Local Search

Alice Gao

Lecture 6

Readings: R & N 4.1

Based on work by K. Leyton-Brown, K. Larson, and P. van Beek

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.
- Given a local search problem, verify whether a state is a local/global optimum.
- Describe strategies for escaping local optima.
- Trace the execution of hill climbing, hill climbing with random restarts, simulated annealing, and genetic algorithms.
- Compare and contrast the properties of local search algorithms.

Why Use Local Search?

- Many search spaces are too big for systematic search.
- For CSPs, we only need to find a goal node. The path to a goal is irrelevant.
- Solution: local search

What is local search?

- Keep track of a single state, which is a complete assignment of values to variables.
- Move to a neighbour of the state based on how good the neighbour is.

When should we use local search?

- The state space is large or infinite.
- Memory is limited.
- To solve pure optimization problems.

Local Search

A local search problem consists of a:

- A state in the search space is
  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

Questions to think about

The problem formulation:

- What is the neighbour relation?
- What is the cost function?

Executing the algorithm:

- Where do we start?
- Which neighbour do we move to?
- When do we stop?

Properties and performance of the algorithm:

- Given enough time, will the algorithm find the global optimum solution?
- How much memory does it require?
- How does the algorithm perform in practice?

Hill climbing

- Where do we start?
  Start with a random solution.
- Which neighbour do we move to?
  Move to a neighbour with the lowest cost. Break ties randomly.
- When do we stop?
  Stop when no neighbour has a lower cost.
- How much memory does it require?
  Only need to remember the current node.
  No memory of where we've been.

Hill climbing in one sentence

Climbing Mount Everest in a thick fog with amnesia

CQ: Will hill climbing find the global optimum?

CQ: Given any problem and any initial state for the problem, will hill climbing always find the global optimum?

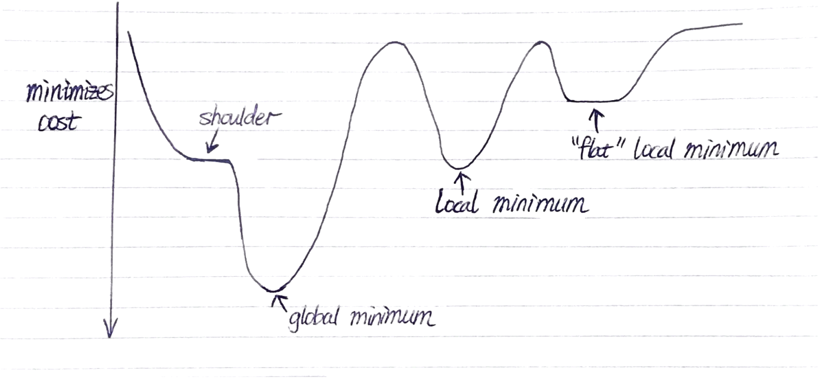
(A) Yes.

(B) No.

Trace the execution of hill climbing

See the notes on the course website.

The State Space Landscape



Where can hill climbing get stuck?

- Local optima: A state s^* is locally optimal if $c(s^*) \leq c(s)$ for every state s in the neighbourhood of s^*.
- Global optimum: A state s^* is globally optimal if $c(s^*) \leq c(s)$ for every state s.
- Plateau: a flat area in the state space
  - A shoulder: possible to improve when we escape.
  - A flat local optimum: cannot improve even if we escape.

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 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 is a global optimum.

(D) 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?

		Q	
			Q
	Q		
Q			

(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 is a global optimum.

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

Escaping a shoulder

- 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 hill climbing

- Perform quite well in practice.
- Makes rapid progress towards a solution.
  Easy to improve a bad state.

8-queens problem: ≈ 17 million states.

- Basic hill climbing

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

- Basic hill climbing $+ \leq 100$ consecutive sideway moves:

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

Choosing the Neighbour Relation

How do we choose the neighbour relation?

- Small incremental change to the variable assignment

There's a trade-off:

- bigger neighbourhoods:
- smaller neighbourhoods:

Dealing with local optima

Hill climbing can get stuck at a local optimum.
What can we do?

- Restart search in a different part of the state space.
  Hill climbing with random restarts

- Move to a state with a higher cost occasionally.
  Simulated annealing

Hill climbing with random restarts

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

Restart the search with a randomly generated initial state when

- we found a local optimum, or
- we've found a plateau and
  made too many consecutive sideway moves, or
- we've made too many moves.

