Genetic Algorithms

- Iterative improvement
- Takes inspiration from evolutionary theory
  - Start with a *population* of solutions and determine *fitness* using evaluation function
  - Individuals are selected for *reproduction* according to fitness
  - Reproduction subject to *cross-over* and *mutation*

Implementation

Function **GA-Solve**(*Fitness function F*)
Initialize $population_0$ to a set of N random solutions
For i = 0 to *MAXGENERATIONS*
  P = ComputeProbabilities*(*$population_i$, *F)*
  For j = 1 to N/2
    u = Select($population_i$, P); v = Select($population_i$, P)
    (u, v) = Crossover(u,v)
    u = Mutate(u); v = Mutate(v)
    $population_{i+1}$ = union($population_{i+1}$, {u, v})
Return individual in $population_{MAXGENERATIONS}$ with greatest fitness

Implementation (cont.)

Function **ComputeProbabilities**(*population*, *fitness_fn*)

$$Total_fitness = \sum_{i \in population} fitness_fn(i)$$

Foreach i in *population*

$$\Pr(i) = \frac{fitness_fn[i]}{total_fitness}$$

Return *Pr*

Function **Select**(*population*, *distribution*)
Return some randomly selected member i of *population*, such that each member's probability is determined by distribution.

Implementation (cont.)

- Solutions are represented as bit strings

Function **Crossover**(*i1, i2*)
With probability *CrossoverRate*
  Select a range 0 <= *r1* < *r2* <= |Bitstring representation|
  Swap *i1[r1..r2]* and *i2[r1..r2]*
Return *i1, i2*

Function **Mutate**(*individual*)
With probability *MutationRate*
  Select a bit in *individual* and flip it

Solution Representation

- All solutions are represented by a bit string
  - Vector of binary attributes
  - Numerically encoded solutions
  - Etc.
- Either all bit strings must be valid solutions or crossover/mutation operations must be careful to produce only valid solutions

GA as Search

- Hill-climbing
  - Top solutions are (likely to be) maintained
- Mutation/crossover is random
  - Plays similar role as in simulated annealing
  - Prevents getting stuck in local minima
- Parallel multiple-restart hill-climbing
  - Hopefully some individual starts on slope of global maximum

Genetic Programming: An Alternate Representation

Function approximation task

- Solution is parse tree of a program
- All trees are valid “programs”
- Crossover exchanges subtrees
- Mutation modifies a constant or adds/removes tree structure
- Fitness determined by execution of program

7

GP Experience

- Attempt to generate “monitoring behaviour”
  - Look before you walk into a wall
- Various tasks/environments
  - Ops: Turn, FaceGoal, **LookAhead**, Move
  - Hitting barriers causes negative fitness
  - Goal has high positive fitness decaying exponentially with distance
  - Tasks: Reach goal or explore maximum area

8

What the system learned

- Big central obstacle
  - Bump into obstacle, then go round it
- Obstacles surround goal
  - Aim at goal, turn a little, then go, missing obstacle
- Few barriers
  - Learns a constant path that maximizes fitness
- Many barriers
  - Don’t move (robot is scared)

9

Summary

- Genetic Algorithms are inspired by natural selection
- Use fitness to improve solution quality
- Randomness avoids local maxima
- Multiple individuals search different parts of search space in parallel
- Need to be careful about encoding solutions and implementing crossover and mutation

10