Wednesday, April 26, 2023
Lecture #39
MSSC 6000
Announcements
* Homework 6 due the last day of class
11:59pm
* Final Exam (take-home) assigned last day of class, due Friday, May 13, 11:59pm
* Course Evaluations are open
* Mormal AH This work
* Normal OH this week > 2=30pm-3=30pm today on Teams
Topic 14 - Tabu Search (invented in 1986)
"Tabu" = "Taboo"

H-C: You walk up a hill and then you get stuck. What do you do?

(1) Random Restarts

(2) Sometin	mes go	downhill	(Sim. Ann	091.) (2)
(3) Go to	the be	st location	nearby	that
	1	already bee		_
		down hill		

Main Idea: Keep a running list of solutions you've tried before.

Do steepest-ascent hill climbing:

move to the best neighbor that

you have not already been to,

even if it's worse.

Only makes souse for discrete problems (unless you make some changes like discretizing a continuous problem or not really doing steepest ascent)

Problems =

Small problem: slow to check if a solution has been seen before

Bigger problem: requires a ten of 3 memory to store every old Golution

Fix #1: When you see a solution, you add it to "the tabu list" for some # of iterations (L), called the "tabu tenure".

This bans a solution from being revisited for the next L steps, but then it's allowed again.

In code: d = dict() keys = solutions

Alkey] = value

Values = the next

time this

Veep track of what # iteration

you're on, and when you see a allowed

Solution S at iteration # N,

we set d[S] = N+L

Whenever you wont to go to a G new solution S, you check dIsI.

If S is not even a key in the dictionary, then it's good. If it is a key in the dictionary, then we check if $dIsJ \leq (current iteration \#)$.

If so, then it's good.

If not, look at next best solution.

Problems

* Cycling - if L=20, you might eventually

end up cycling through the

same 20 solutions

* Storing whole solutions still isn't ideal.

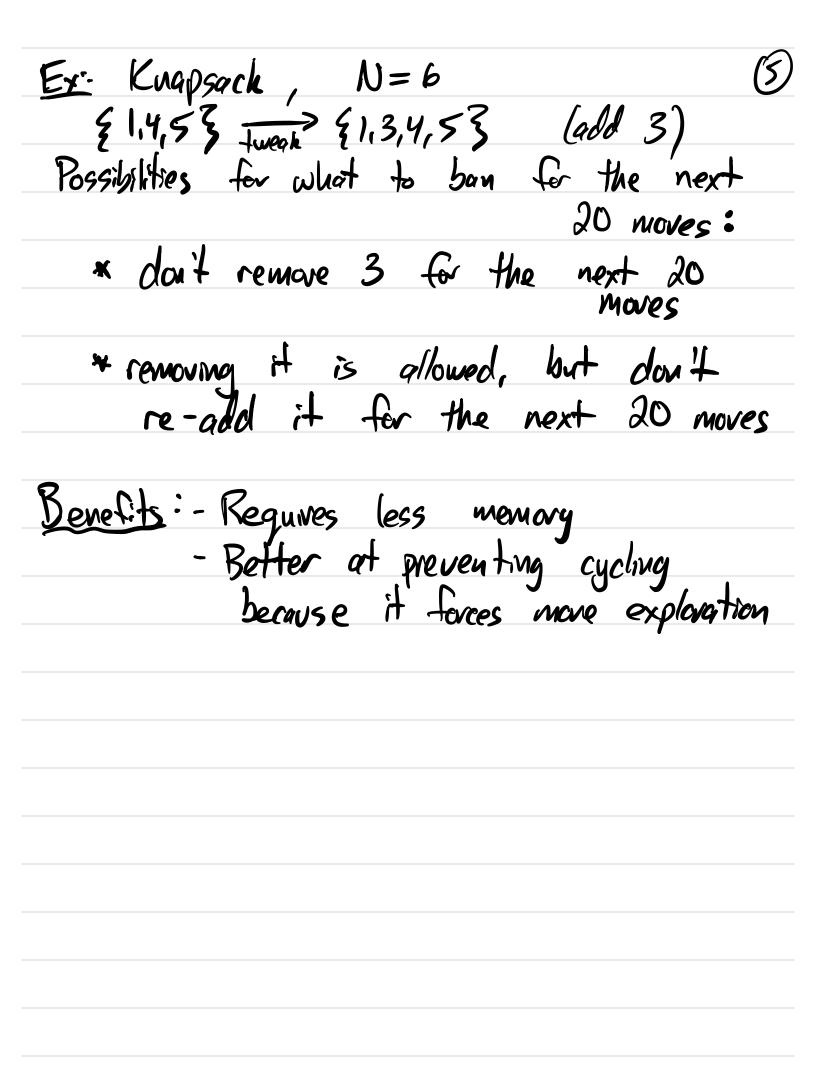
Fix #2: Keep the idea of tabu

tenure, but on top of that, instead

of banning full solutions we'll ban

the type of tweak that was done

(vague!)



Pseudocode: generation = 0 taboo = dict() # track when a move is

Labor = 200

Allowed again tabou_time = 20 x = random element of search space while True: generation = generation +1
neighbors = nbhd(x) # each neighbor is a par (s,m) where s is the solution, and m is the move that turned x mto 5 (new_x, move) = the pair (5,m) in the set of neighbors such that either m is not a key in "taboo" or taboo[m] = generation with the highest score taboo_time