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

Three colour problem

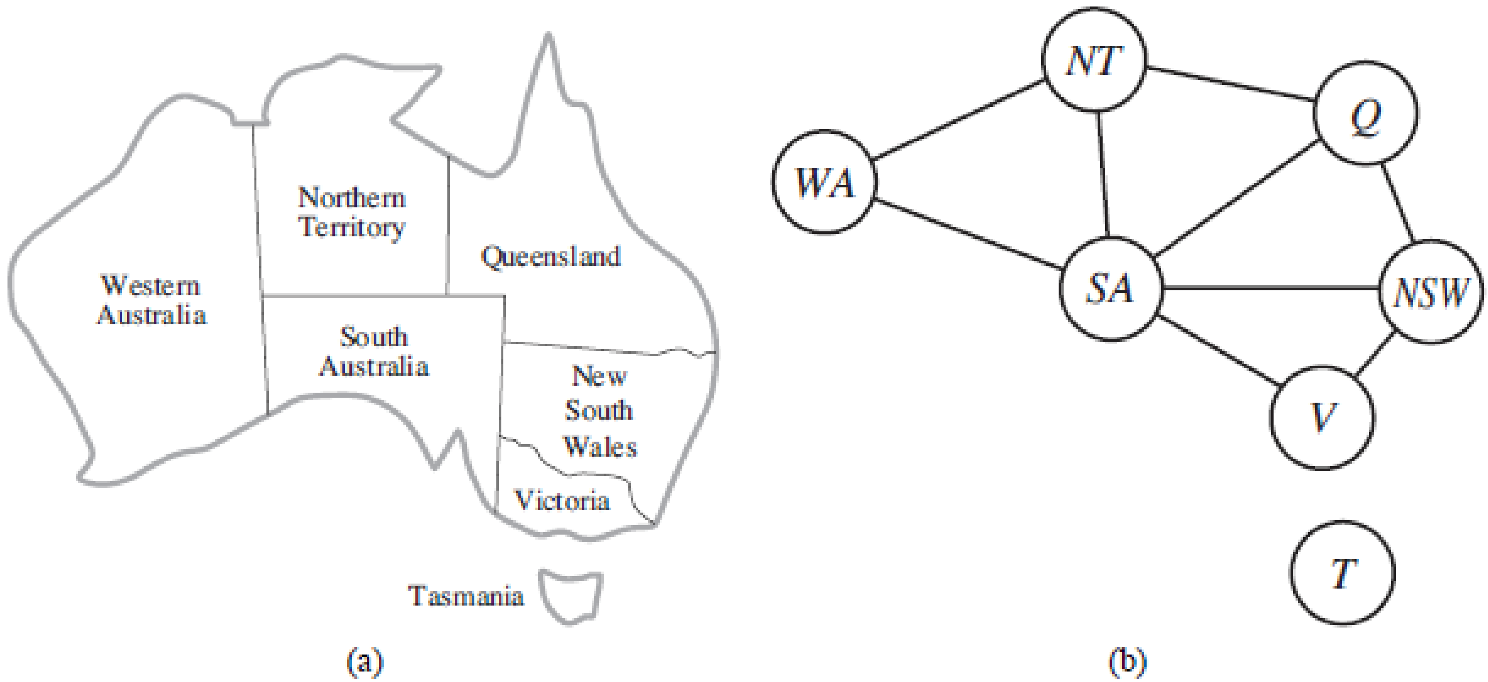

Figure 6.1 FILES: figures/australia.eps (Tue Nov 3 16:22:26 2009) figures/australia-csp.eps (Tue Nov 3 16:22:25 2009). (a) The principal states and territories of Australia. Coloring this map can be viewed as a constraint satisfaction problem (CSP). The goal is to assign colors to each region so that 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

