



Smart Searching Using Constraints

R&N §6. 1-6. 3

Jacques Fleuriot

School of
informatics
University of Edinburgh

(slides adapted from Hwee Tou Ng)

Informatics 2D



Outline

- Constraint Satisfaction Problems (CSP)
- Backtracking search for CSPs
- Making this more efficient

Informatics 2D



Constraint satisfaction problems (CSPs)

- Standard search problem:
 - *state* is a 'black box' – any data structure that supports successor function, heuristic function and goal test.
- CSP:
 - *state* is defined by *variables* X_i with *values* from *domain* D_i
 - goal test is a set of *constraints* specifying allowable combinations of values for subsets of variables.
- Simple example of a *formal representation language*.
- Allows useful *general-purpose* algorithms with more power than standard search algorithms.

Informatics 2D



Example: Map-Colouring



- *Variables* WA, NT, Q, NSW, V, SA, T
- *Domains* $D_i = \{\text{red, green, blue}\}$
- *Constraints*: adjacent regions must have different colours,
 - e.g. $WA \neq NT$,
 - or $\{WA, NT\}$ in $\{(\text{red, green}), (\text{red, blue}), (\text{green, red}), (\text{green, blue}), (\text{blue, red}), (\text{blue, green})\}$.

Informatics 2D



Example: Map-Colouring



- **Solutions** are *complete* and *consistent* assignments,
 - e.g. WA = red, NT = green, Q = red, NSW = green, V = red, SA = blue, T = green.

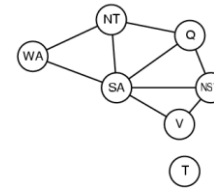
Informatics 2D

5

Constraint graph



- **Binary CSP**: each constraint relates two variables.
- **Constraint graph**: nodes are variables, arcs are constraints.



Informatics 2D

6

Varieties of CSPs



- **Discrete variables**:
 - finite domains:
 - n variables, domain size $d \rightarrow O(d^n)$, complete assignments.
 - e.g. Boolean CSPs, incl. Boolean satisfiability (NP-complete).
 - infinite domains:
 - integers, strings, etc.
 - e.g. job scheduling, variables are start/end days for each job.
 - need a constraint language, e.g. $StartJob_1 + 5 \leq StartJob_3$
- **Continuous variables**:
 - e.g. start/end times for Hubble Space Telescope observations.
 - linear constraints solvable in polynomial time by linear programming.

Informatics 2D

7

Varieties of constraints

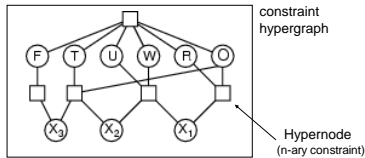


- **Unary** constraints involve a single variable,
 - e.g. $SA \neq \text{green}$.
- **Binary** constraints involve pairs of variables,
 - e.g. $SA \neq WA$.
- **Higher-order** constraints involve 3 or more variables,
 - e.g. crypt-arithmetic column constraints.
- **Global** constraints involve an arbitrary number of variables

Informatics 2D

8

Example: Crypt-arithmic

$$\begin{array}{r} T W O \\ + T W O \\ \hline F O U R \end{array}$$


- **Variables:** $F T U W R O X_1 X_2 X_3$
 - **Domains:** $\{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$.
 - **Constraints:** *Alldiff* (F, T, U, W, R, O).
 - $O + O = R + 10 \cdot X_1$.
 - $X_1 + W + W = U + 10 \cdot X_2$.
 - $X_2 + T + T = O + 10 \cdot X_3$.
 - $X_3 = F, T \neq 0, F \neq 0$.
- Global constraint

Informatics 2D



9

Real-world CSPs

- Assignment problems
 - e.g. who teaches what class.
- Timetabling problems.
 - e.g. which class is offered when and where.
- Transportation scheduling.
- Factory scheduling.

Notice that many real-world problems involve real-valued variables.

Informatics 2D



10

Standard search formulation (incremental)

Let's start with the straightforward approach, then adapt it.

States are defined by the values assigned so far.

- **Initial state:** the empty assignment $\{ \}$.
 - **Successor function:** assign a value to an unassigned variable that does not conflict with current assignment
 - fail if no legal assignments.
 - **Goal test:** the current assignment is complete.
1. This is the same for all CSPs.
 2. Every solution appears at depth n with n variables
 - use depth-first search.

Informatics 2D

11

Backtracking search

- Variable assignments are *commutative*,
 - e.g. [WA = red then NT = green] same as [NT = green then WA = red].
- Only need to consider assignments to a single variable at each node
 - $b = d$ and there are d^* leaves.
- Depth-first search for CSPs with single-variable assignments is called *backtracking* search.
- Backtracking search is the basic uninformed algorithm for CSPs.
- Can solve n -queens for $n \approx 25$.

