CS540 Introduction to Artificial Intelligence
Lecture 17

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

July 22, 2019

Search Algorithms

optimal if c = 1

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

optimal

IDA*

Iterative Deepening A Star Search

Discussion

- $A^{\star}$ 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 $g(s) + h(s) > d$.

$g+h \leq 1$

$g+h \leq 2$

$g+h \leq 3$

$\vdots$

Iterative Deepening A Star Search Performance

Discussion

- IDA* is complete.
- IDA* is optimal.
- IDA* is more costly than A^*.

goal with least cost (shortest path)

Time ↑

Space ↓

Beam Search

Discussion

- 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.

	A	B	C	~~D~~	~~E~~
g+h	1	2	3	~~4~~	~~5~~

$\varepsilon = 2$ if see $g+h > 3$ → do not enqueue

⇒ Small queue during search.

Beam Search Performance

Discussion

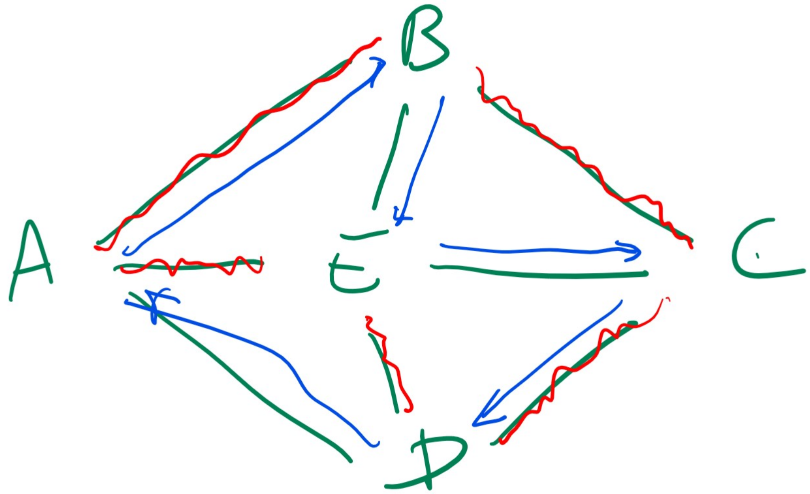
- Beam is incomplete.
- Beam is not optimal.

Traveling Salesperson Example

Motivation

goal $A \xrightarrow{BCDE} A$

tour : $\underbrace{ABCDEA}_{\text{State}}$



State Space too large,

ADCEBA ← always a solution

may not be the best

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), $f(s)$.

local Search

$= g(s) + h(s) \rightarrow$ Search

- 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.

neighbors

gradient descent

→ cost (w)

Local Search Application

Motivation

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

Boolean Satisfiability Example, Part I

Quiz (Graded)

Q2 ABCE

- Assume all variables A, B, C, D, E are set to True. How many of the following clauses are satisfied?

OR not

- A: $A \vee \neg B \vee C$
- B: $\neg A \vee C \vee D$
- C: $B \vee D \vee \neg E$
- D: ~~$\neg C \vee \neg D \vee \neg E$~~
- E: $\neg A \vee \neg C \vee E$

SAT

max # of clauses satisfied

return True.

Boolean Satisfiability Example, Part II

Quiz (Graded)

Q4 CD

- Assume all variables A, B, C, D, E are set to True. Which ones 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 ✓

D ✓

A B C D E

A B C ¬D E

A B ¬C D E

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.

