Creating a Constraint Programming Application

Chris Kuip

April 15, 2015

AIMMS

Agenda

- Brief introduction to Constraint Programming
- Planning the staffing of a Nursery Home

Constraint Programming Modeling

- Three persons staffing the three shifts of one day,
  the first shift cannot be done by persons 1 and 2.

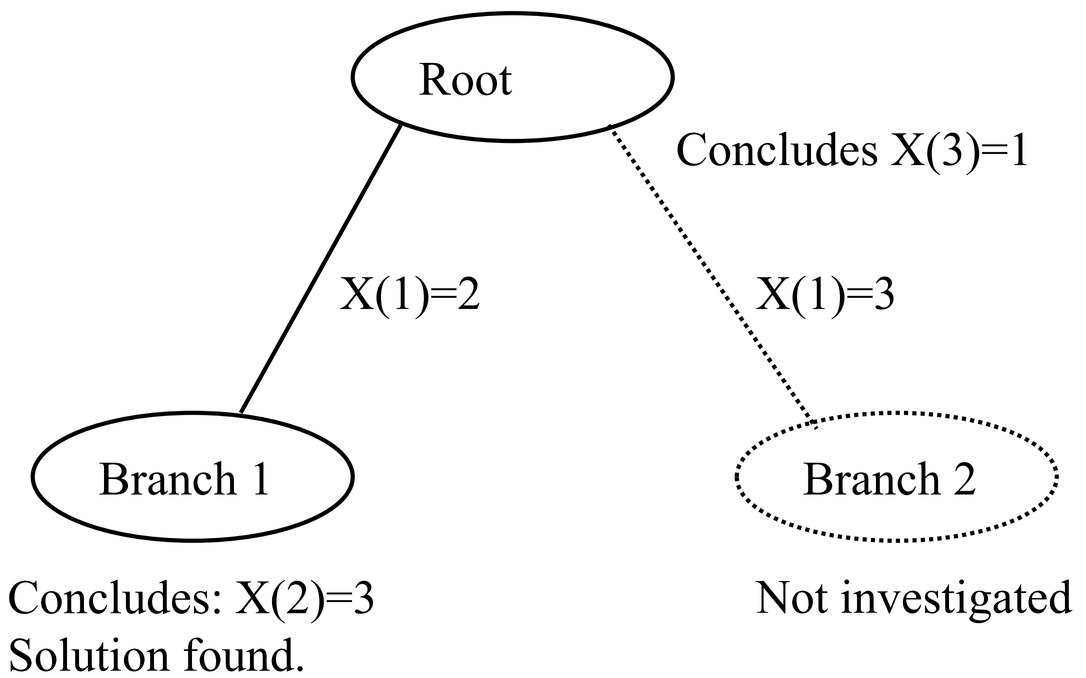
- Modeling
  - i in { 1..3 } ! Shifts
  - X(i) in { 1..3 }, ! Variable
  - X(1) <> 1, ! Constraint, first person does not do shift 1.
  - X(2) <> 1, ! Constraint, second person does not do shift 1.
  - cp::AllDifferent ! Constraint, each person does a different shift.

- Straightforward but naïve solution method: try each combination of possible values of the variables and see whether all constraints are satisfied.
- There are exponentially many possibilities: 3 variables, each 3 possibilities is 3^3=27 possible solutions to consider.

AIMMS

Constraint Programming Solution Method

- Alternates between trying and deduction:



Variables used in Constraint Programming model

Discrete variables

- Integer variables,
  - `VARIABLE:`
  - `   identifier :  x`
  - `   range      :  { lb .. ub }`
- Element variables
  - `ELEMENT VARIABLE:`
  - `   identifier   :  W`
  - `   index domain :  (s,d)`
  - `   range        :  Employees`

AIMMS

Constraints in Constraint Programming, part I

Constraint Programming Constraints are expressions combining:

