UNIVERSITY OF LEEDS

Modelling for Constraint Programming

Barbara Smith

3. Symmetry, Viewpoints

Symmetry in CSPs

UNIVERSITY OF LEEDS

- A symmetry transforms any solution into another
  - Sometimes symmetry is inherent in the problem (e.g. chessboard symmetry in n-queens)
  - Sometimes it's introduced in modelling
- Symmetry causes wasted search effort: after exploring choices that don't lead to a solution, symmetrically equivalent choices may be explored

Example: SONET Rings

UNIVERSITY OF LEEDS

- Split the demand graph into subgraphs (SONET rings):
  - every edge is in at least one subgraph
  - a subgraph has at most 5 nodes
  - minimize total number of nodes in the subgraphs
- Modelled using Boolean variables, x_{ij} , such that $x_{ij} = 1$ if node i is on ring j
- Introduces symmetry between the rings:
  - in the problem, the rings are interchangeable
  - in the CSP, each ring has a distinct number

Symmetry between Values: Car sequencing

UNIVERSITY OF LEEDS

- A natural model has individual cars as the values
  - introduces symmetry between cars requiring the same option
- The model instead has *classes* of car
  - needs constraints to ensure the right number of cars in each class

cars	1	2	3	4	5	6	7	8	9	10
option 1	1	0	0	0	0	0	1	1	1	1
option 2	0	0	1	1	1	1	0	0	1	1
option 3	1	0	0	0	0	0	1	1	0	0
option 4	1	1	0	0	1	1	0	0	0	0
option 5	0	0	1	1	0	0	0	0	0	0

classes	1	2	3	4	5	6
option 1	1	0	0	0	1	1
option 2	0	0	1	1	0	1
option 3	1	0	0	0	1	0
option 4	1	1	0	1	0	0
option 5	0	0	1	0	0	0
no. of cars	1	1	2	2	2	2

Symmetry between Variables: Golfers Problem

UNIVERSITY OF LEEDS

- 32 golfers want to play in 8 groups of 4 each week, so that any two golfers play in the same group at most once. Find a schedule for n weeks
- One viewpoint has 0/1 variables x_{ijkl}:
  - x_{ijkl} = 1 if player i is the j th player in the k th group in week l, and 0 otherwise.
- The players within each group could be permuted in any solution to give an equivalent solution
  - also the groups within each week, the weeks within the schedule and the players themselves

Reformulating to avoid symmetry: Set Variables

UNIVERSITY OF LEEDS

- Eliminate the symmetry between players within a group by using set variables to represent the groups
  - G_{kl} represents the k th group in week l
  - the value of G_{kl} represents the set of players in the group.
- The constraints on these variables are that:
  - the cardinality of each set is 4
  - the sets in any week do not overlap: for all l, the sets G_{kl} , $k = 1,...,8$ have an empty intersection
  - any two sets in different weeks have at most one member in common
- Constraint solvers that support set variables allow constraints of this kind

Symmetry Breaking

- Often, not all the symmetry can be eliminated by remodelling
- Remaining symmetry should be reduced or eliminated:
  - dynamic symmetry breaking methods (SBDS, SBDD, etc.)
  - symmetry-breaking constraints
    - unlike implied constraints, they change the set of solutions
    - can lead to further implied constraints

