BEYOND CLASSICAL SEARCH

Fundamentals of Artificial Intelligence
Session 09

Pramod Sharma
pramod.sharma@prasami.com

Agenda

2

- Local Search Algorithms
- Hill-climbing
- Simulated Annealing
- Local Beam Search
- Genetic Algorithms
- Parallel Search

3

Search Algorithms So Far

- Designed to explore search space systematically
- Keep one or more paths in memory
- Record which have been explored and which have not
- A path to goal represents the solution
- More often than not, are complex in terms on time and space

pra-sâmi

4

Local Search Algorithms

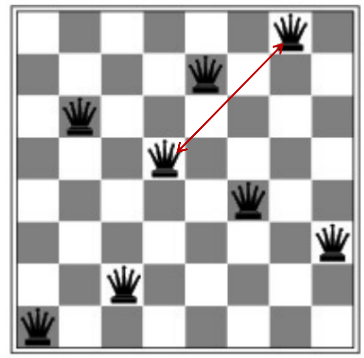
- In many optimization problems, the path to the goal is irrelevant; the goal state itself is the solution
- State space = set of "complete" configurations
- Find configuration satisfying constraints,
  - e.g., n-queens, factory floor layout, job shop scheduling, vehicle routing and portfolio management
- In such cases, we can use local search algorithms
- Keep a single "current" state, try to improve it
  - Use very little memory – usually a constant amount
  - Find reasonable solutions in large or infinite state spaces for which systematic solutions are unsuitable
  - Useful for solving optimization problems, e.g. Darwinian evolution, no “goal test” or “path cost”

pra-sâmi

5

Example: n-Queen Problem

- Put n queens on an n × n board with no two queens under attack
  - Not on the same row, column, or diagonal



- We still have one conflict
  - This is best we could do in present search
- Good neighborhood function is good balance between immediate neighbors and length of path to solution.
  - It needs to be learned. Cannot be guessed!

4/11/2024

pra-sâmi

6

Search Space

- State
  - All queen on board in some configuration
- Successor function
  - Move single queen to another square in the same column
- Objective function
  - No of queens attacking each other
- We are keen in finding a state which minimizes is objective function