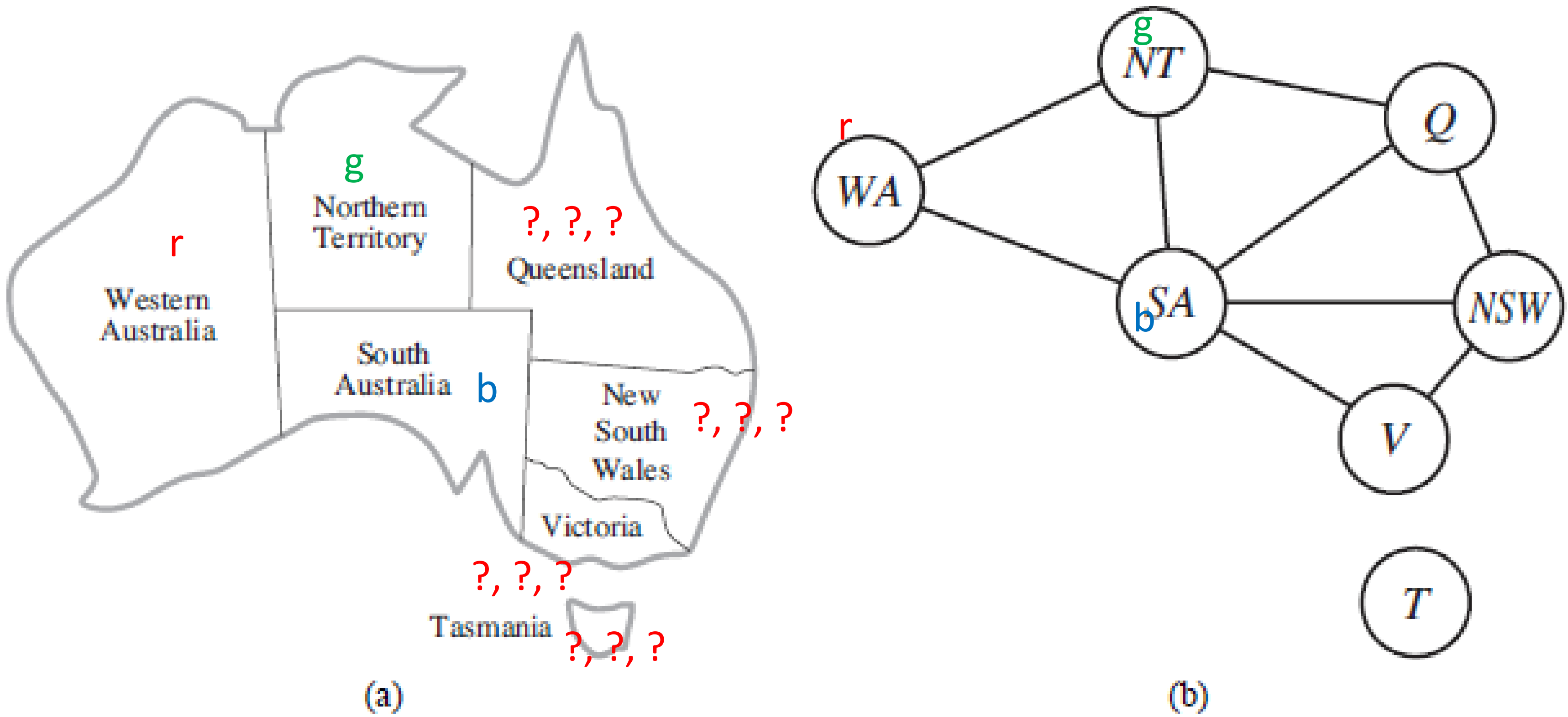

Figure 6.1 **FILES: figures/australia.eps (Tue Nov 3 16:22:26 2009) figures/australia-csp.eps (Tue Nov 3 16:22:25 2009).** (a) The principal states and territories of Australia. Coloring this map can be viewed as a constraint satisfaction problem (CSP). The goal is to assign colors to each region so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

