CSE/ECE 848
Introduction to Evolutionary Computation

Module 2, Lecture 6, Part 2c
More Principles of Evolutionary Computation -- Selection

Erik Goodman
Professor, ECE, ME, and CSE
Michigan State University

Selection – An Important Part of Any Evolutionary Algorithm

- In a SGA, based on fitness, choose the set of individuals (the "intermediate" or parent population) to:
  - survive untouched, or
  - be mutated, or
  - in pairs, be crossed over and possibly mutated

  then evaluated (if new) and kept as part of the next population (maybe replacing parents, or replacing the worst, or if worse than parents, discarded, etc.)
- One individual may appear several times in the intermediate population (or the next population)
- "Selective pressure" or "selection pressure" informally means ratio of chances of survival of best fitness individual to average fitness individuals… but used differently sometimes…

Selection Can Be Done in Various Places, Ways

- There must be selection of some sort done somewhere in an EC algorithm, but, for example, it could be done:
  - To choose individuals to be parents from a population
  - To choose who among the parents and new offspring will survive
  - To choose which parents will be replaced
- These can be done in various combinations, but something must determine who the parents are and what happens to the parents and offspring, and some fitness bias must be introduced somewhere.
- Some algorithms introduce “mating restrictions” that only allow individuals with certain properties or similarities to be chosen as parents for crossover

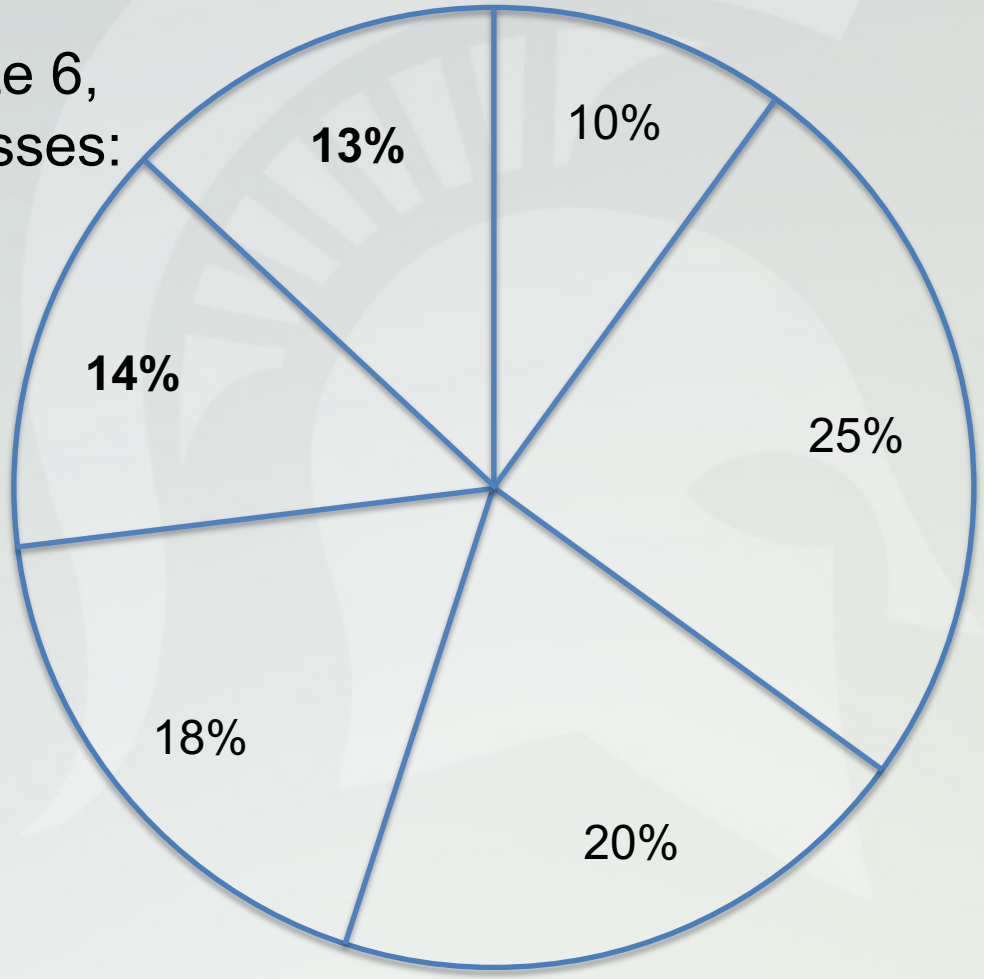
Types of Selection Using Relative Fitness

