

Wednesday, April 26, 2023

Lecture #39

MSSC 6000

①

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

* 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) Sometimes go downhill (Sim. Anneal.) (2)

(3) Go to the best location nearby that you haven't already been to before, even if it's downhill! (Tabu Search)

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 sense 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 ton of
memory to store every old
solution

3

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: $d = \text{dict}()$
 $d[\text{key}] = \text{value}$

keys = solutions
values = the next
time this

Keep track of what # iteration
you're on, and when you see a
solution S at iteration # N ,
we set $d[S] = N + L$

sol. is
allowed

Whenever you want to go to a new solution s , you check $d[s]$.

(4)

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 $d[s] \leq (\text{current iteration \#})$.

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!)

Ex: Knapsack, $N=6$

(5)

$\{1,4,5\} \xrightarrow{\text{tweak}} \{1,3,4,5\}$ (add 3)

Possibilities for what to ban for the next 20 moves:

* don't remove 3 for the next 20 moves

* removing it is allowed, but don't re-add it for the next 20 moves

Benefits :- Requires less memory
- Better at preventing cycling because it forces more exploration

Pseudocode:

(6)

generation = 0

taboo = dict() # track when a move is allowed again

taboo_time = 20

x = random element of search space

while True:

generation = generation + 1

neighbors = nbhd(x) # each neighbor is a pair (s, m) where s is the solution, and m is the move that turned x into s

(new_x, move) = the pair (s, m) in the set of neighbors such that either m is not a key in "taboo" or $\text{taboo}[m] \leq \text{generation}$ with the highest score

taboo[move] = generation + taboo_time