CSPs II

CPS 271
Ron Parr

Review

- CSPs are NP-hard decision problems
- Can formulate SAT instances as CSPs
- Paths don’t matter – often use local search
- What’s special about CSPs?
  - Special problem description language
  - Use CSP specific heuristics, methods instead of problem specific heuristics (A*)
- CSP examples…
  - http://www-users.cs.york.ac.uk/~tw/csplib/

CSP Example

Graph coloring:

- Western Australia (WA)
- Northern Territory (NT)
- Queensland (Q)
- South Australia (SA)
- New South Wales (NSW)
- Victoria (V)

Problem: Assign colors, Red, Green and Blue so that no 2 adjacent Regions have the same color.

Constraint Graph

For CSPs with binary constraints!

Constraint Propagation

- Forward checking
  - Reduce domains after assignments are made to some variables
- Constrains search space
- Forces choices
- Eliminates branching

Arc Consistency

- Stronger than propagation
  - Check all directed arcs for inconsistencies
  - For each value at the start, there must exist a consistent value at the terminus
- Catch inconsistencies early
- Reduces domains
Propagating Arc Consistency

Let Q be a queue of all edges
While not(empty(Q))
    \((i,j) = \text{pop}(Q)\)
    success = remove consistencies(i,(i,j))
    if success then
        for k in neighbors(i)
            push(k,Q)

Generalized Arc Consistency

- k-consistency
  - Consider sets of k variables
  - For each setting of a k-1 subset
  - Must exist a consistent setting for the kth variable
- Check for more distant influences
- Strong k-consistency
  - Strong implies j<k consistency

Facts About Arc Consistency

- What if a graph with n variables is strongly n-consistent?
  Solution exists!
- What is the worst-case cost of checking strong n-consistency?
  \(O(2^n)\)

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
  - Group problems within each class by structural features
  - New measure of problem complexity

Linear Constraint Structures

\(x_1 \rightarrow x_2 \rightarrow x_3 \rightarrow x_4 \rightarrow x_5 \rightarrow x_6\)

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.
I.H. Arc consistent chains of length 1 are strongly consistent.
I.S. Extending an i step arc-consistent chain by 1 new arc consistent link produces an i+1 link arc-consistent chain.
Proof of I.S.: Since the last link is arc-consistent, any choice for variable i ensures a consistent choice for i+1. 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.

General Cutsets

Suppose removing a single node converts graph to tree. Spawn 3 separate subproblems: SA, SA, SA

Properties of Cutsets

- Suppose we find a cutset with m variables
- What is the cost of solving the CSP?
  \[ O(d^m \text{poly}(n - m)) \]
- How hard is it to find the smallest cutset?

NP hard, but fast (decent) poly time approximations and randomized algorithms exist and are under development.

Why Cutsets Are Important

- Cutsets give another structural measure of problem complexity
- Approximate min cutset algorithms give upper bound on run time to solve problem
- Cutsets do not depend upon the content of the constraints, only the structure
- Investment in finding a good cutset can be amortized over several problems with different constraints but same structure

Manipulating Graphs

- Cutsets reduce graphs into a number of smaller problems that we know how to solve in polynomial time.
- Is there a way to work directly with the graph structure?
- Idea: Force graph to be tree-like by eliminating and combining variables
Variable Elimination

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

Eliminate WA

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

Eliminate Q

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

Simplify

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

Finish

Can identify all settings of SA, V, NSW for which there is guaranteed to be a consistent settings of the remaining 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)
  X = merge(X, neighbors(X))
  Simplify X
  remove_from_Q(Q, neighbors(X))
  add_to_Q(Q, X)
  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 - 1.
  (AKA: Induced tree width.)
- 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

Structural Complexity

- Structural complexity is a somewhat different view of computational complexity: depends upon problem features, not problem class
- For many problems structural complexity is quite manageable
- Idea: Why not convert other NP-hard problems to CSPs and use structural complexity measures, CSP algorithms to solve?

\[ 2^{\text{poly}(n)} \gg 2^n \]

Cutsets vs. Elimination

- Cutsets:
  - Linear space
  - Potential wins if cutset values simplify subproblems (Can get speedups that from content of constraints)
- Variable elimination:
  - Space exponential in tree width
  -Insensitive to content of constraints
- Preference depends on taste, problem

Cheating Structural Complexity

- What if structural complexity is too high?
  - Merge(\(x\), neighbors(\(x\))) too big to construct domains
- Cheat: Define multiple variables over smaller domains (mini-buckets)
  - Instead of merging \(x\), neighbors(\(x\)) = \{x_1...x_n\}
  - Merge (e.g.) \{x_1...x_k\}, \{x_k...x_n\}
  - \(2 \cdot 2^{n/2} \ll 2^n\)

Mini-bucket Properties

- Is mini-buckets a sound method?
  - If mini-buckets finds a solution, is there guaranteed to be a solution to the CSP?
    - No.
  - If mini-buckets fails to find a solution, is there guaranteed not to be a solution the CSP?
    - Yes.
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
• We can use structural measures of complexity to figure out which ones are really hard