GENETIC ALGORITHMS
Hill-climbing — depth-first with a heuristic

```plaintext
function hillclimbing(node)
    node = selectrandomnode;
    loop
        if (node is goal) then return(node);
        successors = generatesuccessors(node);
        bestsuccessor = selectbestnode(successors);
        if (value(bestsuccessor) < value(node))
            then return(node);
        node = bestsuccessor;
    end loop
```
Properties of Hill-climbing

- *Hillclimbing* algorithms (also called *gradient descent* if the evaluation function represents cost) only consider the current state. All previous states are forgotten.

- Hillclimbing resembles depth-first search because it tends to steadfastly move up/down one path.

- "Pure" hillclimbing does not support backtracking so each move is irrevocable.

- The initial state is chosen at random! Ties are also broken randomly. *The algorithm may return different solutions depending on the start state!*
Some of the problems....

- **Local Maxima**: Local peaks that aren't as high as the globally maximum peak.

- **Plateaus**: Regions of the state space that are flat (all states have virtually the same evaluation measure).

- **Ridges (1)**: Impossible to get to the higher slope in a single move; must move down first in order to move up later.

- **Ridges (2)**: Areas with steep slopes at the bottom but gentle slopes near the top. Can wander around aimlessly on the top.
Genetic algorithms -- Why

- I’m playing chess and we seem to be moving back and forth to the same squares.
- I’m in a forest and can’t see the end – and there aren’t any landmarks that I can identify…
- My search seems to be going over the same ground – over and over again.
Genetic algorithms

- They are good when your search technique seems to be going nowhere!
- They are good when your goal seems to have changed
Genetic Algorithms

- Genetic Algorithms are a type of machine learning for representing and solving complex problems.
- They provide a set of efficient, domain-independent search heuristics for a broad spectrum of applications.
- A genetic algorithm interprets information that enables it to reject inferior solutions and accumulate good ones, and thus it learns about its universe.
Genetic Algorithms

- An algorithm is a set of instructions that is repeated to solve a problem.

- A genetic algorithm conceptually follows steps inspired by the biological processes of evolution.

- Genetic Algorithms follow the idea of SURVIVAL OF THE FITTEST- Better and better solutions evolve from previous generations until a near optimal solution is obtained.
Genetic Algorithms

- Also known as evolutionary algorithms, genetic algorithms demonstrate self-organization and adaptation similar to the way that the fittest biological organism survive and reproduce.

- A genetic algorithm is an iterative procedure that represents its candidate solutions as strings of genes called chromosomes.
Genetic Algorithms

- Genetic Algorithms are often used to improve the performance of other AI methods such as expert systems or neural networks.
- The method learns by producing offspring that are better and better as measured by a fitness function, which is a measure of the objective to be obtained (maximum or minimum).
Simulated Evolution

- We need the following
  - Representation of an individual
  - Fitness Function
  - Reproduction Method
  - Selection Criteria
Representing an Individual – A data structure

- An individual is a data structure representing the “genetic structure” of a possible solution.
- Genetic structure consists of an alphabet
  - Often \{0,1\}
  - Can be whatever works
Binary Encoding

- Most Common — string of bits, 0 or 1.
  Chrom: A = 1 0 1 1 0 0 1 0 1 1
  Chrom: B = 1 1 1 1 1 1 0 0 0 0

- Gives you many possibilities

- Example Problem: Knapsack problem

- The problem: there are things with given value and size. The knapsack has given capacity. Select things to maximize the total value that fits in knapsack

- Encoding: Each bit says, if the corresponding thing is in the knapsack
Permutation Encoding

- Used in “ordering problems”
- Every chromosome is a string of numbers, which represents a sequence.
  Chrom A: 1 5 3 2 6 4 7 9 8
  Chrom B: 8 5 7 6 2 3 1 4 9
- Example: Traveling Salesperson problem
- The problem: cities that must be visited.
- Encoding says order of cities in which salesperson will visit.
Another Example

