VICTORIA UNIVERSITY OF
WELLINGTON
TE HERENGA WAKA

School of Engineering and Computer Science
Te Kura Mātai Pūkaha, Pūrorohiko

COMP 307/AIML 420 — Lecture 03

Problem Solving and Search Techniques

A/Prof. Yi Mei

Yi.Mei@ecs.vuw.ac.nz

(with thanks to Prof. Mengjie Zhang)

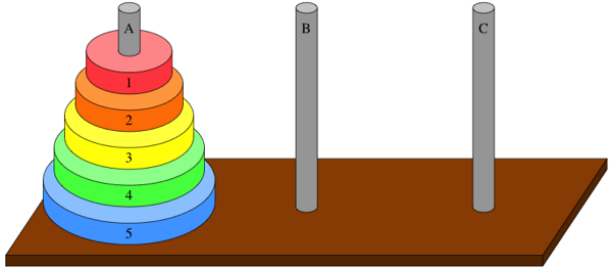
Outline

- *Uninformed/Blind search*
- *Informed search/Heuristic search*
- Local search: Hill Climbing
- Local search in continuous space
- Genetic beam search
- Advanced discussions
- Game Playing

Why Search (1)

Many puzzle and game playing problems need search:

- The Monkey and Bananas Problem
- The Missionaries and Cannibals Problem
- The 8-puzzle
- The Tower of Hanoi
- Wolf, Goat and Cabbage
- Water jug
- Route finding



- Chess, Bridge, Go (AlphaGo, AlphaZero,...all using search!)

Why Search (2)

Many real-world complex and engineering problems need search:

- Touring problems
- Travelling Salesperson Problem (TSP)
- Robot navigation
- University timetabling
- Job shop scheduling

Search is used in almost all AI techniques such as machine learning (ML) and evolutionary computation (EC)

General Search Algorithm

For general tree-search:

```
Initialise the frontier using the initial state of the problem
 loop
   If the frontier/fringe is empty
     then return failure
   choose a leaf node and remove it from the frontier
   If the node represents the goal state
     then return the corresponding solution (as success)
   else expand the node and put the children notes on the frontier
             (and ordering)
 end loop
```

Search Strategies

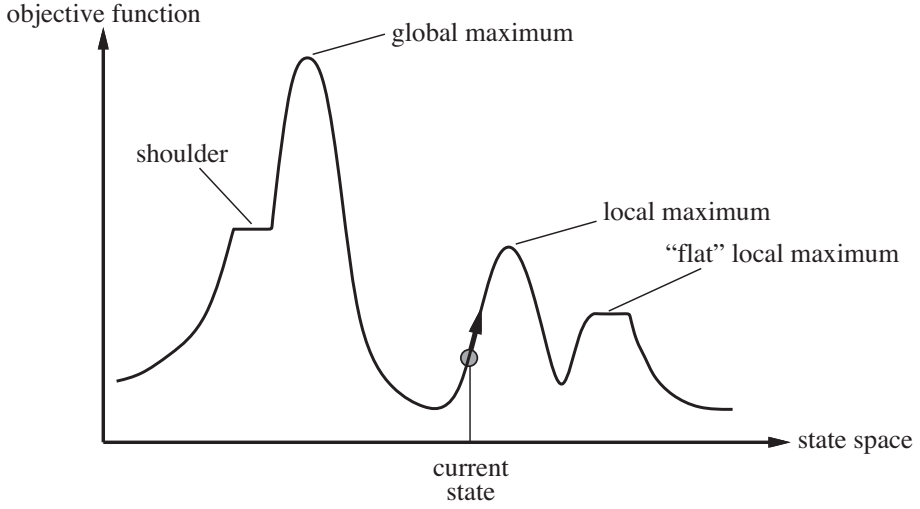
- Uninformed (blind) search
  - Breadth first: time complexity = $O(b^d)$, space complexity = $O(b^d)$
  - Depth first: time complexity = $O(b^m)$, space complexity = $O(bm)$
  - Depth limited: time complexity = $O(b^c)$, space complexity = $O(bc)$
  - Iterative deepening: time complexity = $O(b^d)$, space complexity = $O(bd)$
  - Uniform cost: time complexity = $O(b^d)$, space complexity = $O(b^d)$
  - Bidirectional: time complexity = $O(b^{d/2})$, space complexity = $O(b^{d/2})$
- Informed (Heuristic) search
  - Greedy best-first search
  - A* search
- **Beyond classic search (core to AI/ML!)**
  - **Hill climbing**
  - **Gradient descent**
  - **Simulated Annealing**
  - **Beam search**
  - Bound and bound (x)
  - dynamic programming (x)

