Artificial Intelligence

Constraint Programming 3: The Party

Ian Gent
ipg@cs.st-and.ac.uk

Artificial Intelligence

Constraint Programming 3

Part I : **Formulation**
Part II: **Progressive piss up at a yacht club**

Constraint Satisfaction Problems

- CSP = Constraint Satisfaction Problems
- A CSP consists of:
  - a set of ***variables***, X
  - for each variable x_i in X, a ***domain*** D_i
    - D_i is a finite set of possible ***values***
  - a set of constraints restricting ***tuples*** of values
    - if only pairs of values, it's a **binary** CSP
- A **solution** is an assignment of a value in D_i to each variable x_i such that every constraint satisfied

3

Donald + Gerald = Robert

- We can write one long constraint for the sum:
  - 100000*D + 10000*O + 1000*N + 100*A+ 10*L + D
    + 100000*G + 10000*E + 1000*R + 100*A+ 10*L + D
    = 100000*R + 10000*O + 1000*B + 100*E+ 10*R + T
- But what about the difference between variables?
  - Could write D =/= O, D=/=N, … B =/= T
  - Or express it as a single constraint on all variables
  - *AllDifferent*(D,O,N,A,L,G,E,R,B,T)
- These two constraints
  - express the problem precisely
  - both involve all the 10 variables in the problem

4

Formulation of CSP's

- All-different is an example of the importance of formulation
  - all-different(x,y,z) much better than x≠y, y≠z, z≠x
  - even though logically equivalent
- In general, it's hard to find the best formulation
- Remember DONALD + GERALD = ROBERT
- The formulation I gave had just 2 constraints
  - all-different and a complicated arithmetic constraint
- All-different fine, but neither FC nor MAC can do much with the arithmetic constraint

5

Cryptarithmetic Revisited

- FC cannot propagate until only one variable left in constraint
- AC cannot propagate until only two variables left
- When coded in ILOG Solver, search backtracks 8018 times
- How can we formulate the problem better?
- *Hint: we'd like to consider the sum in each column separately*

6

This shouldn't work ?!?

- We've made the problem ***bigger***, so how can it help?
- Before, there were $9^3\,10^7$ possibilities
  - now there are $2^5 = 32$ times as many!
- The constraints now involve fewer variables
  - constraint propagation can happen sooner
  - variables can be set sooner (reduced to one value)
  - domain wipe out & backtracking occurs earlier
- In ILOG Solver, this encoding needs only 212
  - down from 8,018
  - if that doesn't impress you, call it minutes (or hours)

7

DONALD + GERALD = ROBERT

- One solution is to add *more variables* to the problem
- Variables C1, C2, C3, C4, C5
  - Ci represents carry from previous column i
  - $D_{Ci} = \{0,1\}$
  - Now we can express more constraints
  - D + D = 10*C1 + T
  - C1 + L + L = 10*C2 + R
  - C2 + A + A = 10*C3 + E
  - C3 + N + R = 10*C4 + B
  - C4 + O + E = 10*C5 + O
  - C5 + D + G = R

8

The importance of heuristics

- Remember "minimum remaining value" heuristic
  - check out Constraints lecture 1 if not
- Variable ordering heuristic
  - choose variable to expand next with m.r.v.
    - I.e. smallest number of values left in current domain
  - very important in practical solution of CSPs
- In DONALD + GERALD using ILOG Solver
  - carry variables take 8,018 fails to 212
  - m.r.v. reduces it to 14
  - (multiply it by 1,000,000 if it doesn't seem important)

9

Exercises from Constraints 3

- Solve DONALD + GERALD = ROBERT
  - Try to simulate what constraints program would do
    - use carry variables and m.r.v. heuristic
- What does the progressive party problem tell us?
  - Consider issues such as:
    - relative success and failure of CP/ILP
    - number of variables necessary
    - complication of formulation/heuristics

10

Progressive Piss up at a yacht club

- "The Progressive Party Problem:
  - Integer Linear Programming and Constraint Programming Compared"
  - Barbara M Smith, Sally Brailsford, Peter Hubbard, Paul Williams
- 39 boats at a yachting rally
  - each boat with a known crew size
  - & capacity to entertain a certain number of guests
  - designated host crews stay put, other crews circulate
  - guest crews progress every half hour
  - 3 hours = 6 visits

11

What's a progressive party?