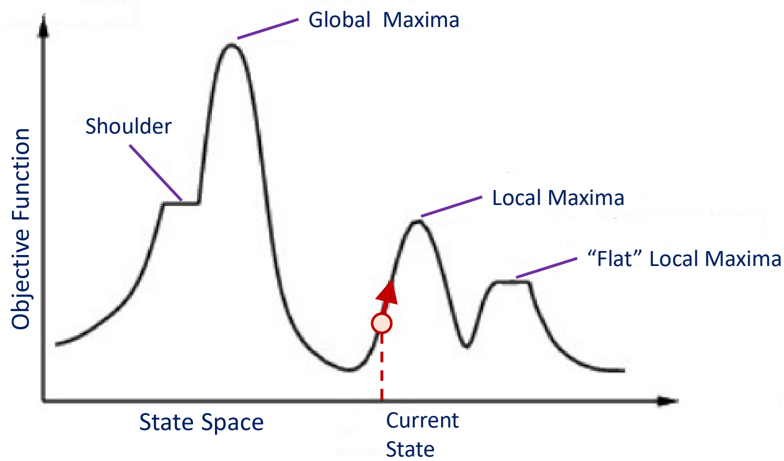
4/11/2024

pra-sâmi

7

State Space Landscape

- Problem: depending on initial state, can get stuck in local maxima/minima

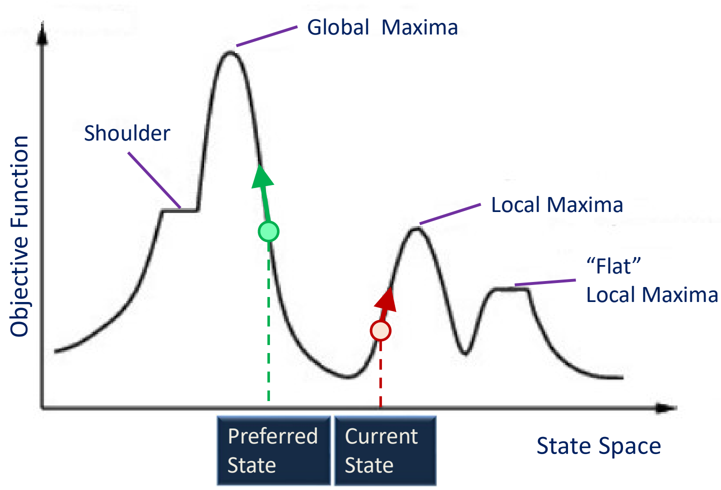


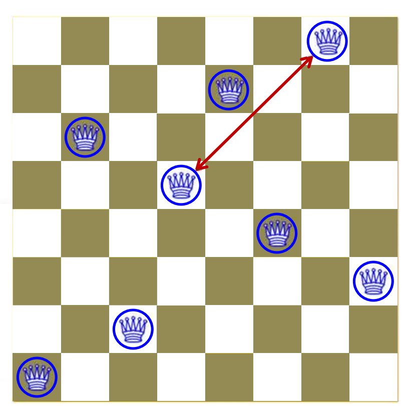
4/11/2024 pra-sâmi

8

State Space Landscape

- Problem: depending on initial state, can get stuck in local maxima/minima





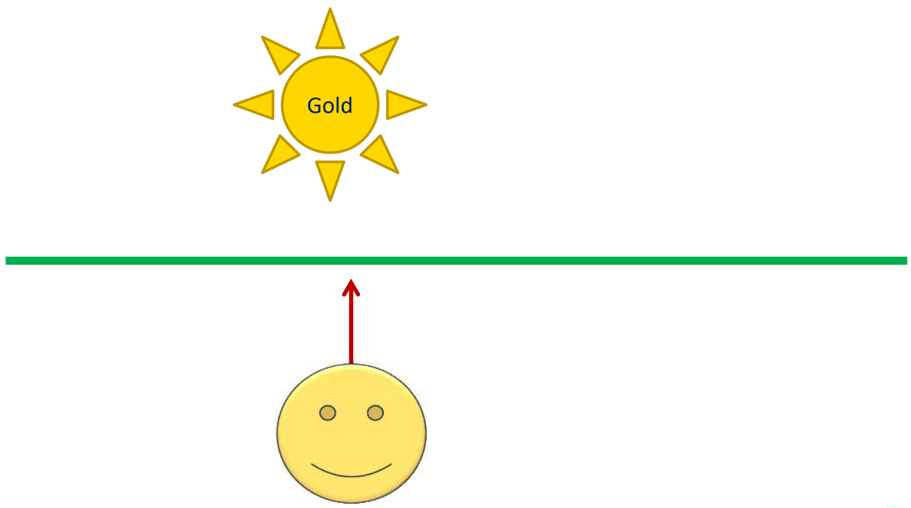
Local Minimum with h = 1

4/11/2024 pra-sâmi

9

Example: Initial State

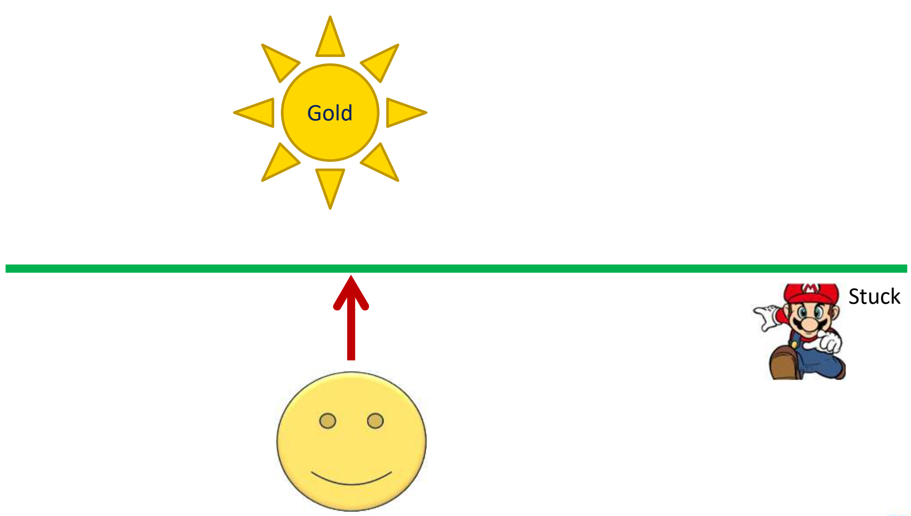
- Assume the objective function measures the straight-line distance



10

Example: Local Minima

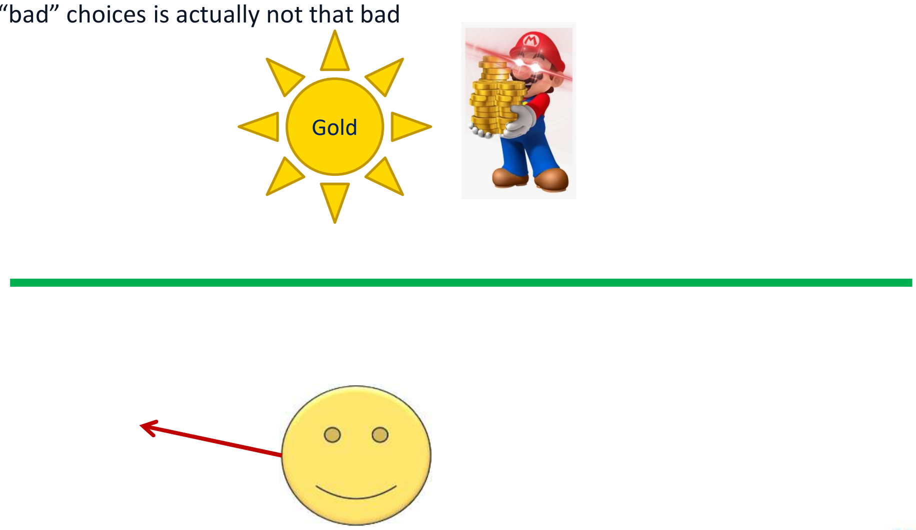
- Assume the objective function measures the straight-line distance



11

Example: A Plausible Solution

- Assume the objective function measure the straight-line distance
- Making some “bad” choices is actually not that bad

