

# Constraint Satisfaction Problems (CSPs)



## Outline

- CSP?
- Backtracking search for CSPs
- Local search for CSPs
- Problem structure

2

These slides are based on the slides at AIMA book webpage

## Constraint satisfaction problems

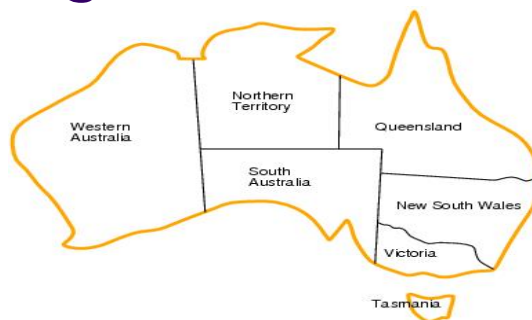


- What is a CSP?
  - Finite set of variables  $V_1, V_2, \dots, V_n$
  - Nonempty domain of possible values for each variable  $D_{V_1}, D_{V_2}, \dots, D_{V_n}$
  - Finite set of constraints  $C_1, C_2, \dots, C_m$
  - Each constraint  $C_i$  specifies allowable combinations of values for subsets of variables, e.g.,  $V_1 \neq V_2$
  - A solution is an assignment of values to all variables that satisfies all constraints.
    - Some CSPs require a solution that maximizes an *objective function* → constrained optimization problems

3

These slides are based on the slides at AIMA book webpage

## CSP example: map coloring

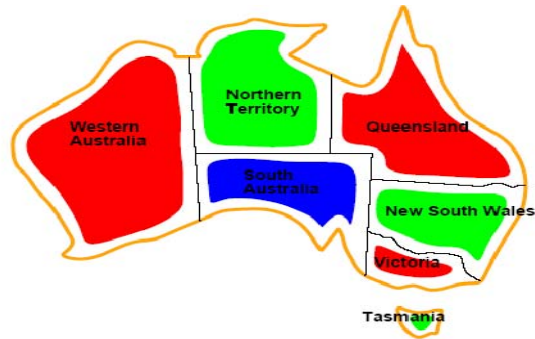


- Variables:  $WA, NT, Q, NSW, V, SA, T$
- Domains:  $D_i = \{red, green, blue\}$
- Constraints: adjacent regions must have different colors.
  - E.g.  $WA \neq NT$ , or  $(WA, NT) \in \{(red, green), (red, blue), (green, red), \dots\}$

4

These slides are based on the slides at AIMA book webpage

# CSP example: map coloring



- Solutions are assignments satisfying all constraints, e.g.  $\{WA=red, NT=green, Q=red, NSW=green, V=red, SA=blue, T=green\}$

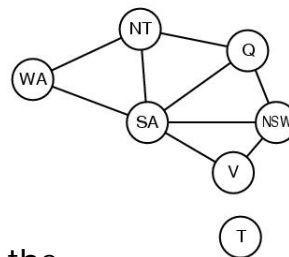
5

These slides are based on the slides at AIMA book webpage

# Constraint graph



- **Binary CSP:** each constraint relates two variables
- **Constraint graph** = nodes are variables, edges show constraints.
- CSP algorithms can make use of the graph structures
  - e.g. Tasmania is an independent subproblem.



6

These slides are based on the slides at AIMA book webpage

# Varieties of CSPs



- Discrete variables
  - Finite domains
    - $n$  variables with 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
- Continuous variables
  - e.g. start/end times for Hubble Telescope observations.

We consider finite domains

7

These slides are based on the slides at AIMA book webpage

# Varieties of constraints



- Unary constraints involve a single variable.
  - e.g.  $SA \neq green$
  - Can be eliminated
- Binary constraints involve pairs of variables.
  - e.g.  $SA \neq WA$
- Higher-order constraints involve 3 or more variables.
  - e.g. cryptarithmic column constraints.
  - can be reduced to binary constraints by introducing auxiliary variables

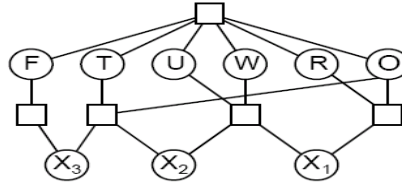
8

These slides are based on the slides at AIMA book webpage

## Example; cryptarithmic



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



**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**

$alldiff(F, T, U, W, R, O)$

$O + O = R + 10 \cdot X_1$ , etc.

