Announcements
PA1 grades should be posted on/by Wednesday night
- Expect PA2 to be posted Thursday
- Expect two weeks to do it

I will post a new reading assignment on Thursday, too

Other questions?

Where are we?
Talking about local search
- State space is space of possible solutions'
- Some may be invalid
- Per iteration:
  - Evaluate neighbors
  - Select one for new state
  - Until goal state or lack of resources
- Finite memory constraint
  - Classic: no memory
  - More common: finite memory
    - Best-so-far
    - Tabu table

Local Search Strategies
Random Restart Search
Parallel Local Search
Beam Local Search
Tabu Search
Dynamic Local Search
- MaxClique Reading Assignment

Searching in Parallel
What is the difference between parallel & beam search?
- Parallel runs N independent searches in parallel
- Beam search pools the neighbors, picks N new states

Both techniques treat states as atomic
- No information passes from one state to another

Most states are complex combinations of solutions to subproblems
- Example: Traveling Salesman
- Every path fragment is a solution to a smaller TSP problem

How does nature do it?
Goal: evolve “best” life form for a given environment
Every individual is a possible solution
- Represented as a sequence of genes
- Fitness to environment determines probability of survival
Sexual reproduction
- Mixes characteristics of two fit solutions
Random mutations
- Introduces new genes
Can we model search on nature?
**Evolutionary Algorithms**

**Analogies**
- Population genetics
- Simulated evolution

**Population based**
- Every member is a feasible solution
- Population as a whole models many points in search space
- Highly parallel

**Decomposable states**
- Feasible solutions not just "points"
- Feasible solutions have good aspects and bad aspects
- Feasible solutions can be combined to create new solutions

---

**Evolutionary Algorithm 0.1**

1. Create initial population
2. While time remaining
   - a. Repeat many times to create children
      i. Randomly select two elements from previous population
      ii. Create "child" element by genetic crossover
      iii. With small probability, mutate child
      iv. add child to pool
   - b. Probabilistically select |P| children from pool based on fitness
   - c. Replace previous population with new population
3. Select best element from final population

>Note: there are a lot of variants on this theme.

---

**Computational Advantages**

- No gradient information needed
  - Fitness function evaluates quality
  - Fitness of neighbors never compared

- Global (not local) search
  - No hill climbing, no local minima

- Potential for massive parallelism

- Easily hybridized
  - E.g. by having population elements do a local search

---

**Visualizing Search Spaces**

Assume \((x,y)\) are features
Assume height is quality

Local search:
- State is a point \((x,y)\)
- Neighborhood is a grid (usually)

Parallel local search:
- State is a set of points \((x,y,...)\)
- Neighborhood still fixed

Evolutionary algorithms:
- State as a set of elements
- Elements are decomposable
- No fixed neighborhood structure

---

**Visualizing Search Spaces (II)**

---

**Example: Touring Salesman**

**Task:**
Find a path that:
- (1) Starts and ends at the same city
- (2) Contains every other city exactly once
- (3) Has minimum total length
Evolutionary Algorithm for TSP

Choose a representation
- Number cities from 1 to N
- Any permutation of the numbers 1 to N is a valid tour
- Assuming wrap-around from last city back to first

Fitness function
\[ f(x) = (x_1 - x_2)^2 + \sum_{i=1}^{N-1} (x_{i+1} - x_i)^2 \]

Initial Population
- \(|P|\) random permutations of the vector \([1, \ldots, N]\)

Evolutionary TSP (cont.)