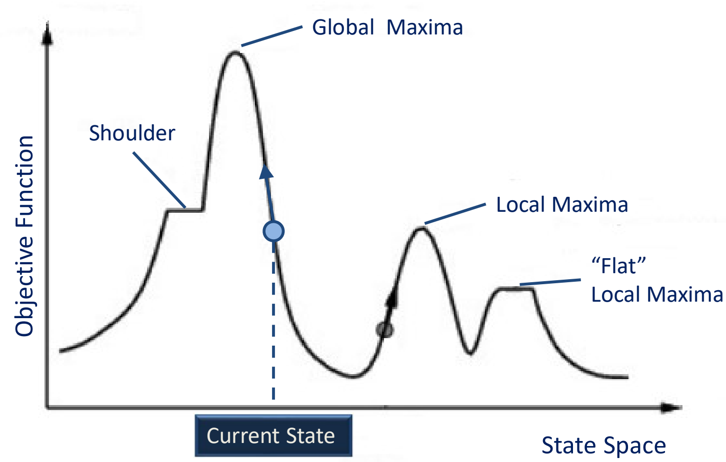


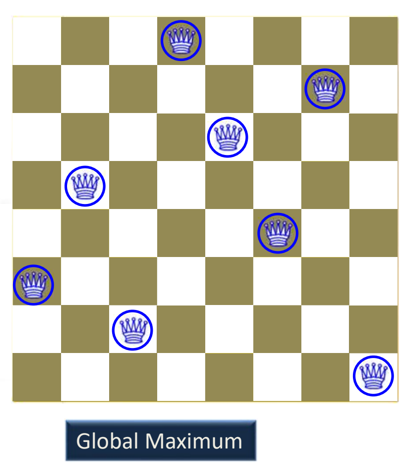
pra-sâmi

12

State Space Landscape

- Right choice will take you to global maxima

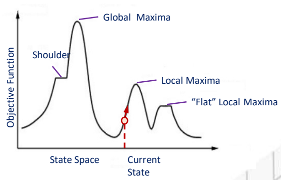


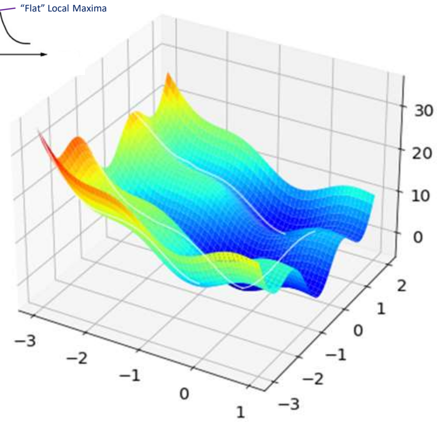


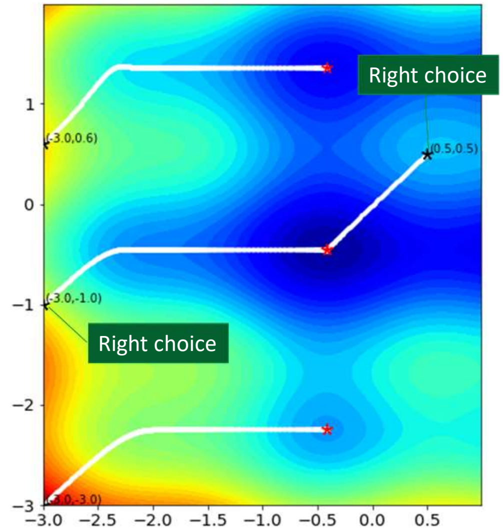
pra-sâmi

13

State Space Landscape

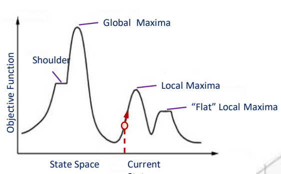


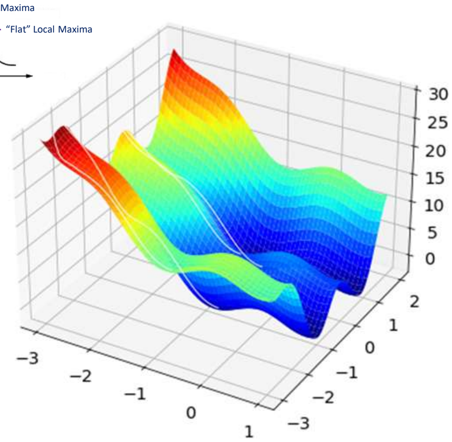


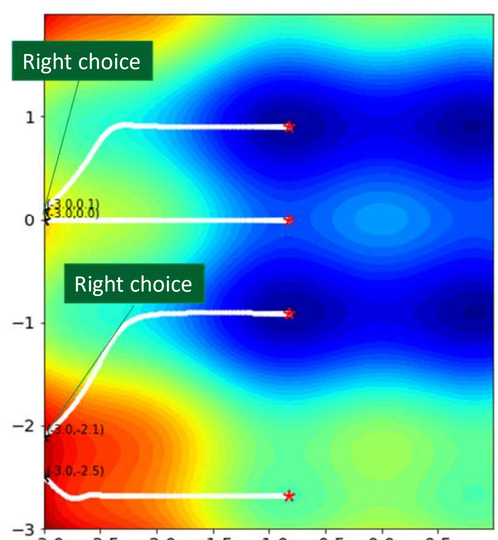


14

State Space Landscape



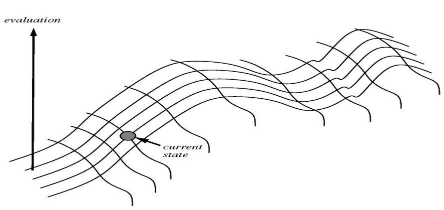


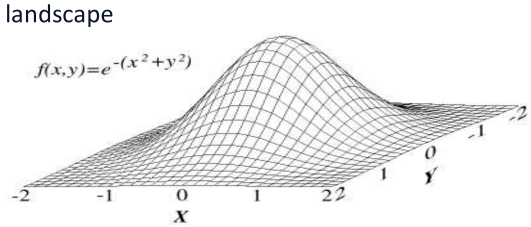


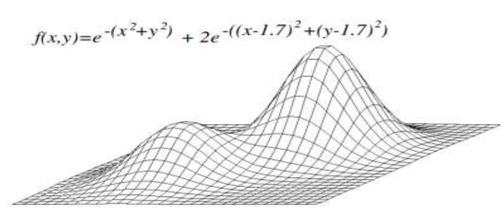
Hill Climbing

Iterative Improvement

- Consider all states laid out on the surface of a landscape







- The height at any point corresponds to the result of the evaluation function

17

Iterative Improvement

- Paths typically not retained - very little memory needed
- Move around the landscape trying to find the highest peaks – optimal solutions (or lowest valleys the if trying to minimise)
  - Useful for hard, practical problems where the state description itself holds all the information needed for a solution
  - Find reasonable solutions in a large or infinite state space
- Example : travelling salesperson, neural network gradient descent,
  - After back propagation, nudge it a bit and see if it converges better

