Outline

• Advanced Search & Hill-climbing
  – More difficult problems, basics, local optima, variations

• Simulated Annealing
  – Basic algorithm, temperature, tradeoffs

• Genetic Algorithms
  – Basics of evolution, fitness, natural selection
Search vs. Optimization

Before: wanted a **path** from start state to goal state
- Uninformed search, informed search

**New setting:** optimization
- States $s$ have values $f(s)$
- Want: $s$ with optimal value $f(s)$ (i.e., **optimize** over states)
- Challenging setting: **too many states** for previous search approaches, but maybe not a differentiable function for SGD.
Examples: $n$ Queens

A classic puzzle:
- Place 8 queens on a 8 x 8 chessboard so that no two have same row, column, or diagonal.
- Can generalize to $n \times n$ chessboard.

- What are states $s$? Values $f(s)$?
  - State: configuration of the board
  - $f(s)$: # of conflicting queens
Examples: TSP

Famous graph theory problem.

- Get a graph $G = (V,E)$. **Goal**: a path that visits each node exactly once and returns to the initial node (a **tour**).
  - **State**: a particular tour (i.e., ordered list of nodes)
  - $f(s)$: total weight of the tour (e.g., total miles traveled)
Examples: Satisfiability

Boolean satisfiability (e.g., 3-SAT)

• Recall our logic lecture. Conjunctive normal form

\[(A \lor \neg B \lor C) \land (\neg A \lor C \lor D) \land (B \lor D \lor \neg E) \land (\neg C \lor \neg D \lor \neg E) \land (\neg A \lor \neg C \lor E)\]

  – Goal: find if satisfactory assignment exists.
  – State: assignment to variables
  – \(f(s)\): # satisfied clauses
Hill Climbing

One approach to such optimization problems.

• Basic idea: move to a neighbor with a better $f(s)$

• Q: how do we define neighbor?
  – Not as obvious as our successors in search
  – Problem-specific
  – As we’ll see, needs a careful choice
Defining Neighbors: n Queens

In n Queens, a simple possibility:

- Look at the **most-conflicting column** (ties? right-most one)
- Move queen in that column vertically to a different location

\[ f(s) = 1 \]

**Neighborhood of** \( s \)
Defining Neighbors: TSP

For TSP, can do something similar:

- Define neighbors by small changes
- Example: 2-change: A-E and B-F

A-B-C-D-E-F-G-H-A

flip

A-E-D-C-B-F-G-H-A
Defining Neighbors: SAT

For Boolean satisfiability,
• Define neighbors by flipping one assignment of one variable

Starting state: TFTTT

\[(A=F, B=F, C=T, D=T, E=T)\]
\[(A=T, B=T, C=T, D=T, E=T)\]
\[(A=T, B=F, C=F, D=T, E=T)\]
\[(A=T, B=F, C=T, D=F, E=T)\]
\[(A=T, B=F, C=T, D=T, E=F)\]

\[A \lor \neg B \lor C\]
\[\neg A \lor C \lor D\]
\[B \lor D \lor \neg E\]
\[\neg C \lor \neg D \lor \neg E\]
\[\neg A \lor \neg C \lor E\]
Hill Climbing Neighbors

Q: What’s a neighbor?

• **Vague definition.** For a given problem structure, neighbors are states that can be produced by a small change.

• **Tradeoff!**
  – Too small? Will get struck.
  – Too big? Not very efficient

• Q: how to pick a neighbor? Greedy

• Q: terminate? When no neighbor has bigger value
Hill Climbing Algorithm

Pseudocode:

1. Pick initial state $s$
2. Pick $t$ in $\text{neighbors}(s)$ with the largest $f(t)$
3. if $f(t) \leq f(s)$ THEN stop, return $s$
4. $s \leftarrow t$. goto 2.

What could happen? Local optima!
Hill Climbing: Local Optima

Q: Why is it called hill climbing?

L: What’s actually going on.

R: What we get to see.

Global optimum, where we want to be

f

state

f

fog

state
Hill Climbing: Local Optima

Note the local optima. How do we handle them?

Done?

Where do I go?
Escaping Local Optima

**Simple idea 1: random restarts**
- Stuck: pick a random new starting point, re-run.
- Do $k$ times, return best of the $k$.

**Simple idea 2: reduce greed**
- “Stochastic” hill climbing: randomly select between neighbors
- Probability proportional to the value of neighbors
Hill Climbing: Variations

Q: neighborhood too large?
• Generate random neighbors, one at a time. Take the better one.

