CPS 270
Alternative Search Techniques
Ron Parr

Overview
- Memory-bounded Search
- Local Search
- Searching with Incomplete Information

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

Attempt 1: IDA*
- Iterative deepening A*
- Idea: Like IDDFS, but use the f cost as a cutoff
  - Cutoff all searches with \( f > 1 \), then \( f > 2 \), \( f > 3 \), etc.
  - Motivation: Cut off bad-looking branches early
- Problems:
  - Excessive node regeneration
  - Can still use a lot of memory

Attempt 2: RBFS
- Recursive best first search
- Objective: Linear space
- Idea: Remember best alternative
- Rewind, try alternatives if “best first” path gets too expensive
- Remember costs on the way back up

RBFS
Assume \( h=1 \), initially along this path.
Replace with \( alt = 11 \)
alt = 12
alt = 13
alt = 9
alt = 16
alt = 15
alt = 14
h=3
Return to best alternate.
**SMA**
- Idea: Use all of available memory
- Discard the worst leaf when memory starts to run out, to make room for new leaves
- Values get backed up to parents
- Optimal if solution fits in memory
- Complete
- Thrashing still possible

**Optimization**
- Solution is more important than path
- Interested in minimizing or maximizing some function of the problem state
  - Find TSP tour with minimum cost
  - Optimize circuit layout
  - Schedule tasks as tightly as possible
- History of visits not worth the trouble

**State Space Landscape**

**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**

**Getting Unstuck**
- Random restarts
- Simulated annealing
  - 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
    - settle into low energy configuration
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 involves crossover:

| Organism 1 | 1 1 0 0 1 | 0 0 1 0 |
| Organism 2 | 0 0 0 1 0 | 1 1 1 0 |
| Offspring  | 1 1 0 0 1 1 1 1 0 |

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

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

\[ x \leftarrow x + \alpha \nabla f (x) \]

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

If gradient is hard to compute:
- Compute empirical gradient
- Compare with classical hill climbing

Accelerating Gradient Ascent

- Many methods for choosing step size
- Newton Raphson method for finding roots:
  \[ x \leftarrow x - g(x) / g'(x) \]
- Application to gradient ascent:
  \[ x \leftarrow x - \nabla f(x) H_f^{-1} (x) \]
What's a Hessian?

\[
H_f = \begin{pmatrix}
\frac{\partial^2 f}{\partial x_1 \partial x_1} & \cdots & \frac{\partial^2 f}{\partial x_1 \partial x_n} \\
\vdots & \ddots & \vdots \\
\frac{\partial^2 f}{\partial x_n \partial x_1} & \cdots & \frac{\partial^2 f}{\partial x_n \partial x_n}
\end{pmatrix}
\]

Constrained Optimization

• Don't forget about the easier cases
  – If the objective function is linear, things are easier
  – If linear constraints, solve as a linear program:
    – Maximize: \( f(x) \)
    – Subject to: \( Ax \leq b \)
  – Can be done in polynomial time
  – Can solve some quadratic programs in poly time

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

• In some situations, 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

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
Conclusions

• There are search algorithms for almost every situation

• Many problems can be formulated as search

• While search is a very general method, it can sometimes outperform special-purpose methods