Artificial Intelligence

CS482, CS682, MW 1 – 2:15, SEM 201, MS 227

Prerequisites: 302, 365

Instructor: Sushil Louis, sushil@cse.unr.edu, http://www.cse.unr.edu/~sushil

Constraint satisfaction problems

- Constraints on the values of variables that define system state
- What's new
  - State is no longer a black box
    - Previously all you could do with states was
      - Test if two states were the same
      - Tell if a state was a goal state
  - Now: State space is defined by the values of a set of variables
  - Each variable's set of values is the variable's **domain**
  - There can be
    - Unary
    - Binary
    - Path
  - Constraints

CSP

- Find values of variables that satisfy all problem constraints
- How?
  - Search, of course
  - Can we use any search method?
  - Hmm, some intuition from considering specific problems will help

Three colour problem

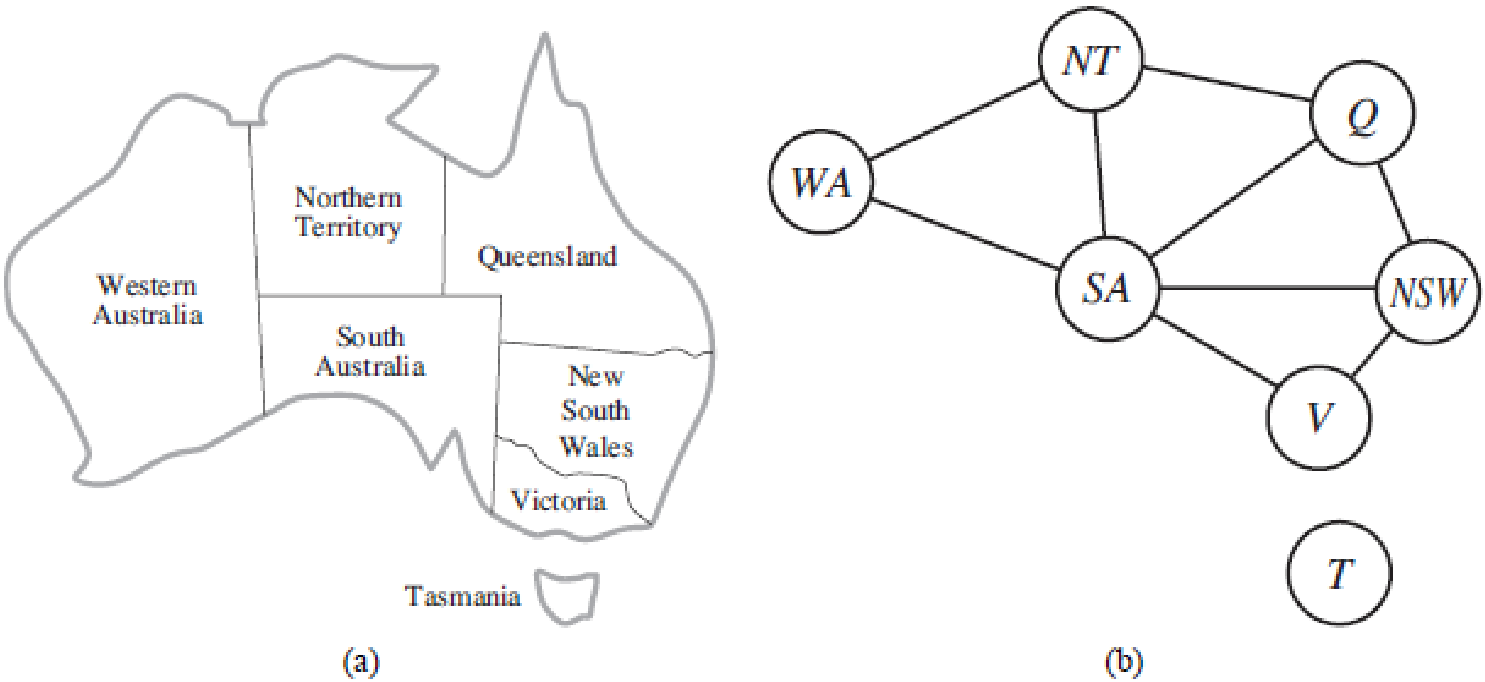

Figure 6.1 FILES: figures/australia.eps (Tue Nov 3 16:22:26 2009) figures/australia-csp.eps (Tue Nov 3 16:22:25 2009). (a) The principal states and territories of Australia. Coloring this map can be viewed as a constraint satisfaction problem (CSP). The goal is to assign colors to each region so that no neighboring regions have the same color. (b) The map-coloring problem represented as a constraint graph.

Neighboring regions cannot have the same color
Colors = {red, blue, green}

Consider using a local search

WA	NT	NSW	Queen	Victoria	SA	Tas
{r, g, b}	{r, g, b}	{r, g, b}	{r, g, b}	{r, g, b}	{r, g, b}	{r, g, b}

- 3 to the power 7 possible states = 2187
- But not all states are legal
- For example: {r, r, r, r, r, r, r} is NOT legal because it violates our constraint

- Suppose we do sequential assignment of values to variables
- Assign r (say) to WA then we can immediately reduce the number of possible values for NT and SA to be {g, b}, and if we chose NT = {g}, then SA has to be {b}.