9

These slides are based on the slides at AIMA book webpage

## Real-world CSPs



- Timetabling problems
  - e.g., class scheduling: which class is offered when and where by whom?
- Transportation scheduling
- Factory scheduling
- Floor planning, VLSI design
- Graph coloring, Map coloring

10

These slides are based on the slides at AIMA book webpage

## CSP benefits



- Standard representation with generic goal and successor functions
  - A state is an assignment of values to some or all variables.
  - Incremental vs. complete-state
- General purpose heuristics (no problem-specific expertise required).
- Can take advantage of the structure of constraint graph

11

These slides are based on the slides at AIMA book webpage

## CSP as a standard search problem



- A CSP can be easily expressed as a standard search problem.
- Incremental formulation
  - *A state is an assignment of values to some or all variables.*
  - *Initial State:* the empty assignment {}.
  - *Successor function:* Assign value to an unassigned variable provided that there is not conflict.
  - *Goal test:* the current assignment is complete.
  - *Path cost:* irrelevant
- This is the same for all CSP's !

12

These slides are based on the slides at AIMA book webpage

## CSP as a standard search problem



- Solution is found at depth  $n$  (if there are  $n$  variables).
  - Hence depth first search can be used.
- Branching factor  $b$  at the top level is  $nd$ .  
 $b=(n-1)d$  at depth  $l$ , hence  $n!d^n$  leaves!
- But only  $d^n$  complete assignments.

13

These slides are based on the slides at AIMA book webpage

## Commutativity



- Variable assignments are commutative.
  - The order of any given set of actions has no effect on the outcome.
  - Example: choose colors for Australian territories one at a time
    - [WA=red then NT=green] same as [NT=green then WA=red]
- CSP search algorithms consider a single variable assignment at a time,  $b=d \Rightarrow$  there are  $d^n$  leaves.

14

These slides are based on the slides at AIMA book webpage

# Backtracking search



- Depth-first search
- Chooses values for one variable at a time and backtracks when a variable has no legal values left to assign, called **backtracking search**

15

These slides are based on the slides at AIMA book webpage

# Backtracking search



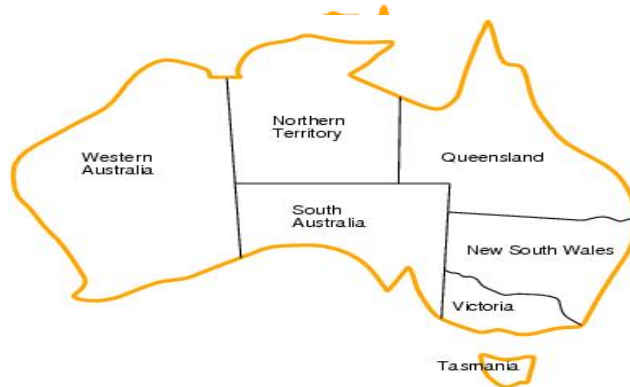
**function** BACKTRACKING-SEARCH(*csp*) **return** a solution or failure  
**return** RECURSIVE-BACKTRACKING(*{}*, *csp*)

**function** RECURSIVE-BACKTRACKING(*assignment*, *csp*) **return** a solution or failure  
**if** *assignment* is complete **then return** *assignment*  
*var* ← SELECT-UNASSIGNED-VARIABLE(VARIABLES[*csp*], *assignment*, *csp*)  
**for each** *value* **in** ORDER-DOMAIN-VALUES(*var*, *assignment*, *csp*) **do**  
    **if** *value* is consistent with *assignment* according to CONSTRAINTS[*csp*] **then**  
        add {*var=**value*} to *assignment*  
        *result* ← RRECURSIVE-BACKTRACKING(*assignment*, *csp*)  
        **if** *result* ≠ *failure* **then return** *result*  
        remove {*var=**value*} from *assignment*  
**return** *failure*