- Discrete values and variables
- Operators/functions:
  - >=, <=, =, <, <>, >
  - *, +, -, /, abs, sqr, div(,), mod(,)
  - min, max (binary and iterative)
  - and, or, forall, exists
- In operator
  - eV in oneDimSet
  - (eU, eV, eW) in threeDimRelation
- Indexing
  - P(eV) Indexing a row of values
  - X(eV) Indexing a row of variables
- If then else endif

Constraints in Constraint Programming, part I

Global Constraints

- Essentials:
    - AllDifferent
    - Channel
    - Count/Cardinality/Sequence
    - BinPacking
    - Basic scheduling

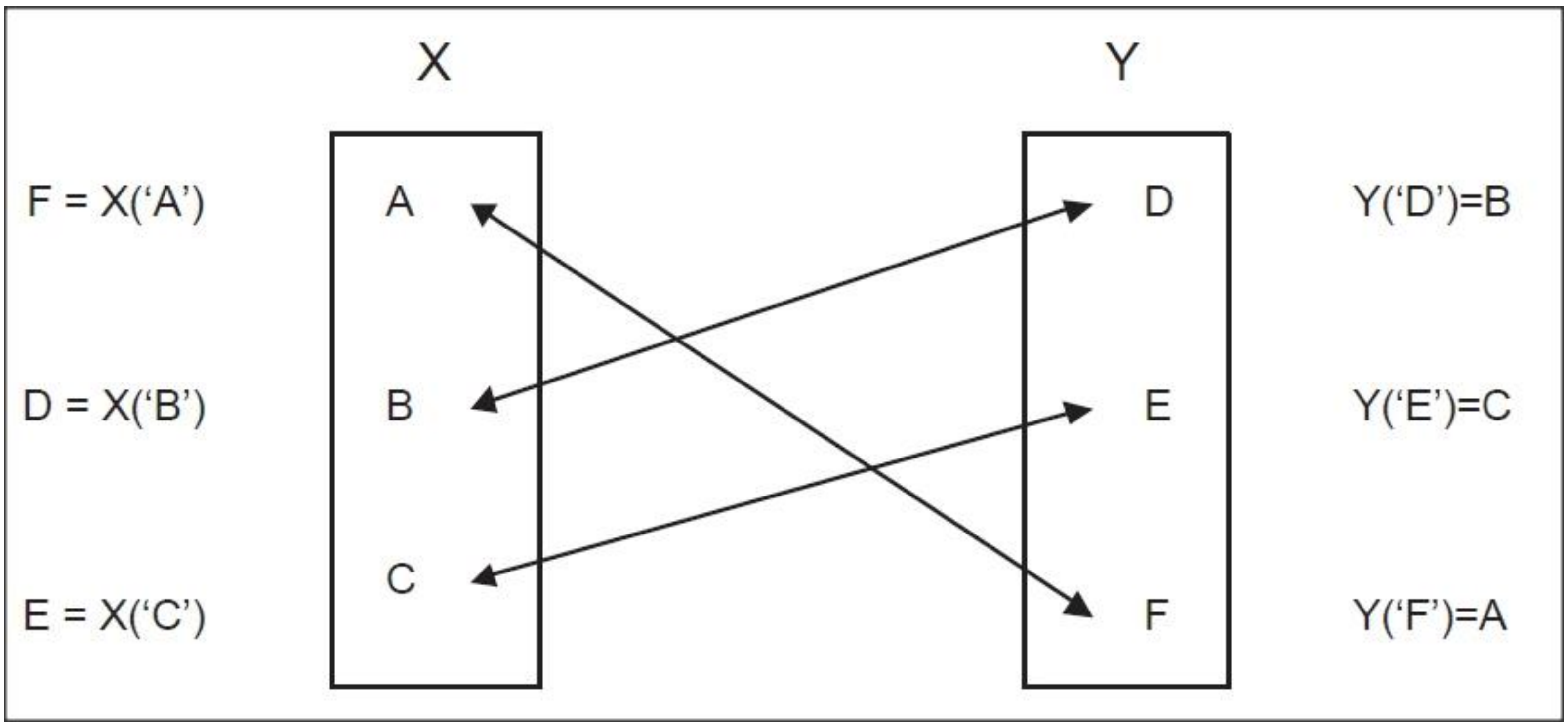
        sequentialResource

        parallelResource
- Scheduling constraints
    - In a later Webinar

Constraints in Constraint Programming, part II

Global Constraints

- cp::AllDifferent
- cp::Channel
  - Y(X(i)) = i & X(Y(j)) = j



Nursing Home Problem Formulation:

