DM841 (10 ECTS - E23)

Heuristics and Constraint Programming for Discrete Optimization

[Heuristikker og Constraint Programmering for Diskret Optimering]

Marco Chiarandini, Asc Prof, IMADA
imada.sdu.dk/~marco

Video (14 min): https://imada.sdu.dk/~marco/Videos/dm841.mp4

Course Formalities

Target students:	computer science, applied math, math and economics and data science at 3rd semester (but challenging) or 5th semester of Bachelor degree or at Master level
Prerequisites:	✔ Programming (Java or Python)

Decision Problems with Discrete Variables

Social Golfer Problem (Combinatorial Design)

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

- 9 golfers: 1, 2, 3, 4, 5, 6, 7, 8, 9
- wish to play in groups of 3 players in 4 days
- such that no golfer plays in the same group with any other golfer more than just once.

Is it possible?

Solution Paradigms

- Dedicated algorithms
  (eg.: enumeration, branch and bound, dynamic programming)
- Integer Linear Programming (DM871/DM545)
- Constraint Programming:
  representation (modeling) + reasoning (search + inference)
- Heuristics & Metaheuristics
  representation (modeling) + reasoning (search)
- Others (SAT, SMT, etc.)

Constraint Programming

Modeling

Modelling in MIP and SAT

Modelling in CP

Constraint Programming

Modeling

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	{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}

Integer variables:

`assign[i,j]` variable whose value is from the domain $\{1, 2, 3\}$

Constraints:

C1: each group has exactly groupSize players

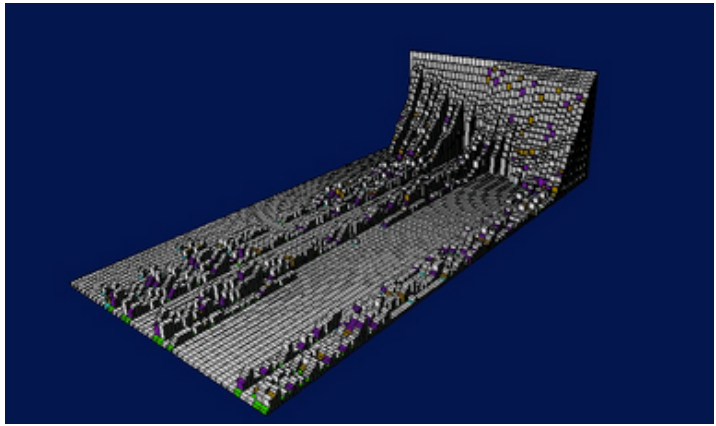
C2: each pair of players only meets once

Constraint Programming

MiniZinc Model with Integer Variables

```
int: golfers = 9;
int: groupSize = 3;
int: days = 4;
int: groups = golfers/groupSize;

set of int: Golfer = 1..golfers;
set of int: Day = 1..days;
set of int: Group = 1..groups;

array[Golfer, Day] of var Group: assign; % Variables

constraint
  % C1: Each group has exactly groupSize players
  forall (gr in Group, d in Day)( % c1
   sum (g in Golfer) (bool2int(assign[g,d] = gr)) = groupSize
  )
  /\
  % C2: Each pair of players only meets at most once
  forall (g1, g2 in Golfer, d1, d2 in Day  where g1 != g2 /\ d1 != d2) (
   (bool2int(assign[g1,d1] = assign[g2,d1]) + bool2int(assign[g1,d2] = assign[g2,d2])) <=1
  );

solve :: int_search([assign[i,j] | i in Golfer, j in Day ],
                    first_fail, indomain_min, complete)  satisfy;
```

Constraint Programming



The solution process proceeds by propagating the constraints on the domanins of the variables (ie, removing values) and tentatively assigning variables until only feasible values are left.