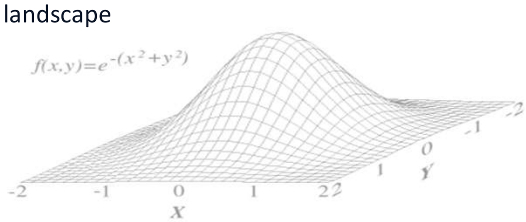
4/11/2024

pra-sâmi

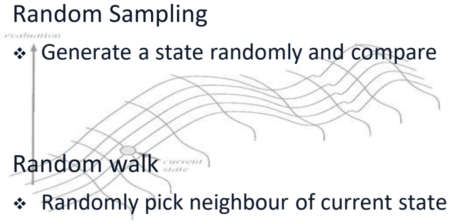
18

Iterative Improvement

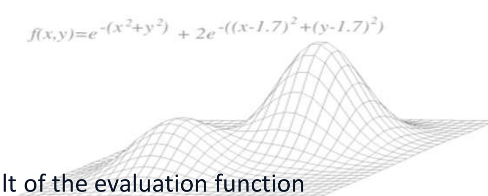
- Consider all states laid out on the surface of a landscape



- Random Sampling
  - Generate a state randomly and compare



- Random walk
  - Randomly pick neighbour of current state



- The height at any point corresponds to the result of the evaluation function

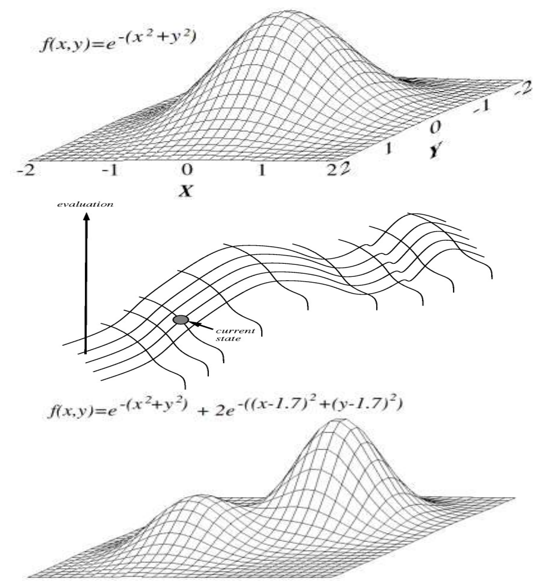
4/11/2024

pra-sâmi

19

Hill-Climbing Search

- Check all neighbours find better and move towards that neighbour
  - Terminate when peak is reached
- Maximize objective function
- Never thinks beyond its neighbours
- Can randomly choose among the best successors
  - If multiple successors have best value

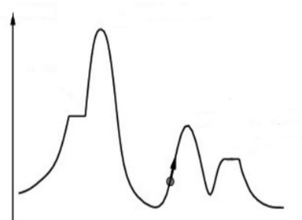


4/11/2024

pra-sâmi

20

Hill-Climbing Search



- “Like climbing Everest in thick fog with amnesia”
- Complete? Optimal?
- Hill climbing search is sometimes called **greedy local search**
- Although greedy algorithms often perform well, hill climbing gets stuck when:
  - Local maxima/minima
  - Ridges
  - Plateau (shoulder or flat local maxima/minima)
- The steepest-ascent hill climbing solves only 14% of the randomly-generated 8-queen problems with an avg. of 4 steps
  - When it gets struck (86% generated problems), it takes only 3 moves
- Allowing sideways move raises the success rate to 94% with an avg. of 21 steps, and 64 steps for each failure

4/11/2024

pra-sâmi

21

Hill- Climbing (Greedy Local)

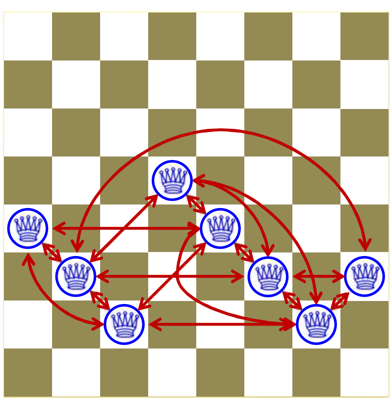
- Start with current-state = initial-state
- Until current-state = goal-state OR there is no change in current-state do:
  - Get the children of current-state and apply evaluation function to each child
  - If one of the children has a better score, then set current-state to the child with the best score
- Loop that moves in the direction of increasing (or decreasing) value
  - Terminates when a “peak” (or “dip”) is reached
  - If more than one best direction, the algorithm can choose at random

pra-sâmi

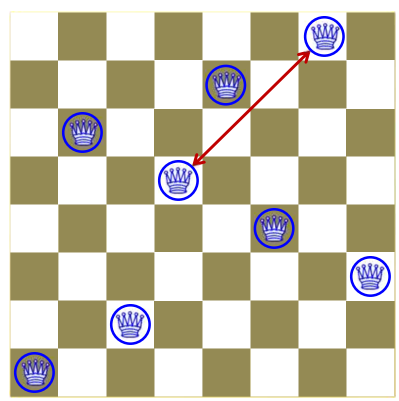
22

Hill Climbing: n-Queen Problem

- Move one queen at time to reduce h



Eight Queen with heuristic cost estimate h =17



Local Minimum with h = 1

Only current state matters. How we got there is of little consequence.

pra-sâmi

23

"A hill climbing algorithm that never makes "downhill" (or "uphill") to a lower (or "higher") value. Does not guarantee complete search!

A purely random walk – moving to a successor chosen uniformly at random – is complete, but extremely inefficient!"