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

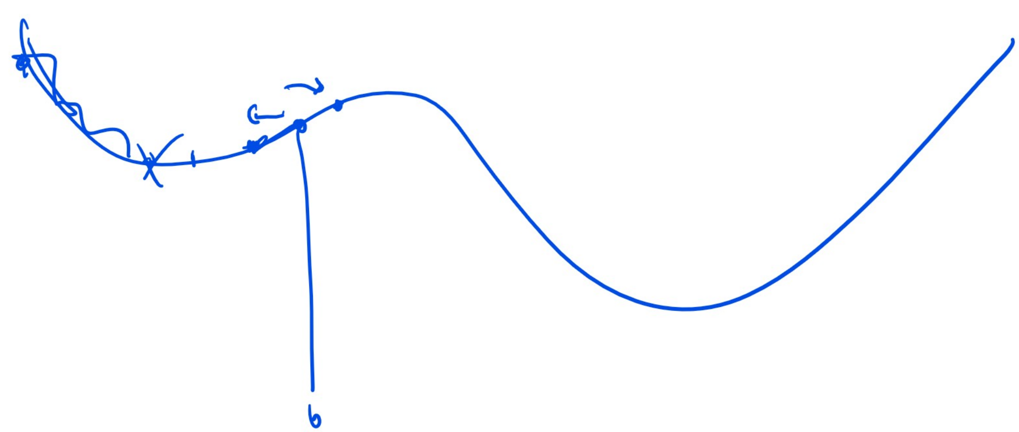
- Stop when none of the neighbors have a lower cost.

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

Hill Climbing Performance

Discussion

- 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 best among the local minima.

First Choice Hill Climbing

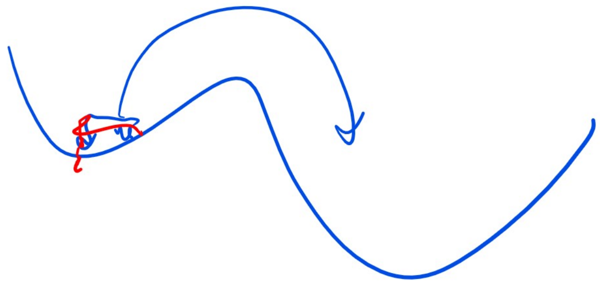
Discussion

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

not for jump out of local min

Walk SAT Example

Discussion



$p = 0.1$

$A\ B\ C\ D\ E = T$

$\neg B \vee \neg C \vee \neg D$

- Pick a random unsatisfied clause.
- Select and flip a variable from that clause:
  1. With probability p, pick a random variable. avoid local min → $\frac{1}{3}B, \frac{1}{3}C, \frac{1}{3}D$
  2. With probability $1 - p$, pick the variable that maximizes the number of satisfied clauses. → check # of clauses satisfied

$B \to F$

$C \to F$

$D \to F$

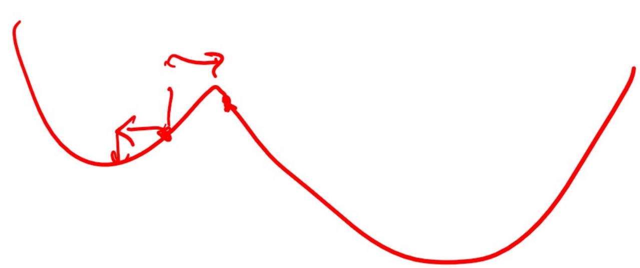
- Repeat until the solution is found.
- Walk SAT uses the idea of stochastic hill climbing.

Simulated Annealing

Description

→ similar to first choice hill climbing

- 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. → jump out of local min.
- Stop until bored.
- It is a version of Metropolis-Hastings Algorithm.



Acceptance Probability

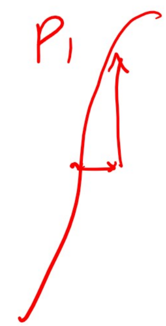
Definition

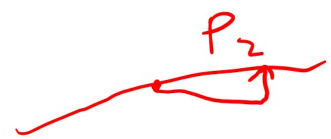
- The probability of moving to a state with a higher cost should be small.

1. Constant: $p = 0.1$
2. Decreases with time: $p = \frac{1}{t}$

cost difference

3. Decreases with time and as the ~~energy difference~~ increases:
$$p = \exp\left(-\frac{|f(s') - f(s)|}{\cancel{t}\ \text{Temp}}\right)$$





- The algorithm corresponding to the third idea is called simulated annealing.