Q: relax requirement to always go up?
• Often useful for harder problems
• 3SAT algorithm: Walk-SAT
Break & Quiz

Q 1.1: Hill climbing and SGD are related by

(i) Both head towards optima
(ii) Both require computing a gradient
(iii) Both will find the global optimum for a convex problem

• A. (i)
• B. (i), (ii)
• C. (i), (iii)
• D. All of the above
Break & Quiz

Q 1.1: Hill climbing and SGD are related by
(i) Both head towards optima
(ii) Both require computing a gradient
(iii) Both will find the global optimum for a convex problem

• A. (i)
• B. (i), (ii)
• C. (i), (iii)
• D. All of the above
Q 1.1: Hill climbing and SGD are related by
(i) Both head towards optima
(ii) Both require computing a gradient
(iii) Both will find the global optimum for a convex problem

• A. (i) (No: (iii) also true since convexity->local optima are global)
• B. (i), (ii) (No: (ii) is false. Hill-climbing looks at neighbors only.)
• C. (i), (iii)
• D. All of the above (No: (ii) false, as above.)
Simulated Annealing

A more sophisticated optimization approach.

- **Idea**: allow some downhill moves at first, then be pickier over time

- **Pseudocode**:
  
  Pick initial state $s$; $T=1$
  For $k = 0$ through $K$:
  
  $T \leftarrow T \times 0.99$ (*cool down*)
  Pick a random neighbour $t \leftarrow \text{neighbor}(s)$
  If $f(s) \leq f(t)$, then $s \leftarrow t$
  Else with prob. $P(f(s), f(t), T)$ still do $s \leftarrow t$

  **Output**: the best state ever seen
Simulated Annealing: Picking Probability

How do we pick probability $P(f(s), f(t), \text{Temp})$?

- Decrease with temperature
- Decrease with gap $f(s) - f(t)$: $\exp\left(-\frac{|f(s) - f(t)|}{\text{Temp}}\right)$

- Temperature cools over time.
  - So: high temperature, accept any $t$
  - But, low temperature, behaves like hill-climbing
  - Still, $f(s) - f(t)$ plays a role: if big, replacement probability low.
Simulated Annealing: Picking Parameters

• Have to balance the various parts, e.g., cooling schedule.
  – Too fast: becomes hill climbing, stuck in local optima
  – Too slow: takes too long.

• Combines with variations (e.g., with random restarts)
  – Probably should try hill-climbing first though.

• Inspired by cooling of metals
  – We’ll see one more alg. inspired by nature
Q 2.1: Which of the following is likely to give the best cooling schedule for simulated annealing?

A. Temp_{t+1} = Temp_t * 1.25
B. Temp_{t+1} = Temp_t
C. Temp_{t+1} = Temp_t * 0.8
D. Temp_{t+1} = Temp_t * 0.0001
Q 2.1: Which of the following is likely to give the best cooling schedule for simulated annealing?

A. $\text{Temp}_{t+1} = \text{Temp}_t \times 1.25$
B. $\text{Temp}_{t+1} = \text{Temp}_t$
C. $\text{Temp}_{t+1} = \text{Temp}_t \times 0.8$
D. $\text{Temp}_{t+1} = \text{Temp}_t \times 0.0001$
Q 2.1: Which of the following is likely to give the best cooling schedule for simulated annealing?

A. $\text{Temp}_{t+1} = \text{Temp}_t \times 1.25$ (No, temperature is increasing)
B. $\text{Temp}_{t+1} = \text{Temp}_t$ (No, temperature is constant)
C. $\text{Temp}_{t+1} = \text{Temp}_t \times 0.8$
D. $\text{Temp}_{t+1} = \text{Temp}_t \times 0.0001$ (Cools too fast---basically hill climbing)
Q 2.2: Which of the following would be better to solve with simulated annealing than A* search?

i. Finding the smallest set of vertices in a graph that involve all edges
ii. Finding the fastest way to schedule jobs with varying runtimes on machines with varying processing power
iii. Finding the fastest way through a maze

• A. (i)
• B. (ii)
• C. (i) and (ii)
• D. (ii) and (iii)
Q 2.2: Which of the following would be better to solve with simulated annealing than A* search?

i. Finding the smallest set of vertices in a graph that involve all edges

ii. Finding the fastest way to schedule jobs with varying runtimes on machines with varying processing power

iii. Finding the fastest way through a maze

• A. (i)
• B. (ii)
• C. (i) and (ii)
• D. (ii) and (iii)
Q 2.2: Which of the following would be better to solve with simulated annealing than A* search?