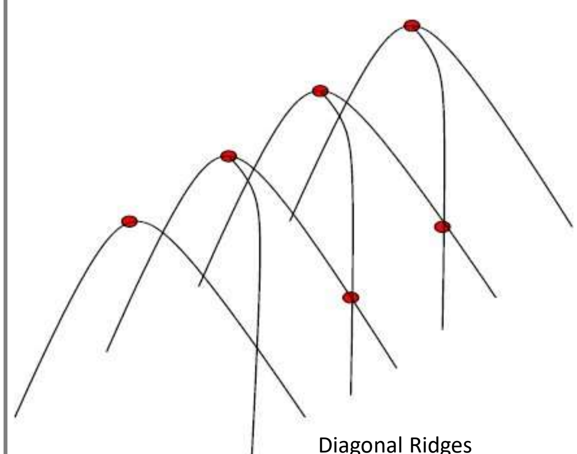
What should we do?

pra-sâmi

24

Hill-climbing Drawbacks

- Local minima (maxima)
  - Local, rather than global minima (maxima)
- Plateau
  - Area of state space where the evaluation function is essentially flat
  - The search will conduct a random walk
- Ridges
  - Causes problems when states along the ridge are not directly connected - the only choice at each point on the ridge requires uphill (downhill) movement



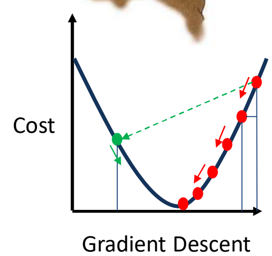
Diagonal Ridges

pra-sâmi

25

Problems with Iterative Improvement

- Given a set pick items to reach a total:
  - A = { 15, 5, 8, 2, 12}
  - Q = 29
  - Start Z = {15, 5, 8} = 28
    - No further swapping/ additions possible
  - However, best solution is {15, 2, 12}
    - Let's start from right side Z = { 8, 2, 12 } = 20
    - Swap 2 with 15 ; Z = { 8, 15, 12 } = 35 > 29
    - Swap 8 with 15; Z = {15, 2, 12 } = 29

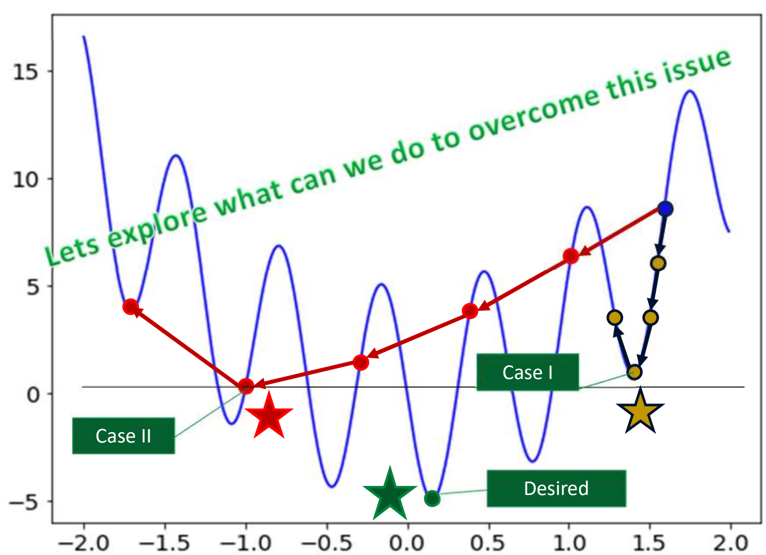


- Hill Climbing
  - Start with any maximal subset Z of A
  - Consider a pair (a_j , a_k) such that $a_j \in Z$ and $a_k \notin Z$.
  - Replace a_j with a_k in Z in subset sum increases but does not exceed Q.
- For gradient descent change increase to decrease

4/11/2024 pra-sâmi

26

Can get struck in local minima too…



4/11/2024 pra-sâmi

27

Variants of Hill Climbing

- Escaping shoulder or local optima
  - May be start with breadth-first search and once we find better optimization function
  - Start climbing the hill again
- Prolonged period of exhaustive search
  - Small period of hill climbing
- Some what a middle ground between local and systematic search

28

Variants of Hill Climbing

- What happens if you run into a plateau or ridge
- Simple solution could be keep moving even if the objective function is same
  - Impose some limit of moves, if it does not start improving, end the search
  - But in larger space, we may be shuttling between couple of points
- Tabu Search
  - Prevent returning to same state again
  - Maintain list of states already visited
  - Fixed length
  - Add latest visited node, drop the oldest one
  - In general a list of size 100 to 500 states is sufficient

29

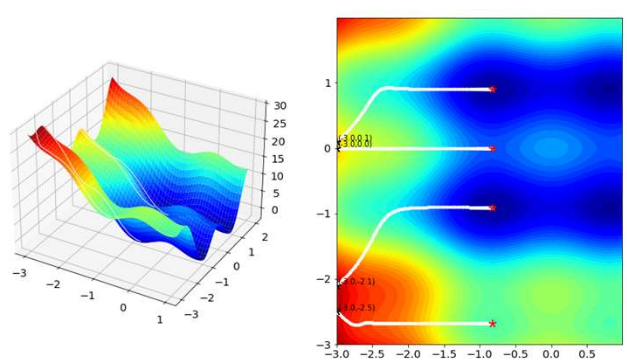
Variants of Hill Climbing

- Stochastic hill climbing:
  - Chooses at random from among uphill moves
  - Converges more slowly, but finds better solutions in some landscapes
- Different Variations
  - For each restart: run till end vs. run for fixed time
  - Run fixed number of restarts vs. run indefinitely
- Stochastic hill climbing with random walk
  - Use a probability p
  - p times use best neighbor
  - (1-p) time move to a random neighbor
- As time progresses
  - Increase p; more greedy less stochastic

