Topic 16 – Firefly Search and Cuckoo Search (continued)

Friday, April 29

Announcements:

→ HW 5 due the last day of class Mon, May 9 11:59pm

→ Final will be takehome, due Mon, May 16, 11:59pm

Cuckoo Search

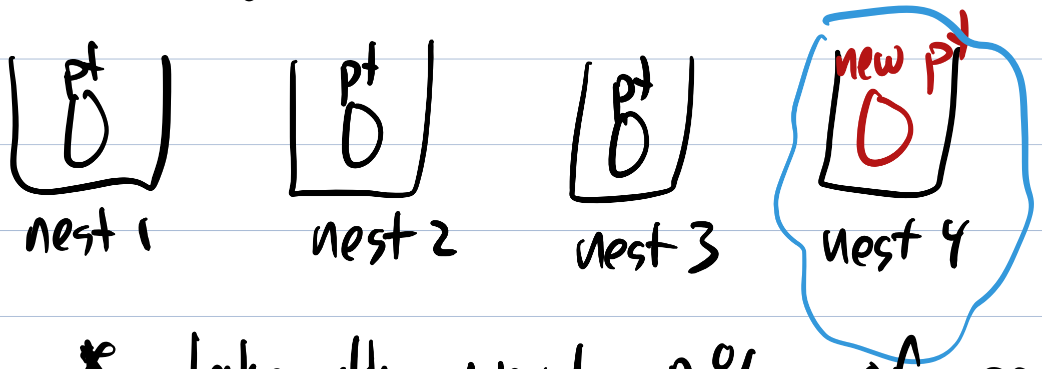
Cuckoos are birds and parasites – they lay their eggs in the nests of other birds, so that the other birds take care of them. The other birds sometimes get mad and fly away to make new nests.

Idea: N nests. Each nest contains one egg. eggs = solutions.

Repeat:

* pick a random nest. Its egg is a sol.
* form a new solution by tweaking with a Lévy flight.

* pick a new random nest, and if the new solution is better than the egg in that nest, replace it



* take the worst p% of eggs in any of the nests, and replace them all by tweaking with a Lévy flight.

Notes:

* Guaranteed to hang on to good solutions.
* Could work for discrete spaces! Just need a Lévy-style tweak, often a small change, occassionally a big change.
* It would be good to incorporate some kind of hill-climbing.

Topic 17 - Greedy Randomized Adaptive Search Procedures (GRASP)

Very fancy name for a very simple metaheuristic.

<u>Idea</u>: (1) Build a greedy solution, <u>but</u> not being as fully greedy as possible, so that you have choices.
(2) Starting at that greedy solution, perform H-C (single tweak or steepest ascent)
(3) Get rid of it, go back to (1).

GRASP is basically H-C with random restarts, but instead of starting from totally random solutions, you start from greedy-ish decently good solutions.

(1) better results than just H-C because you're more likely to start on a good hill
(2) the H-C itself is faster because

you have less far to climb.

Only question: How do we build greedyish solutions?

Regular Greedy: Build up a solution bit-by-bit, picking the best new component to add at each point in time.

Greedyish: At each step, compile some of the best next components, and randomly pick one of them to add.

How? Uniformly. Weighted (good = better or worse)

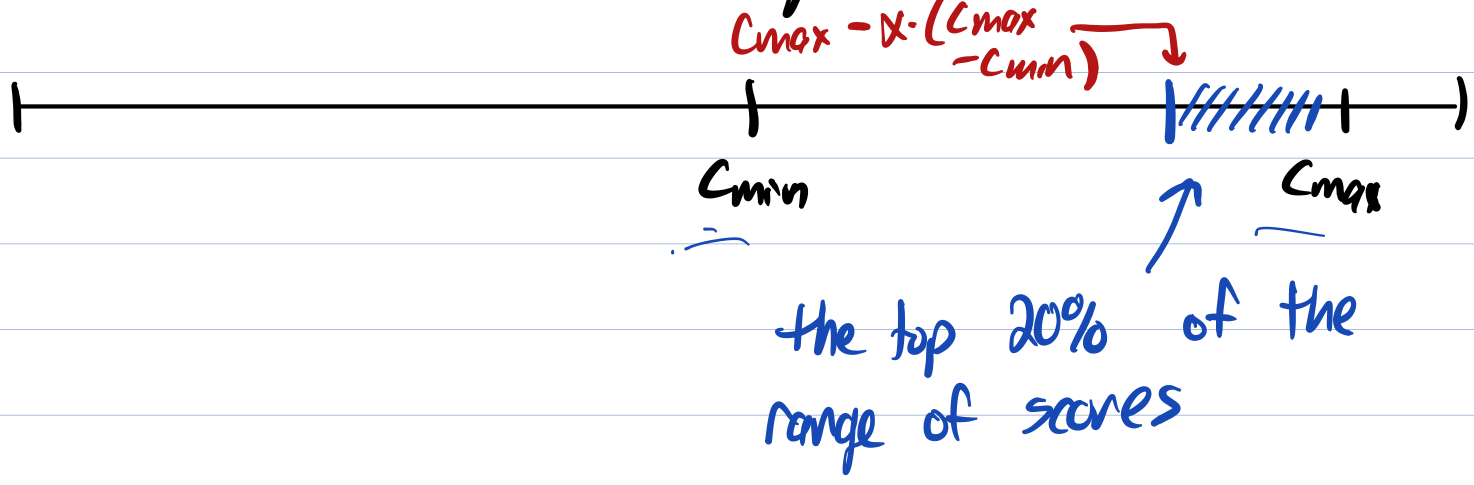
Called the Restricted Candidate List (RCL)

Two options for picking the RCL:
Option 1: Pick a percentage p and just put the top p% of options into the RCL. Good values of p depend on the problem, how

many components you have. Usually 10%-30% is good.

Option 2: Consider all possible next comp. and the value they would add. Let C_{min} and C_{max} to be the smallest and largest possible score. Form the RCL out of all possibilities in top α% of this score range.

$\alpha = 0.2$ (20%)


$C_{max} - \alpha \cdot (C_{max} - C_{min})$
C_{min}
C_{max}
the top 20% of the range of scores

This corresponds to all components whose score is $\geq C_{max} - \alpha \cdot (C_{max} - C_{min})$