Lecture02a: Local Search

CS540 1/29/19

Announcements

Programming Assignment #1 is due one week from today
Multiple questions asked/answered on Piazza
Other questions?
If you haven’t subscribed to the Piazza class board yet, do
Only 27 have joined as of last night

Last Lecture

On Thursday, we reviewed basic search terms
- From CS440 and Russell & Norvig

Concentrating on tree-based methods
- Keep a tree of expanded nodes
- Produces both solution (goal node) and path from root
- Infeasible for large search spaces with long solution paths

Correction: I mis-used the term “Greedy Search”
- Expanding the cheapest node in a tree is a form of best-first search
- So what really is Greedy Search?

Local Search (General Idea)

Goal: find a termination state
- Example: SAT (find a mapping of variables to {T,F})
- Example: Max-SAT (find the mapping of variables to {T,F} that satisfies as many terms as possible)
- Counter-example: Path planning (output is a sequence of states)

Method
- Start with an initial state
- On every step:
  - Evaluate “neighboring” state (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

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 $\pi$:
- 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: $M(\pi)$
- $M(\pi) \subseteq S(\pi)$
- Initial state: $s(0) \in S(\pi)$
- Initialization function: $0 \rightarrow D(0(\pi) \times M(\pi))$
- Step function: $S(\pi) \times M(\pi) \rightarrow D(S(\pi) \times M(\pi))$
- $s_{1}(\pi), s_{2}(\pi) \in S(\pi)$
- $L_1(\pi) = \{ s_{1}(\pi) \}$
- $L_2(\pi) = \{ s_{2}(\pi) \}$
- $L_3(\pi) = \{ M(\pi) \}$
- Evaluation function: $E(\pi) \rightarrow \mathbb{R}$
- Termination predicate $E: S(\pi) \times M(\pi) \rightarrow \{ T,F \}$
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...

It’s 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
◦ every step is expensive
◦ Too small: too slow, poor answers
◦ too many steps to get to local optima
◦ Too many local optima
◦ 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(M)| = 0

Random initialization

\[ s \leftarrow \text{RandomElement}(S(s)) \times \emptyset \]

Update

\[ s' \leftarrow \max_{s \in S(s)} H(s) \]

If \( H(s') > H(s) \), \( s \leftarrow s' \)

Until

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

Discussion: SAT Example

\( (A \lor \neg B) \land (B \lor \neg C) \land (C \lor \neg A) \land \neg(A \lor B \lor \neg C) \)

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. \( \exists \ H(s') > H(s) \))
◦ But there is an equally good neighbor (i.e. \( \exists \ H(s') = H(s) \))
◦ And plateau ctr < threshold
◦ Increment plateau ctr,
◦ \( s \leftarrow s' \)

Randomized Improvement

\[ S' = \{ s' : (s, s') \in H(r) \land H(s') > H(s) \} \]

\( s \leftarrow \text{RandomElement}(S') \)

(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 \( g(s') \)
  - \( s = s' \)

Random initialization
- \( s \leftarrow \text{RandomElement}(S) \times 0 \)
- \( \text{Plateau}_\text{ctr} \leftarrow 0 \)

Update
- \( p \leftarrow \text{RandomNumber}[0,1] \)
- \( S' = \{ x \in X \mid x \neq s \} \)
- If \( p < p_{\text{thresh}} \):
- Else:
- \( s' \leftarrow \text{RandomElement}(S') \)
- If \( H(s') > H(s) \):
- Else:
- \( s' = s \)
- \( \text{Plateau}_\text{ctr} = 0 \)
- If \( H(s') = H(s) \):
- If \( \text{Plateau}_\text{ctr} \geq \text{PlateauThresh} \), STOP
- Else:
- \( s = s' \)
- \( \text{Plateau}_\text{ctr}++ \)
- 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