30

Variants of Hill Climbing

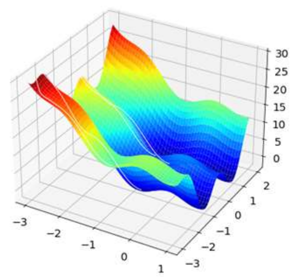
- First-choice hill climbing:
  - Generate successors randomly until one is better than the current
  - Good when a state has many successors

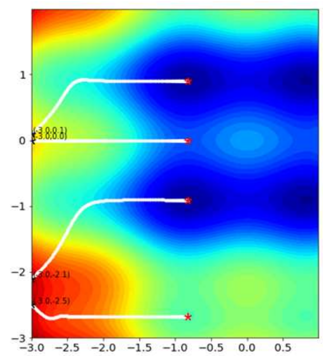


31

Variants of Hill Climbing

- Random-restart hill climbing:
  - Conducts a series of hill climbing searches from randomly generated initial states, stops when a goal is found
  - It's complete with probability approaching 1
  - Assume each hill climbing search has a probability p of success, then the expected number of restarts required is 1/p
  - For 8-queen problem, p = 14%, so we need roughly 7 iterations to find a goal
  - Expected # of steps = cost_to_success + (1-p)/p * cost_to_failure
  - Random-restart hill climbing is very effective for n-queen problem
    - 3 million queens can be solved < 1 min





pra-sâmi

32

Simulated Annealing

- What is Simulated Annealing?
  - *The process used to harden metals and glass by heating them to a high temperature and then gradually cooling them, thus allowing the material to reach a low-energy crystalline state*

- Idea
  - Escape local maxima by allowing some "bad" moves but gradually decrease their frequency
  - *'Shake out of the pit'*

- One can prove: If T decreases slowly enough, then simulated annealing search will find a global optimum with probability approaching 1
  - Proposed in 1983 by IBM (Kirkpatric et al) for solving VLSI layout problem
  - Effective in Traveling salesperson, Graph Partitioning , Airline Scheduling, Facility Layout, etc.
  - Later used in Image processing as well.

pra-sâmi

33

Simulated Annealing

- Temperature
  - Let T denote the temperature
  - Initially Set T to a very high value
  - Reduce T to Zero gradually with some cooling scheme
    - Reduce it to half after n iterations (exponential)
    - Reduce by x % after n iteration (linear)

- Algorithm
  - Initialize T
  - Set next = randomly selected successor of current
    - Calculate Δ E = value (current) – value (next)
    - If ΔE > 0 then set current = next (minimization)
    - Else set current = next with probability $e^{\Delta E/T}$
    - Update T as per the schedule and loop

Boltzmann Function
- T being in the division, more exploration is permitted at high temperatures
- More exploitation at , lower temperatures

34

Simulated Annealing : Temperature T

- High T : probability of locally bad move is high

- Low T : probability of locally bad move is low

- Depreciate T as time progresses
  - Over subsequent epochs
  - Code in a Temperature schedule
    - Reduce 10 % after every 10 epochs

35

Local Beam Search

- Keeping one node (current state) in the memory!
  - Is it enough??
  - Is it not a bit of over reaction
- Idea:
  - Keep track of k states rather than just one
  - Start with k randomly generated states
  - At each iteration, all the successors of all k states are generated
  - If anyone is a goal state, stop;
  - Else select the k best successors from the complete list and repeat

36

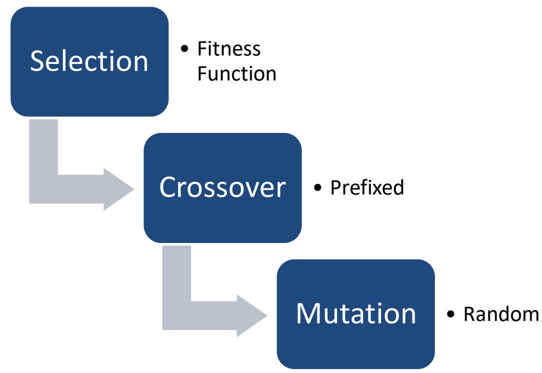
Local Beam Search

- Is it the same as running k random-restart searches in parallel?
  - No
  - Useful information is passed among the k parallel search threads
- Challenge: frequently, all successor end up on same hill
- Stochastic beam search:
  - Choose K successors randomly, biased towards good ones
  - Similar to natural selection, offspring of an organism populate the next generation according to its fitness