$P_2 > P_1$

Temperature

Definition

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

Temp

- ~~The t in the above expression does not have to be time.~~
- It can represent temperature which decreases over time. For example, the temperature can change geometrically.

T

$$t = 0.9t$$

$T = 0.9, 0.8, 0.7$ $\quad$ $T = \frac{1}{1}, \frac{1}{2}, \frac{1}{3}, \dots$

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

Simulated Annealing

Algorithm

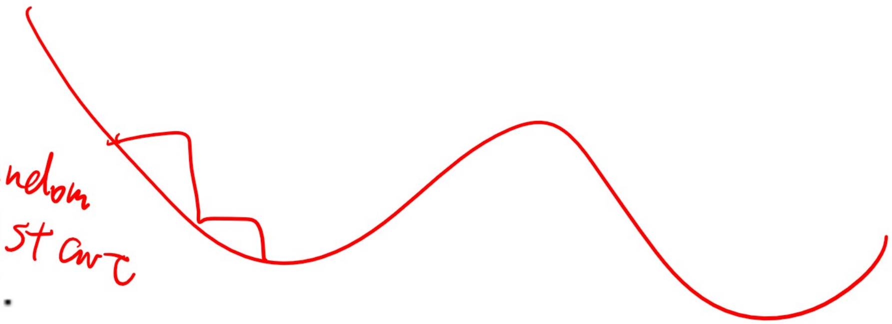
- Input: state space S, temperature function Temp , and cost function f.
- Output: $s^\star \in S$ that minimizes $f(s)$.
- Start at a random state $s_{0.}$
- At iteration t, generate a random neighbor s', and update the state according to the following rule.

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

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.

Genetic Algorithm

Description

- Start with a fixed population of initial states.
- Find the successors by:

1. Cross over.
2. Mutation.

Reproduction Probability

Definition

- Each state in the population has probability of reproduction proportional to the fitness. Fitness is the opposite of the cost: higher cost means lower fitness. Use F to denote the fitness function, for example, $F(s) = \dfrac{1}{f(s)}$ is a valid fitness function.

$$p_i = \frac{F(s_i)}{\sum_{j=1}^{N} F(s_j)}, i = 1, 2, ..., N$$

- A pair of states are selected according to the reproduction probabilities (using CDF inversion).

Cross Over

Definition

- The states need to be encoded by strings.
- Cross over means swapping substrings.
- For example, the children of 10101 and 01010 could be the same as the parents or one of the following variations.

$$(11010, 00101), (10010, 01101)$$
$$(10110, 01001), (10100, 01011)$$

1 0101
0 1010

10 101
01 010

Mutation

Definition

- The states need to be encoded by strings.
- Mutation means randomly updating substrings. Each character is changed with small probability q, called the mutation rate.
- For example, the mutated state from 000 could stay the same or be one of the following.

$$\text{one of } 001, 010, 100, \text{ with probability } q\left(1-q\right)^{2}$$
$$\text{one of } 011, 101, 110, \text{ with probability } q^{2}\left(1-q\right)$$
$$\text{and } 111, \text{ with probability } q^{3}$$

Cross Over, Modifications

Definition

1 0 1 0 1
0 1 0 1 0

2-point cross over

- The previous cross over method is called 1 point cross over.
- It is also possible to divide the string into N parts. The method is called N point cross over.
- It is also possible to choose each character from one of the parents randomly. The method is called uniform cross over.

1 0 1 0 1
0 1 0 1 0

1 50% 0 50%
0 50% 1 50% — ...

1 0 0 1 1 1
0 1 1 1 1

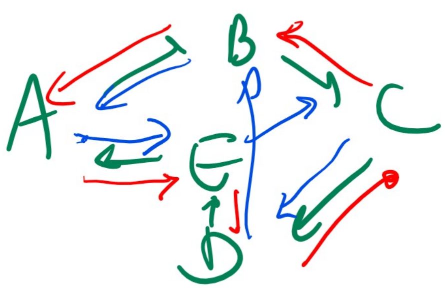
Mutation, Modifications

Definition

- For specific problems, there are ways other than flipping bits to mutate a state.

