CPS 270
Advanced Search and CSPs
Ron Parr

Memory-bounded Search: Why?

• We run out of memory before we run out of time
• Problem: Need to remember entire search horizon
• Solution: Remember only a partial search horizon
• Issue: Maintaining optimality, completeness
• Issue: How to minimize time penalty

Searching with Partial Information

• Multiple state problems
  – Several possible initial states
• Contingency problems
  – Several possible outcomes for each action
• Exploration problems
  – Outcomes of actions not known \textit{a priori}, must be discovered by trying them

Example

• Initial state may not be detectable
  – Suppose sensors for a nuclear reactor fail
  – Need \textit{safe} shutdown sequence despite ignorance of some aspects of state
• This complicates search \textit{enormously}

• In the worst case, contingent solution could cover the entire state space
State Sets

- Idea:
  - Maintain a set of candidate states
  - Each search node represents a set of states
  - Can be hard to manage if state sets get large
- If states have probabilistic outcomes, we maintain a probability distribution over states

Searching in Unknown Environments

- What if we don’t know the consequences of actions before we try them?
- Often called on-line search
- Goal: Minimize competitive ratio
  - Actual distance/distance traveled if model known
  - Problematic if actions are irreversible
  - Problematic if links can have unbounded cost

Checking for Solution Existence

- In some problems, we don’t care about a path, but about a configuration that has a desired property
- Instead of a goal, we have a target, which can be a set of states that satisfy some property
- We call the set of properties that legal solutions must obey constraints
- We call these problems constraint satisfaction problems (CSPs)

CSP Examples

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

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

---

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?

---

CSP Example

Graph coloring:

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

---

Constraint Graph

- Enumerate all Legal combinations Of WA and SA (ignoring other regions)
CSPs as Search

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
  - (Can you formulate this using the terminology of search algorithms from the previous lecture?)

- Embellishments
  - Methods for picking next variable to assign (e.g. most constrained)
  - Backjumping

Digression: NP-Hardness

- NP hardness is not an AI topic
- You will not be tested on it explicitly, but
- It’s important for all computer scientists
- Understanding it will deepen your understanding of AI (and other CS) topics
- You will be expected to understand its relevance and use for AI problems
- Eat your vegetables; they’re good for you

P and NP

- P and NP are about decision problems
- P is set of problems that can be solved in polynomial time
- NP is a superset of P
- NP is the set of problems that:
  - Have solutions which can be verified in polynomial time or, equivalently,
  - can be solved by a non-deterministic Turing machine in polynomial time (OK if you don’t know what that means yet)
- Roughly speaking:
  - Problems in P are tractable – can be solved in a reasonable amount of time, and Moore’s law helps
  - Some problems in NP might not be tractable
NP-hardness

• Many problems in AI are NP-hard (or worse)
• What does this mean?
• These are some of the hardest problems in CS
• Identifying a problem as NP hard means:
  – You probably shouldn’t waste time trying to find a polynomial time solution
  – If you find a polynomial time solution, either
    • You have a bug
    • Find a place on your shelf for your Turing award
• NP hardness is a major triumph (and failure) for computer science theory

Understanding the class NP

• A class of decision problems (Yes/No)
• Solutions can be verified in polynomial time
• Examples:
  – Graph coloring:
  – Sortedness: [1 2 3 4 5 8 7]

What is NP hardness?

• An NP hard problem is at least as hard as the hardest problems in NP
• The hardest problems in NP are NP-complete
• Demonstrate hardness via reduction
  – Use one problem to solve another
  – A is reduced to B, if we can use B to solve A:

  \[
  \text{A instance} \xrightarrow{\text{Poly-time transformation}} \text{B Solver}
  \]
Hardness vs. Completeness

• For something to be NP-complete, must be in NP
• If something is NP-hard, it could be even harder than the hardest problems in NP
• Proving completeness is stronger theoretical result – says more about the problem

P=NP?

• Biggest open question in CS
• Can NP-complete problems be solved in polynomial time?
• Probably not, but nobody has been able to prove it yet
• Recent attempt at proof detailed in NY Times, one of many false starts: http://www.nytimes.com/2009/10/08/science/Wpolynom.html

How challenging is “P=NP?”