Choose the best solution out of all the local optima found.

Will hill climbing + random restarts find the global optimum?

Will hill climbing with random restarts find the global optimum given enough time?

Simulated Annealing

- Where do we start?
  Start with a random solution and a large T.
- Which neighbour do we move to?
  Choose a random neighbour.
  If the neighbour is better than current, move to the neighbour.
  If the neighbour is not better than the current state,
  move to the neighbour with probability $p = e^{\Delta E/T}$.
- When do we stop?
  Stop when $T = 0$.

Simulated Annealing

Algorithm 1 Simulated Annealing

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

CQ: How does T affect $p = e^{\Delta E/T}$?

CQ: Consider a neighbour with a higher cost than the current node ($\Delta E < 0$).

As T decreases, how does $p = e^{\Delta E/T}$ change?
($p = e^{\Delta E/T}$ is the probability of moving to this neighbour.)

(A) As T decreases, $p = e^{\Delta E/T}$ increases.
(B) As T decreases, $p = e^{\Delta E/T}$ decreases.

CQ: How does ΔE affect $p = e^{\Delta E/T}$?

CQ: Assume that T is fixed. Consider a neighbour where $\Delta E < 0$

As ΔE decreases (becomes more negative),
how does $p = e^{\Delta E/T}$ change?
($p = e^{\Delta E/T}$ is the probability of moving to this neighbour.)

(A) As ΔE decreases, $p = e^{\Delta E/T}$ increases.
(B) As ΔE decreases, $p = e^{\Delta E/T}$ decreases.

Annealing Schedule

How should we decrease T?

- Linear
- Logarithmic
- Exponential

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

Examples of Simulated Annealing

- Example: getting a tennis ball into the deepest hole.
- Exploration versus exploitation

Parallel Search

- Idea: maintain k nodes instead of one.
- At every stage, update each node.
- Whenever one node is a solution, report it.
- Like k restarts, but uses k times the minimum number of steps.

There's not really any reason to use this method. Why not?

Beam Search

- Maintain k nodes instead of one.
- Choose the k best nodes out of all of the neighbors.
- When $k = 1$, it is hill climbing.
- The value of k lets us limit space and parallelism.

Do you see any potential problem with beam search?

Stochastic Beam Search

- Choose the k nodes probabilistically.
- The probability that a neighbor is chosen is proportional to the fitness of the neighbour.
- Maintains diversity amongst the nodes.
- Asexual reproduction: each node mutates and the fittest offsprings survive.

Genetic algorithm

- Like stochastic beam search, but with sexual reproduction
  Pairs of nodes are combined to create an offspring.
- Keep track of a set of states. Each state has a fitness.
- Randomly choose two states to reproduce.
  The fitter a state, the most likely it's chosen to reproduce.
- Two parent states crossover to produce a child state.
- The child state mutates with a small independent probability.
- Add the child state to the new population. Repeat the steps above until we produce a new population. Replace the old population with the new one.
- Repeat until one state in the population has high enough fitness.

Genetic Algorithm

Algorithm 2 Genetic Algorithm

$i = 0$
create initial population $pop(i) = \{X_1, ..., X_n\}$
while true **do**
    if $\exists x \in pop(i)$ with high enough $f(x)$ **then**
        break
    end if
    for each $X_i \in pop(i)$ calculate $pr(X_i) = f(X_i)/\sum_i f(X_i)$
    for j from 1 to n **do**
        choose a randomly based on $pr(X_i)$
        choose b randomly based on $pr(X_i)$
        child $\leftarrow$ crossover(a, b)
        child mutates with small probability
        add child to $pop(i+1)$
    end for
    $i = i + 1$
end while
return $x \in pop(i)$ with highest fitness

Comparing hill climbing and genetic algorithm

- How does each algorithm explore the state space?

  Hill climbing generates neighbours of the state based on the neighbour relation.
  Genetic algorithm ...

- How does each algorithm optimize the quality of the state/population?

  Hill climbing 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.
- Given a local search problem, verify whether a state is a local/global optimum.
- Describe strategies for escaping local optima.
- Trace the execution of hill climbing, hill climbing with random restarts, simulated annealing, and genetic algorithms.
- Compare and contrast the properties of local search algorithms.