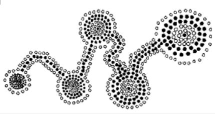
- I am Patrick Prosser
- I am a senior lecturer at Glasgow
- I teach algorithms & data structures in java
- I am a member of
  - the algorithms group
  - the apes (distributed, not disbanded)
- I am a Glaswegian

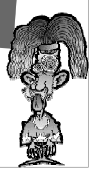
This is all that I am allowed to tell you

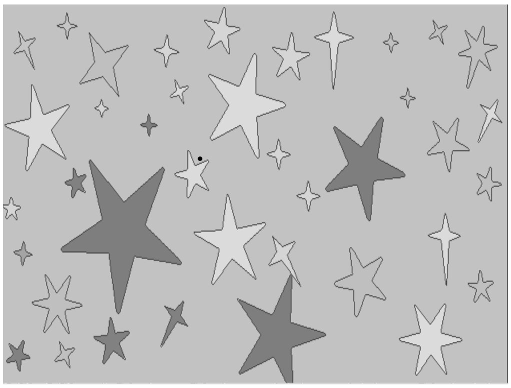
Patrick Prosser

CI9(4)
Hybrid Algorithms for the Constraint Satisfaction Problem

- Still using that old greasy stuff?
- Who cares?
- So?







What's he going to cover!

Stop wasting time

get on with it!

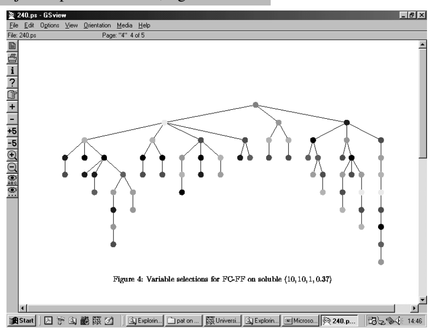
Com...

- What do we want from the algorithm?
  - Minimise consistency checks
  - minimise nodes visited
  - fast (wooooooosh!)
  - pretty

Preamble

- v[i] is a variable
- d[i] is a set of values for variable v[i]
- x is a value in d[i] (and y in d[j])
- c[i,j] is a binary relation/constraint such as
  - >, <, =, nextto, f(x,y), nil (no constraint)
- check(i,j) tests if v[i]=x is compatible with v[j]=y
- cd[i] is the current domain of v[i]
  - i.e. the working domain of v[i]

It's all just depth first search, right?



BT

bt−label
iterate over x in cd[i] until an x is found that is compatible with all past variables i.e. check(h,i) true for all 0 < h < i

On route to finding this compatible value if any incompatible value x is found remove it from cd[i]

return (i+1,true) or (i,false)

BT

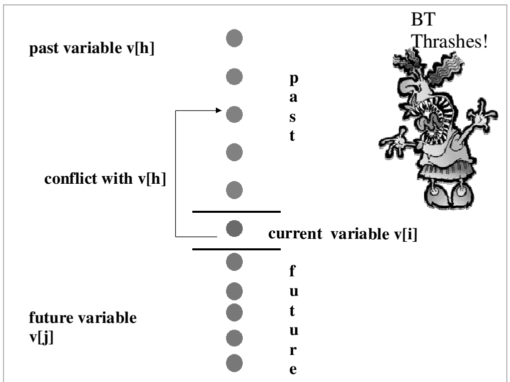
bt−unlabel
bt−label(i) returned (i,false) or
bt−unlabel(i) returned (i,false)

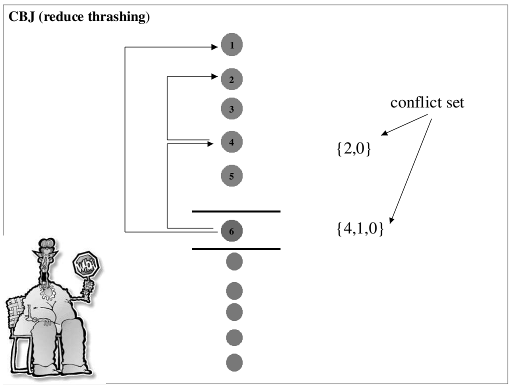
the variable to backtrack to is h = i−1
i.e. the previous variable

remove the value v[h] from cd[h]

