Local Search

Alice Gao

Lecture 6

Readings: RN 4.1, PM 4.7 - 4.8

Outline

Learning Goals

By the end of the lecture, you should be able to

- Describe the advantages of local search over other search algorithms.
- Formulate a real world problem as a local search problem.
- Verify whether a state is a local/global optimum.
- Describe strategies for escaping local optima.
- Trace the execution of greedy descent, greedy descent with random restarts, simulated annealing, and genetic algorithms.
- Compare and contrast the properties of local search algorithms.

Why use local search?

So far, the search algorithms explore the space systematically and keep track of one or more paths.

- The search space can be too big or infinite.
- For solving CSPs, the path to a goal is irrelevant.

Solution: local search

Properties of local search

- Explores only a portion of the search space.
- Requires little memory.

Advantages:

- Can find solutions quickly on average.
- Works for CSPs and general optimization problems.

Disadvantages:

- No guarantee that a solution will be found if one exists. Cannot prove that no solution exists.

What is local search?

- Start with a complete assignment of values to variables.
- Take steps to improve the solution iteratively.

Local Search

A local search problem consists of:

- A state : a complete assignment to *all* of the variables.
- A neighbour relation: which states do I explore next?
- A cost function: how good is each state?

4-Queens Problem as a Local Search Problem

Variables: x_0, x_1, x_2, x_3 where x_i is the row position of the queen in column i. Assume that there is one queen per column. $i \in \{0, 1, 2, 3\}$.

Domains: $x_i \in \{0, 1, 2, 3\}$ $\forall i$.

Initial state: 4 queens on the board in random row positions.

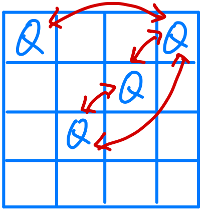
Goal state: 4 queens on the board.
No pair of queens are attacking each other.

Neighbour relation:

- Version A: move a single queen to a different row in the same column.
- Version B: swap the row positions of two queens.

4-Queens Problem as a Local Search Problem

Cost function: the number of pairs of queens attacking each other, directly or indirectly.



cost = 4

Greedy descent

If a neighbour is an improvement,
then move to the best neighbour.

Greedy: does not look beyond the immediate neighbours.

a.k.a. hill climbing or greedy ascent.

- Start with a random state.
- Move to a neighbour with the lowest cost if it's better than the current state.
- Stop when no neighbour has a lower cost than current state.
- Only remembers the current node.

requires very little memory.

If no neighbour is an improvement, stop!
current state is one of the best states in the neighbourhood.

Greedy descent in one sentence

① descend : minimize cost.
move to a neighbour w/ a lower cost.

② thick fog : can only see immediate neighbours.

① ② ③

Descend into a canyon in a thick fog with amnesia

③ amnesia : no memory of where we've been.
may stumble on the same state multiple
times w/ realizing it.

Properties of Greedy Descent

- Performs quite well in practice.
  Makes rapid progress towards a solution.

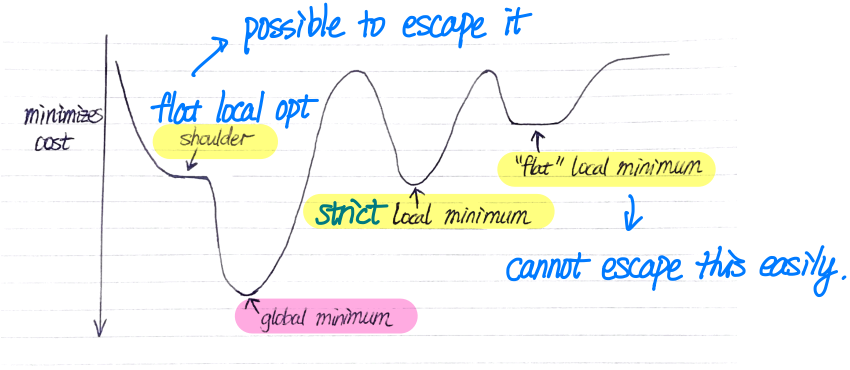
the state w/ lowest cost
among all the states.

- Given enough time,
  will greedy descent find the global optimum?

No. Greedy descent can get
stuck at local optimums.

Where can Greedy Descent get stuck?

- Local optimum: No neighbour has a (strictly) lower cost.
- Global optimum: A state that has the lowest cost among all the states. (a special type of local optimum.)