- Problem formulation:
  - Roster 4 nurses/employees, index e
  - Weekly repeating roster of 7 days, index d
  - Each day three shifts of 8 hours and a ‘free’ shift, index s
- Extra requirements:
  - Divide free days almost equally
  - Ensure at least 16 hours between worked shifts
  - For each working shift, minimize the number of nurses that staff that shift
  - For each working shift, minimize the number of transitions from one day to the next
- Pay attention to application performance

AIMMS

First question: choice of variables

How to represent the solution:

- Variable W(s,d) that assigns a nurse to a shift, looks as:

	W - The employee assigned to shift s on day d						
	Mon	Tue	Wed	Thu	Fri	Sat	Sun
free	A	C	D	D	C	B	B
day	B	B	B	B	B	C	C
evening	C	A	A	A	A	A	A
night	D	D	C	C	D	D	D

- Variable Y(e,d) that assigns a shift to a nurse, looks as:

	Y - The shift assigned to employee e on day d						
	Mon	Tue	Wed	Thu	Fri	Sat	Sun
A	free	evening	evening	evening	evening	evening	evening
B	day	day	day	day	day	free	free
C	evening	free	night	night	free	day	day
D	night	night	free	free	night	night	night

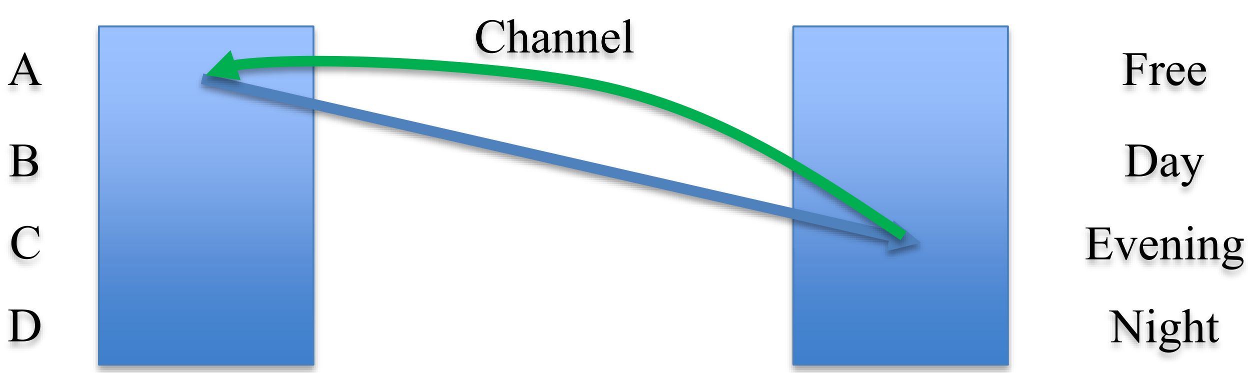
First question, which decision variables should we use in the model for ease of modeling:

- A) Assign nurse to a shift: W(s,d)
- B) Assign shift to a nurse: Y(e,d)
- C) Both W(s,d) and Y(e,d)
- D) Other variables

AIMMS

Answer to first question

- Choice C
- Perhaps non-intuitive; it doubles the number of variables
- No problem, because
    - cp::Channel links the variables; any choice made in the search, will also fix the corresponding other variable



- Handy, because we can now build using two perspectives.