1. Two-swap: ABCDE to EBCDA
2. Two-interchange: ABCDE to EDCBA

TSP :



A B C D E A

swap A E C D B A

interchange A E D C B A

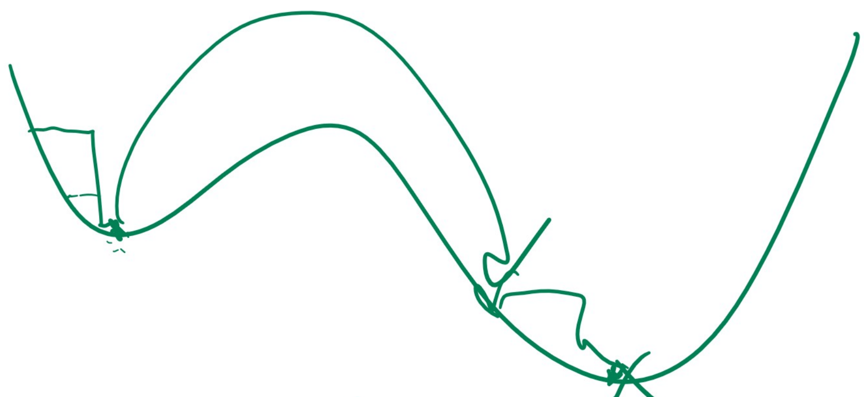
A B C D E F G H I

G F E D

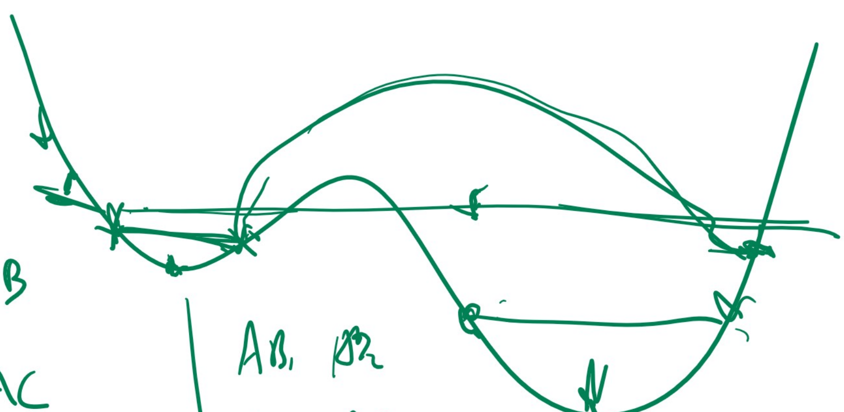
Genetic Algorithm SAT Example

Discussion

0 1 0 1 0



T ↓



000

010

A

B

C

AB

AC

AB_1 AB_2

AC_1 AC_2

Genetic Algorithm TSP Example

Discussion

0 1   0 0
1 0   1 1
1 1   1 1
          0 1

Fitness Example

Quiz (Graded)

population of 5

- Fall 1999 Final Q5
- Which ones (multiple) of the following states have the highest reproduction probability?

max   not cost

- The fitness function is $5A + 3BC - D + 2E$. $\Rightarrow F(s)$

(A, B, C, D, E)

A B C D E

- A: $(1, 1, 0, 1, 1)$ $5 + 0 - 1 + 2 = 6$ $\frac{6}{27}$
- B: $(0, 1, 1, 0, 1)$ 5
- C: $(1, 1, 0, 0, 0)$ 5
- D: $(1, 0, 1, 1, 1)$ $5 + 0 - 1 + 2 = 6$ $\frac{5}{27}$
- E: $(1, 0, 0, 0, 0)$ 5

$$p = \frac{F(s)}{\sum F(s')}$$

pick parent 1 → $\frac{6}{27}$ prob A $\frac{5}{27}$ prob B ......

