Lecture04b: Evolutionary Algorithms

CS540 2/8/18

Announcements

On-campus students:
- Make sure I am wearing the microphone
- We are going to keep using the 2nd microphone for questions

All students:
- No paper this week – work on your projects!
- Next Thursday will be project presentations
  - Send me Powerpoint or pdf files by Wednesday night
- By now, you should think you know what your insight will be
- Any questions?

Back to the beginning

Assume an optimization problem

Space of feasible solutions: $\mathbb{R}^N$ for some N

Infinite solution space (real valued variables)

Makes crossover & mutation easy…

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
      $x_i' = x_i + \sigma N_i(0,1)$
   2. If (f(x') ≥f(x) then
      x := x'
   3. Update **σ**

N(0,1) indicates a normal distribution with 0 mean and 1 standard deviation

(1+1)-ES Update

How to set/update **σ**?

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

$$\frac{\#\ of\ "better"\ steps}{all\ or\ last\ t\ tsteps}$$

If ratio is > 1/5, increase **σ**, else decrease

Or keep it fixed (better for escaping local optima)

$(\mu+\lambda)$-ES

μ solutions in population

Generate λ new solutions

Choose the best μ from original + new at each iteration

Can add recombination before mutation

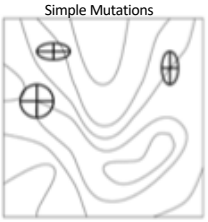
Can have different σ per variable

Evolutionary Strategies (cont.)

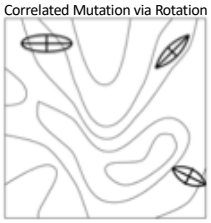
Self-adaptive mutation & rotation

- Log-normal distribution for mutation
- Adaptive through strategy (σ) and rotation angle (α) parameters added through chromosome: $\langle x_1, x_2, \sigma_1, \sigma_2, \alpha \rangle$

Simple Mutations



Correlated Mutation via Rotation

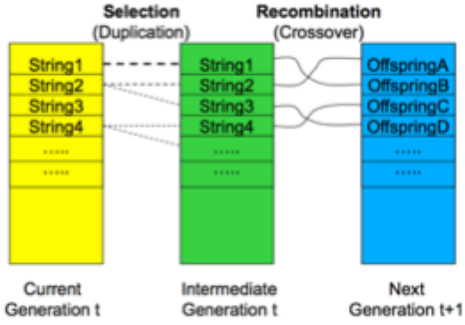


Figures courtesy of D. Whitley

Simple Genetic Algorithm

1. Create initial population P with N elements x^i
2. For I = 1 to N do
   1. Calculate $f(x^i)$
3. While (not termination criterion)
   1. Create M with N individuals selected from P (resampling based on f)
   2. Index = 1
   3. Repeat
      1. If random(0,1) ≤ p recombine x^i and x^{i+1} and add to P'
      2. Else copy x^i and x^{i+1} to P'
      3. Index += 2
   4. Until index > N
   5. For i =1 to N
      for j = 1 to |x|
      if random(0,1) ≤ p then mutate(x^i_j) in P']
   6. Calculate $f(x^i)$
   7. for P'
   8. P = P'

Genetic Algorithm Illustrated



Simple GA Selection

Proportionate Selection *with replacement*

- Whole population evaluated according to $f(x^i)$
- Elements selected by relative fitness:

$$f_i = \frac{f(x^i)}{\sum_{j=1}^{N} f(x^j)}$$

- Roulette analogy: Individuals get more "space on the wheel" if they are more fit
  - Think of f_i as defining how many slots a sample gets

Implementing Proportional Selection

String	Fit	Space	copies
001000000	2.0	.095	
101010101	1.9	.186	
111110011	1.8	.271	
010001100	1.7	.352	
111100000	1.6	.429	
101000110	1.5	.5	
011001110	1.4	.567	
001111000	1.3	.629	
000110100	1.2	.686	
100100011	1.1	.738	
010000111	1.0	.786	

String	Fit	Space	copies
011001111	0.9	.829	
000100110	0.8	.867	
110001101	0.7	.9	
110000111	0.6	.929	
100100100	0.5	.952	
011011011	0.4	.971	
000111000	0.3	.986	
001100100	0.2	.995	
100111010	0.1	1.0	
010010011	0.0	--	

Alternate Selection Methods

Tournament Selection
- Randomly select two population elements, add the more fit to P'

Rank-based Selection
- Order individuals by rank, not fitness.

Remainder Stochastic Sampling
- Proportional mapping to roulette wheel
- Outer ring as N evenly spaced pointers
- Spin outer ring once, take samples that are pointed to

What are the advantages and disadvantages of each method?