Propagation of constraints

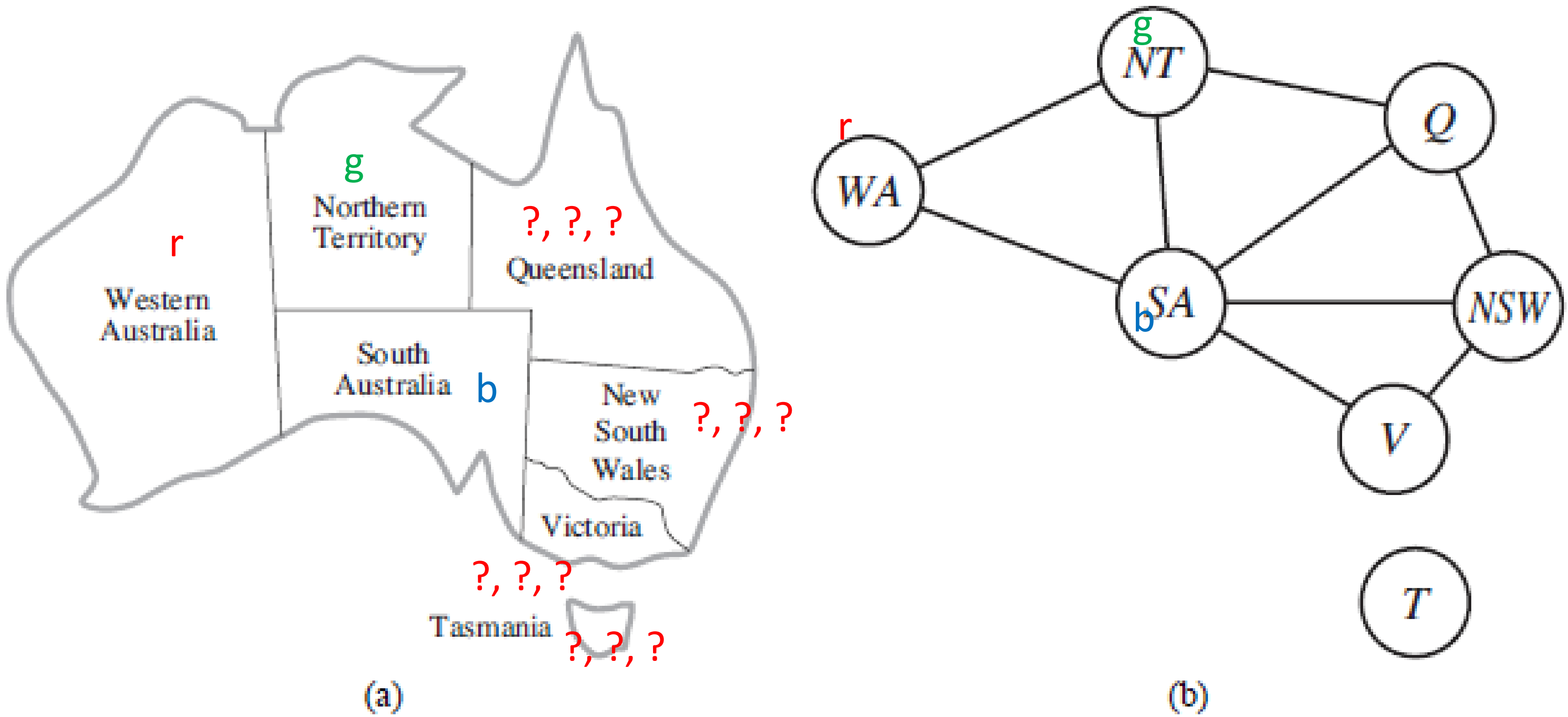

Figure 6.1 **FILES: figures/australia.eps (Tue Nov 3 16:22:26 2009) figures/australia-csp.eps (Tue Nov 3 16:22:25 2009).** (a) The principal states and territories of Australia. Coloring this map can be viewed as a constraint satisfaction problem (CSP). The goal is to assign colors to each region so that no neighboring regions have the same color. (b) The map-coloring problem represented as a constraint graph.

Types

- Discrete finite domains
  - Map coloring, scheduling with time limits
  - Can enumerate all legal value combinations (that specify constraints)
- Discrete infinite domains
  - Ex: Variable values can be the set of integers
  - Needs a constraint language to specify constraints
  - We have solutions for linear constraints over integers
  - We can prove that no algorithm exists for solving general nonlinear constraints over integers
- Continuous domains
  - Hubble telescope scheduling is continuous over time and must obey a variety of astronomical, precedence, and power constraints
  - Linear programming → poly time algorithms