- To find optimal quantity of three major ingredients (sugar, wine, sesame oil) denoting ounces.
  - Use an alphabet of 1-9 denoting ounces.
  - Solutions might be 1-1-1, 2-1-4, 3-3-1.
How offspring are produced – Reproduction – Changing the states

- *Reproduction*- Through reproduction, genetic algorithms produce new generations of improved solutions by selecting parents with higher fitness ratings or by giving such parents a greater probability of being contributors (possibly multiple times) and by using random selection.
How offspring are produced

- **Crossover**- Many genetic algorithms use strings of binary symbols for chromosomes, as in our Knapsack example, to represent solutions. Crossover means choosing a random position in the string (say, after 2 digits) and exchanging the segments either to the right or to the left of this point with another string partitioned similarly to produce two new offspring.
Crossover Example

- Parent A 011011
- Parent B 101100
- “Mate the parents by splitting each number as shown between the second and third digits (position is randomly selected)
- 01*1011 10*1100
Crossover Example

- Now combine the first digits of A with the last digits of B, and the first digits of B with the last digits of A
- This gives you two new offspring
  - 011100
  - 101011
- If these new solutions, or offspring, are better solutions than the parent solutions, the system will be more likely to keep these as more optimal solutions and they will become parents. This is repeated until some condition (for example, number of populations or improvement of the best solution) is satisfied.
Crossover Operators

- Single point crossover:
  - Parent A: 1 0 0 1 0 | 1 1 1 0 1
  - Parent B: 0 1 0 1 1 | 1 0 1 1 0

  - Child AB: 1 0 0 1 0 1 0 1 1 0
  - Child BA: 0 1 0 1 1 1 1 0 1

- Two point crossover:
  - Parent A: 1 0 0 1 | 0 1 1 | 1 0 1
  - Parent B: 0 1 0 1 | 1 1 0 | 1 1 0

  - Child AB: 1 0 0 1 1 1 0 1 0 1
  - Child BA: 0 1 0 1 0 1 1 1 1 0
How offspring are produced cont.

- *Mutation*- *Mutation* is an arbitrary change in a situation. Sometimes it is used to prevent the algorithm from getting stuck. The procedure changes a 1 to a 0 or 0 to a 1 instead of duplicating them. This change occurs with a very low probability (say 1 in 1000)
Intermediate Population

- One generation: current population $\rightarrow$ intermediate population $\rightarrow$ next population
- Selection applied to current population to generate intermediate population
- Crossover and mutation applied to intermediate population to generate next population
Genetic Algorithm Operators
Mutation and Crossover
Uniform Crossover and Mutation

- Uniform crossover:
- Parent A: 1 0 0 1 0 1 1 1 0 1
- Parent B: 0 1 0 1 1 1 0 1 1 0

- Child AB: 1 1 0 1 1 1 1 1 0 1
- Child BA: 0 0 0 1 0 1 0 1 1 0

- Mutation: randomly toggle one bit
- Individual A: 1 0 0 1 0 1 1 1 0 1
- Individual A': 1 0 0 0 0 1 1 1 0 1
Examples

- **Mutation:**

  The recipe example:
  1-2-3 may be changed to 1-3-3 or 3-2-3, giving two new offspring. How often? How many digits change? How big? (parameters to adjust)
More examples:

- Crossover
  
  **Recipe:**
  
  Parents 1-3-3 & 3-2-3. Crossover point after the first digit. Generate two offspring: 3-3-3 and 1-2-3.
  