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

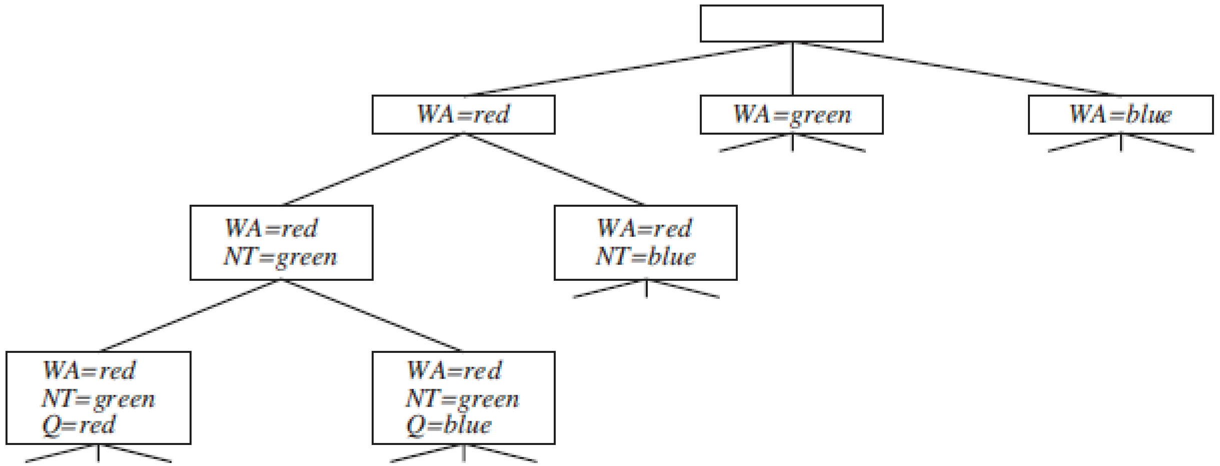

Figure 6.6 FILES: figures/australia-search.eps (Tue Nov 3 16:22:25 2009). Part of the search tree for the map-coloring problem in Figure 6.1.

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.

CSP heuristics

- For **all** CSPs
- Depends on the answer to the following:
  - Which var should be assigned next, and what order should it be assigned a value from the set of values available?
  - What inference should be performed at each step of search?
  - When the search arrives at an assignment that violates a constraint, can the search avoid repeating this failure?

Variable and value ordering

- Choosing which variable:
  - Minimum Remaining Value (MRV) heuristic aka fail-fast
    - Choose the variable with the fewest remaining “legal” values
  - Degree heuristic
    - Choose variable that is involved in the largest number of constraints
- Choosing which value:
  - Least constraining value (fail-last)

Interleaving search & inference

- AC-3 infers reductions in set of possible values before search
- Inference is also powerful during search
- Consider backtracking search + Forward checking
- FC: After X assigned,
  - For each unassigned var Y that is connected to X, delete any values from Y's domain that is inconsistent with the value chosen for X
  - After WA = red
    - Forward check
  - After Q = green
    - Forward check
  - NT = {blue}, SA = {blue}
  - V = {blue} ➔ SA = {}

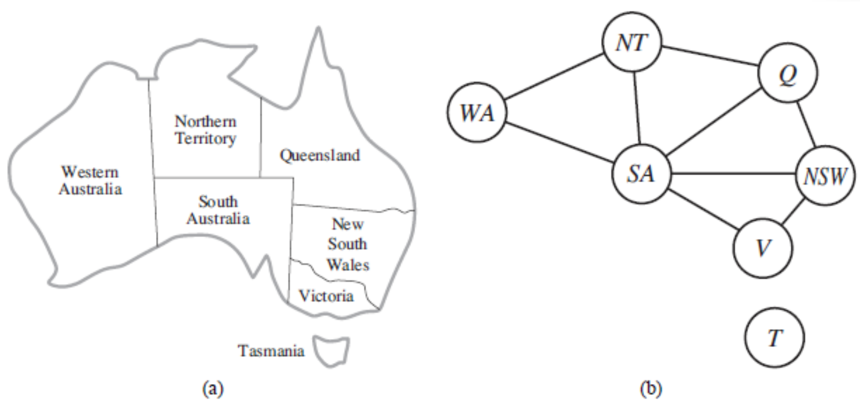

Figure 6.1 **FILES: figures/australia.eps (Tue Nov 3 16:22:26 2009) figures/australia-csp.eps (Tue Nov 3 16:22:25 2009).** (a) The principal states and territories of Australia. Coloring this map can be viewed as a constraint satisfaction problem (CSP). The goal is to assign colors to each region so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