Types

- Unary constraints
  - Truck height < 14 feet
- Binary constraints
  - WA != SA

Cryptarithmetic puzzles

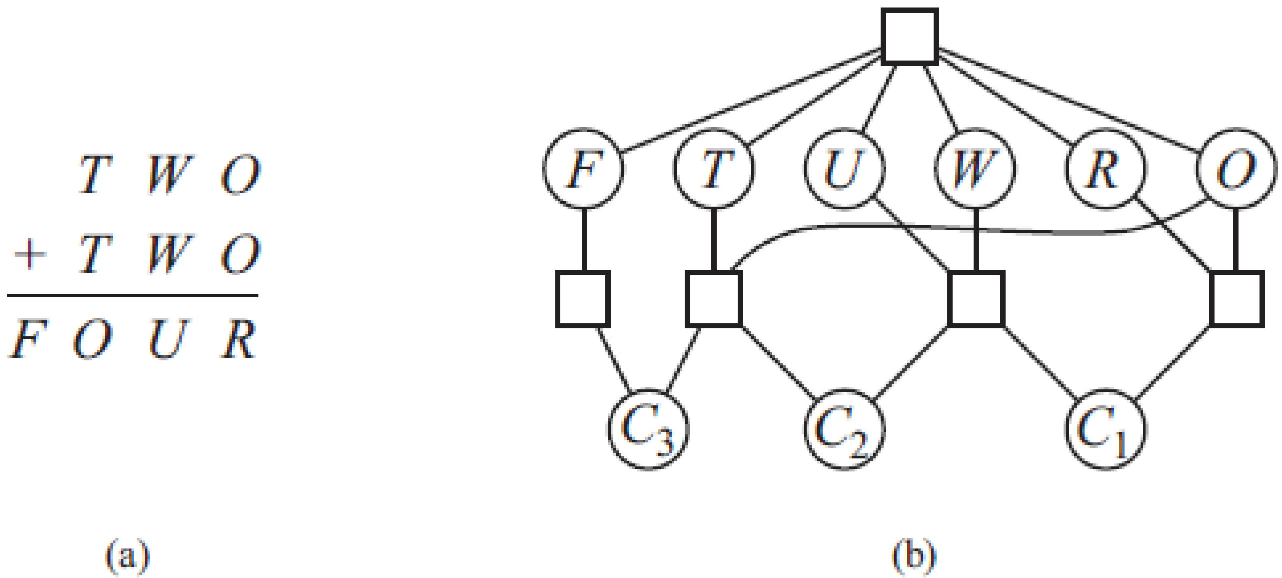

Figure 6.2 FILES: figures/cryptarithmetic.eps (Tue Nov 3 13:31:28 2009). (a) A cryptarithmetic problem. Each letter stands for a distinct digit; the aim is to find a substitution of digits for letters such that the resulting sum is arithmetically correct, with the added restriction that no leading zeroes are allowed. (b) The constraint hypergraph for the cryptarithmetic problem, showing the *Alldiff* constraint (square box at the top) as well as the column addition constraints (four square boxes in the middle). The variables C_1, C_2, and C_3 represent the carry digits for the three columns.

Wouldn't it be nice to have a constraint propagation algorithm?

```
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) ← 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
```

```
function REVISE(csp, X_i, X_j) returns true iff we revise the domain of X_i
   revised ← 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 ← true
   return revised
```

Figure 6.3 The arc-consistency algorithm AC-3. After applying AC-3, either every arc is arc-consistent, or some variable has an empty domain, indicating that the CSP cannot be solved. The name "AC-3" was used by the algorithm's inventor (?) because it's the third version developed in the paper.

Properties

- Node consistency (unary)
- Arc consistency (binary)
  - Network arc consistency (all arcs are consistent)
- ACS3 is the most popular arc consistency algorithm
  - **Fails quickly if no consistent set of values found**
  - Start:
    - Considers all pairs of arcs
    - If making an arc (xi, xj) consistent causes domain reduction
      - **Add** all neighboring arcs that go to xi to set of arcs to be considered
  - Success leaves a much smaller search space for search
    - Domains will have been reduced
  - Suppose n variables, max domain size is d, then complexity is O(cd^3) where c is number of binary constraints

More constraint types and approache

- Path (triples)
- Global constraints (n variables)
  - Special purpose algorithms (heuristics)
  - Alldiff constraints (Sudoku)
    - Remove any variable with singleton domain
    - Remove that value from the domains of all other variables
    - Repeat
      - While
        - singletons values remain
        - No domains are empty
        - Not more variables than domain values
- Resource constraints (Ex: Atmost 100)
- Bounds and bounds propagation

Search

- Constraints have been met and propagated
- But the problem still remains to be solved (multiple values in domains)
  - Search through remaining assignments
- For CSPs **Backtracking search** is good
  - Choose a value for variable, x
  - Choose a subsequent legal value for next variable, y
  - Backtrack to x if no legal value found for y