This is what GA's started with:

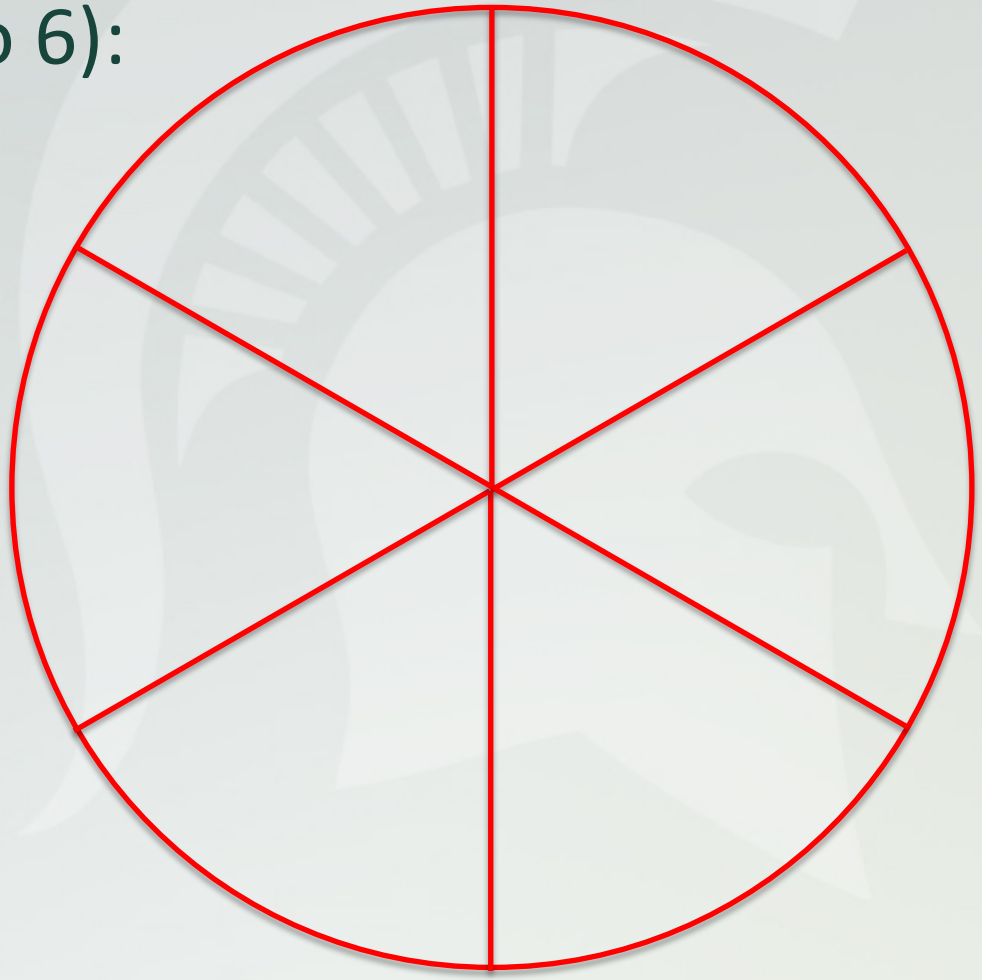
- "Roulette wheel" -- classical Holland *fitness proportional selection* -- chunk of wheel ~ *relative* fitness—now rarely used
- Stochastic universal sampling -- better sampling + fewer calls to random number generator –
  - Divide wheel into fitness "pie slices" as before (dividing each fitness by sum of fitnesses)
  - Pick O = rand [0, 1/M)
  - Step from there in increments of 1/M (uniform slices offset by offset O) – see following figures
  - Still fitness proportional, but lower variance than RWS

Illustration of Stochastic Universal Sampling (Baker, 1987)

