Constraint Satisfaction Problems (CSPs)

CPS 170

Ron Parr

CSPs

- What is a CSP?
- One view: Search with special goal criteria
- CSP definition (general):
  - Variables $X_1,\dots,X_n$
  - Variable X_i has domain D_i
  - Constraints $C_1,\dots,C_m$
  - Solution: Each variable gets a value from its domain such that no constraints violated
- CSP examples…
  - http://www.csplib.org/

Other CSP Examples

- Satisfying curriculum/major requirements
- Sudoku
- Seating arrangements at a party
- LSAT Questions: http://www.lsac.org/JD/pdfs/SamplePTJune.pdf

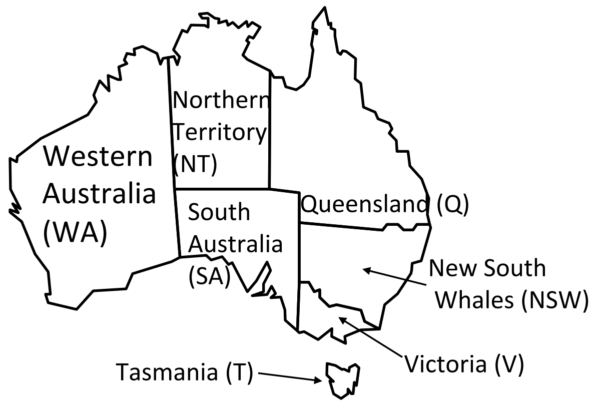
A Restricted View

- Variables $X_1, \ldots, X_n$
- A binary constraint, lists permitted assignments to pairs of variables
- A binary constraint between binary variables is a table of size 4, listing legal assignments for all 4 combinations.
- A k-ary constraint lists legal assignments to k variables at a time.
- How large is a k-ary constraint for binary variables?

Note: More expressive languages are often used.

CSP Example

Graph coloring:



Problem: Assign Red, Green and Blue so that no 2 adjacent regions have the same color. (3-coloring)

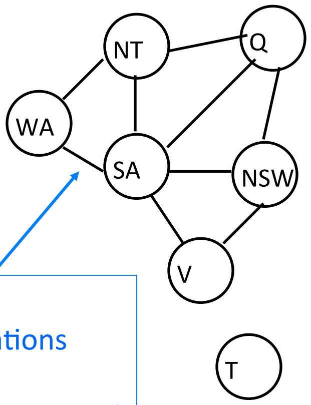
Example Contd.

- Variables: {WA, NT, Q, SA, NSW, V, T}
- Domains: {R,G,B}
- Constraints:

  For WA – NT:{(R,G), (R,B), (G,B), (G,R), (B,R), (B,G)}
- We have a table for each adjacent pair
- Are our constraints binary?
- Can every CSP be viewed as a graph problem?

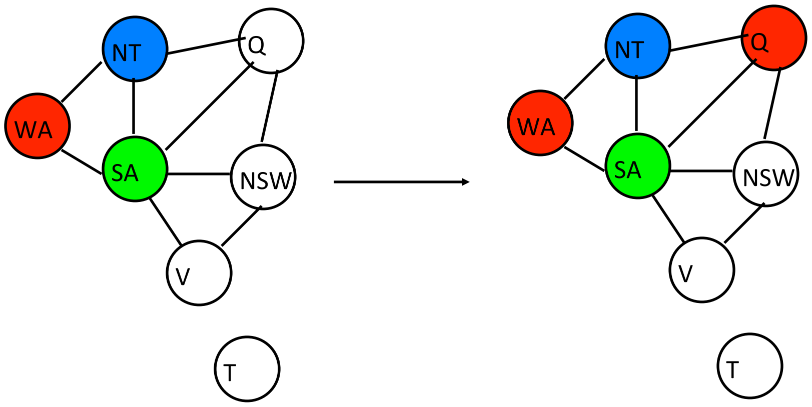
Constraint Graph

WA
NT
Q
SA
NSW
V
T



Enumerate all
Legal combinations
Of WA and SA
(ignoring other regions)

CSPs as Search



Nodes: Partial Assignments

Actions: Make Assignments

Backtracking

- Backtracking is the most obvious (and widely used) method for solving CSPs:
  - Search forward by assigning values to variables
  - If stuck, undo the most recent assignment and try again
  - Repeat until success or all combinations tried
- Embellishments
  - Methods for picking next variable to assign (e.g. most constrained)
  - Backjumping

NP-Completeness of CSPs

- Are CSPs in NP?
- Are they NP-hard?