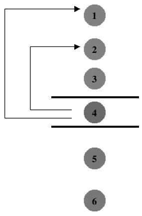
reset cd[i]

return (h,true) or (h,false)





CBJ (reduce thrashing)



{2,1,0}

Jump back to deepest past variable in confSet (call it h) and then combine confSet[i] with confSet[h]

- History:
  - Konkrat and V Beek,
  - Gent and Underwood

CBJ implementation

- Implementation
  - confSet[0..n][0..n] is a boolean
  - confSet[i][h] = true if v[i] in conflict with v[h]
  - on backjumping do logical OR
  - … and then clear out confset[i]
- For DVO
  - h is the depth in the search tree of the variable v[i] conflicts with!

What is forward checking?

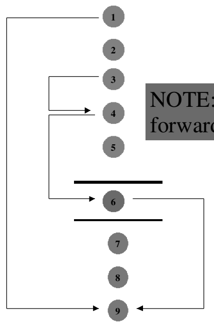
OZ

Show me!

Check Forwards

- instantiate v[i] = x
- for all variables v[j]
  - where v[j] is adjacent to v[i] & v[j] is in the future of v[i]
- remove all values from cd[j] inconsistent with v[i]=x with respect to constraint c[i][j]
- do this every time we instantate a variable
  - a lot of work!
  - does it pay off?
    - Can you bear the suspense?

Forward Checking



NOTE: arrows go forward!

Forward Checking, a marking scheme

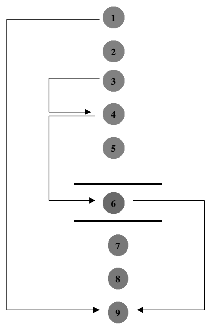
- associate with each variable v[i]
  - future[i][j] = true
    if v[i] = x removes values from cd[j]
  - past[j][y] = i
    if v[i] = x removes value y from cd[j]
  - cd[i][x] = true
    if x is currently in current domain of i

- checkForwards(i,j) does as follows
  - for y in (1 .. m)
    - if cd[j][y] and not(check(i,x,j,y))
      - cd[j][y] = false
      - past[j][y] = i
      - future[i][j] = true

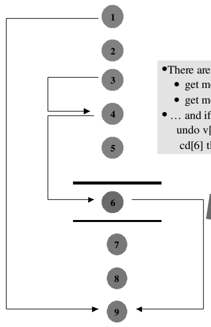
Forward checking by marking

- Marking scheme uses static space!
- Makes it easy to retract/backtrack
- It's neat
- But it still thrashes :(

Forward Checking Can Thrash



Check Forwards, Jump Back!



- There are no values in cd[6] compatible with v[9]
  - get more values into cd[9] (undo v[1]?) OR
  - get more values into cd[6] (undo v[4])
- … and if that doesn't work?
  undo v[3] so cd[4] gets value compatible with
  cd[6] that is then compatible with cd[9]

All the information is already there

Check forwards, jump back!

- assume v[i] = x causes dwo on v[j]
  v[i] is now in conflict with all variables forward checking against v[j]
  - h = past[j][y] means that v[h] removes y from cd[j]
  - therefore confSet[i][h] = true
  - do this for all y!

- assume no more values to try in cd[i], must backjump to v[h] where
  - h is the deepest variable that is in confSet[i] or in past[i]
  - confSet[h] is united with confSet[i] and past[i]

What is a dvo heuristic? (you kind of skipped that)

Does it make any difference

Show me!

OZ

So?

- Paper rejected from IJCAI91 (written in 1990)
- I was a Lisp programmer at the time (it shows)
- I think the experiments were very good (so there!)
- Nice study of influence of topological parameter on search cost.
- In conclusion I forgot to say CBJ was new … why?
- I like BMJ, it is cool (I was smart for 1 day)
- I think CBJ is pretty (natural, discovered, not invented)
- I like FC−CBJ (I can understand it)
- I identify work to be done (researchers love that (why?))
- … and I make errors re−dvo's (researchers love that (why?))
- I put my results in perspective (trash them :)
- I got encouragement (Nadel) and help (Ole and Peter)
- I got a whole load of background (Rina)
- But it hurt … why did it take 3 years to get somebody to read it?