  - Backtrack because there is no assignment for SA

Inference + search

- Backtracking + AC3 = Maintaining Arc Consistency (MAC algorithm)
  - Fails faster than Backtracking + forward checking

	WA	*NT*	*Q*	*NSW*	*V*	*SA*	*T*
Initial domains	R G B	R G B	R G B	R G B	R G B	R G B	R G B
After *WA=red*	R	G B	R G B	R G B	R G B	G B	R G B
After *Q=green*	R	B	G	R B	R G B	B	R G B
After *V=blue*	R	B	G	R	B		R G B

Figure 6.7 FILES: figures/australia-fc.eps (Tue Nov 3 16:22:25 2009). The progress of a map-coloring search with forward checking. $WA = red$ is assigned first; then forward checking deletes *red* from the domains of the neighboring variables NT and SA. After $Q = green$ is assigned, *green* is deleted from the domains of NT, SA, and NSW. After $V = blue$ is assigned, *blue* is deleted from the domains of NSW and SA, leaving SA with no legal values.

Heuristic backtracking

- Q = red, NSW = green, V = blue, T = red, SA = ?
  - Every value of SA violates a constraint
  - Should we backtrack to T = red?
  - But T = red does not have anything to do with SA
- Carry around a **conflict set**, a set of **prior** assignments that affects SA
- {Q=red, NSW=green, V = blue} == **conflict set** for SA
- FC may specify a conflict set!
- Conflict set
  - tells us not to backtrack to T
  - instead to V
- **Back Jumping** algorithm

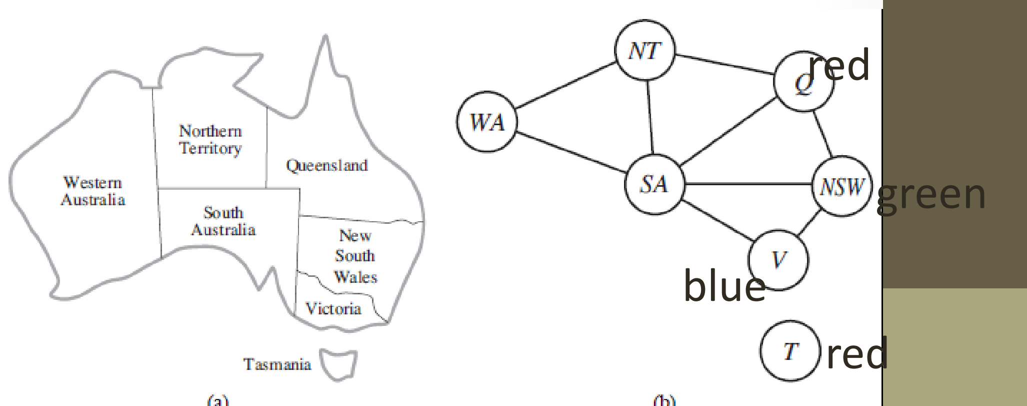

Figure 6.1 **FILES: figures/australia.eps (Tue Nov 3 16:22:26 2009) figures/australia-csp.eps (Tue Nov 3 16:22:25 2009).** (a) The principal states and territories of Australia. Coloring this map can be viewed as a constraint satisfaction problem (CSP). The goal is to assign colors to each region so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

Conflict-directed back jumping

- Not that simple:
- Consider {WA = red, NSW = red}
  - Is this possible?
  - Now, assign to T,
  - then to NT, Q, V, SA
  - Because of earlier inconsistency
    - No possible assignment
    - So we backtrack to NT
      - Try other values and still fail!
      - NT's conflict set {WA} is not complete