- CSPs and graph coloring are equivalent
  - Convert any graph coloring problem to CSP
  - Convert any CSP to graph coloring
- Known: Graph coloring is NP-complete
- CSPs are NP-complete
- End of the story or just the beginning?

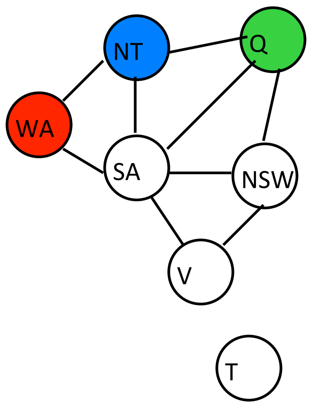
Issues

- What are good heuristics?
  - N.B.: Here we use the term "heuristic" to refer to a *procedure* for selecting next variables, **not** an h(x) function as in A*
  - Often good to think of this as a local search
  - Focus on choosing actions carefully, instead of pruning nodes carefully (as in A* or alpha-beta)
- Can we develop heuristics that apply to the entire class of problems, not just specific instances?
- What's the best we can hope for?

Constraint Graphs

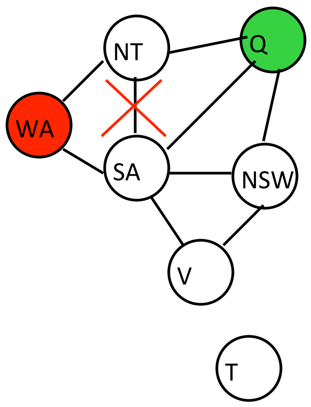
- Constraint graphs are important because they capture the structural relationships between the variables
- IMPORTANT CONCEPT:
  Not all instances of a hard problem class are hard
  - Structural features give insight into hardness
  - Example: Planar graphs are known to be 4-colorable
  - Group problems within class by structural features
  - New measure of problem complexity

Node Consistency



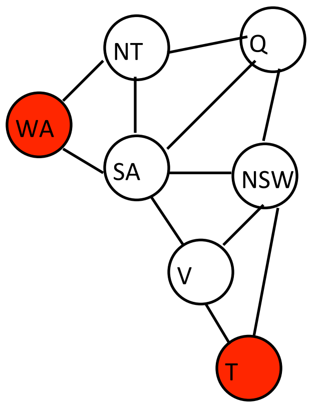
- Check all nodes to verify that set of possible values is non-empty
- How can a set become empty?
- Constraint propagation:
  - After assigning R to WA, we can remove R from the set of legal assignments to SA
  - Constraint propagation w/node consistency checking can discover bad choices quickly

Arc Consistency



- Check all arcs for inconsistencies
- For each value at the start, there must exist a consistent value at the terminus
- Catches many inconsistencies
- Can use to iteratively reduce number of possible assignments to each variable
  (constraint propagation)

K-Consistency



- k-consistency
  - Consider sets of k variables
  - For each legal setting of a k-1 subset
  - Check for legal setting for the kth variable
- Checks for more distant influences

- 1-consistency = node consistency
- 2 consistency = arc consistency

Is this 3-consistent? (assume we've done constraint propagation)

Facts About Arc Consistency

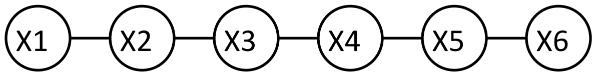
- Strong k-consistency: Consistent for all i<k
- What if a graph with n variables is strongly n-consistent?

Solution exists!

- What is the worst-case cost of checking n-consistency?

$O(2^n)$

Linear Constraint Structures



Are these easy or hard?

Suppose our chain is arc consistent…

Properties of Chains

Theorem: Arc consistent linear constraint graphs are strongly n consistent.

Proof: Induction on n.

Base: Arc consistent chains of length 1 are consistent.

I.H. Arc consistent chains of length i are strongly i consistent

I.S. Extending an i step arc-consistent chain by 1 new arc consistent link produces an i+1 link strongly i+1 consistent chain.

Proof of I.S.: Since the last link is strongly arc-consistent, any choice for variable i ensures a consistent choice for 0…i. Newly added node is consistent. No other variables participate in constraints for i+1.

Properties of Trees

Theorem: Arc consistent constraint trees are strongly n consistent.

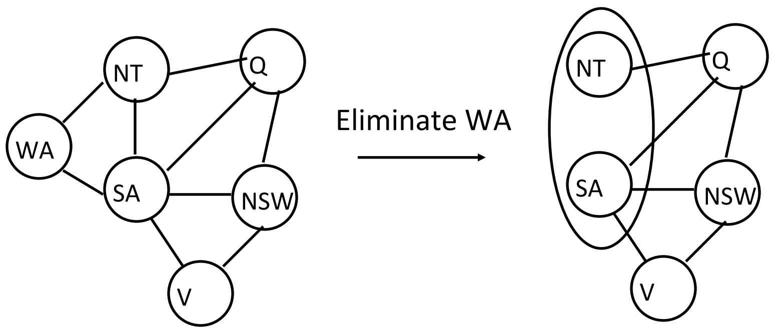
Proof: Same as chain case...