Informatics 2D

12



Backtracking search

```

function BACKTRACKING-SEARCH(csp) returns a solution, or failure
return BACKTRACK({ }, csp)

function BACKTRACK(assignment, csp) returns a solution, or failure
if assignment is complete then return assignment
var ← SELECT-UNASSIGNED-VARIABLE(csp)
for each value in ORDER-DOMAIN-VALUES(var, assignment, csp) do
  if value is consistent with assignment then
    add {var = value} to assignment
    inferences ← INFERENCE(csp, var, value) ← Optional: Can be used to
    impose arc-consistency (more on this later)
    if inferences ≠ failure then
      add inferences to assignment
      result ← BACKTRACK(assignment, csp)
      if result ≠ failure then
        return result
    remove {var = value} and inferences from assignment
return failure

```



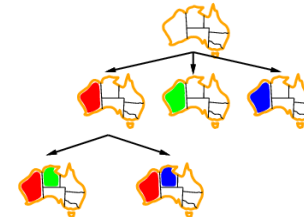
Backtracking example



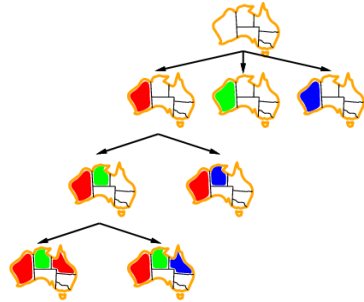
Backtracking example



Backtracking example



Backtracking example

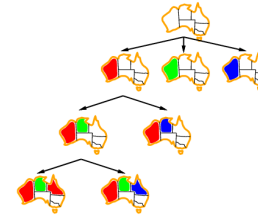


Informatics 2D



17

Which nodes can be eliminated on symmetry grounds?

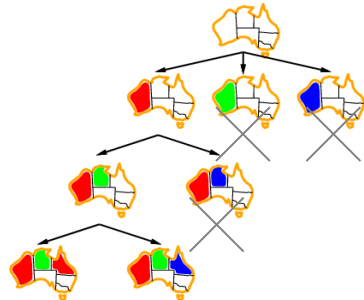


Informatics 2D



18

Solution



Informatics 2D



19

Improving backtracking efficiency

- **General-purpose** methods can give huge gains in speed:
 - Which **variable** should be assigned next?
 - **SELECT-UNASSIGNED-VARIABLE**
 - Then, in what order should its **values** be tried?
 - **ORDER-DOMAIN-VALUES**
 - What inferences should be performed at each step of the search?
 - **INFERENCE**
 - Can we detect inevitable failure early?

Informatics 2D

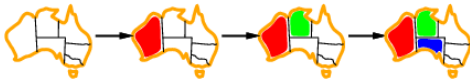


20

Most constrained variable

`var ← SELECT-UNASSIGNED-VARIABLE(csp)`

- Most constrained variable:
choose the variable with the fewest legal values.



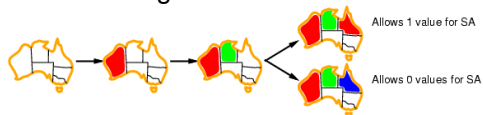
- a.k.a. *minimum-remaining-values* (MRV) heuristic.

Informatics 2D

21

Least constraining value

- Given a variable, choose the least constraining value:
– the one that rules out the fewest values in the remaining variables.



- Combining these heuristics makes 1000 queens feasible.

Informatics 2D

23

Most constraining variable

- Tie-breaker among **most constrained** variables.
- Most constraining variable:
– choose the variable with the most constraints on remaining variables – thus reducing branching.
- a.k.a. *degree heuristic*

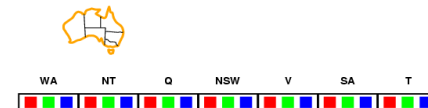


Informatics 2D

22

Inference: Forward checking

- Idea:
– Keep track of remaining legal values for unassigned variables.
– Terminate search when any variable has no legal values.



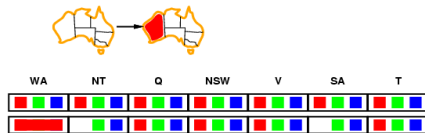
Informatics 2D

24



Forward checking

- Idea:
 - Keep track of remaining legal values for unassigned variables.
 - Terminate search when any variable has no legal values.



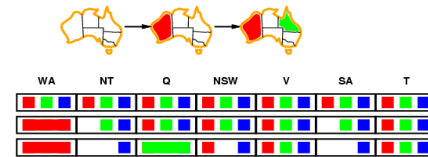
Informatics 2D

25



Forward checking

- Idea:
 - Keep track of remaining legal values for unassigned variables.
 - Terminate search when any variable has no legal values.