37

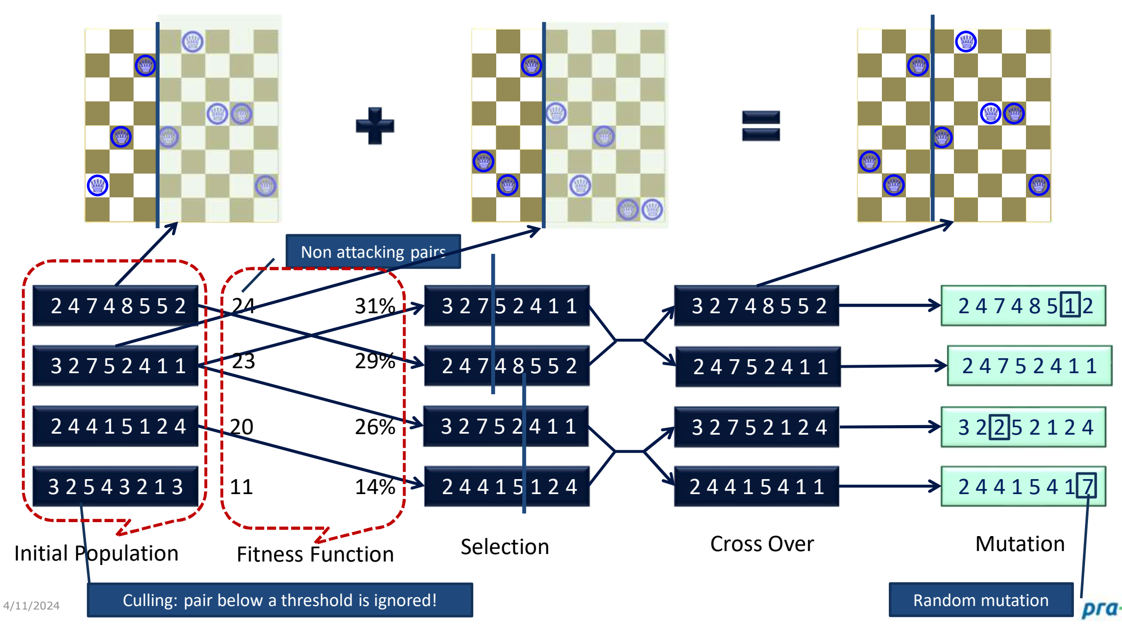
Genetic Algorithms

- Start with k randomly generated states (population)
- A successor state is generated by combining two parent states
- Give some representation to state for ease of manipulation
  - A state is represented as a string over a finite alphabet (often a string of 0s and 1s or digits)
- Evaluation function (fitness function)
  - Higher values for better states
- Produce the next generation of states by selection, crossover, and mutation



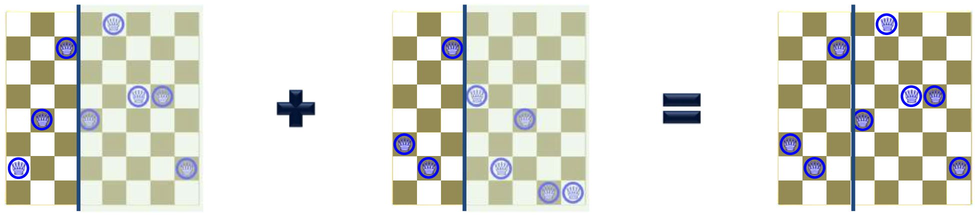
38

Example



39

Fitness Function



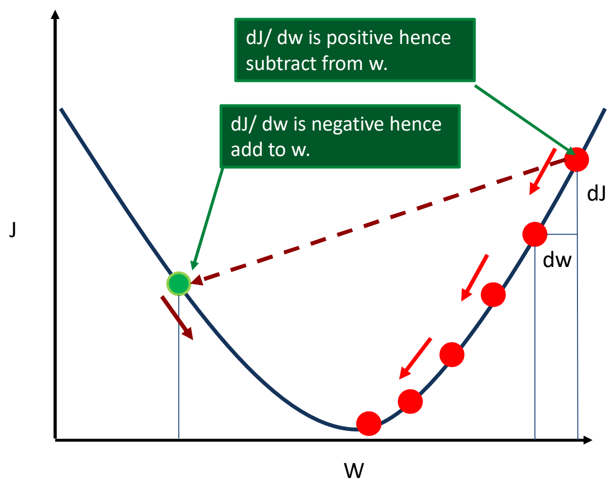
- Fitness function:
  - Number of non-attacking pairs of queens (min = 0, max = 8 × 7/2 = 28)
  - 24 / (24+23+20+11) = 31%
  - 23 / (24+23+20+11) = 29% etc…
- Genetic algorithms combine an uphill tendency with random exploration and exchange of information among parallel search threads
- Advantages come from "crossover", which raise the level of granularity

4/11/2024

pra-sâmi

40

Gradient Descent



4/11/2024

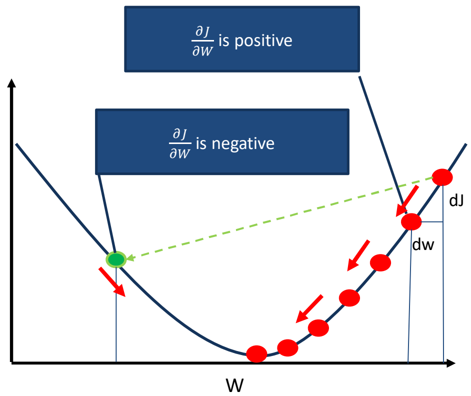
pra-sâmi

41

Gradient Descent

- Solve it as a optimization problem
- Gradient is actually Error Gradient
- Create some stable loss function
  - Loss function : ℓ (a, y),   a in turn is function of W and b
- Compute gradient $\frac{\partial J}{\partial W}$
- Update weights W = W - α . $\frac{\partial J}{\partial W}$
- Similarly b = b - α . $\frac{\partial J}{\partial b}$

Where α is defined as learning rate.