Local Search

- So far, the search techniques we have discussed assume observable, deterministic, known environments where the solution is a sequence of actions
- In those algorithms, the *path* is a solution to the problem
- In many other problems, however, the path is *irrelevant*
- e.g. integrated circuit design (EEEN), job shop scheduling (COMP), automatic programming (SWEN/COMP), telecommunication network optimisation (NWEN/COMP), many data mining tasks (AIML/DATA)
- Local search algorithms operate using a single current node (in general) to move to neighbours of that node
- Local search algorithms are *not* systematic:
  – use very little memory
  – can often find **reasonable solutions** in large or even infinite state space

Local Search — "Hill Climbing"

- Local search is useful for solving optimisation problems
- Aim to find the best state according to an **objective function**
- Only keep *one* state (node) and its evaluation h(n): the *quality*
- Continually move in the direction of increasing h(n)
- Choose the best successor; if more than one, choose at random



Simulated Annealing (1)

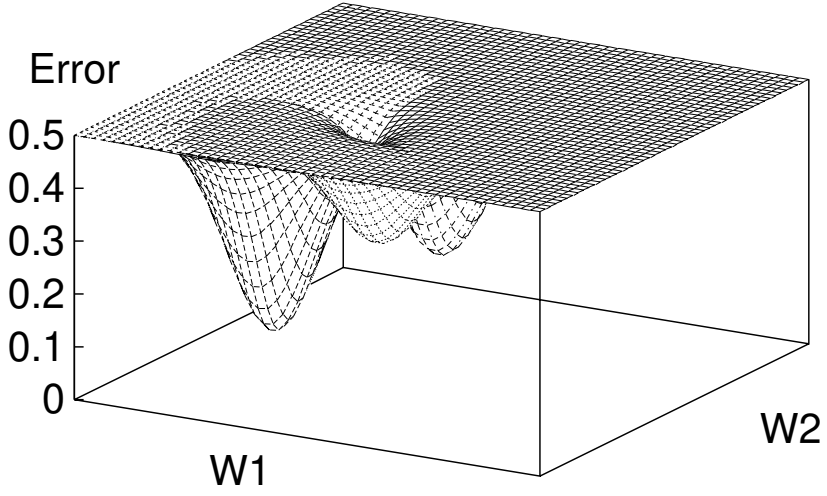
- Hill climbing (HC) never makes "worse" moves toward states with a higher cost. (*It is greedy*)
- HC can easily get stuck in a local optimum (maximum) and is (nearly always) incomplete.
- Purely random walk is complete in general, but can be extremely inefficient (*Halting problem?*)
- Can we learn anything from the "real-world" to improve HC?
- This is the idea of simulated annealing (SA)!

Simulated Annealing (2)

- SA borrows an idea from metal annealing in Physics which is used to temper or harden metals
  – heat them to a high temperature then *gradually* cool them
  – allows them to reach a low energy crystalline state
- Instead of using the best move, SA uses a **random** move
- But still uses the HC idea — if the move improves the situation, accept it; otherwise, accept it with some probability less than 1.
  – It uses the probability and "temperature" to control the loop