Example: Template Design

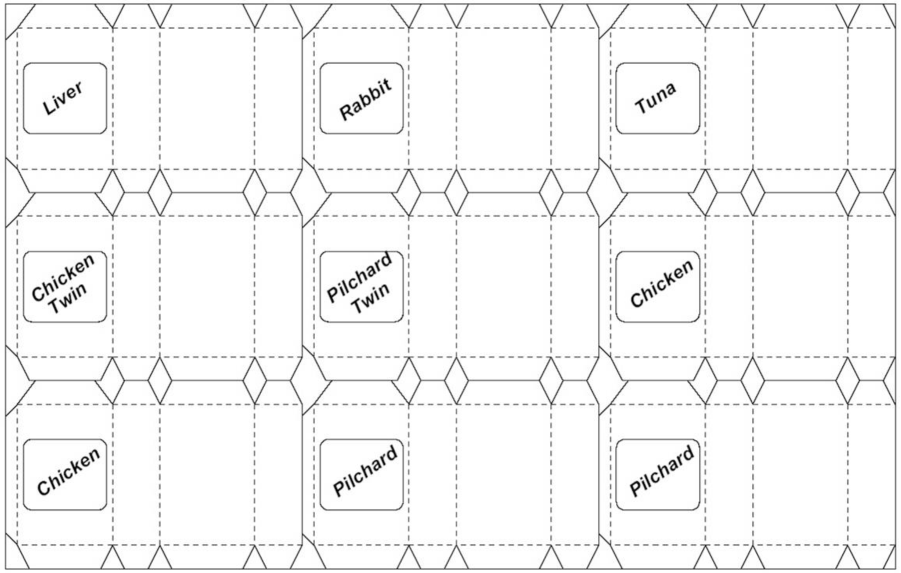
UNIVERSITY OF LEEDS

- Plan layout of printing templates for catfood boxes
- Each template has 9 slots
  - 9 boxes from each sheet of card
- Choose best layout for 1, 2, 3,… templates to minimize waste in meeting order
  - templates are expensive

Flavour	Order (1000s)
Liver	250
Rabbit	255
Tuna	260
Chicken Twin	500
Pilchard Twin	500
Chicken	800
Pilchard	1,100

One Template Solution

UNIVERSITY OF LEEDS



- Could meet the order using only one template
  - print it 550,000 times
  - but this wastes a lot of card

Flavour	Order (1000s)
Liver	250
Rabbit	255
Tuna	260
Chicken Twin	500
Pilchard Twin	500
Chicken	800
Pilchard	1,100

CP Model for Template Design

UNIVERSITY OF LEEDS

- For a fixed number of templates:
- x_{ij} = number of slots allocated to design j in template i
- r_i = run length for template i (number of sheets of card printed from this template)
- $\sum_i x_{ij} r_i \geq d_j \qquad j = 1, 2, \ldots, 7$ where d_j is the order quantity for design j
- minimize $p = \sum_i r_i$ (p = total sheets printed)

Symmetry Breaking & Implied Constraints

UNIVERSITY OF LEEDS

- The templates are indistinguishable
- So add $r_1 \leq r_2 \leq \ldots \leq r_t$
- If there are 2 templates:
  - at most half the sheets are printed from one template, at least half from the other
  - so $r_1 \leq p/2; \; r_2 \geq p/2$
- For 3 templates:
  - $r_1 \leq p/3; \; r_2 \leq p/2; \; r_3 \geq p/3$
- These are *useful* implied constraints
  - they allow tighter constraints on the objective to propagate to the search variables

Changing Viewpoint

UNIVERSITY OF LEEDS

- We can *improve* a CSP model of a problem
  - express the constraints better
  - break the symmetry
  - add implied constraints
- But sometimes it's better just to use a different model
  - i.e. a different viewpoint

Different Viewpoints

UNIVERSITY OF LEEDS

- Reformulate in a standard way, e.g.
  - non-binary to binary translations
  - dual viewpoint for permutation problems
  - Boolean to integer or set viewpoints
- Find a new viewpoint by viewing the problem from a different angle
  - the constraints may express different insights into the problem

Permutation Problems

- A CSP is a permutation problem if:
  - it has the same number of values as variables
  - all variables have the same domain
  - each variable must be assigned a different value
- Any solution assigns a permutation of the values to the variables
- Other constraints determine *which* permutations are solutions
- There is a *dual* viewpoint in which the variables and values are swapped

Example: *n*-queens

UNIVERSITY OF LEEDS

- Standard model
  - a variable for each row, $x_1, x_2, \ldots, x_n$
  - values represent the columns, 1 to n
  - $x_i = j$ means that the queen in row i is in column j
  - n variables, n values, allDifferent($x_1, x_2, \ldots, x_n$)
- Dual viewpoint
  - a variable for each column, $d_1, d_2, \ldots, d_n$; values represent the rows
- In this problem, both viewpoints give the same CSP