  Can have one or two point crossover.
Crossover – Permutation Encoding

**Single point crossover** - one crossover point is selected, till this point the permutation is copied from the first parent, then the second parent is scanned and if the number is not yet in the offspring it is added

\[(1\ 2\ 3\ 4\ 5\ 6\ 7\ 8\ 9) + (4\ 5\ 3\ 6\ 8\ 9\ 7\ 2\ 1) = (1\ 2\ 3\ 4\ 5\ 6\ 8\ 9\ 7)\]

**Mutation**

**Order changing** - two numbers are selected and exchanged

\[(1\ 2\ 3\ 4\ 5\ 6\ 8\ 9\ 7) \Rightarrow (1\ 8\ 3\ 4\ 5\ 6\ 2\ 9\ 7)\]
Selection Criteria

- Fitness proportionate selection, rank selection methods.
  - Fitness proportionate – each individual, $I$, has the probability $\frac{\text{fitness}(I)}{\sum_{all\_individual\_j} \text{Fitness}(j)}$, where $\text{Fitness}(I)$ is the fitness function value for individual $I$.
  - Rank selection – sorts individual by fitness and the probability that an individual will be selected is proportional to its rank in this sorted list.
Fitness Function

- Represents a rank of the “representation”
- It is usually a real number.
- The function usually has a value between 0 and 1 and is monotonically increasing.
- Similarly the length of the route in the traveling salesperson problem is a good measure, because the shorter the route, the better the solution.
Outline of the Basic Genetic Algorithm

1. **[Start]** Generate random population of \( n \) chromosomes (suitable solutions for the problem)

2. **[Fitness]** Evaluate the fitness \( f(x) \) of each chromosome \( x \) in the population

3. **[New population]** Create a new population by repeating following steps until the new population is complete
Outline of the Basic Genetic Algorithm

4. **[Selection]** Select two parent chromosomes from a population according to their fitness (the better fitness, the bigger chance to be selected) The idea is to choose the better parents.

5. **[Crossover]** With a crossover probability cross over the parents to form a new offspring (children). If no crossover was performed, offspring is an exact copy of parents.

6. **[Mutation]** With a mutation probability mutate new offspring at each locus (position in chromosome).
Outline of the Basic Genetic Algorithm

7. **[Accepting]** Place new offspring in a new population

8. **[Replace]** Use new generated population for a further run of algorithm

9. **[Test]** If the end condition is satisfied, stop, and return the best solution in current population

10. **[Loop]** Go to step 2
Flow Diagram of the Genetic Algorithm Process

1. Describe Problem
2. Generate Initial Solutions
3. Test: is initial solution good enough?
   - Yes: Stop
   - No: Select parents to reproduce
4. Apply crossover process and create a set of offspring
5. Apply random mutation
Example: The Knapsack Problem

- You are going on an overnight hike and have a number of items that you could take along. Each item has a weight (in pounds) and a benefit or value to you on the hike (for measurements sake let’s say, in US dollars), and you can take one of each item at most. There is a capacity limit on the weight you can carry (constraint). This problem only illustrates one constraint, but in reality there could be many constraints including volume, time, etc.
GA Example: The Knapsack Problem

- Item:  1 2 3 4 5 6 7
- Benefit:  5 8 3 2 7 9 4
- Weight:  7 8 4 10 4 6 4
- Knapsack holds a maximum of 22 pounds
- Fill it to get the maximum benefit
- Solutions take the form of a string of 1’s and 0’s
- Solutions: Also known as strings of genes called Chromosomes
  - 1. 0101010
  - 2. 1101100
  - 3. 0100111
Example: The Knapsack Problem

- We represent a solution as a string of seven 1s and 0s and the fitness function as the total benefit, which is the sum of the gene values in a string solution times their representative benefit coefficient.

- The method generates a set of random solutions (initial parents), uses total benefit as the fitness function and selects the parents randomly to create generations of offspring by crossover and mutation.
Knapsack Example

- Typically, a string of 1s and 0s can represent a solution.

- Possible solutions generated by the system using Reproduction, Crossover, or Mutations
  1. 0101010
  2. 1101100
  3. 0100111
### Knapsack Example: Solution 1

<table>
<thead>
<tr>
<th>Item</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Solution</strong></td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td><strong>Benefit</strong></td>
<td>5</td>
<td>8</td>
<td>3</td>
<td>2</td>
<td>7</td>
<td>9</td>
<td>4</td>
</tr>
<tr>
<td><strong>Weight</strong></td>
<td>7</td>
<td>8</td>
<td>4</td>
<td>10</td>
<td>4</td>
<td>6</td>
<td>4</td>
</tr>
</tbody>
</table>