i. Finding the smallest set of vertices in a complete graph (i.e., all nodes connected)
ii. Finding the fastest way to schedule jobs with varying runtimes on machines with varying processing power
iii. Finding the fastest way through a maze

• A. (i) (No, (ii) better: huge number of states, don’t care about path)
• B. (ii) (No, (i) complete graph might have too many edges for A*)
• C. (i) and (ii)
• D. (ii) and (iii) (No, (iii) is good for A*: few successors, want path)
Genetic Algorithms

Another optimization approach based on nature

• Survival of the fittest!
Evolution Review

Encode genetic information in DNA (four bases)

• A/C/T/G: nucleobases acting as symbols

• Two types of changes
  – Crossover: exchange between parents’ codes
  – Mutation: rarer random process
    • Happens at individual level
Natural Selection

Competition for resources

• Organisms better fit → better probability of reproducing
• Repeated process: fit become larger proportion of population

Goal: use these principles for optimization

– New terminology: state s ‘individual’
– Value $f(s)$ is now the ‘fitness’
Genetic Algorithms Setup I

Keep around a fixed number of states/individuals

- A bit like beam search
- Call this the population

For our n Queens game example, an individual:

\[(3 \ 2 \ 7 \ 5 \ 2 \ 4 \ 1 \ 1)\]
Genetic Algorithms Setup II

Goal of genetic algorithms: optimize using principles inspired by mechanism for evolution

- E.g., analogous to **natural selection, cross-over, and mutation**
Genetic Algorithms Pseudocode

Just one variant:

1. Let $s_1, \ldots, s_N$ be the current population
2. Let $p_i = \frac{f(s_i)}{\sum_j f(s_j)}$ be the reproduction probability
3. for $k = 1; k < N; k += 2$
   - parent1 = sample with replacement according to $p$
   - parent2 = sample with replacement according to $p$
   - randomly select a crossover point, swap strings of parents 1, 2 to generate children $t[k], t[k+1]$
4. for $k = 1; k <= N; k++$
   - Randomly mutate each position in $t[k]$ with a small probability (mutation rate)
5. The new generation replaces the old: $\{ s \} \leftarrow \{ t \}$. Repeat
Reproduction: Proportional Selection

Reproduction probability: \( p_i = \frac{f(s_i)}{\sum_j f(s_j)} \)

- **Example**: \( \sum_j f(s_j) = 5 + 20 + 11 + 8 + 6 = 50 \)
- \( p_1 = 5/50 = 10\% \)

<table>
<thead>
<tr>
<th>Individual</th>
<th>Fitness</th>
<th>Prob.</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>5</td>
<td>10%</td>
</tr>
<tr>
<td>B</td>
<td>20</td>
<td>40%</td>
</tr>
<tr>
<td>C</td>
<td>11</td>
<td>22%</td>
</tr>
<tr>
<td>D</td>
<td>8</td>
<td>16%</td>
</tr>
<tr>
<td>E</td>
<td>6</td>
<td>12%</td>
</tr>
</tbody>
</table>
Example: Scheduling Courses

Let’s run through an example:

- **5 courses:** A, B, C, D, E
- **3 time slots:** Mon/Wed, Tue/Thu, Fri/Sat
- Students wish to enroll in three courses
- Goal: maximize student enrollment

<table>
<thead>
<tr>
<th>Courses</th>
<th>Students</th>
</tr>
</thead>
<tbody>
<tr>
<td>A B C</td>
<td>2</td>
</tr>
<tr>
<td>A B D</td>
<td>7</td>
</tr>
<tr>
<td>A D E</td>
<td>3</td>
</tr>
<tr>
<td>B C D</td>
<td>4</td>
</tr>
<tr>
<td>B D E</td>
<td>10</td>
</tr>
<tr>
<td>C D E</td>
<td>5</td>
</tr>
</tbody>
</table>
Example: Scheduling Courses

Let’s run through an example:

• State: course assignment to time slot

<table>
<thead>
<tr>
<th>M</th>
<th>M</th>
<th>F</th>
<th>T</th>
<th>M</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>B</td>
<td>C</td>
<td>D</td>
<td>E</td>
</tr>
</tbody>
</table>

= MMFTM

• Here:
  – Courses A, B, E scheduled Mon/Wed
  – Course D scheduled Tue/Thu
  – Course C scheduled Fri/Sat