- Constraints
  - no guest crew may visit the same host boat twice
  - no two guest crews may meet twice
  - crews cannot be split up (neither host nor guest)
  - no boat's capacity can be exceeded
  - want to minimise the number of host boats
    - and find a way of organising the party with this number
- In the particular problem, we definitely need 13 boats
  - the largest 12 boats are too small
  - Integer L.P. techniques found solution with 14 boats
    - but not 13 boats using 189 cpu hours in 1994/5

12

Formulation & Heuristics critical

- Smith designated the 13 host boats
  - those with the largest spare capacity
- This means that failure to find solution not definitive
  - might be a solution with different choice of boats
  - largest 13 boats might be better irrespective of spare capacity
    - large boat with large crew may be best staying put as host
- Now we know *h* boats 1-13, *g* boats 1=26, times 1-6
- Primary variables will be h_{gt}, domain D = { 1 … 13 }
  - variable gives location of guest crew g at time t

13

Secondary variables

- Like carry's in Donald + Gerald, useful for search
  - domains will be {0,1}
  - also helpful for formulation
- $v_{ght} = 1 \leftrightarrow h_{gt} = h$
  - I.e. $v_{ght} = 1$ iff guest crew g visits host h at time t
- $m_{gft} = 1 \leftrightarrow h_{gt} = h_{ft}$
  - *I.e.* $m_{gft} = 1$ iff guest crews g and f meet at time t

14

Constraints for a party

- Automatically have that crews do not split
  - we have to allocate location of whole crew at once
- Need to link up primary and secondary variables
  - using constraints summarised on previous slide
    - e.g. $v_{ght} = 1 \leftrightarrow h_{gt} = h$
- Other constraints now expressible
- First one does not need secondary vars:
  - *Crews do not visit same boat twice*
    - *AllDifferent*(h_{g1},h_{g2}, …, h_{g6})
      - one for all g = 1, 2, … 26

15

More constraints for a party

- Use variables about crews meeting:
  - *Two guest crews do not meet twice (or more)*
    - $m_{gf1} + m_{gf2} + \ldots + m_{gf6} < 2$
      - one for all pairs f, g
- Use variables about visits by guest crews
  - assume that S_h is spare capacity of host boat h
  - and that C_g is size of crew of guest boat g
    - $c_1v_{1ht} + c_2v_{2ht} + \ldots + c_{26}v_{26ht} \leq S_h$
      - one for all pairs h, t
- Some "symmetry" constraints I won't detail
  - e.g. to distinguish guest boats with same size crew

16

Heuristics

- Both variable and value ordering heuristics used
- Variable ordering heuristic had 5 levels …
- Only considered primary variables h_{gt}
  - Consider time periods in order (e.g. h_{71} before h_{32})
    - use minimum remaining value within that so that guest crews with fewest possible hosts allocated first
      - break ties by picking variables in most constraints
        - if still tied, pick biggest guest crew
- Value ordering was much simpler
  - try host crews (values) in descending order of spare capacity

17

We have ourselves a party

- Constraint programming (ILOG Solver) won!
- This managed to schedule the party in 27 mins
  - 1994/5 cpu times
- In general, Constraints not always better than ILP
- In this case, constraints were very tight
- Where constraints looser, often ILP better

18

And Finally …

- The ***real*** party was scheduled by hand
  - since the CP solution was not done until months later
- BUT
  - the real party used more host boats than it needed to
  - and ILOG Solver found a solution for a 7th half hour
  - So with constraint programming …
    - ***They could have had a longer party!!***

19