Australia coloring

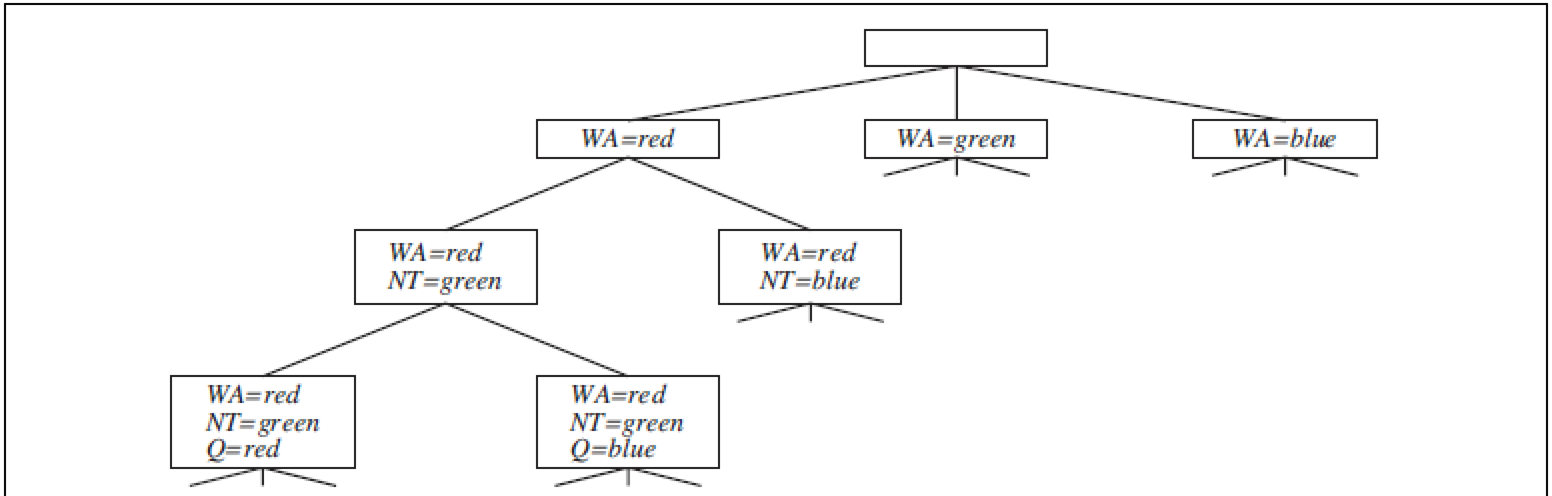

Figure 6.6 **FILES: figures/australia-search.eps (Tue Nov 3 16:22:25 2009).** Part of the search tree for the map-coloring problem in Figure 6.1.

Backtracking search algorithm

```
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)
         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
```

Figure 6.5 A simple backtracking algorithm for constraint satisfaction problems. The algorithm is modeled on the recursive depth-first search of Chapter ??. By varying the functions SELECT-UNASSIGNED-VARIABLE and ORDER-DOMAIN-VALUES, we can implement the general-purpose heuristics discussed in the text. The function INFERENCE can optionally be used to impose arc-, path-, or *k*-consistency, as desired. If a value choice leads to failure (noticed either by INFERENCE or by BACKTRACK), then value assignments (including those made by INFERENCE) are removed from the current assignment and a new value is tried.

Local search for CSPs

```
function MIN-CONFLICTS(csp, max_steps) returns a 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 csp.VARIABLES
     value ← the value v for var that minimizes CONFLICTS(var, v, current, csp)
     set var = value in current
  return failure
```

Figure 6.8 The MIN-CONFLICTS algorithm for solving CSPs by local search. The initial state may be chosen randomly or by a greedy assignment process that chooses a minimal-conflict value for each variable in turn. The CONFLICTS function counts the number of constraints violated by a particular value, given the rest of the current assignment.

SudoKu

	1	2	3	4	5	6	7	8	9
A			3		2		6		
B	9			3		5			1
C			1	8		6	4		
D			8	1		2	9		
E	7								8
F			6	7		8	2		
G			2	6		9	5		
H	8			2		3			9
I			5		1		3		

(a)

	1	2	3	4	5	6	7	8	9
A	4	8	3	9	2	1	6	5	7
B	9	6	7	3	4	5	8	2	1
C	2	5	1	8	7	6	4	9	3
D	5	4	8	1	3	2	9	7	6
E	7	2	9	5	6	4	1	3	8
F	1	3	6	7	9	8	2	4	5
G	3	7	2	6	8	9	5	1	4
H	8	1	4	2	5	3	7	6	9
I	6	9	5	4	1	7	3	8	2

(b)

Figure 6.4 **FILES: figures/sudoku.eps (Tue Nov 3 13:49:46 2009).** (a) A Sudoku puzzle and (b) its solution.