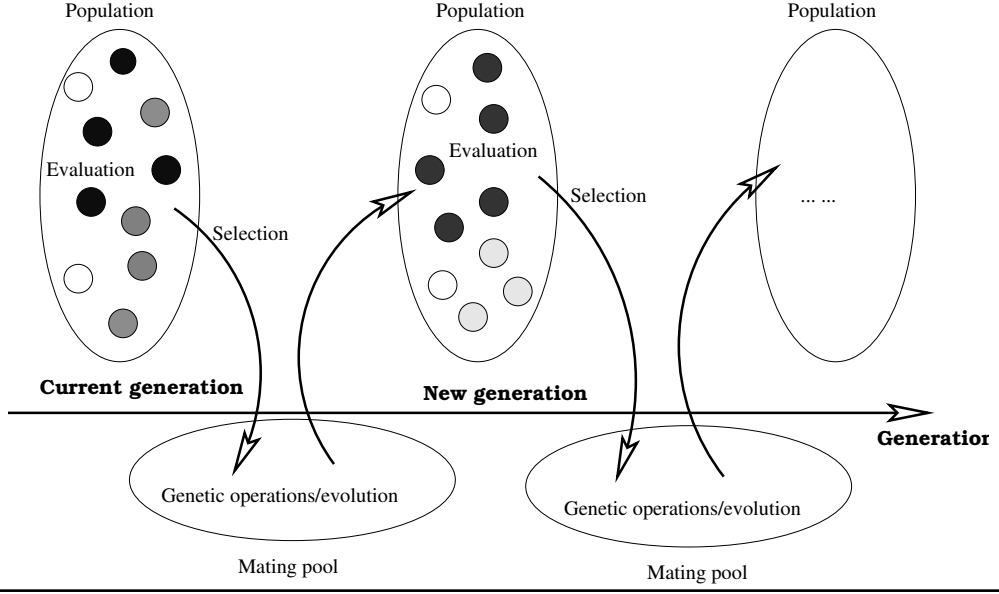
Gradient Descent Search

- HC only considers *discrete* steps
- If the problem (objective function) is continuous, then the idea of HC can still work but the mechanism will need to change
- Gradient Descent (or Ascent) search
- e.g. neural network training using back propagation



(Genetic) Local Beam Search

- Like HC, beam search (BS) considers only the solution space.
- Instead of considering only one neighbour, BS considers one or more neighbours
- Maintains a beam of multiple best *candidate solutions* (initially, randomly generated) and modifies the candidate solutions



Building Solutions vs Searching Solutions?

- Building solutions step by step
  - search path/space is partial solutions
  - classic search
  - all uninformed/blind search: BFS, UC, DFS, ...
  - all classic/heuristic search: greedy BFS, A*, ...
- Searching solution space
  - path does NOT matter or irrelevant
  - modifying solutions step by step
  - HC, GDS, SA, BS
- # Search solutions
  - A single solution per run: classic search, HC, GD, SA
  - Multiple solutions per run: Beam search
- Partial solutions vs candidate solutions
  - Partial: HC, GD, SA
  - Candidate: (genetic) beam search

More Discussions

- How many states would be checked (as the current) for neighbours during search?
  - 1 states/nodes: HC
  - 1 or more states: BS — mutation (1), crossover (>=2), ...
  - in gradient direction: GD
  - random? SA, ...
- Fringe/frontier
  - pruning and ordering
  - genetic operators?
- When to stop?
  - goal? classical search
  - local optima? HC, GD
  - random temperature? SA
  - convergence? BS
  - after some amount of time?

Further Discussions

- Paths and solutions: Explicit graphs (including trees)?
- Paths and solutions: Implicit graphs (construct as you go)?

- Local search vs Global search

- Online search vs off-line search

- Satisficing vs. Optimising?

- Dynamic environments?

(*Lots of things...we will discuss **some** of them in this course!*)

Game Playing

- States are ‘board’ positions
- Operators are moves of the game
- Initial state is initial board position
- Final state is when the game is finished

- COMP313: Computer Game Development

Summary

- Applications of Search
- Search strategies and techniques
  - Uninformed
  - Informed
  - Local search
- Other search techniques
  - Bound and Bound
  - Dynamic programming
  - ...
- Characteristics

- Suggested readings: Chapters 3 and 4
- Next Topic: Machine Learning