Optimization Theory
MT 610

2011/12 Semester I

Evolutionary Algorithms

Metaheuristics

- Recall heuristic methods find “good enough” solutions by successively improving on the current solution
- Two categories
  - *Trajectory* methods – a single solution
  - *Population* methods – multiple, simultaneous solutions
- We've seen trajectory methods with Simulated Annealing and Tabu Search

Population Methods

- Differ from trajectory methods
  - Maintain a sample of candidate solutions rather than a single candidate solution
- Changes to one solution affect them all
  - Poor solutions rejected / new created
  - Remaining solutions tweaked for improvement
- Most mimic biological systems
  - Evolutionary Computation → Evolutionary Algorithm
    - Evolution Strategies
    - Genetic Algorithms

Common Terms

individual	a candidate solution
child and parent	a *child* is the Tweaked copy of a candidate solution (its *parent*)
population	set of candidate solutions
fitness	quality
fitness landscape	quality function
fitness assessment or evaluation	computing the fitness of an individual
selection	picking individuals based on their fitness
mutation	plain Tweaking. This is often thought as "asexual" breeding.
recombination or crossover	A special Tweak which takes two parents, swaps sections of them, and (usually) produces two children. This is often thought as "sexual" breeding.
breeding	producing one or more children from a population of parents through an iterated process of selection and Tweaking (typically mutation or recombination)
genotype or genome	an individual's data structure, as used during breeding
chromosome	a genotype in the form of a fixed-length vector
gene	a particular slot position in a chromosome
allele	a particular setting of a gene
phenotype	how the individual operates during fitness assessment
generation	one cycle of fitness assessment, breeding, and population re-assembly; or the population produced each such cycle

Abstraction Generational EA

Algorithm 17 *An Abstract Generational Evolutionary Algorithm (EA)*

$P \leftarrow$ Build Initial Population
$Best \leftarrow \square$ ▷ $\square$ means "nobody yet"
repeat
    AssessFitness(P)
    for each individual $P_i \in P$ **do**
        if $Best = \square$ or Fitness(P_i) $>$ Fitness($Best$) **then** ▷ Remember, Fitness is just Quality
            $Best \leftarrow P_i$
    $P \leftarrow$ Join(P, Breed(P))
until $Best$ is the ideal solution or we have run out of time
return $Best$

Population Initialization

- Create λ individuals "at random"
- Likely to know "good" regions so
  - *Bias* random generation to favor those regions
  - *Seed* initial population with specific individuals
- Do not overly bias / seed, random is good
- Do not include duplicates in the population
  - Techniques simplify that

Evolutionary Strategy: μ, λ

- Developed by Ingo Rechenberg and Hans-Paul Schwefel at the Technical University of Berlin in the mid 1960s.

```
μ ← number of parents selected
λ ← number of children generated by the parents

P ← {}
for λ times do                                              ▷ Build Initial Population
    P ← P ∪ {new random individual}
Best ← □
repeat
    for each individual P_i ∈ P do
        AssessFitness(P_i)
        if Best = □ or Fitness(P_i) > Fitness(Best) then
            Best ← P_i
    Q ← the μ individuals in P whose Fitness( ) are greatest            ▷ Truncation Selection
    P ← {}                          ▷ Join is done by just replacing P with the children
    for each individual Q_j ∈ Q do
        for λ/μ times do
            P ← P ∪ {Mutate(Copy(Q_j))}            Mutate = Tweak
until Best is the ideal solution or we have run out of time
return Best
```

The Knobs for Tweaking

- The μ, λ algorithm has 3 knobs to adjust
  - Size of λ
    - Sample size of each population
  - Size of μ, the number of parents selected
    - How *selective* is the algorithm
      - μ/λ low means more exploitative, only the best survive
  - Degree of *Mutation*
    - Larger the noise in the tweaking, the more random the children, regardless of selectivity μ

Mutation

- May be considered *unary reproduction*
  - Hence some children are recombinations of two parents and some are mutations of a single parent
- Decision variables converted to a string of binary digits
- Mutation is random changing of bits in the string, i.e. the bits are like chromosomes
  - Example 4-bit integer, 10 $\rightarrow$ 1010
  - Mutate one bit to 1110 $\rightarrow$ 14

Evolutionary Strategy: μ + λ

$\mu \leftarrow$ number of parents selected
$\lambda \leftarrow$ number of children generated by the parents

$P \leftarrow \{\}$
for λ times **do**
    $P \leftarrow P \cup \{$new random individual$\}$
$Best \leftarrow \square$
repeat
    for each individual $P_i \in P$ **do**
        AssessFitness(P_i)
        if $Best = \square$ or Fitness(P_i) $>$ Fitness($Best$) **then**
            $Best \leftarrow P_i$
    $Q \leftarrow$ the μ individuals in P whose Fitness( ) are greatest
    $P \leftarrow Q$ ▷ The Join operation is the only difference with (μ, λ)
    for each individual $Q_j \in Q$ **do**
        for λ / μ times **do**
            $P \leftarrow P \cup \{$Mutate(Copy(Q_j))$\}$
until $Best$ is the ideal solution or we have run out of time
return $Best$