AIMMS

Study solution by solving problem

- After a first solve we see that:

Y - The shift assigned to employee e on day d	Mon	Tue	Wed	Thu	Fri	Sat	Sun
A	free	night	free	free	night	day	evening
B	evening	free	day	night	evening	evening	day
C	day	day	night	day	day	free	night
D	night	evening	evening	evening	free	night	free

W - The employee assigned to shift s on day d	Mon	Tue	Wed	Thu	Fri	Sat	Sun
free	A	B	A	A	D	C	D
day	C	C	B	C	C	A	B
evening	B	D	D	D	B	B	A
night	D	A	C	B	A	D	C

- The problem with this solution is that the free days are not evenly distributed; Nurse A has three free days, Nurse B and Nurse C both 1 free day, Nurse D two free days.

- So the next question is: How do we ensure a proper distribution of the free days?

AIMMS

Second question: How to distribute the free days?

- A) Fix free days, as sequences of two days, as follows:
  Nurse A to Mon, Tue,
  Nurse B to Wed, Thu,
  Nurse C to Fri, Sat,
  Nurse D to Sun
- B) Fix free days, three days apart, as follows:
  Nurse A to Mon, Fri,
  Nurse B to Tue, Sat,
  Nurse C to Wed, Sun,
  Nurse D to Thu
- C) Add constraint: Max the number of free days to 2 per nurse:
  For each nurse e: sum(d, Y(e,d) = 'free') <= 2
- D) Add variable y0(e) in {1..2} Number of free days
  Add global constraint: cp::Count(d,Y(e,d),'free','=',y0(e))
- E) What are we talking about?
  This isn't an actual problem!
  No need to model this!

AIMMS

My answer to second question:

- A, B: This is not constraint programming, this is heuristic programming potentially excluding interesting solutions.
- C, D: two equally correct approaches of formulating the constraint.
- E: Always good to take a step back and ask ourselves whether we are solving a problem or creating a new one. In this case, I think we are solving a problem.

- For the remainder of this Webinar, I'm using answer D.

- Note that we have taken the perspective of Employees, not shifts.

AIMMS

Look at solution, limit number of nurses per shift.

- Look at the solution

	Y - The shift assigned to employee e on day d						
	Mon	Tue	Wed	Thu	Fri	Sat	Sun
A	night	free	day	night	night	night	day
B	evening	evening	free	evening	evening	day	free
C	free	day	evening	free	day	evening	night
D	day	night	night	day	free	free	evening

	W - The employee assigned to shift s on day d						
	Mon	Tue	Wed	Thu	Fri	Sat	Sun
free	C	A	B	C	D	D	B
day	D	C	A	D	C	B	A
evening	B	B	C	B	B	C	D
night	A	D	D	A	A	A	C

- To the clients of the nursing home, the above planning may come across as chaotic:
  - Day shift, staffed by 4 different nurses
  - Evening shift, staffed by 3 different nurses
  - Night shift, staffed by 3 different nurses
- Can we bring this down to 2 different nurses per shift?

Can we limit the number of nurses per shift?

- A) For each day d:
  - W('day',d) = DayNurse1 or W('day',d) = DayNurse2
  - W('evening',d) = EveningNurse1 or W('evening',d) = EveningNurse2
  - W('night',d) = NightNurse1 or W('night',d) = NightNurse2
- B) For each day d, shift s, s <> free:
  - W(s,d) = ShiftNurse1(s) or W(s,d) = ShiftNurse2(s)
- C) For each day d, shift s, s <> free:
  - Exists(i | W(s,d) = ShiftNurse(s,i)), i in { 1..2 }
- D) No we can't

AIMMS

Answer third question:

- Answers A, B, C are all correct and equivalent given the current data.
- C is just more generic and flexible
(if 2 nurses doesn't work, it is easy to extend to 3 nurses)
- So I'm going for C.
- I just hope you didn't choose answer D.

Question 4: Ensure at least 16 hours between work shifts.

Imagine working the night shift on Monday and the day shift on Tuesday: 16 hours in a row. How can we prevent this (fourth question)?