Genetic Algorithm, Part I

Algorithm

- Input: state space S represented by strings s and cost function f or fitness function F.
- Output: $s^\star \in S$ that minimizes $f(s)$.
- Randomly generate N solutions as the initial population.

$$s_1, s_2, \ldots, s_N$$

- Compute the reproduction probability.

$$p_i = \frac{F(s_i)}{\sum_{j=1}^{N} F(s_j)}, i = 1, 2, \ldots, N$$

Genetic Algorithm, Part II

Algorithm

- Randomly pick two states according to p_i, say s_a, s_b. Randomly select a cross over point c, swap the strings.

$$s'_a = s_a\left[0...c\right) s_b\left[c...m\right)$$
$$s'_b = s_b\left[0...c\right) s_a\left[c...m\right)$$

- Randomly mutate each position of each state s_i with a small probability (mutation rate).

$$s'_i\left[k\right] = \begin{cases} s_i\left[k\right] & \text{with probability } 1-q \\ \text{random} & \text{with probability } q \end{cases}, k = 1, 2, ..., m$$

- Repeat with population s'.

Variations

Discussion

F(s)	6	5	5	6	5
ranking	4	1	1	4	1
prob	$\frac{4}{11}$	$\frac{1}{11}$	---		

- Parents can survive.
- Use ranking instead of $F(s)$ to compute reproduction probabilities.
- Cross over random bits instead of chunks.

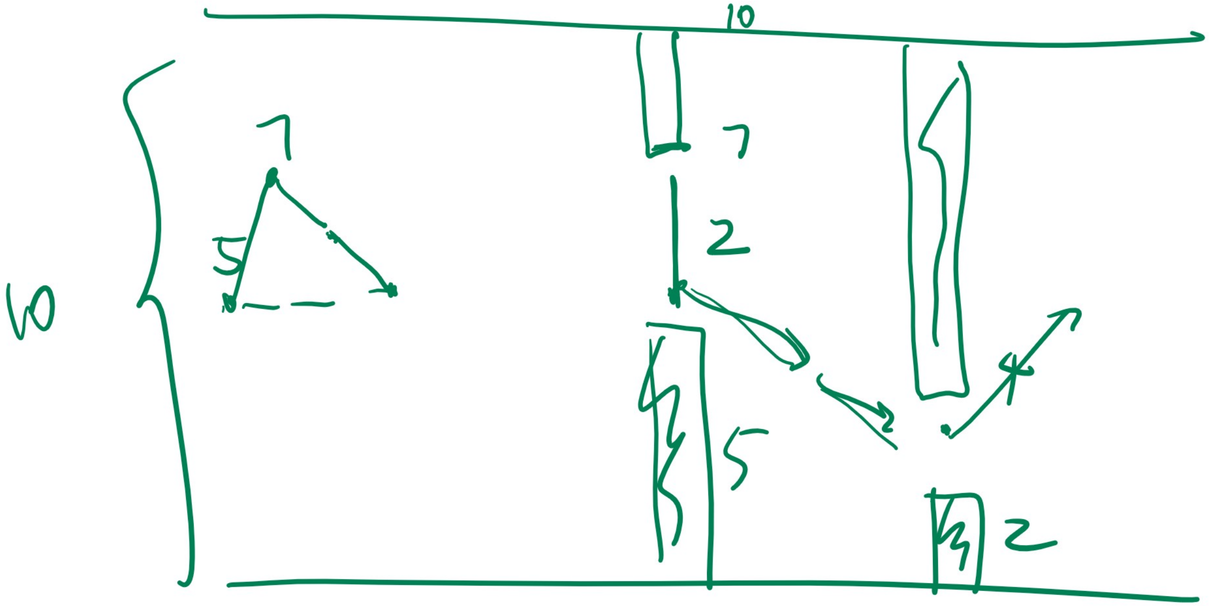
uniform cross over

Genetic Algorithm Performance

Discussion

with random restart

- Use hill-climbing first.
- State design is the most important.
- In theory, cross over is much more efficient than mutation.



0 0 0 5 0 2

time 0 1 2 3

x,	y
11	4
2	1

Score = # times it's flying.