Searching with Partial Information
(not a focus of this class, but good to be aware of)

- 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 *safe* shutdown sequence despite ignorance of some aspects of state

- This complicates search *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
(not a focus of this class, but good to be aware of)

- 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

Optimization
(Not directly a topic of this class, but used later)

- Want to find the “best” state
- 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
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 optimum w.p. 1
  - Motivated by the annealing of metals and glass
Genetic Algorithms

- GAs run hot and cold
- 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
  \[
  \nabla f = \left( \frac{\partial f}{\partial x_1}, \frac{\partial f}{\partial y_1}, \frac{\partial f}{\partial x_2}, \frac{\partial f}{\partial y_2}, \frac{\partial f}{\partial x_3}, \frac{\partial f}{\partial y_3} \right)
  \]
Gradient Descent in Continuous Space

- Minimize $y = f(x)$
- Move in opposite direction of derivative $\frac{df}{dx}(x)$

![Gradient Descent Diagram](image)
Gradient Descent in Continuous Space

• Minimize $y = f(x)$
• Move in opposite direction of derivative $\frac{df}{dx}(x)$
Gradient Descent in Continuous Space

- Minimize $y = f(x)$
- Move in opposite direction of derivative $df/dx(x)$
**Gradient**: analogue of derivative in multivariate functions $f(x_1,\ldots,x_n)$

Direction that you would move $x_1,\ldots,x_n$ to make the steepest increase in $f$

**Algorithm for Gradient Descent**

- **Input**: continuous *objective function* $f$, initial point $x^0=(x_1^0,\ldots,x_n^0)$
- For $t=0,\ldots,N-1$:
  - Compute gradient vector $g^t=(\partial f/\partial x_1(x^t),\ldots,\partial f/\partial x_n(x^t))$
  - If the length of $g^t$ is small enough [convergence]
    - Return $x^t$
  - Pick a *step size* $\alpha^t$
  - Let $x^{t+1}=x^t - \alpha^t g^t$

“Industrial strength” optimization software uses more sophisticated techniques to use higher derivatives, handle constraints, deal with particular function classes, etc.
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