- A) Ensure that shift changes go only via a free shift in between
  for all n,d: if Y(e,d) <> 'free' then
  Y(e,d++1) = Y(e,d) or Y(e,d++1) = 'free'
  endif ;
- B) The problem is with evening and night shifts:
  - **Evening**: no day shift after a night shift
    if Y(e,d) = 'evening' then Y(e,d++1) in { free, evening, night } endif ;
  - **Night**: no evening, nor day shift after a night shfit
    if Y(e,d) = 'night') then Y(e,d++1) in { free, night } endif ;
- C) Determine, in advance, for each shift s and s' following directly after, whether this is allowed or not, resulting in relation R. For instance ('day','evening') in R, but ('evening','day') not in R.   Constraint:
  (y(e,d),y(e,d++1)) in R.

AIMMS

And the answer to question 4 is:

- Both answers B and C are correct.
- A will give enforce much longer periods free
- B and C are precisely implementing the restriction.
- Note that C is a so-called Table-constraint.
  - Sometimes much more efficient, especially when the table is very sparse.

Question 5: Minimize the number of Staffing Changes

- 10 Staffing Changes are marked in the solution below:

	W - The employee assigned to shift s on day d						
	Mon	Tue	Wed	Thu	Fri	Sat	Sun
free	C	A	B	C	D	A	D
day	B	B	A	B	B	B	B
evening	D	D	D	D	C	D	C
night	A	C	C	A	A	C	A

How to minimize the number of staffing changes? (fourth question)

- A) Minimize sum((e,d)|Y(e,d)<>Y(e,d++1))
- B) Minimize sum((s,d)|s<>’free’, W(s,d) <> W(s,d++1))
- C) Introduce WN(s,d), WP(s,d) in { 0 .. 4 } to measure deviations:
  For all (s,d): WN(s,d)-WP(s,d)= W(s,d)-W(s,d++1)
  Minimize sum((s,d), WN(s,d)+WP(s,d))
- D) Introduce WN(s,d), WP(s,d) in { 0 .. 4 } to measure deviations:
  For all (s,d)|s<>’free’: WN(s,d)-WP(s,d)= W(s,d)-W(s,d++1)
  Minimize sum((s,d)|s<>’free’, WN(s,d)+WP(s,d))

And the answer to question 5 is:

- B
- A The wrong variable
- C and D measure distance, while only difference is to be counted.

Question 6: Reduce solve time (via solution space)

- Solving this problem already takes more than 10 seconds. How to improve?
- A first technique: reduce the solution space:
- Now: Y(e,d) 4 X 7 = 28 individual variables, 4 possible values →
  4^28 = 7.2e+16
  (as shown before: W(s,d) does not really count extra here).

Sixth question:

- A) Formalize heuristic to construct as we used to create the
  roster by hand, and leave some freedom thereafter.
- B) Fix the first day:
  For all (e,s)|ord(e) = ord(s): Y(e,d) = s
- C) Make sure the first nurse only does the same or later shifts,
  otherwise we can re-order the days:
  eFirst = first(Employees)
  for all d: Y(eFirst,d) <= Y(eFirst,d+1)
- D) All of the above

AIMMS

And the answer to the sixth question is…

- A may exclude solutions and doesn’t fit the Constraint Programming paradigm
- B is correct, there is symmetry in the problem, employees only coded A, B, C and D; and no restrictions on a particular employee.
- C is not correct; there is no symmetry in days, as we already have restrictions on going from one day to the next
- D is not correct, as A and C are not.

Can we further reduce the solution time (question 7).

Second technique; add constraints that are implied by other constraints, so-called redundant constraints

Seventh and last question, can you derive other constraints

- A) Sum(e, Y0(e)) = 7
- B) obj >= 6
- C) cp::AllDifferent(e, Y(e,d)) and cp::AllDifferent(s, W(s,d))
- D) All of the above

And the answer is…

- D, all of the above, but B is most helpful.

Next webinar

- The Mathematical Program Inspector.
- Peter Nieuwesteeg
- May 20, 2015

AIMMS

Questions?