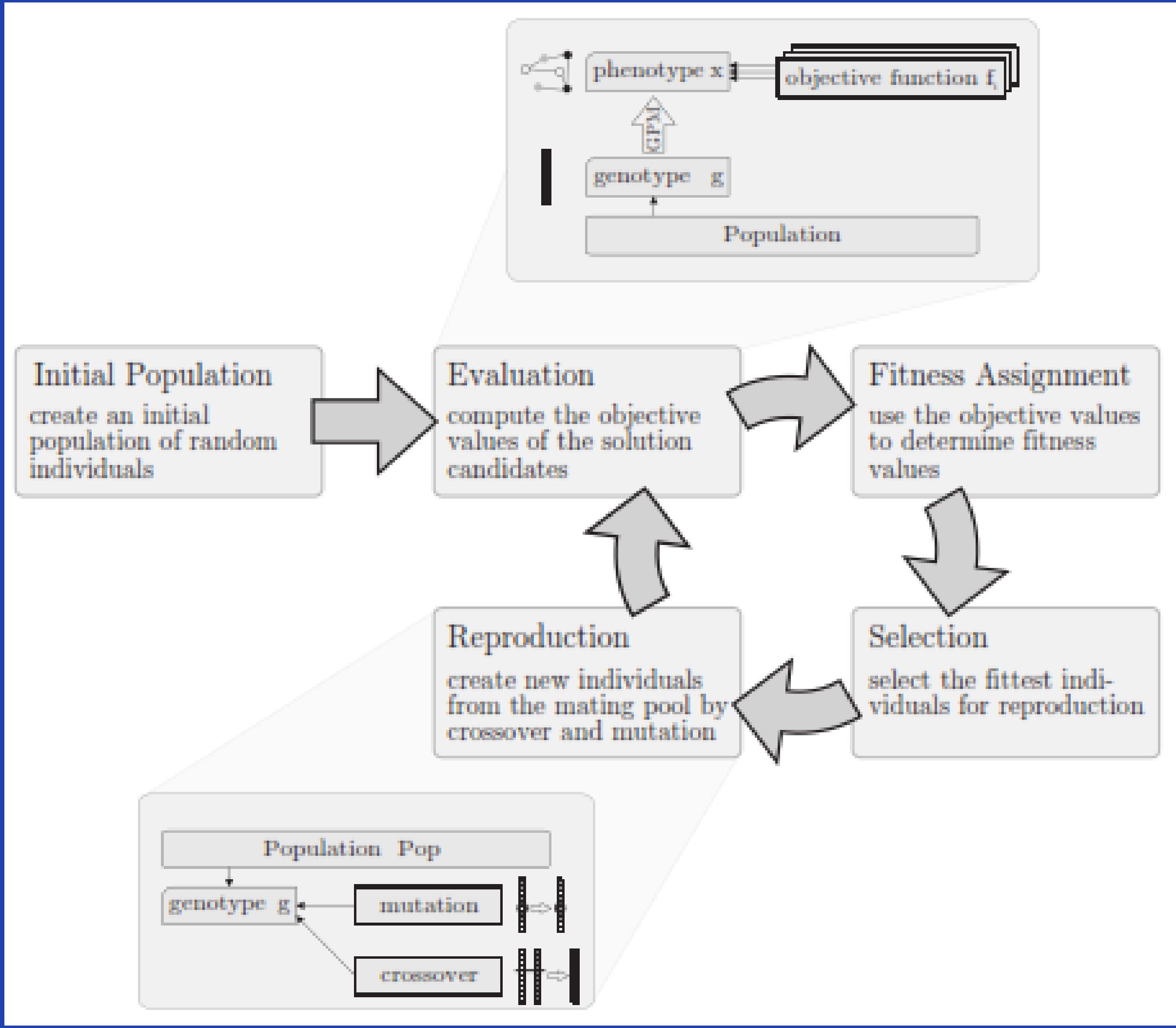
Strategy Difference

- First case, μ, λ
  - Children replace parents
- Second case, μ + λ
  - High-fit parents persist
- 2^{nd} is more exploitative, but with a risk
  - Sufficiently fit parent may defeat others in the population and solution may converge too quickly to a local minimum, not a global

Genetic Algorithm

- Invented in 1970's by John Holland, University of Michigan
- Similar to μ, λ EA
- Difference in how selection and breeding take place
  - EA: Selects parents then creates children
  - GA: Selects a few parents, creates children and continues until enough children created