CQ: Local and global optimum (1)

CQ: Consider the following state of the 4-queens problem. Consider neighbour relation B: swap the row positions of two queens. Which of the following is correct?

state : 3201 (2)
2301 (4)
0231 (1)
1203 (1)

0			Q	
1				Q
2		Q		
3	Q			

3021 (1)
3102 (1)
3210 (6)

(A) This state is a local optimum and is a global optimum.

(B) This state is a local optimum and is NOT a global optimum.

(C) This state is NOT a local optimum and NOT a global optimum.

CQ: Local and global optimum (2)

CQ: Consider the following state of the 4-queens problem. Consider neighbour relation A: move a single queen to another square in the same column. Which of the following is correct?

2	2	Q	4
3	3	4	Q
3	Q	3	2
Q	3	3	2

cost = 2

(A) This is a local optimum and is a global optimum.

(B) This is a local optimum and is NOT a global optimum.

(C) This is NOT a local optimum and NOT a global optimum.

Escaping flat local optimums

stop after a # of consecutive sideway moves

- Sideway moves: allow the algorithm to move to a neighbour that has the same cost.
- Tabu list: keep a small list of recently visited states and forbid the algorithm to return to those states.

  some short term memory.

Performance of Greedy Descent with sideway moves

8-queens problem: ≈ 17 million states.

- Greedy descent

  % of instances solved: 14%
  # of steps until success/failure: 3-4 steps on average until success or failure.

- Greedy descent $+ \leq 100$ consecutive sideway moves:

  % of instances solved: 94%
  # of steps until success/failure: 21 steps until success and 64 steps until failure.

Random restarts and random walks

Greedy descent can get stuck at a local optimum (that is not a global optimum). What can we do?

- Random restarts:
  restart search in a different part of the space.
  Example: Greedy descent with random restarts

- Random walks:
  move to a state with a higher cost occasionally.
  Example: Simulated annealing

Random restarts vs random walks

(1) Which type of randomization is better for landscape (a) ?

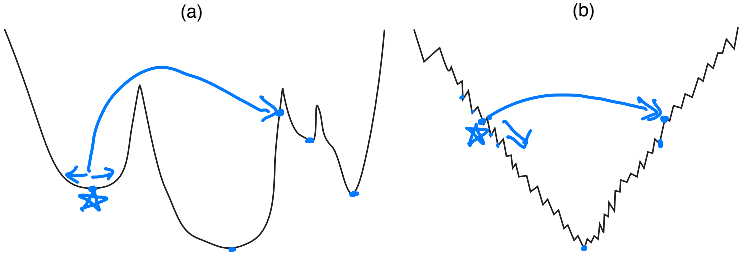
(A) Random restarts — a global random move.

(B) Random walks

(2) Which type of randomization is better for landscape (b) ?

(A) Random restarts

(B) Random walks — a local random move.



Greedy descent with random restarts

If at first you don't succeed, try, try again.

- Performs multiple greedy descents from randomly generated initial states.
- Will greedy descent with random restarts find the global optimum?

It will find the global optimum w/ probability approaching 1. Eventually, it will generate the global optimum as the initial state.

So far...

Greedy descent focuses on optimization/exploitation,
whereas random moves allow us to explore the search space.

Can we combine exploration and optimization into one algorithm?

Simulated Annealing

- Annealing: slowly cool down molten metals to make them stronger.
- Start with a high temperature and reduce it slowly.
- At each step, choose a random neighbour.
  If the neighbour is an improvement, move to it.
  If the neighbour is not an improvement, move to the neighbour probabilistically depending on
  - the current temperature T
  - how much worse is the neighbour compared to current state

How likely do we move to a worse neighbour?

A is the current state and A' is the worse neighbour.
Let $\Delta C = cost(A') - cost(A)$. The current temperature is T.

We move to the neighbour A' with probability

$$e^{-\frac{\Delta C}{T}}$$

Gibbs distribution / Boltzmann distribution.

CQ: Probability of moving to a worse neighbour

CQ 1: A is the current state and A' is the worse neighbour.
Let $\Delta C = cost(A') - cost(A)$.

As T decreases, how does the probability of moving to the worse neighbour ($e^{-\frac{\Delta C}{T}}$) change?

(A) As T decreases, we are more likely to move to the neighbour.

(B) As T decreases, we are less likely to move to the neighbour.

