Wednesday, April 28 Lecture #39/42

→ Homework Questions?

Topic 20 - Genetic and Evolutionary Algorithms

These will be population MHs inspired by evolution. Unlike PSO, Firefly, Cuckoo, the solution candidates don't need to represent points in space. These will work with any kind of space, continuous or discrete.

First group - "evolutionary strategies", we just do some tweaks to all of the things in our population, and keep the best ones.

$(10, 50)$

"$(\mu, \lambda)$ Evolution Strategy"

* Starts with a population of $\mu$ random solutions.
* Tweak each one of them $\lambda/\mu$

times, and keep the results whether they are better worse. Now we have λ solutions.

* Keep the best μ of those λ and throw the rest away.
* Repeat

Pseudocode:

```
pop = [μ random solutions]
while True:
    best = best solution we've ever
                seen
    next_gen = [ ]
    for sol in pop:
        repeat λ/μ times:
            new_sol = tweak(sol)
            next_gen.append(new_sol)
    # len(next_gen) is now λ
    pop = [μ best things in next_gen]
```

It's possible that nothing in next_gen is as good as the best thing in pop.

$(\mu, \lambda)$ always throws away the $\mu$ parents even if they were better than the $\lambda$ children.

Variant: "$(\mu+\lambda)$ Evolution Strategy"

In this variant we start with $\mu$ parents, generate $\lambda$ children, then we pick the best $\mu$ solutions out of the $\mu + \lambda$ parents and children together.

<u>Ex</u>: $(10+50)$

10 parents → 50 children → pick 10 best out of 60

<u>Pseudocode:</u>

```
pop = [μ random solutions]
while True:
    best = best solution we've ever
               seen
    next_gen = [ ]
    for sol in pop:
        repeat λ/μ times:
```

```
        new_sol = tweak(sol)
        next_gen.append(new_sol)
    # len(next_gen) is now λ
    pop = [μ best things in next_gen
                          or pop combined]
```

$(\mu, \lambda)$ – more explorative
$(\mu + \lambda)$ – more exploitative

good solutions tend to stick around longer

It is common to use these in conjunction with a tweak function whose intensity can be dialed up or down to find a good balance between exploration and exploitation.

"One Fifth Rule"   "One $n^{th}$ Rule"

* Aiming for about 1/5 of the children to be better than their parents.
* If more, too much exploitation, dial

up exploration with bigger tweaks.
* If less, too much exploration, dial down exploration with smaller tweaks.

Examples: continuous functions
Gaussian walk - bigger or smaller standard deviation
TSP - k-opt with smaller or larger k

## Genetic Algorithms

Tied for the most famous MH w/ Simulated Annealing.

Adds one critical idea to the Evol. Strats we just saw: crossover.
* Single parents can create children with a tweak (terminology: "mutation")
* Two parents can combine to produce one or more offspring that some qualities from each parent.

**Big Idea:** Start with a population of $\mu$ solutions. To form the next generation: we'll pick two solutions to be parents, then cross them over to form children. Each child is given some probability of mutating.

Do this a bunch of times with the $\mu$ parents. Now you have a bunch of children. Pick the best ones, and they become the next generation.

**Pseudocode:**

```
pop = [μ random solutions]
while True:
    best = best thing you've ever seen
    next_gen = []
    while len(next_gen) < λ:
*       select two parents P1, P2 in pop
*       cross them over to get some children
*       allow each child to mutate with
            some probability
        add to next_gen
```

pop = [best $\mu$ of next_gen]