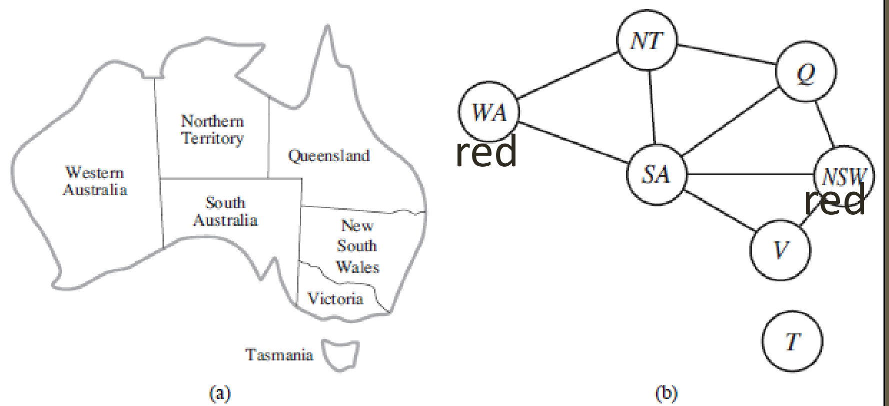

Figure 6.1 **FILES: figures/australia.eps (Tue Nov 3 16:22:26 2009) figures/australia-csp.eps (Tue Nov 3 16:22:25 2009).** (a) The principal states and territories of Australia. Coloring this map can be viewed as a constraint satisfaction problem (CSP). The goal is to assign colors to each region so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

- FC does not always provide enough information
- Consider:
  - SA fails and SA's conflict set is (say) {WA, NSW, NT, Q}
  - We backjump to Q and Q **absorbs** SA's conflict set – Q
    - Q's conflict set = {NT, NSW} (we haven't seen SA yet)
    - SAcs Union Qcs - Q = {WA, NT, NSW} → no solution forward from Q given Qcs
    - Backtrack to NT which absorbs {WA, NT, NSW} – {NT} = {WA, NSW}
    - Back jump to NSW

Constraint learning

- Can we learn sets of variable assignments that lead to conflicts?
  - **NO GOOD** == {min set of variable and their values in a conflict set that lead to contradiction}

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.

CSP problem structure

- Independent sub-problems
  - Very nice
- Tree structure (any two variables are only connected by one path)
  - Linear time!   O(nd^2)
- Can we convert a constraint graph to a tree structure?
  - 1. Removing nodes (delete SA!)
    - By assigning a value to SA and removing that value from all other nodes' domains
    - In general, find a cycle cutset, and return cutset's assignment and remaining tree CSP
    - d^c * (n-c)d^2