Corollary: Hardness of CSPs with constraint trees

Polynomial!

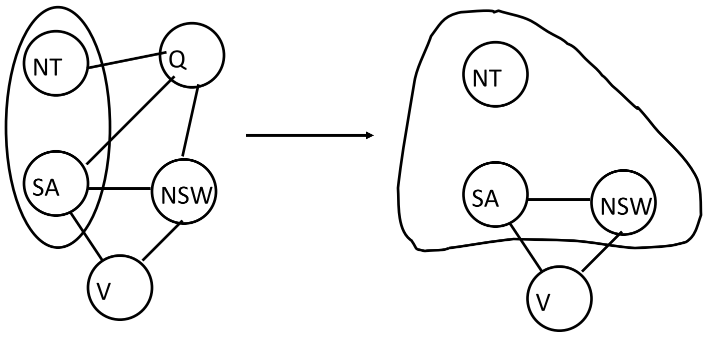
Cool fact: We now have a graph-based test for separating out some of the hard problems from the easy ones.

Variable Elimination



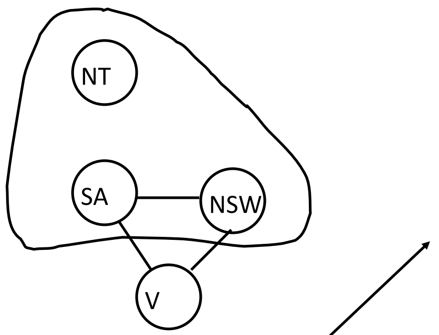
Domain(NT,SA) = {(blue, green), (blue, red), (green, blue), (green, red), (red, blue), (red, green)}

Eliminate Q



Domain(NT,SA,NSW) = {(blue, green, blue), (blue, red, blue),
(red, blue, red), (red, green, red), (green, blue, green),
(green, red, green)}

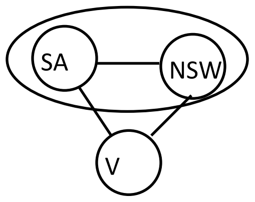
Simplify



Domain(SA, NSW) =
{(blue, green), (blue, red),
(green, blue), (green, red),
(red, blue), (red, green)}

Domain(NT,SA,NSW) = {(blue, green, blue), (blue, red, blue),
(red, blue, red), (red, green, red), (green, blue, green),
(green, red, green)}

Finish



Domain(SA, NSW) =
{(blue, green), (blue, red),
(green, blue), (green, red),
(red, blue), (red, green)}

Can identify all settings of SA, V, NSW for which there is guaranteed to be a consistent setting of the remaining variables.

Q: How do we get the settings of the other variables?

Variable Elimination

```
Var_elim_CSP_solve (vars, constraints)
Q = queue of all variables
i = length(vars)+1
While not(empty(Q))
    X = pop(Q)
    Xi = merge(X, neighbors(X))
    Simplify Xi (remove variables w/o external connections)
    remove_from_Q(Q, neighbors(X))
    add_to_Q(Q, Xi)
    i=i+1
```

Note: Merge operation can be tricky to implement, depending upon constraint language.

Variable Elimination Issues

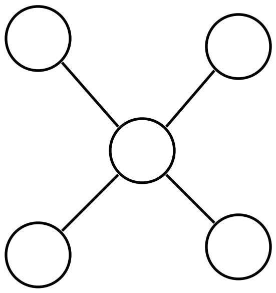
- How expensive is this?

  Exponential in size of largest merged variable set.

- Is it sensitive to elimination ordering?

  Yes!

Variable Elimination Ordering



Is it better to start at the edges and work in, or at the center and work out?

Edges!

Variable Elimination Facts

- You can figure out the cost of a particular elimination ordering without actually constructing the tables
- Finding optimal elimination ordering is NP hard
- Good heuristics for finding near optimal orderings
- Another structural complexity measure
- Investment in finding good ordering can be amortized

CSP Summary

- CSPs are a specialized language for describing certain types of decision problems
- We can formulate special heuristics and methods for problems that can be described in this language
- In general, CSPs are NP hard – no general, fast solutions on the horizon
- In some cases, we can use structural measures of complexity to figure out which ones are really hard