16

These slides are based on the slides at AIMA book webpage

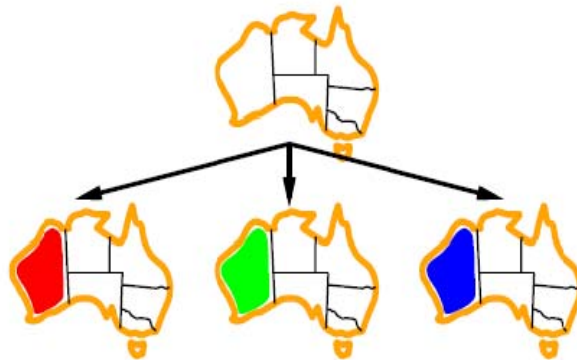
# Backtracking example



17

These slides are based on the slides at AIMA book webpage

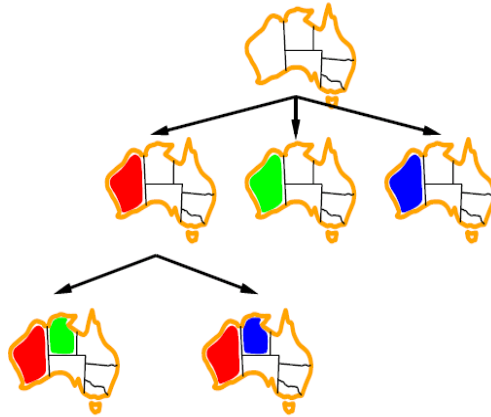
# Backtracking example



18

These slides are based on the slides at AIMA book webpage

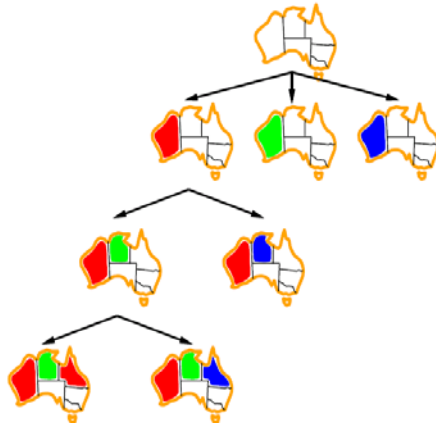
# Backtracking example



These slides are based on the slides at AIMA book webpage

19

# Backtracking example



These slides are based on the slides at AIMA book webpage

20

## Improving backtracking efficiency

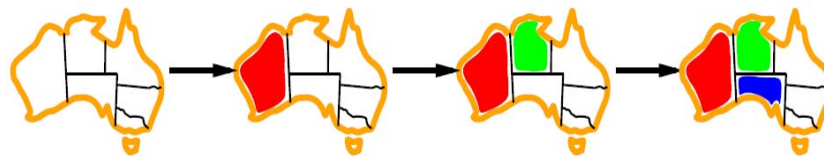


- Uninformed algorithm
  - general performance not good
  - Can solve  $n$ -queens for  $n \approx 25$
- Previously → introduce problem-specific heuristics to improve uninformed search
- General-purpose methods without problem-specific knowledge can give huge gains in speed:
  - Which variable should be assigned next?
  - In what order should its values be tried?
  - Can we detect inevitable failure early?

21

These slides are based on the slides at AIMA book webpage

## Minimum remaining values (MRV)

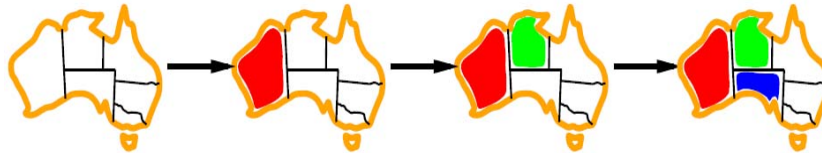


*Which variable shall we try next?*

22

These slides are based on the slides at AIMA book webpage

## Minimum remaining values (MRV)

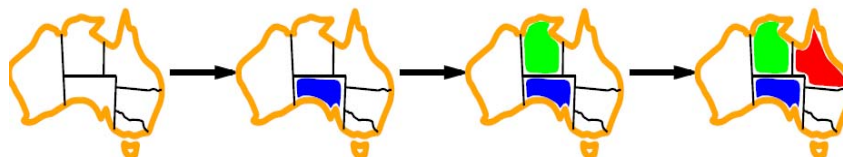


- A.k.a. *most constrained variable* heuristic
- *Rule*: choose variable with the fewest legal values