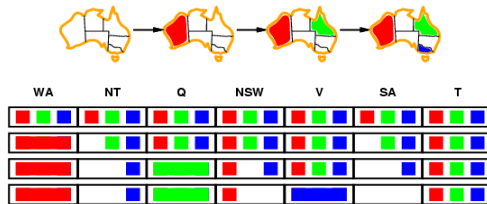
Informatics 2D

26



Forward checking

- Idea:
 - Keep track of remaining legal values for unassigned variables.
 - Terminate search when any variable has no legal values.



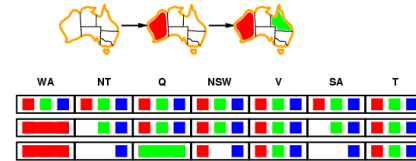
Informatics 2D

27



Constraint propagation

- Forward checking propagates information from assigned to unassigned variables, but doesn't provide **early** detection for all failures:



- NT and SA cannot both be blue!
- Constraint propagation repeatedly enforces constraints locally.

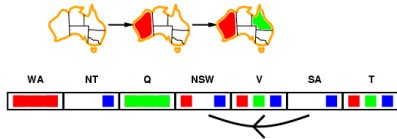
Informatics 2D

28

Arc consistency



- Simplest form of propagation makes each arc **consistent**.
- $X \rightarrow Y$ is consistent iff
for **every** value x of in the domain of X there is **some** allowed y in the domain of Y .



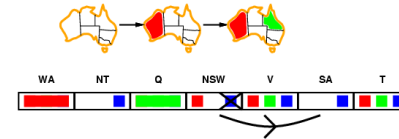
Informatics 2D

29

Arc consistency



- Simplest form of propagation makes each arc **consistent**.
- $X \rightarrow Y$ is consistent iff
for **every** value x of in the domain of X there is **some** allowed y in the domain of Y .



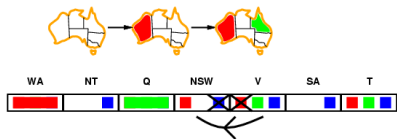
Informatics 2D

30

Arc consistency



- Simplest form of propagation makes each arc **consistent**
- $X \rightarrow Y$ is consistent iff
for **every** value x of in the domain of X there is **some** allowed y in the domain of Y .



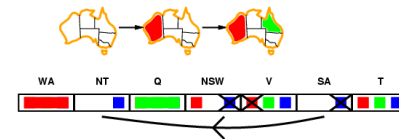
Informatics 2D

31

Arc consistency



- Simplest form of propagation makes each arc **consistent**
- $X \rightarrow Y$ is consistent iff
for **every** value x of in the domain of X there is **some** allowed y in the domain of Y .



Informatics 2D

32

- If X loses a value, neighbours of X need to be rechecked

- If X loses a value, neighbours of X need to be rechecked
- Arc consistency detects failure earlier than forward checking.
- Can be run as a preprocessor or after each assignment.

Arc consistency algorithm AC-3



```

function AC-3(csp) returns false if an inconsistency is found and true otherwise
inputs: csp, a binary CSP with components ( $X, D, C$ )
local variables: queue, a queue of arcs, initially all the arcs in csp

while queue is not empty do
    ( $X_i, X_j$ )  $\leftarrow$  REMOVE-FIRST(queue)
    if REVISE(csp,  $X_i, X_j$ ) then
        if size of  $D_i = 0$  then return false
        for each  $X_k$  in  $X_i$ .NEIGHBORS -  $\{X_j\}$  do
            add ( $X_k, X_i$ ) to queue
    return true

```

Make X_i arc-consistent with respect to X_j
 No consistent value left for X_i , so fail
 Since revision occurred, add all neighbours of X_i for consideration (or reconsideration)

```

function REVISE(csp,  $X_i, X_j$ ) returns true iff we revise the domain of  $X_i$ 
    revised  $\leftarrow$  false
    for each  $x$  in  $D_i$  do
        if no value  $y$  in  $D_j$  allows ( $x, y$ ) to satisfy the constraint between  $X_i$  and  $X_j$  then
            delete  $x$  from  $D_i$ 
            revised  $\leftarrow$  true
    return revised

```

- Time complexity: $O(cd^3)$, where d is maximum size of each domain and c is the number of binary constraints (arcs).



Summary



- CSPs are a special kind of problem:
 - states defined by values of a fixed set of variables
 - goal test defined by constraints on variable values
- Backtracking = depth-first search with one variable assigned per node
- Variable ordering and value selection heuristics help significantly
- Forward checking prevents assignments that guarantee later failure
- Constraint propagation (e.g. arc consistency) does additional work to constrain values and detect inconsistencies