Removing nodes

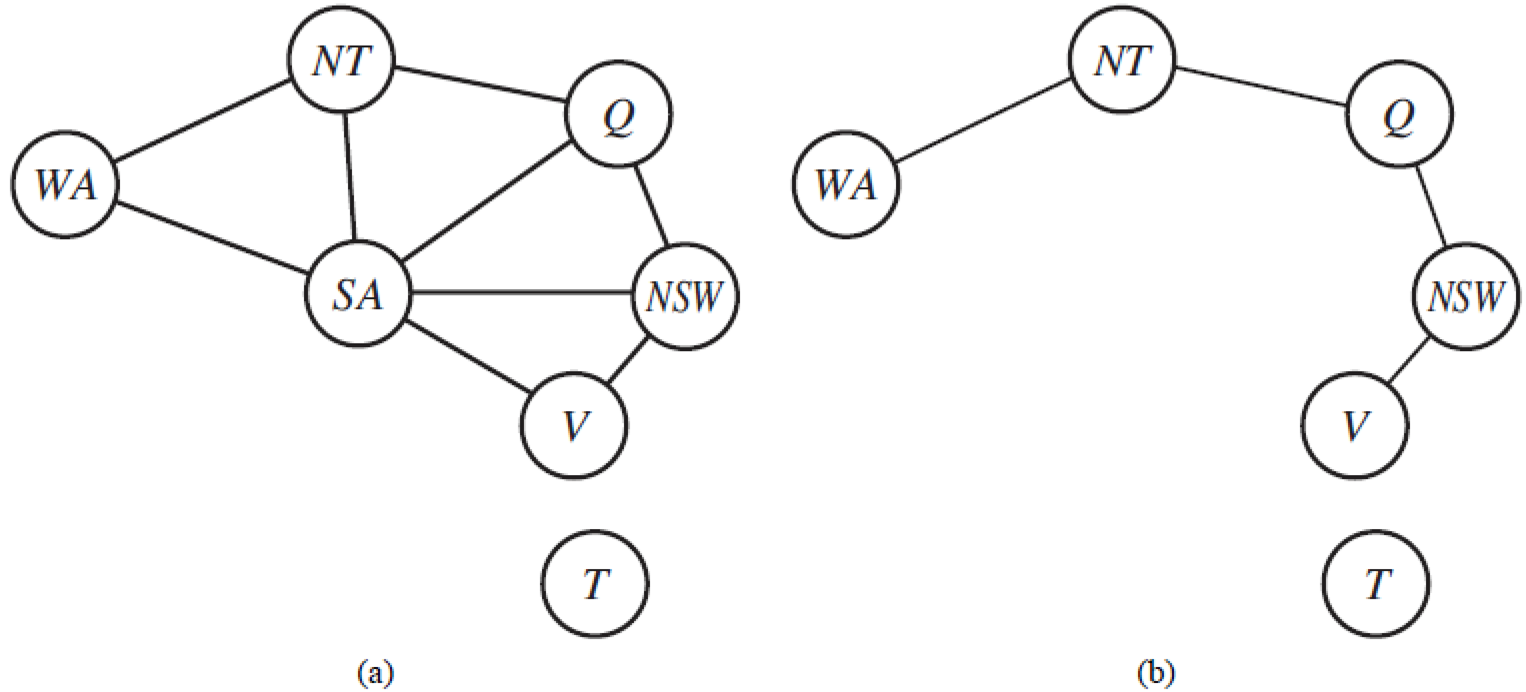

Figure 6.12 **FILES: figures/australia-csp.eps (Tue Nov 3 16:22:25 2009) figures/australia-tree.eps (Tue Nov 3 16:22:26 2009).** (a) The original constraint graph from Figure 6.1. (b) The constraint graph after the removal of SA.

Collapsing nodes

- Tree decomposition of constraint graph into a set of connected sub-problems.
  - Great if **tree width** of Constraint Graph is small
  - But
    - Many possible decompositions

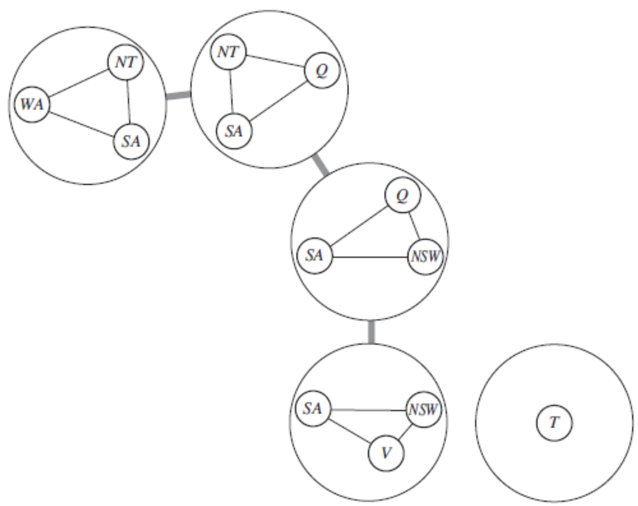

Figure 6.13 FILES: figures/australia-decomposition.eps (Tue Nov 3 16:22:25 2009). A tree decomposition of the constraint graph in Figure 6.12(a).

CSP Puzzle

- In five houses, each with a different color, live five persons of different nationalities, each of whom prefer a different brand of candy, a different drink, and a different pet.
- Where does the zebra live?
- Which house do they drink water?

- What are possible representations of this CSP problem?
- Which is best?

- The Englishman lives in the red house
- The Spaniard owns the dog
- The Norwegian lives in the first house on the left
- The green house is immediately to the right of the ivory house
- The man who eats Hershey bars lives in the house next to the man with the fox
- Kits Kats are eaten in the yellow house
- The Norwegian lives next to the blue house
- The Smarties eater owns snails
- The Snickers eater drinks OJ
- The Ukranian drinks tea
- The Japanese eats Milky Ways
- Kit Kats are eaten in a house next to the house where the horse is kep
- Coffee is drunk in the green house
- Milk is drunk in the middle house

Logical Agents