Local Search

Modeling

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

- Variables = solution representation, tentative solution
- Constraints:
  - implicit
  - soft
- Evaluation function

Local Search

Solution: Trial and Error

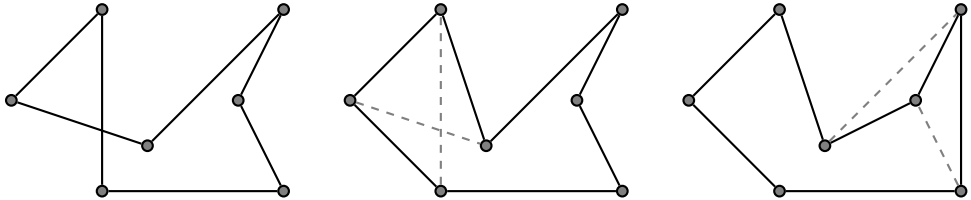
	Group 1	Group 2	Group 3
Day 0	0 1 2	3 4 5	6 7 8
Day 1	**0 4** 6	1 **3 7**	2 5 8
Day 2	**0 4** 8	1 5 6	2 **3 7**
Day 3	0 5 7	1 3 8	2 4 6

Heuristic algorithms: compute, efficiently, good solutions to a problem (without caring for theoretical guarantees on running time and approximation quality).

Local Search

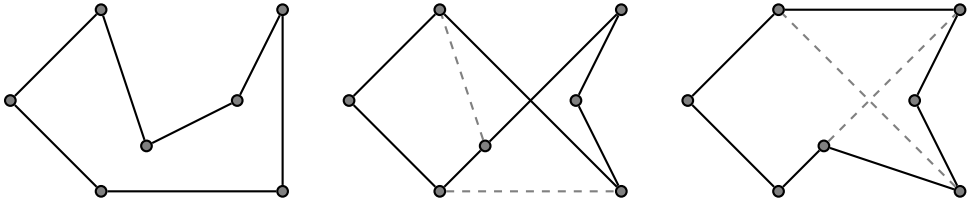
Solution process: local changes

Example on Traveling Salesman Problem:

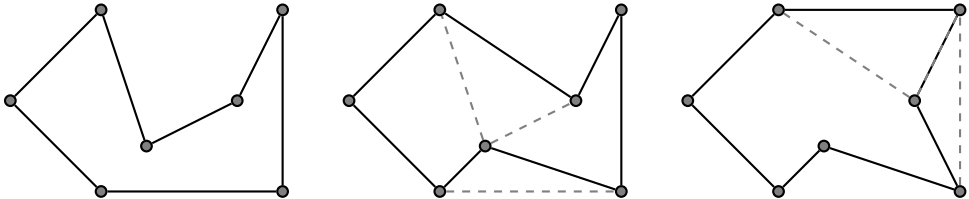


Metaheuristics

Accepting worsening changes



Trying different changes



Contents: Constraint Programming

- Modelling and Applications
  Integer variables, set variables, float variables, constraints
- Principles
  Consistency levels
- Filtering Algorithms
  Alldifferent, cardinality, regular expressions, etc.
- Search:
  Backtracking, Strategies
- Symmetry Breaking
- Restart Techniques
- CP Systems: Minizinc

Contents: Heuristics

- Construction Heuristics
- Local Search
- Metaheuristics
  - Simulated Annealing
  - Iterated Local Search
  - Tabu Search
  - Variable Neighborhood Search
  - Evolutionary Algorithms
  - Ant Colony Optimization
- Programming
  (Python)

Aims & Contents

- modeling discrete optimization problems with constraint programming
- design heuristic algorithms
- implement the algorithms
- assess the programs
- describe with appropriate language
- look at different problems

Assessment (10 ECTS)

Five obligatory assignments:

- individual
- deliverables: program + short written report
- Two graded with external censor,
  final grade given by weighted average

DM841 (10 ECTS - E23)

Heuristics and Constraint Programming for Discrete Optimization

[Heuristikker og Constraint Programmering for Diskret Optimering]

Marco Chiarandini, Asc Prof, IMADA
imada.sdu.dk/~marco

Video (14 min): https://imada.sdu.dk/~marco/Videos/dm841.mp4