23

These slides are based on the slides at AIMA book webpage

## Degree heuristic

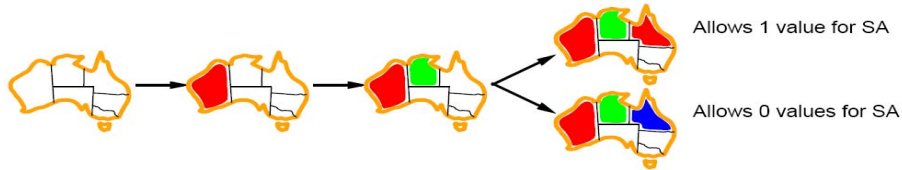


- *Rule*: select variable that is involved in the largest number of constraints on other unassigned variables. (Most constraining variable)
- Degree heuristic is very useful as a tie breaker among MRV variables.
- *In what order should its values be tried?*

24

These slides are based on the slides at AIMA book webpage

# Least constraining value

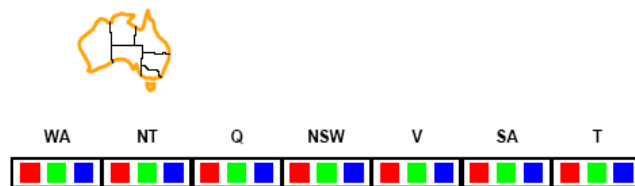


- Least constraining value heuristic
- Rule: given a variable choose the least constraining value,
  - i.e. the one that rules out the fewest values in the remaining variables

25

These slides are based on the slides at AIMA book webpage

# Forward checking

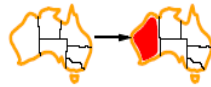


- Can we detect inevitable failure early?
- *Forward checking idea*: keep track of remaining legal values for unassigned variables.
  - Whenever a variable is assigned, delete from neighbors' domain any inconsistent value
  - Terminate search when any variable has no legal values.
- Provides an efficient way to incrementally compute the information that the MRV heuristic needs

26

These slides are based on the slides at AIMA book webpage

# Forward checking



- Assign  $\{WA=red\}$
- Effects on other variables connected by constraints with WA
  - *NT can no longer be red*
  - *SA can no longer be red*

27

These slides are based on the slides at AIMA book webpage

# Forward checking

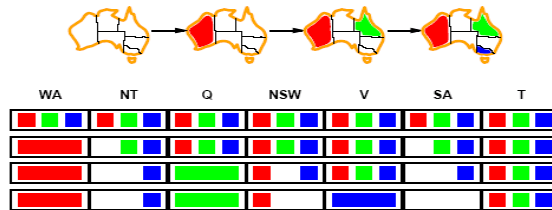


- Assign  $\{Q=green\}$
  - Effects on other variables connected by constraints with WA
    - *NT can no longer be green*
    - *NSW can no longer be green*
    - *SA can no longer be green*
- (MRV heuristic will automatically select NT or SA next)

28

These slides are based on the slides at AIMA book webpage

# Forward checking

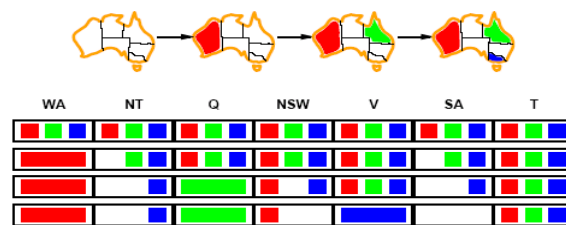


- If *V* is assigned *blue*
- Effects on other variables connected by constraints with *WA*
  - *SA is empty*
  - *NSW can no longer be blue*
- FC has detected that partial assignment is *inconsistent* with the constraints and backtracking can occur.
- Combining these heuristics makes 1000 queens feasible