Population Size 6,
Relative Fitnesses:
0.10
0.25
0.20
0.18
0.14
0.13

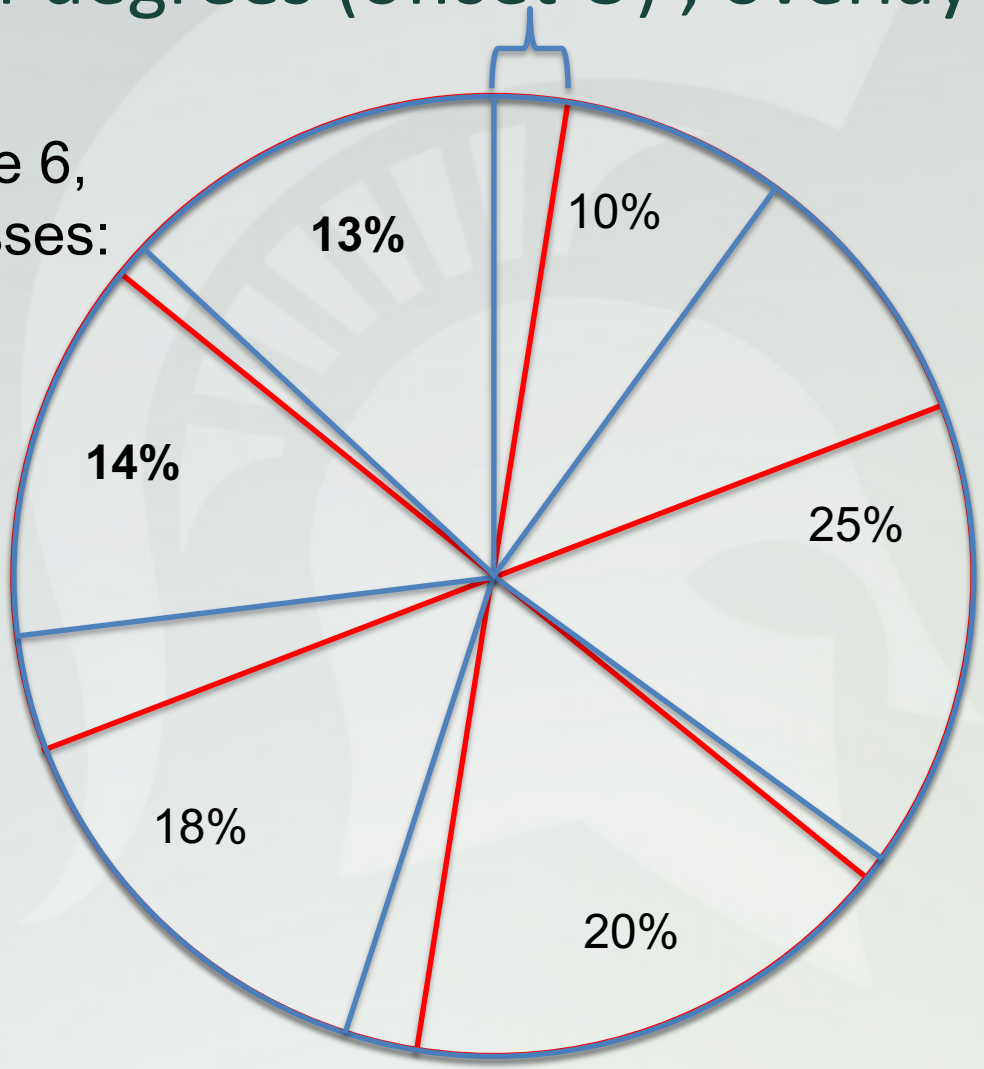


Make a circle with the number of samples we want to generate, equally divided (here, into 6):



Now rotate the red circle a random number of degrees (offset O) , overlay on blue circle:

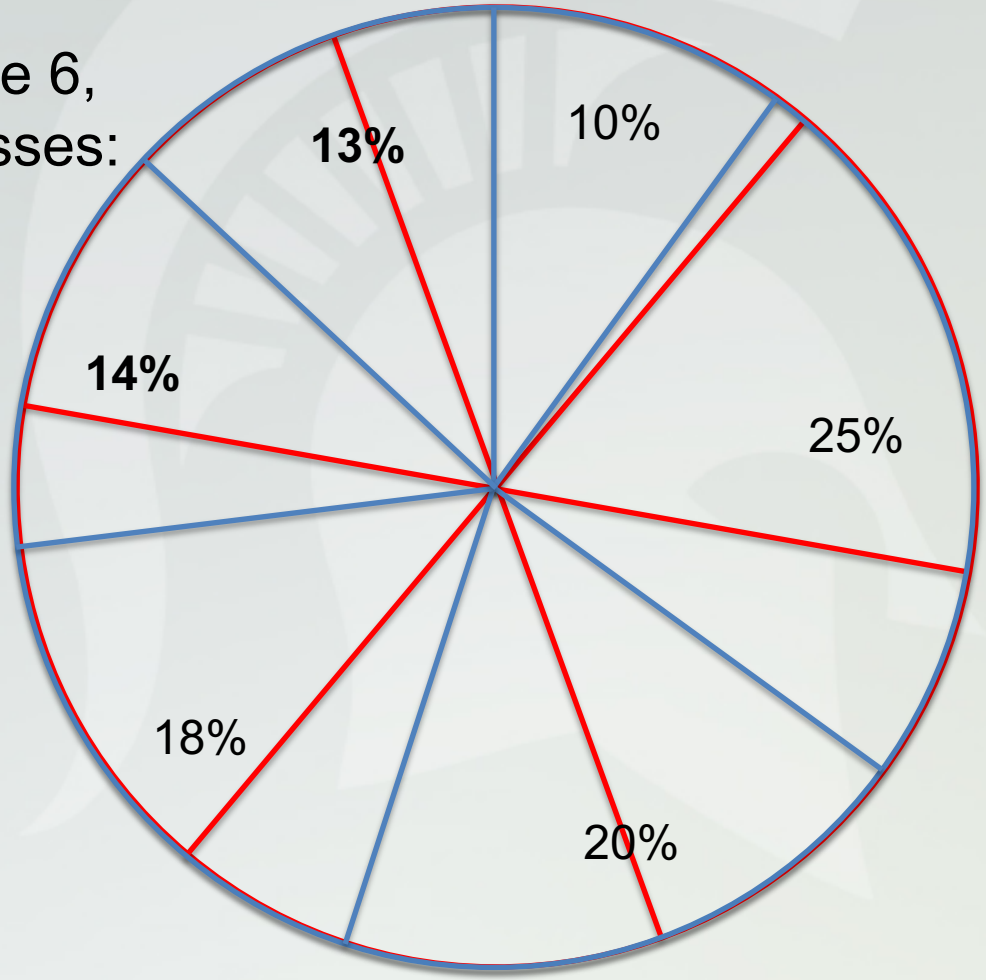
Population Size 6,
Relative Fitnesses:
0.10
0.25
0.20
0.18
0.14
0.13



This configuration samples all individuals but the sixth, and samples the third twice. Only one random number call.

If choose a different random offset, might instead get:

Population Size 6,
Relative Fitnesses:
0.10
0.25
0.20
0.18
0.14
0.13



This configuration samples all individuals but the first, and samples the second twice.

So Stochastic Universal Sampling Is Cheaper & BETTER than Roulette Wheel!

- SUS provides fitness-proportional selection, but with a lower variance than Roulette Wheel Sampling.
- It is universally preferred over Roulette Wheel!
- But why is it now rarely used?
- Because it suffers the same problem as Roulette Wheel Sampling—what happens to relative fitnesses as the population evolves?

