Monday, May 3 Lecture #41/42

→ Homework Questions?

→ No class on Wednesday.

Topic 20 - Genetic and Evolutionary Algorithms

Pseudocode:

```
pop = [μ random solutions]
while True:
  best = best solution in pop
  next_gen = []
  while len(next_gen) < len(pop):
*     select two parents P1, P2 in pop              (how?)
*     perform crossover on P1 and P2                (how?)
          to get some children
*     allow each child to mutate with               (how?)
          some probability
      add the children to next_gen
  pop = next_gen
```

Selection Methods

How do we choose pairs of parents

to crossover.

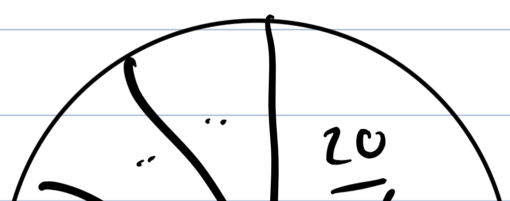
(1) Completely Randomly, with replacement
Each time you need two parents, pick randomly from the list. Don't remove. Make sure you don't pick the same parent twice.

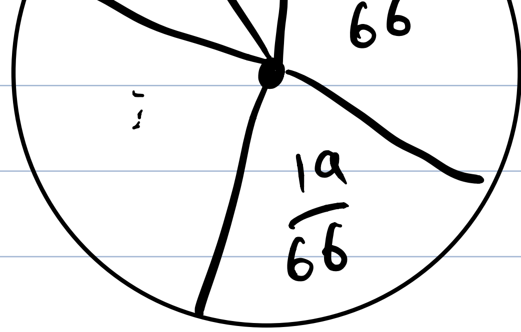
(2) Fitness Proportionate Section
(aka Roulette Selection)
Select at random, but not with equal probability. Set probabilities based on score.

<u>Ex</u>: Suppose we're maximizing, and our population has fitness values 20, 19, 15, 10, 2.
Total fitness = 66
So the probabilities will be:

$30\% \approx \frac{20}{66}, \frac{19}{66}, \frac{15}{66}, \frac{10}{66}, \frac{2}{66}$ $\approx 3\%$

(they add up to 1)



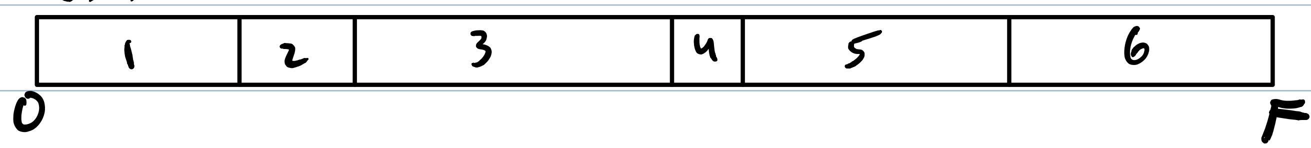


random.sample ?

(3) Stochastic Universal Sampling

Pick n parents all at once, possibly with repititions.

Line up the population in intervals whose width corresponds with their fitness.

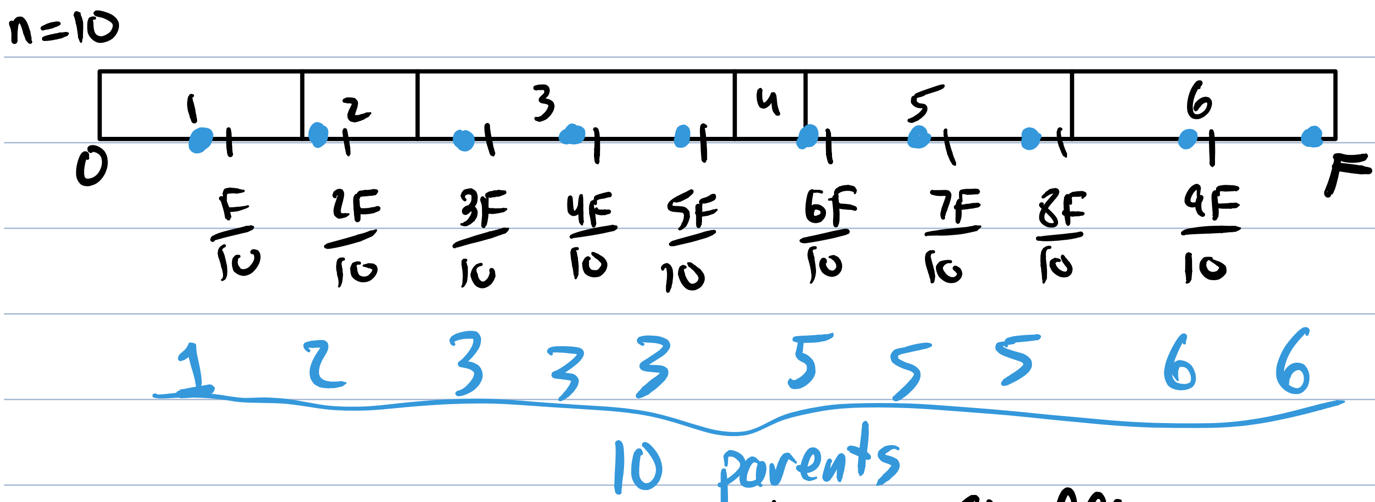


Let F be the sum of the fitnesses (width of the full interval).

To pick n times:

(1) pick a random start point between 0 and F/n.

(2) at every point F/n away, pick the corresponding parent



These are your selections. Shuffle.

Pros: Good things still get picked more, but now it's <u>guaranteed</u> that anything decently good (fitness $\geq \frac{F}{n}$) will get picked at least once.

Cons: If you don't want to crossover the same parent with itself, you have to be a little careful.

(4) Tournament Selection

(2) and (3) are very dependent on numerical values of the fitness. Often we only care about the relative ranking of the solutions.

<u>Ex</u>: If you have 3 solutions with

0 50 100

Scores 99.8, 99.9, 100, then (2) and (3) will pick them basically equally often.

Tournament Selection (parameter t)

To pick one thing from a population:

```
Best = random solution in the population
for i=2, ..., t:
    sol = random solution in the population
    if sol is better than Best:
        Best = sol
return Best
```

$t=2$ is usually good
$t=1$ is just random selection

If you need to be really selective, maybe $t=10$ is good.

One more topic:

Eliteism - When you pare down to

make the next generation, you can choose to include the parents so that they stick around if they're better.

$((\mu + \lambda)$ vs. $(\mu, \lambda))$

Big Example: TSP

Selection: Tournament Selection ($t=2$)

Mutation: Go through each edge one-by-one and delete it with prob. $\frac{1}{\#\text{cities}}$ $\left(\text{or } \frac{2}{\#\text{cities}}\right)$. Reconnect

in the best way possible, just like with k-opt.

Crossover ??

How do you blend two tours together?