• Princeton University CS department
• See: http://www.cs.princeton.edu/general/bricks.php
• Photo from: http://stackinthedabble.blogspot.com/2009/07/three-interesting-points-on-princeton.html

NP-Completeness of CSPs

• Are CSPs in NP?
• Are they NP-hard?
• Graph coloring is known to be NP-complete
  • Can solve graph coloring using a CSP solver
  • Therefore CSPs are NP-hard
  • Since CSPs are in NP, CSPs are therefore NP-complete
Dealing with NP-hard problems

- If there is little hope of having a generally efficient solver, what do we do?

- Even if worst case requires exponential time, not all problems will
  - Identify classes of problems that might be easier than the general case
  - Develop algorithms that work well most of the time
  - Refine understanding of problems and algorithms so that we can avoid expensive computations more predictably and reliably
  - Lather, rinse, repeat

Optimization

- As with CSPs, solution is more important than path, but
- Some solutions are better than others
- Interested in minimizing or maximizing some function of the problem state
  - Find a protein with a desirable property
  - Optimize circuit layout

- History of search steps not worth the trouble

State Space Landscape

![State Space Landscape Diagram]

Goal: Find values of problem features that maximize objective function.

Note: This is conceptual. Often this function is not smooth.

Hill Climbing

- Idea: Try to climb up the state space landscape to find a setting of the problem features with high value.

- Approaches:
  - Steepest ascent
  - Stochastic – pick one of the good ones
  - First choice
- This is a greedy procedure
Limitations of Hill Climbing

- Local maxima
- Ridges – direction of ascent is at 45 degree angle to any of the local changes
- Plateaux – flat expanses

Getting Unstuck

- Random restarts
- Simulated annealing (maximization)
  - Take downhill moves with small probability
  - Probability of moving downhill decreases with
    - Number of iterations
    - Steepness of downhill move
  - If system is “cooled” slowly enough, will find global optimal w.p. 1
  - Motivated by the annealing of metals and glass

Genetic Algorithms

- GAs are hot in some circles
- Biological metaphors to motivate search
- Organism is a word from a finite alphabet (organisms = states)
- Fitness of organism measures its performance on task (fitness = objective)
- Uses multiple organisms (parallel search)
- Uses mutation (random steps)

Crossover

Crossover is a distinguishing feature of GAs:

Randomly select organisms for “reproduction” in accordance with their fitness. More “fit” individuals are more likely to reproduce.

Reproduction is sexual and involves crossover:

Organism 1: 110010010
Organism 2: 000101110
Offspring: 110011110
Is this a good idea?

- Has worked well in some examples
- Can be very brittle
  - Representations must be carefully engineered
  - Sensitive to mutation rate
  - Sensitive to details of crossover mechanism
- For the same amount of work, stochastic variants of hill climbing often do better
- Hard to analyze; needs more rigorous study

Continuous Spaces

- In continuous spaces, we don’t need to “probe” to find the values of local changes
- If we have a closed-form expression for our objective function, we can use the calculus
- Suppose objective function is: \( f(x_1, y_1, x_2, y_2, x_3, y_3) \)
- Gradient tells us direction and steepness of change
  \[ \nabla f = \left( \frac{\partial f}{\partial x_1}, \frac{\partial f}{\partial y_1}, \frac{\partial f}{\partial x_2}, \frac{\partial f}{\partial y_2}, \frac{\partial f}{\partial x_3}, \frac{\partial f}{\partial y_3} \right) \]

Following the Gradient

\[
\mathbf{x} = (x_1, y_1, x_2, y_2, x_3, y_3)
\]

\[
\mathbf{x} \leftarrow \mathbf{x} + \alpha \nabla f(\mathbf{x})
\]

For sufficiently small step sizes, this will converge to a local optimum.

Search Conclusions

- Search = most general purpose technique in existence
- Everything can be formulated as a search problem, from sorting to curing cancer
- Search techniques have been specialized to match different types of problems
- Be a smart consumer of search:
  - Specifying your problem clearly
  - Find the technique that matches your problem