CS540 Introduction to Artificial Intelligence
Lecture 17

Young Wu
Based on lecture slides by Jerry Zhu and Yingyu Liang

July 25, 2019

Search Algorithms

- Breadth-First, BiDirectional
- Depth-First, Iterative Deepening
- Uniform Cost
- Best First Greedy
- A (or $A^{\star}$), Iterative Deepening A, Beam

uninformed search

informed search

Iterative Deepening A Star Search

Discussion

$$\text{initial} \xrightarrow{g(s)} s \xrightarrow[\geq\, h(s)]{h^*(s)} \text{goal}$$

- $A^{\star}$ can use a lot of memory.
- Do path checking without expanding any vertex with $g(s) + h(s) > 1$.
- Do path checking without expanding any vertex with $g(s) + h(s) > 2$.
- ...
- Do path checking without expanding any vertex with $g(s) + h(s) > d$.

Iterative Deepening A Star Search Performance

Discussion

- IDA* is complete.
- IDA* is optimal.
- IDA* is more costly than $A^{\star}$.

↑ Time complexity
↓ Space complexity

Beam Search

Discussion

→ reduces space complexity

- Version 1: Keep a priority queue with fixed size k. Only keep the top k vertices and discard the rest.
- Version 2: Only keep the vertices that are at most ε worse than the best vertex in the queue. ε is called the beam width.

$\varepsilon = 2$

1 2 3 ~~4 5 6~~

$> 1 + \varepsilon = 3$

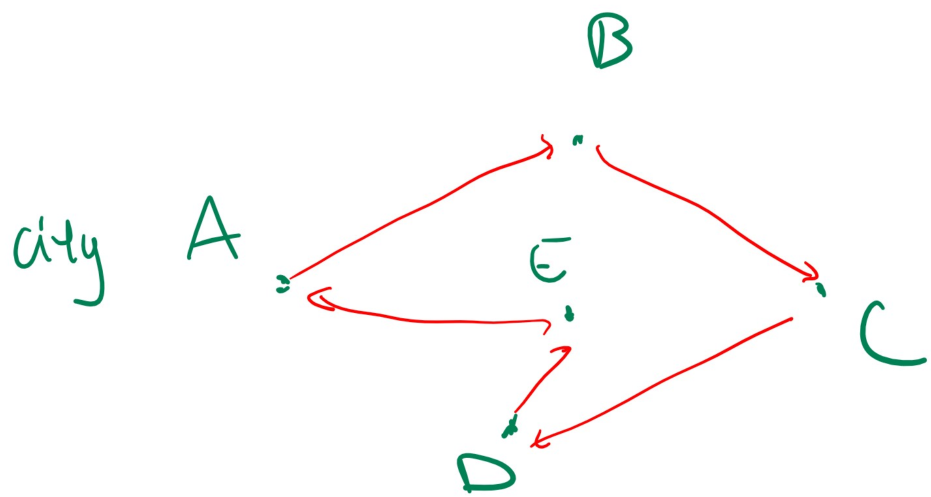
Beam Search Performance

Discussion

- Beam is incomplete.
- Beam is not optimal.

Traveling Salesperson Example

Motivation



tour ABCDEA

cost = total distance.

all states are solutions

→ find best state.

optimization.

Search vs. Local Search

Motivation

- Some problems do not have an initial state and a goal state.
- Every state is a solution. Some states are better than others, defined by a cost function (sometimes called score function in this setting), $f(s)$.
- The search strategy will go from state to state, but the path between states is not important.
- There are too many states to enumerate, so standard search through the state space methods are too expensive.

Local Search

Motivation

- Local search is about searching through a state space by iteratively improving the cost to find an optimal or near-optimal state.
- The successor states are called the neighbors (sometimes move set).
- The assumption is that similar (nearby) solutions have similar costs.

Local Search Application

Motivation

- Optimization problems (gradient descent methods are all local search methods)
- Traveling salesman
- Boolean satisfiability (SAT)
- Scheduling

discrete

Boolean Satisfiability Example, Part I

Quiz (Graded)

or not

- Assume all variables A, B, C, D, E are set to True. How many of the following clauses are satisfied?
- A: $A \vee \neg B \vee C$ T v F v T = T
- B: $\neg A \vee C \vee D$
- C: $B \vee D \vee \neg E$
- D: $\neg C \vee \neg D \vee \neg E$
- E: $\neg A \vee \neg C \vee E$

Choose A, B, C, D, E T, F

max # of clauses satisfied

T

Boolean Satisfiability Example, Part II

Quiz (Graded)

Q3

- Assume all variables A, B, C, D, E are set to True. Which one of the variables should changed to False to maximize the number of clauses satisfied?
  - $A \vee \neg B \vee C$ ✓
  - $\neg A \vee C \vee D$ ✓
  - $B \vee D \vee \neg E$ ✓
  - $\neg C \vee \neg D \vee \neg E$ ✓
  - $\neg A \vee \neg C \vee E$ ✓

C = F

D = F

C

D

Hill Climbing (Valley Finding)

Description

- Start at a random state.
- Move to the best neighbor state (one of the successors).
- Stop when all neighbors are worse than the current state.
- The idea is similiar to gradient descent.