29

These slides are based on the slides at AIMA book webpage

# Constraint propagation

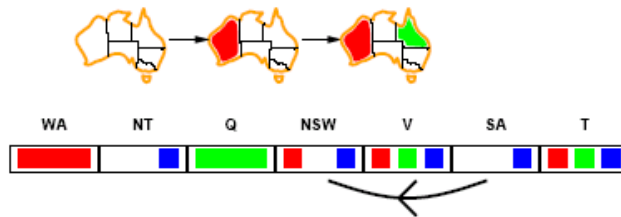


- FC propagates information from assigned to unassigned variables but does not provide early detection for all failures.
  - NT and SA cannot both be blue!
- *Constraint propagation* repeatedly enforces constraints locally

30

These slides are based on the slides at AIMA book webpage

# Arc consistency

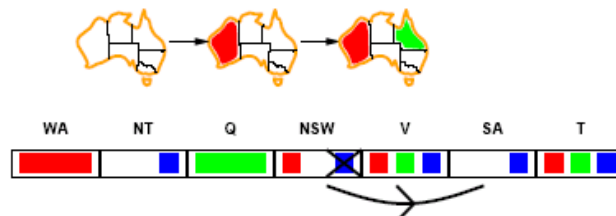


- Simplest form of propagation makes each arc **consistent** (in constraint graph)
- $X \rightarrow Y$  is consistent iff for **every** value  $x$  of  $X$  there is **some** allowed  $y$
- $SA \rightarrow NSW$  is consistent?

31

These slides are based on the slides at AIMA book webpage

# Arc consistency



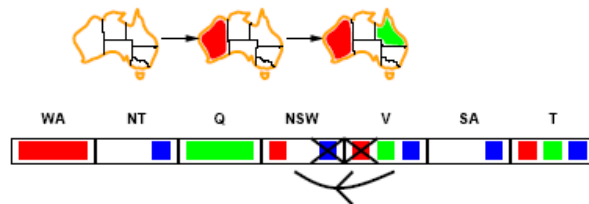
- $X \rightarrow Y$  is consistent iff for **every** value  $x$  of  $X$  there is some allowed  $y$
- $NSW \rightarrow SA$  is consistent?

Arc can be made consistent by removing *blue* from *NSW*

32

These slides are based on the slides at AIMA book webpage

# Arc consistency



- Arc can be made consistent by removing *blue* from *NSW*
- If  $X$  loses a value, neighbors of  $X$  need to be rechecked
- RECHECK neighbours of *NSW*!!
  - Remove red from *V*

33

These slides are based on the slides at AIMA book webpage

# Arc consistency algorithm



**function** AC-3(*csp*) **return** the CSP, possibly with reduced domains

**inputs:** *csp*, a binary csp with variables  $\{X_1, X_2, \dots, X_n\}$

**local variables:** *queue*, a queue of arcs initially the arcs in *csp*

**while** *queue* is not empty **do**

$(X_i, X_j) \leftarrow$  REMOVE-FIRST(*queue*)