<table>
<thead>
<tr>
<th>Courses</th>
<th>Students</th>
</tr>
</thead>
<tbody>
<tr>
<td>A B C</td>
<td>2</td>
</tr>
<tr>
<td>A B D</td>
<td>7</td>
</tr>
<tr>
<td>A D E</td>
<td>3</td>
</tr>
<tr>
<td>B C D</td>
<td>4</td>
</tr>
<tr>
<td>B D E</td>
<td>10</td>
</tr>
<tr>
<td>C D E</td>
<td>5</td>
</tr>
</tbody>
</table>
Example: Scheduling Courses

Value of a state? Say $MMFTM$

<table>
<thead>
<tr>
<th>Courses</th>
<th>Students</th>
<th>Can enroll?</th>
</tr>
</thead>
<tbody>
<tr>
<td>A B C</td>
<td>2</td>
<td>No</td>
</tr>
<tr>
<td>A B D</td>
<td>7</td>
<td>No</td>
</tr>
<tr>
<td>A D E</td>
<td>3</td>
<td>No</td>
</tr>
<tr>
<td>B C D</td>
<td>4</td>
<td>Yes</td>
</tr>
<tr>
<td>B D E</td>
<td>10</td>
<td>No</td>
</tr>
<tr>
<td>C D E</td>
<td>5</td>
<td>Yes</td>
</tr>
</tbody>
</table>

- Here $4+5=9$ students can enroll in desired courses
Example: Scheduling Courses

First step:

• Randomly initialize and evaluate states

<table>
<thead>
<tr>
<th>Courses</th>
<th>Students</th>
</tr>
</thead>
<tbody>
<tr>
<td>A B C</td>
<td>2</td>
</tr>
<tr>
<td>A B D</td>
<td>7</td>
</tr>
<tr>
<td>A D E</td>
<td>3</td>
</tr>
<tr>
<td>B C D</td>
<td>4</td>
</tr>
<tr>
<td>B D E</td>
<td>10</td>
</tr>
<tr>
<td>C D E</td>
<td>5</td>
</tr>
</tbody>
</table>

• Calculate reproduction probabilities

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>MMFTM</td>
<td>TTFMM</td>
</tr>
<tr>
<td>9</td>
<td>4</td>
</tr>
<tr>
<td>MMFTM</td>
<td>TTFMM</td>
</tr>
<tr>
<td>26%</td>
<td>11%</td>
</tr>
<tr>
<td>FMTTF</td>
<td>MTTTF</td>
</tr>
<tr>
<td>19</td>
<td>3</td>
</tr>
<tr>
<td>FMTTF</td>
<td>MTTTF</td>
</tr>
<tr>
<td>54%</td>
<td>9%</td>
</tr>
</tbody>
</table>
Example: Scheduling Courses

Next steps:

• Select parents using reproduction probabilities
• Perform crossover
• Randomly mutate new children
Example: Scheduling Courses

Continue:

• Now, get our function values for updated population
• Calculate reproduction probabilities

<table>
<thead>
<tr>
<th>Courses</th>
<th>Students</th>
</tr>
</thead>
<tbody>
<tr>
<td>A B C</td>
<td>2</td>
</tr>
<tr>
<td>A B D</td>
<td>7</td>
</tr>
<tr>
<td>A D E</td>
<td>3</td>
</tr>
<tr>
<td>B C D</td>
<td>4</td>
</tr>
<tr>
<td>B D E</td>
<td>10</td>
</tr>
<tr>
<td>C D E</td>
<td>5</td>
</tr>
</tbody>
</table>

\[
\text{FMFTTT} = 11 \quad \text{FMFTTT} = 39\%
\]
\[
\text{MMTTTF} = 13 \quad \text{MMTTTF} = 46\%
\]
\[
\text{MMTFFF} = 4 \quad \text{MMTFFF} = 14\%
\]
\[
\text{FTTTTF} = 0 \quad \text{FTTTTF} = 0\%
\]
Variations & Concerns

Many possibilities:

- Parents survive to next generation
- Use ranking instead of exact value of $f(s)$ for reproduction probabilities (reduce influence of extreme $f$ values)

Some challenges

- State encoding
- Lack of diversity: converge too soon
- Must pick a lot of parameters
Summary

• Challenging optimization problems
  – First, try hill climbing. Simplest solution

• Simulated annealing
  – More sophisticated approach; helps with local optima

• Genetic algorithms
  – Biology-inspired optimization routine