Example: Magic Square

UNIVERSITY OF LEEDS

- First viewpoint:
  - variables $x_1, x_2, \ldots, x_9$
  - values represent the numbers 1 to 9
  - The assignment (x_i, j) means that the number in square i is j
- Dual viewpoint
  - a variable for each number, $d_1, d_2, \ldots, d_9$
  - values represent the squares
- Constraints are much easier to express in the first viewpoint
  - see earlier

x_1	x_2	x_3
x_4	x_5	x_6
x_7	x_8	x_9

Boolean Models

UNIVERSITY OF LEEDS

- Permutation problems: another viewpoint has a Boolean variable b_{ij} for every variable-value combination
  - e.g. in the n-queens problem, $b_{ij} = 1$ if there is a queen on the square in row i and column j, 0 otherwise
- A Boolean viewpoint can be derived from a CSP viewpoint with integer or set variables (or v.v.)
  - in an integer viewpoint, $b_{ij} = 1$ is equivalent to $x_i = j$
  - in a set-variable viewpoint, $j \in X_i$ is equivalent to $b_{ij} = 1$
- The Boolean viewpoint often gives a less efficient CSP than the integer or set model
  - the reverse translation can be useful

Different Perspectives: Example

UNIVERSITY OF LEEDS

- Constraint Modelling Challenge, IJCAI 05
- "Minimizing the maximum number of open stacks"
- A manufacturer has a number of orders from customers to satisfy
  - each order is for a number of different products, and only one product can be made at a time
  - once a customer's order is started (i.e. the first product in the order is made) a *stack* is created for that customer
  - when all the products that a customer requires have been made, the stack is closed
  - the number of stacks that are in use simultaneously i.e. the number of customer orders that are in simultaneous production, should be minimized

Minimizing Open Stacks – Example

UNIVERSITY OF LEEDS

- The product sequence shown needs 4 stacks
- But if all customer 3’s products are made before (or after) customer 4’s, only 3 are needed
- 3 is the minimum possible because products 2 and 6 are each for 3 customers

products	1	2	3	4	5	6
customer 1	0	1	0	0	0	1
customer 2	1	1	1	0	0	1
customer 3	1	0	0	1	0	1
customer 4	0	1	0	0	1	0

products	1	4	6	2	3	5
customer 1	0	0	1	1	0	0
customer 2	1	0	1	1	1	0
customer 3	**1**	**1**	**1**	0	0	0
customer 4	0	0	0	**1**	0	**1**

Open Stacks – Possible Viewpoints

UNIVERSITY OF LEEDS

- Variables are positions in production sequence, values are products
  - a permutation problem
  - so has a dual viewpoint

products	1	2	3	4	5	6
customer 1	0	1	0	0	0	1
customer 2	1	1	1	0	0	1
customer 3	1	0	0	1	0	1
customer 4	0	1	0	0	1	0

- In constructing the product sequence, at any point, products that are only for customers that already have open stacks can be inserted straightaway
  - e.g. if product 1 is first, products 3 & 4 can follow
  - the next *real* decision is whether to open a stack for customer 1 or 4 next (or both)
  - leads to a viewpoint based on customers

Open Stacks – Customer Viewpoints

- Variables are positions in *customer* sequence, values are customers
  - $r_i = j$ if the i th customer to have their order completed is j
- A variable for each customer, values are stack locations
  - customers ordering the same product cannot share a stack location
  - a graph colouring problem with additional constraints
- A Boolean variable for each *pair* of customers
  - 0 means they share a stack location, 1 means that they don't
  - NB we want to maximize the number of customers that can share a stack location

Summary

UNIVERSITY OF LEEDS

- Symmetry
  - Look out for symmetry in the CSP
    - avoid it if possible by changing the model
    - eliminate it e.g. by adding constraints
    - does this allow more implied constraints?
- Viewpoints
  - don’t stick to the first viewpoint you thought of, without considering others
    - think of standard reformulations
    - think about the problem in different ways