# Lecture02a: Local Search

CS540 1/23/18

## Announcements

Off-campus students:
- Double check your access to cs540 on Canvas
- This is your access point for videos

Everyone (on-campus and off)
- Send a message to cs540@cs.colostate.edu (if you haven't already)
- This will get you added to Piazza

The Piazza site is up
- Accessible through class web site
- First reading discussion has already begun

Lecture notes go up on class web site (progress page) after class
- Ping me if I forget

First reading assignment: R. Qu, et. al. *A Survey of Methodologies and Automated System Development for Examination Timetabling,* Journal of Scheduling, 12(1):55-89.

## Local Search (General Idea)

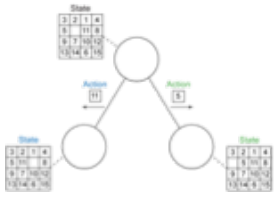
Goal: find a termination state
- Example: SAT (find a mapping of variables to {T,F})
- Counter-example: Path planning (output is a sequence of states)

Method
- Start with an initial state
- On every step:
  - Evaluate "neighboring" states (small variations of current state)
  - Move to neighboring state with better evaluation score
- Variations on strategy
  - First improvement
  - Steepest ascent
- Variations on evaluation
  - Evaluation function is the function to be optimized
  - Evaluation function is a cheaper heuristic.

Constraints
- Finite memory
- Classic form: "memory-less"
- Modifications: finite (but non-zero) memory

## Local Search Visualization

Visualize as a graph
- Elements of $S(\pi)$ are nodes in a graph
- Neighborhood function defines edges between nodes



https://www.ibm.com/developerworks/library/j-ai-search/

Example: 15-tile problem
- States: tile configurations
- Goal: tiles in order
- Neighborhood: slide 1 tile into open space
- Evaluation: ?

In 3D: height = evaluation

## Local Search Definition

For a problem instance π:
- Search space: $S(\pi)$
- Solution set: $S'(\pi) \subseteq S(\pi)$
- Neighborhood relation: $N(\pi) \subseteq S(\pi) \times S(\pi)$
- Set of memory states: $M(\pi)$
  - $M(\pi) \subseteq S(\pi)$
  - $|M(\pi)| = k, k \in \mathbb{Z}^+$
- Initialization function: $\emptyset \rightarrow D(S(\pi) \times M(\pi))$
  - May be stochastic or fixed
- Step function: $S(\pi) \times M(\pi) \rightarrow D(S(\pi) \times M(\pi))$
  - $S_s(\pi) \times M_s(\pi) \rightarrow D(S_d(\pi) \times M_d \subseteq \{M_s \cup S_s(\pi)\})$
  - $(S_s(\pi), S_d(\pi)) \in N(\pi)$
- Evaluation function: $H(s \in S(\pi)) \rightarrow \mathbb{R}$
- Termination predicate E: $S(\pi) \times M(\pi) \rightarrow D(F, T)$

## Local Search Terminology

Local optima: state without improving or equal-value neighbors

Plateau: state with no improving neighbors, but neighbors of equal value

Global optima: local optima or plateau such that no other local optima or plateau is better

Basin of attraction: set of states that lead to the same local optima (via hill-climbing)

## Discussion

Before discussing search strategies, let's admit…

Its all about the representation

- The state space must describe all possible answers
  - Does it have to be discrete?
  - If discrete, does it have to be finite?
- What if the goal is a path, not a destination?

- The neighborhood function is an art form
  - Too large -> too slow
  - Too small -> too slow
  - Too regular -> failure (falls into the same traps over and over)

- The evaluation function drives the search
  - Needs to reflect small improvements, even far from goal
  - Needs to be quick to evaluate

## Classic Local Search

No memory, i.e. $|M(\pi)| = 0$

Random initialization

- $s \leftarrow RandomElement(S(\pi)) \times \emptyset$

Update

- $s' \leftarrow \max_{s_d:(s,s_d)\in N(\pi)} H(s_d)$
- If $H(s') > H(s), s \leftarrow s'$

Until

- $E(s) = T$ or $H(s') \leq H(s)$

## Discussion: SAT Example

(A v ~B) ^ (B v ~C) ^ (C v ~A) ^ (~A v ~B v ~C)

What is the search space?

What is the evaluation function?

Assume the neighborhood changes one variable
Does local search work? Why or why not?

## Local Search Variations

Random Plateau Walk

- If no neighbor is an improvement (i.e. ∄ H(s') > H(s))
- But there is an equally good neighbor (i.e. ∃ H(s') = H(s))
- And plateau ctr < threshold
  - Increment plateau ctr,
  - s ← s'

Randomized Improvement

- $S' = \{s_d : (s, s_d,) \in N(\pi)\ \&H(s_d) > H(s)\}$
- $s \leftarrow RandomElement(S')$

(Randomized) First Improvement

- Generate neighbors in random order
- Update state to first neighbor with higher H score

## Discussion: Path Finding

What if the goal is the path to a final state?

Can we still do local search? How?

What is the cost?
In time?
In space?

## Randomized Iterative Improvement

*With some fixed probability, choose randomly from the neighborhood. Otherwise, choose improving move.*

Determine initial candidate solution s

While termination condition is not satisfied:

- With probability p:
  - Choose a neighbor s' of s randomly from neighborhood
- Otherwise (will occur with probability 1-p):
  - Choose s' from neighborhood of s such that g(s') > g(s)
  - If no such s' exists, pick s' with maximum g(s')
- s = s'

## Randomized Iterative Improvement

Random initialization
- $s \leftarrow RandomElement(S(\pi))$×∅
- Plateau_ctr ←0

Update
- p ← RandomNumber[0,1]
- $S' = \{s_d : (s, s_d,) \in N(\pi)\}$
- If (p < pthresh)
  - s ← RandomElement(S')
- Else
  - $s' \leftarrow \max_{s'} H(s_d)$
  - If H($s'$) > $H(s)$
    - s ←s'
    - Plateau_ctr ←0
  - Else if (H(s') = H(s)
    - If Plateau_ctr >= PlateauThreh, STOP
    - Else
      - s ←s'
      - Plateau_ctr++
  - Else STOP

## Simulated Annealing

Based on randomized iterative improvement

Still uses no memory

EXCEPT the probability of a random move is a function of time
- Probability of random move is called *temperature*
- Temperature begins very high (near 1)
- Over time, temperature is reduced
  - 'annealing schedule'

AND EXCEPT stop conditions go away
- Run for a fixed number of moves

## Discussion

Does simulated annealing make any sense?

Is simulated annealing complete?

When is simulated annealing appropriate?

## Adding Memory

Add just one state of memory ("best so far")

Random Restart Local Search
- Run local search from random start position
- Save result as "best so far"
- Run local search N more times
- After each, if result is better than "best so far", replace

Note: any local search strategy will do
- Generally run lots of fast searches