Genetic Algorithm Flow



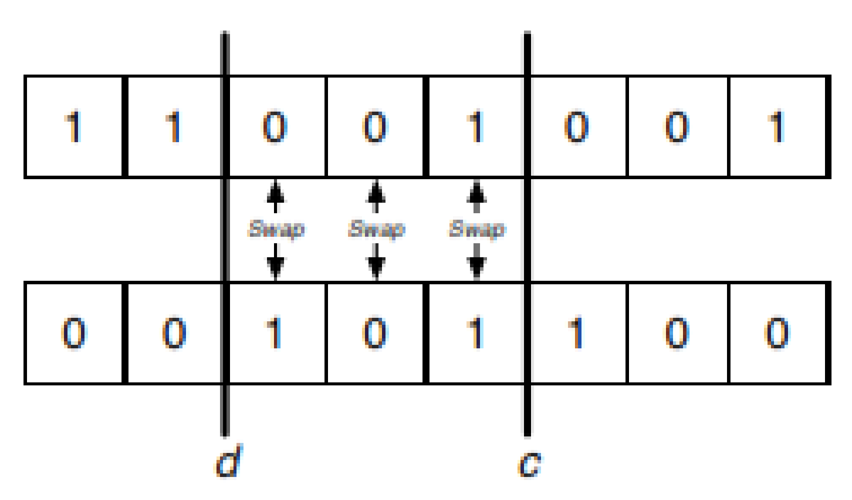
Genetic Algorithm Pseudocode

```
popsize ← desired population size                ▷ This is basically λ. Make it even.

P ← {}
for popsize times do
    P ← P ∪ {new random individual}
Best ← □
repeat
    for each individual P_i ∈ P do
        AssessFitness(P_i)
        if Best = □ or Fitness(P_i) > Fitness(Best) then
            Best ← P_i
    Q ← {}                                       ▷ Here's where we begin to deviate from (μ, λ)
    for popsize/2 times do
        Parent P_a ← SelectWithReplacement(P)
        Parent P_b ← SelectWithReplacement(P)
        Children C_a, C_b ← Crossover(Copy(P_a), Copy(P_b))
        Q ← Q ∪ {Mutate(C_a), Mutate(C_b)}
    P ← Q                                        ▷ End of deviation
until Best is the ideal solution or we have run out of time
return Best
```

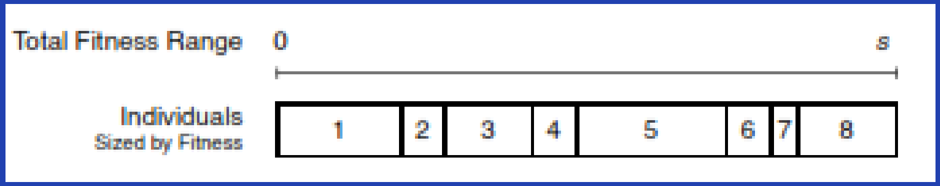
Cross-over and Mutation

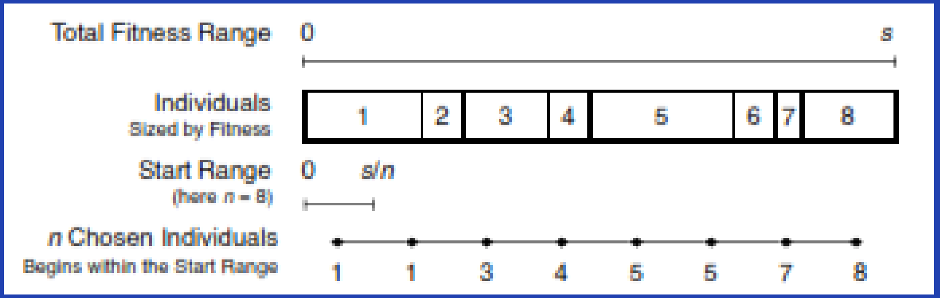
- Key to GA is in the breeding phase
  - Select with Replacement (next slide)
  - Crossover
    - Mixing/Matching parts of parents to form children



SelectionWithReplacement

- Reinforces the “fittest” parents
- Not everyone selected to be parents
  - Fitness-proportionate selection
    - Random pick, but more fit get more
  - Stochastic Universal Sampling
    - Adds bias so fittest get selected AT LEAST once





References / Further Reading

- Luke, Sean. *Essentials of Metaheuristics*, 1st Ed. http://cs.gmu.edu/~sean/book/metaheuristics/, 2009-2011, Ch 3.
- Rao, Singiresu S. *Engineering Optimization: Theory and Practice*, 3rd Ed. New Dehli: New Age International (P) Ltd, 2010, Ch 12.7, pp 676-678.
- Baudin, Michael and Vincent Couvert. *Optimization in Scilab*. Paris: Scilab Consortium, 2010, Chapter 5.
- Weise, Thomas. *Global Optimization*. www.it-weise.de, 2009, Ch 3-4.

- Scilab is available for free, non-commercial use at www.scilab.org