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

Modeller og løsningsmetoder for optimeringsproblemer med sidebetingelser

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

Context: Solving Problems like these

Social Golfer Problem

9 golfers 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	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

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

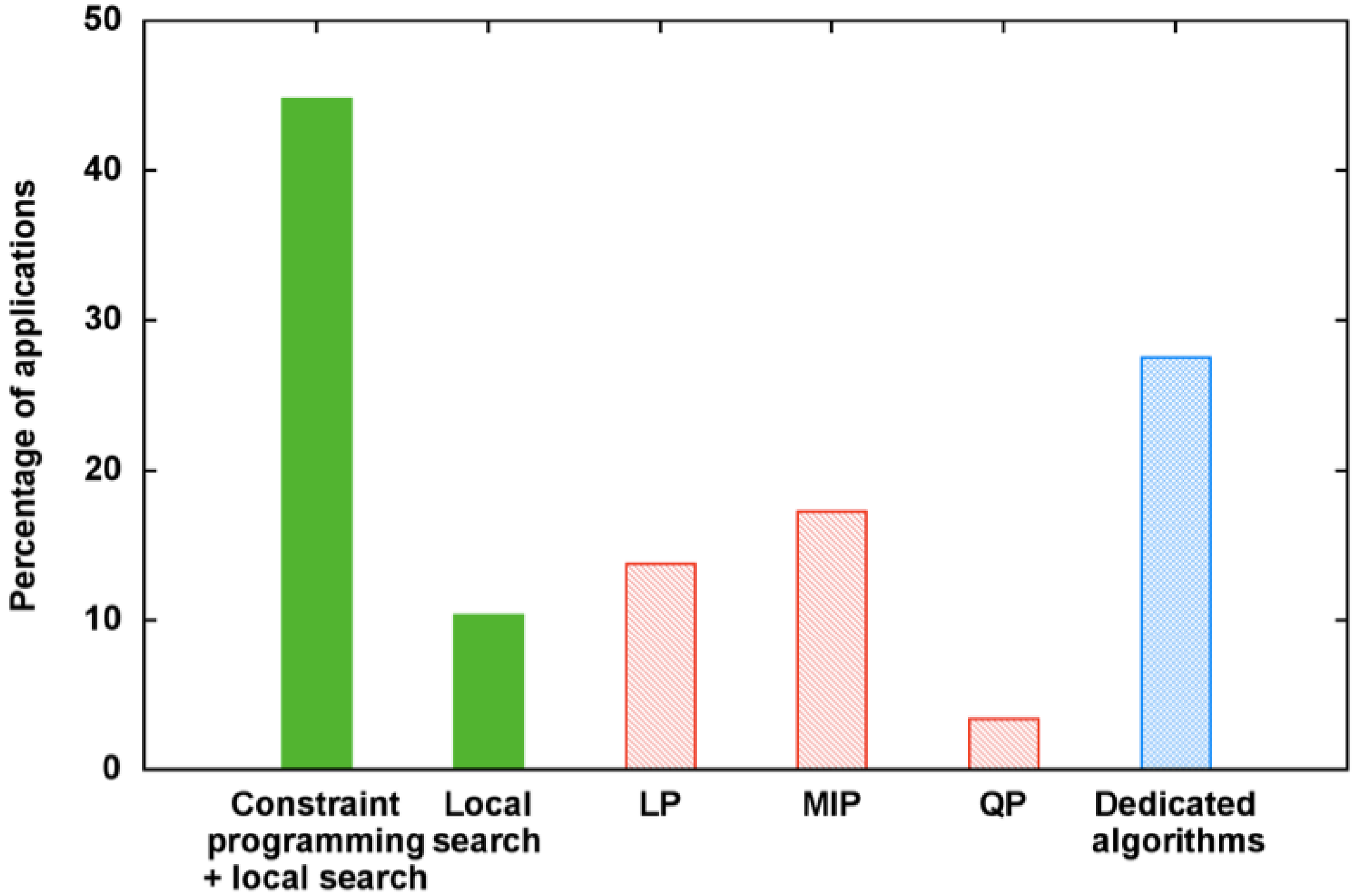
	Day 0	Day 1	Day 2	Day 3
Golfer 0	1	1	{1,2,3}	{1,2,3}
Golfer 1	1	2	{1,2,3}	{1,2,3}
Golfer 2	1	{3}	{1,2,3}	{1,2,3}
Golfer 3	2	{1,2,3}	{1,2,3}	{1,2,3}
Golfer 4	2	{1,2,3}	{1,2,3}	{1,2,3}
Golfer 5	2	{1,2,3}	{1,2,3}	{1,2,3}
Golfer 6	3	{1,2,3}	{1,2,3}	{1,2,3}
Golfer 7	3	{1,2,3}	{1,2,3}	{1,2,3}
Golfer 8	3	{1,2,3}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

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;

Solver<CP> m();
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);
}
```

Main Goal

Distribution of technology used at Google for optimization applications developed by the operations research team



Constraint Programming

Programming with Constraints =

representation (language) +

reasoning (search + propagation)

- Programming languages (DM509)
  logic based representation
- Mixed Integer Programming and Networks (DM515)
  modeling and search

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

Constraint Propagators

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

Course Organization

Aims

- understanding the fundamental concepts underlying constraint programming,
- developing skills in modeling 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 consistency, search techniques)

- Algorithmics

  (to propagate constraints efficiently)

- Applications

  (academic problems)

- Programming

  (Comet and Gecode, modelling and C++)

Course Organization

Contents

- Principles

  (modelling, domain consistency, search techniques)

- Algorithmics

  (to propagate constraints efficiently)

- Applications

  (academic problems)

- Programming

  (Comet and Gecode, modelling and C++)

14 Lectures

8 Exercises

Prerequisites

The content of DM515 and DM509 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
  - F. Rossi, P. van Beek and T. Walsh (ed.). Handbook of Constraint Programming, Elsevier, 2006
- Photocopies
- Slides
- Comet and Gecode manuals and data sets
- www.imada.sdu.dk/~marco/DM826