**if** REMOVE-INCONSISTENT-VALUES( $X_i, X_j$ ) **then**

**for each**  $X_k$  **in** NEIGHBORS[ $X_i$ ] **do**

            add  $(X_k, X_i)$  to *queue*

**function** REMOVE-INCONSISTENT-VALUES( $X_i, X_j$ ) **return** *true* iff we remove a value

*removed*  $\leftarrow$  *false*

**for each**  $x$  **in** DOMAIN[ $X_i$ ] **do**

**if** no value  $y$  in DOMAIN[ $X_j$ ] allows  $(x, y)$  to satisfy the constraints between  $X_i$  and  $X_j$

**then delete**  $x$  from DOMAIN[ $X_i$ ]; *removed*  $\leftarrow$  *true*

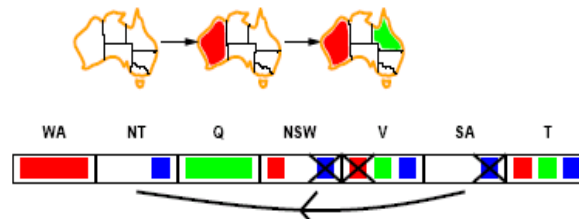
**return** *removed*

- Time complexity:  $O(n^2d^3)$

34

These slides are based on the slides at AIMA book webpage

## Arc consistency



- Arc consistency detects failure earlier than FC
- Can be run as a preprocessing step or after each assignment.
  - Repeated until no inconsistency remains
  - Much more expensive than FC, the extra cost is often worthwhile
- Stronger forms of propagation? k-consistency

35

These slides are based on the slides at AIMA book webpage

## Local search for CSP

- Typically use complete-state representation
- For CSPs
  - A state is an assignment of values to all variables - allow unsatisfied constraints
  - operators change the value of one variable at a time
- Min-conflicts algorithm
  - Variable selection: randomly select any conflicted variable
  - Value selection: ***min-conflicts heuristic***
    - Select new value that results in a minimum number of conflicts with the other variables
    - i.e., hill-climb with  $h(n) = \text{total number of violated constraints}$

36

These slides are based on the slides at AIMA book webpage

# Local search for CSP



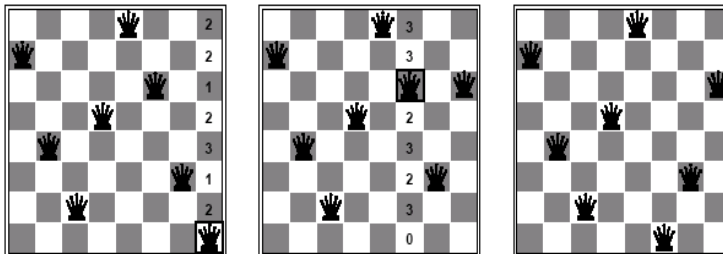
```
function MIN-CONFLICTS(csp, max_steps) return solution or failure
  inputs: csp, a constraint satisfaction problem
           max_steps, the number of steps allowed before giving up

  current ← an initial complete assignment for csp
  for i = 1 to max_steps do
    if current is a solution for csp then return current
    var ← a randomly chosen, conflicted variable from VARIABLES[csp]
    value ← the value v for var that minimize CONFLICTS(var, v, current, csp)
    set var = value in current
  return failure
```

37

These slides are based on the slides at AIMA book webpage

# Min-conflicts example



- A two-step solution for an 8-queens problem using min-conflicts heuristic.
- At each stage a queen is chosen for reassignment in its column.
- The algorithm moves the queen to the min-conflict square breaking ties randomly.

38

These slides are based on the slides at AIMA book webpage

## Advantages of local search

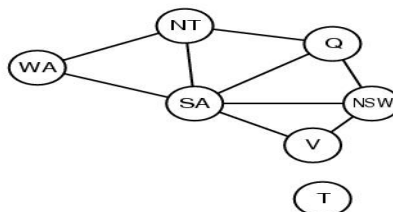


- Min-conflicts is surprisingly effective for many CSPs
  - Used to schedule observations for the Hubble Space Telescope
- Given random initial state, can solve  $n$ -queens in almost constant time for arbitrary  $n$  with high probability (e.g.,  $n = 10,000,000$ )
  - Solving the millions-queen problem in an average of 50 steps.

39

These slides are based on the slides at AIMA book webpage

## Problem structure

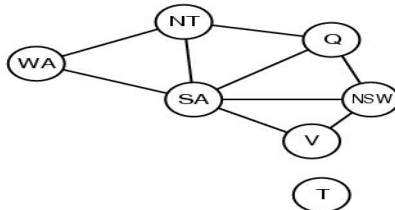


- *How can the problem structure help to find a solution quickly?*
- Subproblem identification is important:
  - Coloring Tasmania and mainland are independent subproblems
  - Identifiable as *connected components* of constraint graph.
- Improves performance

