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

Modeller og løsningsmetoder for optimeringsproblemer med sidebetingelser

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

Main Goal

To cover the only missing technique for solving combinatorial problems not yet treated at IMADA:

Constraint Programming

and to present an unifying framework for

- Mixed Integer Programming
- Constraint Programming and
- Heuristics

Example: Tournament Scheduling

Social Golfer Problem

9 golfers wish to play in g=3 groups of s=3 players for d=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	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

Solution: Assign and Propagate

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

	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

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

	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

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

	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

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

	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

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

	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}

Constraint Programming

```
int days = 4;
int groups = 3;
int groupSize = 3;
int golfers = groups * groupSize;

range Golfer = 1..golfers;
range Days = 1..days;
range Group = 1..groups;

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

explore<m> {
 // C1: Each group has exactly groupSize players
 forall (gr in Group, d in Days)
    m.post(sum (g in Golfer) (assign[g,d] == gr) == groupSize);
 // C2: Each pair of players only meets once
 forall (g1 in Golfer, g2 in Golfer: g1 != g2, d1 in Days, d2 in Days: d1!=d2)
    m.post((assign[g1,d1] == assign[g2,d1]) + (assign[g1,d2] == assign[g2,d2])
== 1);
} using {
  label(m);
}
```

Constraint Satisfaction Model

Input:
a set of variables $X_1, X_2, \ldots, X_n$ each variable has a non-empty domain D_i of possible values
a set of constraints. Each constraint C_i involves some subset of the variables and specifies the allowed combination of values for that subset.

Task:
find an assignment of values to all the variables
$\{X_i = v_i, X_j = v_j, \ldots\}$
such that it is consistent, that is, it does not violate any constraint

Constraint Programming

- Modelling
  - variables, values, constraints, heuristics, symmetries, ...
- Compute with possible values
  - rather than enumerating assignments
- Prune inconsistent values
  - constraint propagation
- Search
  - branch (define the search tree)

Constraint Propagators

arithmetic and logical constraints

- element constraints
- table constraint
- alldifferent and minAssignment
- atleast, atmost, cardinality
- binary- and multi-knapsack
- scheduling constraints: disjunctive and cumulative
- graph constraints: circuit and minCircuit
- sequence
- stretch and regular

All-Intervals Problem

Given the twelve standard pitch-classes (c, c#, d, ...), represented by numbers 0,1,...,11,
find a series in which each pitch-class occurs exactly once and in which the musical intervals between neighbouring notes cover the full set of intervals from the minor second (1 semitone) to the major seventh (11 semitones).

```
int n = 12;
int sum_distinct = ((n+1)*n) / 2;

var<CP>{int} x[1..n](m, 1..n);
var<CP>{int} diffs[1..n-1](m, 1..n-1);

exploreall<m> {
  forall(k in 1..n-1)
    m.post(diffs[k] == abs(x[k+1] - x[k]));
    m.post(alldifferent(x));
    m.post(alldifferent(diffs));

    // symmetry breaking
    m.post(x[1] < x[n-1]);
    m.post(diffs[1] < diffs[2]);
} using {
   label(m)
}
```

Course Organization

Aims

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

Course Organization

Contents

- Principles

  (modelling, domain consisteny, search techniques)

- Algorithmics

  (to propagate constraints efficiently)

- Applications

  (academic problems)

- Programming

  (Gecode or Comet, modelling + C++)

14 Lectures

4 Tutorials

Prerequisites

The content of DM515 (Intro to Linear and Integer Programming) must be known

Final Assessment (5 ECTS)

- Three mandatory assignments:
  - Two during the course (pass/fail)
  - One at the end
- External examiner

Course Material

- Text book
  - Hooker, J. N. Integrated Methods for Optimization. Springer, 2007
  - F. Rossi, P. van Beek and T. Walsh (ed.). Handbook of Constraint Programming, Elsevier, 2006
  - Christian Schulte, Guido Tack, Mikael Z. Lagerkvist, Modelling and Programming with Gecode, 2010
- Photocopies
- Slides
- Gecode and data sets
- www.imada.sdu.dk/~marco/DM826