}
Golfer 8	3	{1,2,3}	{1,2,3}	{1,2,3}

Solution: Assign and Propagate

Golfers

	Group 1	Group 2	Group 3
Day 0	0 1 2	3 4 5	6 7 8
Day 1	0	1	
Day 2			
Day 3			

Groups

	Day 0	Day 1	Day 2	Day 3
Golfer 0	1	1	{1,2,3}	{1,2,3}
Golfer 1	1	2	{1,2,3}	{1,2,3}
Golfer 2	1	{3}	{1,2,3}	{1,2,3}
Golfer 3	2	{1,2,3}	{1,2,3}	{1,2,3}
Golfer 4	2	{1,2,3}	{1,2,3}	{1,2,3}
Golfer 5	2	{1,2,3}	{1,2,3}	{1,2,3}
Golfer 6	3	{1,2,3}	{1,2,3}	{1,2,3}
Golfer 7	3	{1,2,3}	{1,2,3}	{1,2,3}
Golfer 8	3	{1,2,3}	{1,2,3}	{1,2,3}

Solution: Assign and Propagate

Golfers

	Group 1	Group 2	Group 3
Day 0	0 1 2	3 4 5	6 7 8
Day 1	0 3 6	1 4 7	2 5 8
Day 2	0 4 8	1 5 6	2 3 7
Day 3	0 5 7	1 3 8	2 4 6

Groups

	Day 0	Day 1	Day 2	Day 3
Golfer 0	1	1		
Golfer 1	1	2		
Golfer 2	1	{3}		
Golfer 3	2			
Golfer 4	2			
Golfer 5	2			
Golfer 6	3			
Golfer 7	3			
Golfer 8	3			

Contents of the Course

- **Modelling and Applications**

  Integer variables, set variables, float variables, constraints

- **Principles**

  Consistency levels

- **Filtering Algorithms**

  Alldifferent, cardinality, regular expressions, …

- **Search**

  Backtracking, Strategies

- **Symmetry Breaking**
- **Restart Techniques**
- **Programming**

  Gecode and python (or C++)

14 Introductions

8 Training classes

Course Organization - Aims

- understanding the fundamental concepts underlying constraint programming
- developing skills in modeling and solving problems
- developing skills in taking advantage of strong algorithmic techniques
- getting acquainted with a CP system (Gecode) and develop applications

Prerequisites

The content of DM545 and DM509 would help

Final Assessment (5 ECTS)

Three mandatory assignments:

- Two during the course (pass/fail) with feedback
- One at the end, graded

External examiner

Course Material

- Text book
  - F. Rossi, P. van Beek and T. Walsh (ed.). Handbook of Constraint Programming, Elsevier, 2006
- Photocopies
- Slides
- Gecode manuals and data sets
- www.imada.sdu.dk/~marco/DM826