Local Search

Alice Gao

Lecture 5

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.

Learning Goals

Introduction to Local Search

Local Search Algorithms

Revisiting the Learning goals

Why use local search?

So far, the search algorithms

- explore the space systematically.

Why use local search?

So far, the search algorithms

- explore the space systematically.

  What if the search space is large or infinite?

Why use local search?

So far, the search algorithms

- explore the space systematically.

  What if the search space is large or infinite?

- remember a path from the initial state.

Why use local search?

So far, the search algorithms

- explore the space systematically.

  What if the search space is large or infinite?

- remember a path from the initial state.

  What if we do not care about the path to a goal? e.g. CSP

Why use local search?

So far, the search algorithms

- explore the space systematically.

  What if the search space is large or infinite?

- remember a path from the initial state.

  What if we do not care about the path to a goal? e.g. CSP

Solution: local search

Properties of local search

- Does not explore the search space systematically.

Properties of local search

- Does not explore the search space systematically.

  Can find reasonably good states quickly on average.

Properties of local search

- Does not explore the search space systematically.

  Can find reasonably good states quickly on average.

  Not guaranteed to find a solution even if one exists.
  Cannot prove that no solution exists.

Properties of local search

- Does not explore the search space systematically.

  Can find reasonably good states quickly on average.

  Not guaranteed to find a solution even if one exists.
  Cannot prove that no solution exists.

- Does not remember a path to the current state.

Properties of local search

- Does not explore the search space systematically.

  Can find reasonably good states quickly on average.

  Not guaranteed to find a solution even if one exists.
  Cannot prove that no solution exists.

- Does not remember a path to the current state.

  Requires very little memory.

Properties of local search

- Does not explore the search space systematically.

  Can find reasonably good states quickly on average.

  Not guaranteed to find a solution even if one exists.
  Cannot prove that no solution exists.

- Does not remember a path to the current state.

  Requires very little memory.

- Can solve pure optimization problems.

What is local search?

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

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

4-Queens Problem as a Local Search Problem

State:

- 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.
- Domain for each variable: $x_i \in \{0, 1, 2, 3\}, \forall i$.

4-Queens Problem as a Local Search Problem

State:

- 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.
- Domain for each variable: $x_i \in \{0, 1, 2, 3\}, \forall i$.

Initial state: a random state.

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

4-Queens Problem as a Local Search Problem

State:

- 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.
- Domain for each variable: $x_i \in \{0, 1, 2, 3\}, \forall i$.

Initial state: a random state.

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

Neighbour relation:

- A: Move one queen to another row in the same column.
- B: Swap the row positions of two queens.

4-Queens Problem as a Local Search Problem

State:

- 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.
- Domain for each variable: $x_i \in \{0, 1, 2, 3\}, \forall i$.

Initial state: a random state.

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

Neighbour relation:

- A: Move one queen to another row in the same column.
- B: Swap the row positions of two queens.

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

4-Queens Problem as a Local Search Problem

Greedy descent

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.

Greedy descent in one sentence

Descend into a canyon in a thick fog with amnesia

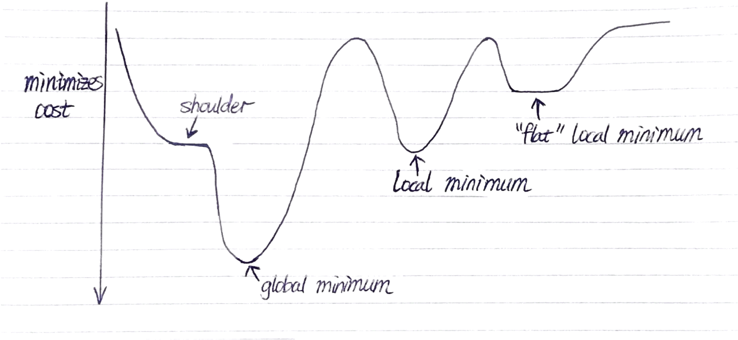
Properties of Greedy Descent

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

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

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.



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?

		Q	
			Q
	Q		
Q			

(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.

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?

		Q	
			Q
	Q		
Q			

(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

- 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.

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.

Learning Goals

Introduction to Local Search

Local Search Algorithms

- Greedy descent
- Escaping local optimums
- Greedy descent with random moves
- Simulated annealing
- Population-based algorithms

Revisiting the Learning goals

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

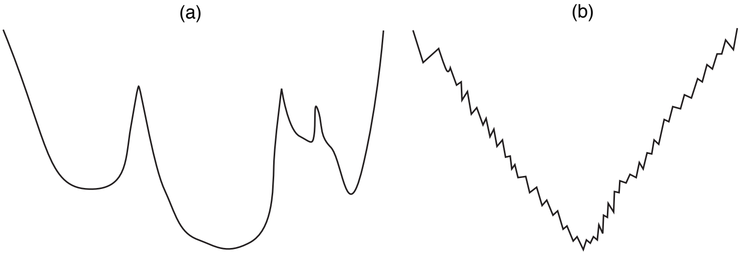
- Random walks:
  move to a random neighbour.
  Example: Simulated annealing

Random restarts vs random walks

Which random move is better for search space (a) or (b) ?

(A) Random restarts

(B) Random walks



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?

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

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

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.

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.

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.

- In practice, a popular schedule is geometric cooling.

Simulated annealing is like life...

Population-Based Algorithms

- The local search algorithms so far only remember a single state.
- What if we remember multiple states at a time?

Beam Search

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

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

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

- 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.

A Fun Genetic Algorithm Car Simulator

https://rednuht.org/genetic_cars_2/

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 ...

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

  Greedy descent moves to the best neighbour.
  Genetic algorithm ...

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.