Topic 13- Tabu Search (Continued)

Wednesday, April 13

Main Idea: Keep a list of solutions you've tried so far.

Do steepest ascent hill-climbing: move to the best neighbor that you have not already been to, even if that means going downhill.

Problems:

small problem: can be slow to check if a solution has been seen before

huge problem: this would use way too much memory

Fix (quick sketch):

#1) Keep a list only of the N most recent things seen.

#2) Don't even keep the whole solution, just keep the tweak that created it.

Fix #1

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

In code: d = dict()

key = solutions

values = next time this solution is allowed to appear

Keep track of which iteration you're on. When you go to a solution S at iteration N, you set $d[S] = N+L$

Ex: If you're on it. 1,252 and $L=100$ then you set $d[S] = 1352$.

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

- If $d[S]$ doesn't exist, you're good.
- If $d[S]$ exists and is $\leq$ the current iteration #, you're also good.

Problems?

- Cycling behavior - if $L=20$, you could eventually end up cycling through the same 20 solutions over and over again.

- Storing whole solutions still is not totally ideal.

Fix #2: Keep the idea of tabu tenure, but on top of that, instead of remembering whole solutions, just remember the tweak you did.

Ex: Knapsack $W=6$

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

Could do:

* don't remove 3 for 20 moves.
* could remove, but don't readd for 20 moves
* or both.

Ex: $L=500$ $\{1,2,3,4,\ldots\ldots,20\}$

all combinations of 10 items being in or out $-$ $2^{10}=1024$

Benefits:

- Less to remember
- Prevents repeatedly changing just a few components to go in a cycle, and forces more exploration.

Vague Pseudocode:

```
generation = 0
taboo = dict()
taboo_tenure = 20        (or whatever)
x = random element of search space
while True:
    generation += 1
    neighbors = nbhd(x)        # each neighbor is
                                 a pair (s,m) where
                                 s is the solution,
                                 and m is the
                                 move that turns
                                 x -> s.
    x , move = the pair (s,m) in
               [neighbors such that m is
                not a key in "taboo" or
               taboo[m] ≤ generation]
               with the highest score
```

taboo[move] = generation + taboo_tenure

Advanced Topics:

* Sometimes using just the "move" as the taboo is too restrictive. In this case, you can try keeping the taboo list in terms of (move, score). So you only prevent a move if it would lead to the same score you had the last time you did the move.

* Aspiration Criteria: You can decide ahead of time to ignore the taboo list under certain circumstances, e.g., if the new solution is the best you've ever seen.

* If neighborhoods are too large:
  (1) change the tweak function, possibly allowing solutions that violate the constraints

Ex: Knapsack

old = zero or 1 items out and zero or 1 items in

$O(n^2)$

new = add 1 item OR remove 1 item

$O(n)$

might allow solutions over capacity, so penalize them

(2) instead of generating the whole neighborhood, just generate K random tweaks and pick the best of those.