40

These slides are based on the slides at AIMA book webpage

## Problem structure

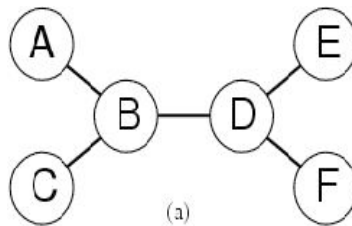


- Suppose each problem has  $c$  variables out of a total of  $n$ .
- Worst case solution cost is  $O(n/c d^c)$ 
  - Instead of  $O(d^n)$
- E.g.  $n = 80, c = 20, d = 2$ 
  - $2^{80} = 4$  billion years at 1 million nodes/sec.
  - $4 * 2^{20} = .4$  second at 1 million nodes/sec

41

These slides are based on the slides at AIMA book webpage

## Tree-structured CSPs

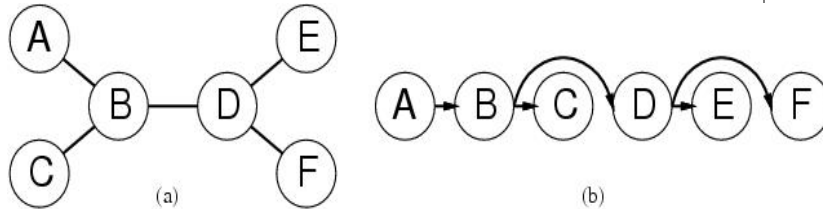


- Theorem: if the constraint graph has no loops then CSP can be solved in  $O(nd^2)$  time
- Compare difference with general CSP, where worst case is  $O(d^n)$

42

These slides are based on the slides at AIMA book webpage

## Tree-structured CSPs

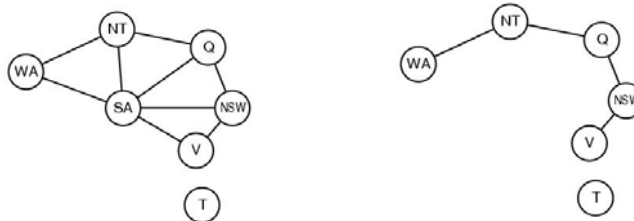


1. Choose a variable as root, order variables from root to leaves such that every node's parent precedes it in the ordering. (label var from  $X_1$  to  $X_n$ )
2. For  $j$  from  $n$  down to 2, apply arc consistency to  $\text{Parent}(X_j) \rightarrow X_j$ , removing values from domain of  $\text{Parent}(X_j)$  as necessary
3. For  $j$  from 1 to  $n$  assign  $X_j$  consistently with  $\text{Parent}(X_j)$

43

These slides are based on the slides at AIMA book webpage

## Nearly tree-structured CSPs

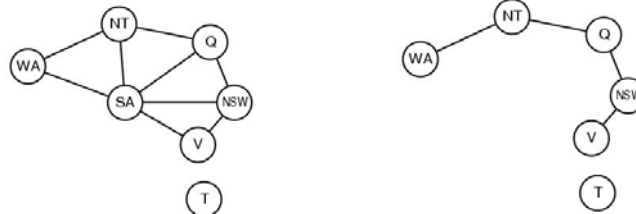


- One idea: assign values to some variables so that the remaining variables form a tree.
- Assume that we assign  $\{SA=x\} \leftarrow \text{cycle cutset}$ 
  - And remove any values from the other variables that are inconsistent.
  - The selected value for SA could be the wrong one so we have to try all of them

44

These slides are based on the slides at AIMA book webpage

## Nearly tree-structured CSPs



- This approach is worthwhile if cycle cutset is small.
  - $O(d^c (n-c)d^2)$
- Finding the smallest cycle cutset is NP-hard
  - Approximation algorithms exist
- This approach is called *cutset conditioning*.

45

These slides are based on the slides at AIMA book webpage

## 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 lead to failure.
- Constraint propagation (e.g., arc consistency) does additional work to constrain values and detect inconsistencies.
- Min-conflicts local search is usually effective in practice.
- The CSP representation allows analysis of problem structure.
  - Tree structured CSPs can be solved in linear time.

46

These slides are based on the slides at AIMA book webpage