Hill Climbing

Algorithm

- Input: state space S and cost function f.
- Output: $s^\star \in S$ that minimizes $f(s)$.
- Start at a random state $s_{0.}$
- At iteration t, find the neighbor that minimizes f.

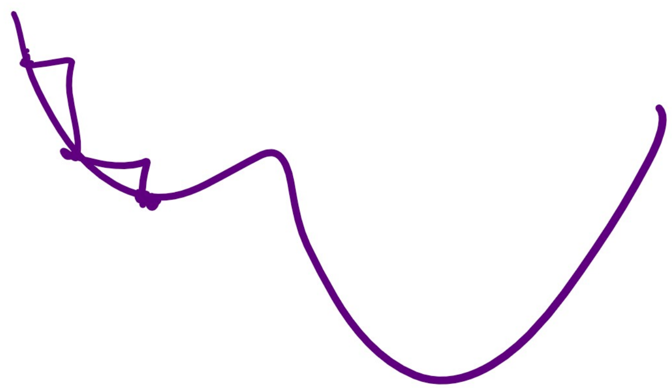
$$s_{t+1} = \arg \min_{s \in s'(s_t)} f(s)$$

- Stop when none of the neighbors have a lower cost.

$$\text{stop if } f(s_{t+1}) \leqslant f(s_t)$$

Hill Climbing Performance

Discussion



escape local minimum?

- It does not keep a frontier, so no jumping and no backtracking.
- It is simple, greedy, and stops at a local minimum.

Random Restarts

Discussion

- A simple modification is picking random initial states multiple times and finding the $\underbrace{\text{best among the local minima.}}_{\approx \text{ global min}}$

First Choice Hill Climbing

Discussion

αDf — GD

— hill climbing

s s'

- If there are too many neighbors, randomly generate neighbors until a better neighbor is found.
- This method is called first choice hill climbing.

Walk SAT Example

Discussion

ABCDE

Start with random assignment → Only SAT

- Pick a random unsatisfied clause.
- Select and flip a variable from that clause:

hill climbing

small

1. With probability p, pick a random variable.
2. With probability $1 - p$, pick the variable that maximizes the number of satisfied clauses.

also costly

try to escape local min

- Repeat until the solution is found.
- Walk SAT uses the idea of stochastic hill climbing.

Simulated Annealing

Description

- Each time, a random neighbor is generated.
- If the neighbor has a lower cost, move to the neighbor.
- If the neighbor has a higher cost, move to the neighbor with a small probability.
- Stop until bored.
- It is a version of Metropolis-Hastings Algorithm.

Acceptance Probability

Definition

- The probability of moving to a state with a higher cost should be small.

1. Constant: $p = 0.1$
2. Decreases with time: $p = \frac{1}{t}$
3. Decreases with time and as the energy difference increases:

$$p = \exp\left(-\frac{|f(s') - f(s)|}{\text{Temp}\ (t)}\right)$$

f(s') f(s) f(s) f(s')

cost difference

decrease in t

- The algorithm corresponding to the third idea is called simulated annealing. Temp should be a decreasing in time (iteration number).

Temperature

Definition

- Temp represents temperature which decreases over time. For example, the temperature can change arithmetically or geometrically.

$$\text{Temp }(t+1) = \max\{\text{ Temp }(t) - 1, 1\}, \ \text{Temp }(0) = \text{ large}$$
$$\text{Temp }(t+1) = 0.9 \text{ Temp }(t), \ \text{Temp }(0) = \text{ large}$$

- High temperature: almost always accept any s'.
- Low temperature: first choice hill climbing.

Simulated Annealing

Algorithm

- Input: state space S, temperature function Temp , and cost function f.
- Output: $s^\star \in S$ that minimizes $f(s)$.
- Start at a random state $s_{0.}$
- At iteration t, generate a random neighbor s', and update the state according to the following rule.

$$s_{t+1} = \begin{cases} s' & \text{if } f(s') > f(s_t) \\ s' & \text{with probability } \exp\left(-\dfrac{|f(s') - f(s_t)|}{\text{Temp}(t)}\right) \\ s_t & \text{otherwise} \end{cases}$$

Simulated Annealing Performance

Discussion

random restart

- Use hill-climbing first.
- Neighborhood design is the most important.
- In theory, with infinitely slow cooling rate, SA finds global minimum with probability 1.

infinite # times