Simple Genetic Algorithm (redux)

1. Create initial population P with N elements x^i
2. For I = 1 to N do
   1. Calculate $f(x^i)$
3. While (not termination criterion)
   1. Create M with N individuals selected from P (resampling based on f)
   2. Index = 1
   3. Repeat
      1. If random(0,1) ≤ p recombine x^i and x^{i+1} and add to P'
      2. Else copy x^i and x^{i+1} to P'
      3. Index += 2
   4. Until index > N
   5. For i =1 to N
      for j = 1 to |x|
      if random(0,1) ≤ p then mutate(x^i_j) in P']
   6. Calculate $f(x^i)$
   7. for P'
   8. P = P'

Recombination / Crossover

Two parents: binary strings representing an encoding of 5 parameters that are used in some optimization problems.

```
 10010101    11011001    01110100    10100101    10000101
 xyxyyyxx    xyyyyxyx    xyyxyyxy    yyxyxxxy    yxyxyxyx
```

Recombination occurs as follows:

```
10010 \ / 10111011001011101 \ / 001010010110000101
xyxyy / \ yxxxyyyyyxyxxyyxy  / \ xyyyyxyxxxyyxyxyxyx
```

Producing the following *offspring*:

```
10010yxxxyyyyyxyxxyyxyy001010010110000101
xyxyy10111011001011101xyyyyxyxxxyyxyxyxyx
```

Desirable Characteristics of Recombination Operators

Respect
- Any commonalities of the parents are inherited by the offspring

Transmission
- All components of the offspring come from one of the parents

Assortment
- All offspring are feasible (all components are compatible)

Ergodicity
- Can eventually create any offspring from any initial population

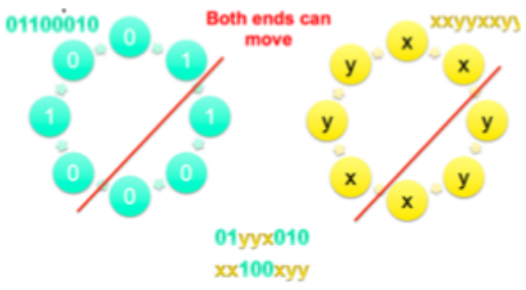
Combination Operators

1 point crossover
- Pick single crossover point
- Split both strings at this point
- Recombine (before split from one parent, after from the other)

2 point crossover
- Think of the strings as rings
- Pick two crossover points (dividing parents into 3 pieces each)
- Child contains first and last section of one parent, middle of the other

Uniform Crossover
- Randomly pick each element from one of the two parents

2 point crossover illustrated



HUX: Another Combination Operator

HUX: exactly half of the differing bits are swapped

Given parents:

1001101011110000101 10

1011000111000011000 01

a new individual is:

1001001111100001100 10

Reduced Surrogate Combination

Reduced Surrogate: Crossover points chosen within differing bits only

1001101011110000101 10

1011000111000011000 01

may become:

1001101111100001100 01

Why (under what conditions)are HUX and reduced surrogate sampling important?
Why (under what conditions) might reduced surrogate be "better" than HUX

Simple Genetic Algorithm (redux 2)

1. Create initial population P with N elements x^i
2. For I = 1 to N do
   1. Calculate $f(x^i)$
3. While (not termination criterion)
   1. Create M with N individuals selected from P (resampling based on f)
   2. Index = 1
   3. Repeat
      1. If random(0,1) ≤ p recombine x^i and x^{i+1} and add to P'
      2. Else copy x^i and x^{i+1} to P'
      3. Index += 2
   4. Until index > N
   5. For i =1 to N
      for j = 1 to |x|
      if random(0,1) ≤ p then mutate(x^i_j) in P']
   6. Calculate $f(x^i)$
   7. for P'
   8. P = P'

Mutation

For each bit/number in population

- Mutate with probability p
- p should be low, typically ≤ .01

Mutation means

- If bits, flip
- If numbers, add $\sigma N(0,1)$

Which Strings in New Generation?

Replace with offspring

- Canonical GA

Elitism: Keep best parent in population and (all but one) offspring

- Anytime algorithm: guaranteed improvement over time

Keep best of parents and offspring

- Anytime algorithm: guaranteed improvement over time
- Intense pressure on improvement
- Less exploration?

Termination Criteria

Quality of Solution

- Best individual passes pre-set threshold

Time

- Iterate for a fixed number of generations

Diminishing returns

- Generation to generation improvement is below a threshold

Convergence

- Too little variation among members of population