- Benefit \(8 + 2 + 9 = 19\)
- Weight \(8 + 10 + 6 = 24\)
# Knapsack Example: Solution 2

<table>
<thead>
<tr>
<th>Item</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Solution</strong></td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td><strong>Benefit</strong></td>
<td>5</td>
<td>8</td>
<td>3</td>
<td>2</td>
<td>7</td>
<td>9</td>
<td>4</td>
</tr>
<tr>
<td><strong>Weight</strong></td>
<td>7</td>
<td>8</td>
<td>4</td>
<td>10</td>
<td>4</td>
<td>6</td>
<td>4</td>
</tr>
</tbody>
</table>

- Benefit $5 + 8 + 7 = 20$
- Weight $7 + 8 + 4 = 19$
Knapsack Example: Solution 3

<table>
<thead>
<tr>
<th>Item</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Solution</strong></td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td><strong>Benefit</strong></td>
<td>5</td>
<td>8</td>
<td>3</td>
<td>2</td>
<td>7</td>
<td>9</td>
<td>4</td>
</tr>
<tr>
<td><strong>Weight</strong></td>
<td>7</td>
<td>8</td>
<td>4</td>
<td>10</td>
<td>4</td>
<td>6</td>
<td>4</td>
</tr>
</tbody>
</table>

- Benefit \(8 + 7 + 9 + 4 = 28\)
- Weight \(8 + 4 + 6 + 4 = 22\)
Knapsack Example

- Solution 3 is clearly the best solution and has met our conditions, therefore, item number 2, 5, 6, and 7 will be taken on the hiking trip. We will be able to get the most benefit out of these items while still having weight equal to 22 pounds.
- This is a simple example illustrating a genetic algorithm approach.
The Knapsack Problem

- The knapsack problem, though simple, has many important applications including determining what items to take on a space shuttle mission.
Another Problem

- The Maze search as a genetic algorithm....
- zeus.csci.unt.edu/swigger/csce3210/genetic-maze.doc
Another Problem

- Antenna design
Figure 1.7. Maximum and minimum gain at 7.2 GHz for antennas (a) ST5-3-10; (b) QFH.
Figure 1.8. Maximum and minimum gain at 8.47 GHz for antennas (a) STS-3-10; (b) QFH.
Genetic Algorithm Application Areas

- Dynamic process control
- Induction of rule optimization
- Discovering new connectivity topologies
- Simulating biological models of behavior and evolution
- Complex design of engineering structures
- Pattern recognition
- Scheduling
- Transportation
- Layout and circuit design
- Telecommunication
- Graph-based problems
Business Applications

- Schedule Assembly lines at Volvo Truck North America
- Channel 4 Television (England) to schedule commercials
- Driver scheduling in a public transportation system
- Jobshop scheduling
- Assignment of destinations to sources
- Trading stocks
- Productivity in whisky-making is increased
- Often genetic algorithm hybrids with other AI methods
In Genetic Programming, programs are evolved instead of bit strings. Programs are often represented by trees. For example:

$$\sin(x) + \sqrt{x^2 + y}$$
Crossover in Genetic Programming

[Diagram of crossover operation involving expressions with variables and functions]
Baldwin Effect

Assume
Individual learning has no direct influence on individual DNA
But ability to learn reduces need to "hard wire" traits in DNA
  -- can perform local search!

Then
  Ability of individuals to learn will support more diverse gene pool
More diverse gene pool will support faster evolution of gene pool

individual learning (indirectly) increases rate of evolution
Summary: Evolutionary Programming

- Conducts randomized, parallel, hillclimbing search through hypothesis space $H$
- Approaches learning as an optimization problem (optimize fitness)
- Nice property: evaluation of fitness can be very indirect
  - consider learning rule set for multistep decision making
  - no issue of assigning credit/blame to individual steps
- **Crowding** can occur when an individual that is much more fit than others reproduces like crazy, which reduces diversity in the population.
A good site example

Questions?