Crossover Operator
- Takes two "parent" elements, produces a "child"
- Example: ordered crossover
  - Select random interval of first parent, copy it in place to child
  - Insert missing elements into child, in order they appear in second parent
- Parent 1: \([1,2,3,4,5,6,7,8,9]\)
- Parent 2: \([9,8,7,6,5,4,3,2,1]\)
- Child: \([5,4,3,2,6,7,8,9]\)

Mutation operator
- Must produce valid element, given an element
- Example: swap mutation
  - Randomly swap two elements in the tour
- Input: \([1,3,4,2,5]\)
  - Output: \([4,2,3,5,1]\)

Example

Let’s consider just five cities (lat, long):
- 1: Cherokee (42.7, 95.5)
- 2: Davenport (41.5, 90.6)
- 3: Des Moines (41.6, 93.6)
- 4: Dubuque (42.5, 90.7)
- 5: Sioux City (42.5, 96.4)

Pairwise distances precomputed in a table:

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>5.0</td>
<td>2.2</td>
<td>4.8</td>
<td>0.9</td>
</tr>
<tr>
<td>2</td>
<td>5.0</td>
<td>0</td>
<td>2.1</td>
<td>1.0</td>
<td>5.9</td>
</tr>
<tr>
<td>3</td>
<td>2.2</td>
<td>2.1</td>
<td>0</td>
<td>3.0</td>
<td>2.9</td>
</tr>
<tr>
<td>4</td>
<td>4.8</td>
<td>1.0</td>
<td>3.0</td>
<td>0</td>
<td>5.7</td>
</tr>
<tr>
<td>5</td>
<td>0.9</td>
<td>5.9</td>
<td>2.9</td>
<td>5.7</td>
<td>0</td>
</tr>
</tbody>
</table>

Example (II)

Initial Population
- Chosen randomly, size 3 (fitness in red)
  - \([1,3,4,2,5]\) 12.8
  - \([4,3,5,1,2]\) 11.9

Generate a child
- \([1,3,4,2,5]\) + \([4,3,5,1,2]\) = \([2,3,4,5,1]\) (16.7)

Mutate child (only do this sometimes)
- \([2,3,4,5,1]\) = \([4,2,3,5,1]\) (11.8)

Add child to pool
- \([2,3,4,5,1]\) (20.5)

Example (III)

Generate another child
- \([4,3,5,1,2]\) + \([1,3,4,2,5]\) = \([3,4,5,1,2]\) (15.6)
- Don’t mutate this one (probabilistic)
  - Add to pool \([1,3,4,5,3]\) (20.5), \([3,4,5,1,2]\) (15.6)

And another child
- \([2,4,5,1,3]\) + \([4,3,5,1,2]\) = \([2,4,3,5,1]\) (12.8)
  - Mutate \([2,4,3,5,1]\) = \([4,2,3,5,1]\) (11.8)

Select new population, with preference by fitness
- \([4,2,3,5,1]\) (11.8), \([2,1,4,5,3]\) (20.5)
Simplest Evolutionary Strategy

(1+1)-ES

- 1 “parent” solution plus
- 1 “offspring” solution
- Keep the best

(1+1)-ES Algorithm

1. Create initial solution $x$
2. While termination criterion is not met do
   1. For $i = 1$ to $n$ do
      - $x'_i = x_i + \sigma N(0,1)$
      - If ($f(x') \geq f(x)$) then
        - $x := x'$
      - Update $\sigma$
   3. Update $N(0,1)$ indicates a normal distribution with 0 mean and 1 standard deviation

(1+1)-ES Update

How to set/update $\sigma$?

Convergence proofs on two simple problems define a 1/5 rule:

$$\frac{\text{# of "better" steps}}{\text{all or last 1 steps}} > \frac{1}{5}$$

If ratio is $> 1/5$, increase $\sigma$, else decrease

Or keep it fixed (better for escaping local optima)

(\mu+\lambda)-ES

$\mu$ solutions in population
Generate $\lambda$ new solutions
Choose the best $\mu$ from original + new at each iteration
Can add recombination before mutation
Can have different $\sigma$ per variable

Evolutionary Strategies (cont.)

Self-adaptive mutation & rotation
- Log-normal distribution for mutation
- Adaptive through strategy ($\sigma$) and rotation angle ($\theta$) parameters added through chromosome: $\{x_1, x_2, \sigma_1, \sigma_2, \theta\}$

Simple Mutations

Correlated Mutation via Rotation

Figures courtesy of D. Whitley