The Trouble with Fitness-Proportional Selection: *Fitnesses Must Be SCALED*

- As evolution progresses, relative fitness differences get smaller (as population gets more similar to each other). So best and worst guys have near same probability of becoming parents—not good! (Loss of *selection pressure*)
- So we can fight this by SCALING relative fitnesses to keep about same ratio of best guy/average guy, for example
- **OR, Even better:** use tournament or rank-based or elitist selection, NOT proportional (relative) fitness

Types of Selection NOT Using Relative Fitness

- Rank-based selection, so amounts of difference don't matter—just #1, #2, etc.—can distribute probabilities as you like (***but must sort…)***
- Tournament selection (picking each parent as the best of a new sample of k individuals)
  - k = 2 → weak selection pressure; k = 5 → strong pressure
  - A variation divides the entire population N into N/k tournaments ("sampling without replacement"), to provide less randomness in who participates in tournaments—this way, each individual participates in exactly one tournament. Shuffle population first to avoid any deliberate or inadvertent ordering of the population by fitness.

Types of Selection NOT Using Relative Fitness

- "Elitist selection" (upper k% of a population), from ES
  - (μ, λ) makes $\lambda > \mu$ kids, best μ replace parent pop,
  - or ($\mu + \lambda$) makes λ kids, added to parents, best μ survive
  - Either scheme can work better, depending on the class of problems being optimized
- Also called "truncation selection," meaning the top k% of some population is selected
- Note: *elitism* is also used to mean guaranteeing that the best ONE (or k) of the parent population always survive, even though some other method is used to determine other survivors. "With elitism" usually means this.

Steady-State Evolutionary Algorithms

In any kind of evolutionary algorithm, instead of generating a whole generation's kids, then deciding who to keep when all have been evaluated—

- Generate one kid, send for evaluation, work into population (or discard) as soon as it's been evaluated
- Can generate and evaluate more kids before that finishes—no need to wait for it…
- Strongly improves capability to parallelize calculations, as is now ASYNCHRONOUS
- Gen gap = 0, vs Gen gap = N for generational GA

EC Differences from Other Search Methods

- Maintain a set or "population" of solutions at each stage
- Typically use a *recombination* operator (operating on two or more solutions to generate an offspring)
- All we have learned up to time t is represented by time *t*'s POPULATION of solutions
- BIG advantage over single-point methods in ease of parallelization!

Contrast of EC with Other Search Methods

- "indirect" -- setting derivatives to 0
- "direct" -- hill climber
- enumerative
- random
- simulated annealing
- Tabu
- Response Surface Modeling (RSM) -- fits approx. surface to set of points, searches on that cheap-to-evaluate surface, avoids full evaluations until finds local optima on response surface; repeats

BEWARE of Claims about ANY Algorithm's Asymptotic Behavior – "Eventually" is a LONG Time

- LOTS of methods can guarantee to find the best solution, probability 1, eventually…
  - Enumeration
  - Random search (even better without resampling)
  - SA (properly configured)
  - Any GA that avoids "absorbing states" in a Markov chain
  - Any EA that continues to add new random individuals
- The POINT: you can't afford to wait that long, if the problem is anything interesting, so this claim doesn't do much for you!!!

When Might a GA Be Useful?

- Highly multimodal functions
- Discrete or discontinuous functions
- High-dimensionality functions, including many combinatorial ones
- Non-additive dependencies among parameters -- "epistasis" makes it hard for other approaches
- Often used for approximating solutions to NP-hard combinatorial problems
- DON'T USE if a hill-climber, etc., will work well

The Limits to Search

- No search method is best for all problems – per the No Free Lunch Theorem
- BUT fortunately, most real-world problems have some simple relationships among their variables that a suitable algorithm can discover and exploit! (See Kolmogorov complexity)
- Needle-in-a-haystack is just *hard*, in fact
- Don't let anyone tell you a GA (or THEIR favorite method) is best for all problems!!!
- Efficient search must be able to EXPLOIT correlations/relationships in the search space, or it is no better than random search or enumeration
- Must balance with EXPLORATION, so don't just find nearest local optimum

Examples of Successful Real-World GA/EC Application

- Antenna design
- Drug design
- Chemical classification
- Electronic circuits (Koza)
- Factory floor scheduling (Volvo, Deere, others)
- Turbine engine design (GE)
- Crashworthy car design (GM/Red Cedar)
- Protein folding prediction
- Network design
- Control systems design
- Production parameter choice
- Satellite constellation orbital design
- Stock/commodity analysis/trading
- VLSI partitioning/ placement/routing
- Cell phone factory tuning
- Data Mining
- Deep Learning Network Architecture