$\Delta C > 0$, $T > 0$, $\frac{\Delta C}{T} > 0$. $T \downarrow$, $\frac{\Delta C}{T} \uparrow$, $-\frac{\Delta C}{T} \downarrow$, $e^{-\frac{\Delta C}{T}} \downarrow$

$\Delta C = 10$ $\quad T = 100 \quad e^{-10/100} = e^{-0.1} = 0.9$

$T = 10 \quad e^{-10/10} = e^{-1} = 0.36$

CQ: Probability of moving to a worse neighbour

CQ 2: A is the current state and A' is the worse neighbour.
Let $\Delta C = cost(A') - cost(A)$.

As ΔC increases (the neighbour becomes worse), how does the probability of moving to the worse neighbour ($e^{-\frac{\Delta C}{T}}$) change?

(A) As ΔC increases, we are more likely to move to the neighbour.

(B) As ΔC increases, we are less likely to move to the neighbour.

$\Delta C > 0,\ T > 0,\ \frac{\Delta C}{T} > 0.\quad \Delta C \uparrow,\ \frac{\Delta C}{T} \uparrow,\ -\frac{\Delta C}{T} \downarrow,\ e^{-\frac{\Delta C}{T}} \downarrow$

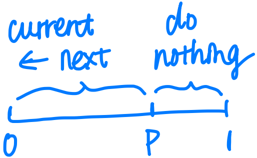
$T = 10 \quad \Delta C = 10 \quad e^{-10/10} = 0.36.$

$\Delta C = 100 \quad e^{-100/10} = 0.000045.$

Simulated Annealing Algorithm

Algorithm 1 Simulated Annealing

```
current ← initial-state
T ← a large positive value
while T > 0 do
    next ← a random neighbour of current
    ΔC ← cost(next) - cost(current)
    if ΔC < 0 then
        current ← next
    else
        current ← next with probablity p = e^(−ΔC/T)
    decrease T
return current
```



Annealing Schedule

How should we decrease T?

- In theory, we want to decrease the temperature very slowly.

If the temperature decreases slowly enough, simulated annealing is guaranteed to find the global optimum with probability approaching 1.

- In practice, a popular schedule is geometric cooling.

start w/ $T = 10$. multiply by 0.99 at each step.

Simulated annealing is like life...

Learning Goals

Introduction to Local Search

Local Search Algorithms

- Iterative Best Improvement
- Greedy descent with randomization
- Simulated Annealing
- Population-Based Algorithms

Revisiting the Learning goals

Beam Search

- Remember k states.
- Choose the k best states out of **all of the neighbors**.
- k controls space and parallelism.

1. What is beam search when $k = 1$? greedy descent
2. How is beam search different from k random restarts in parallel?
3. Are there problems with beam search?

2. w/ random restarts, each search is independent of the others.
for beam search, useful info is passed among parallel searches.

3. suffers from a lack of diversity among the k states.
can quickly become concentrated in a small region.

Stochastic Beam Search

- Choose the k states probabilistically.
- Probability of choosing a neighbour is proportional to its fitness.
- Maintains diversity in the population of states.
- Mimics natural selection.

Genetic Algorithm

stochastic beam search – asexual reproduction
genetic algorithm – sexual reproduction.

- Maintain a population of k states.
- Randomly choose two states to reproduce.
  Probability of choosing a state for reproduction is proportional to the fitness of the state.
- Two parent states crossover to produce a child state.
- The child state mutates with a small probability.
- Repeat the steps above to produce a new population.
- Repeat until the stopping criteria is satisfied.

Comparing greedy descent and genetic algorithm

- How do the algorithms explore the state space?

  Greedy descent generates neighbours of the state based on the neighbour relation.
  Genetic algorithm ... crosses over parent states randomly to produce a child state and mutates the child state.

- How do the algorithms optimize the quality of the population?

  Greedy descent moves to the best neighbour.
  Genetic algorithm ... chooses parent states probabilistically based on the fitness of the states.

Revisiting the Learning Goals

By the end of the lecture, you should be able to

- Describe the advantages of local search over other search algorithms.
- Formulate a real world problem as a local search problem.
- Verify whether a state is a local/global optimum.
- Describe strategies for escaping local optima.
- Trace the execution of greedy descent, greedy descent with random restarts, simulated annealing, and genetic algorithms.
- Compare and contrast the properties of local search algorithms.