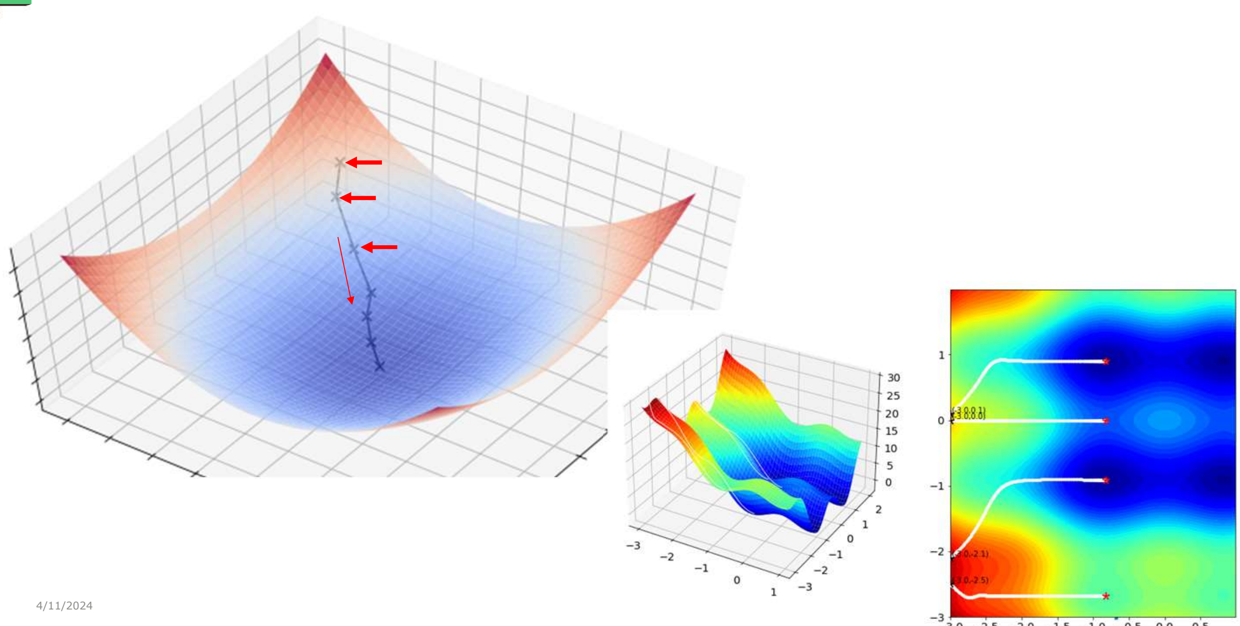


4/11/2024

pra-sâmi

42

Loss / Cost Optimization



4/11/2024

43

Learning Rate : Tough Terrain

- Finding most optimal gradient descent can be difficult

Question: How to select right learning rate?

- Too fast and you can miss global minima!
- Too slow, you can be struck at local minima!
- Need to look for learning rate that converges smoothly and avoids local minima!
- Need a learning that 'adapts' to the terrain
- No Fixed learning rate
- To change as per the change in gradient
- Popular algorithms
  - SGD
  - Adam
  - Adadelta
  - Adagrad
  - RMSProp

44

Measuring problem-solving performance

The evaluation of a search strategy

- Completeness:
  - Is the strategy guaranteed to find a solution when there is one?
- Optimality:
  - Does the strategy find the highest-quality solution when there are several different solutions?
- Time complexity:
  - How long does it take to find a solution?
- Space complexity:
  - How much memory is needed to perform the search?

45

Measuring problem-solving performance

- In AI, complexity is expressed in
  - b, branching factor, maximum number of successors of any node
  - d, the depth of the shallowest goal node (depth of the least-cost solution)
  - m, the maximum length of any path in the state space

- Time and Space is measured in
  - Number of nodes generated during the search
  - Maximum number of nodes stored in memory

4/11/2024

pra-sâmi

46

Measuring problem-solving performance

- For effectiveness of a search algorithm
  - we can just consider the total cost
  - The total cost = path cost (g) of the solution found + search cost
    - search cost = time necessary to find the solution

- Tradeoff:
  - *< long time, optimal solution with least g >* vs. *< shorter time, solution with slightly larger path cost g >*

4/11/2024

pra-sâmi

47

Which method?

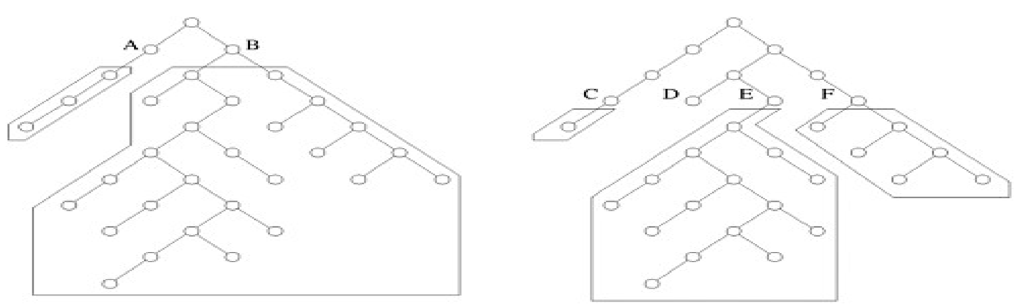
- Exhaustive search for small finite spaces when it is essential that the optimal solution is found
- A* for medium-sized spaces if heuristic knowledge is available
- Random search for large evenly distributed homogeneous spaces
- Hill climbing for discrete spaces where a sub-optimal solution is acceptable

pra-sâmi

48

Parallel Depth First Search:

- The critical issue in parallel depth-first search algorithms is the distribution of the search space among the processors



(a) (b)

pra-sâmi

49

Reflect…

- In heuristic hill climbing , paths typically not retained - very little memory needed
- Depending on initial state, can get stuck in local maxima/minima
  - Right choice will take you to global maxima
  - Making some “bad” choices is actually not that bad
- Variants of Hill Climbing
  - Stochastic
  - First-choice
  - Random-restart
- In many optimization problems, the path to the goal is irrelevant; the goal state itself is the solution

pra-sâmi

50

Reflect

- We are looking at AI problems as a search problems
  - Defined the problem as State Space Search
  - Looked at solutions as:
    - Path to Goal state
    - Or Goal as some better state
- We looked at:
  - Informed search strategies
    - Greedy best-first search
    - A* search
    - Memory Bound Search
  - Blind Search
    - Breadth-first search
    - Depth-first search
    - Bidirectional search
  - Local search strategies- Hill climbing, simulated annealing.
  - Performance measurement

pra-sâmi

51

Next Session

- Matrix Multiplication
- Recursion
- Dynamic Programming
- Branch and Bound

4/11/2024

pra-sâmi

52

4/11/2024

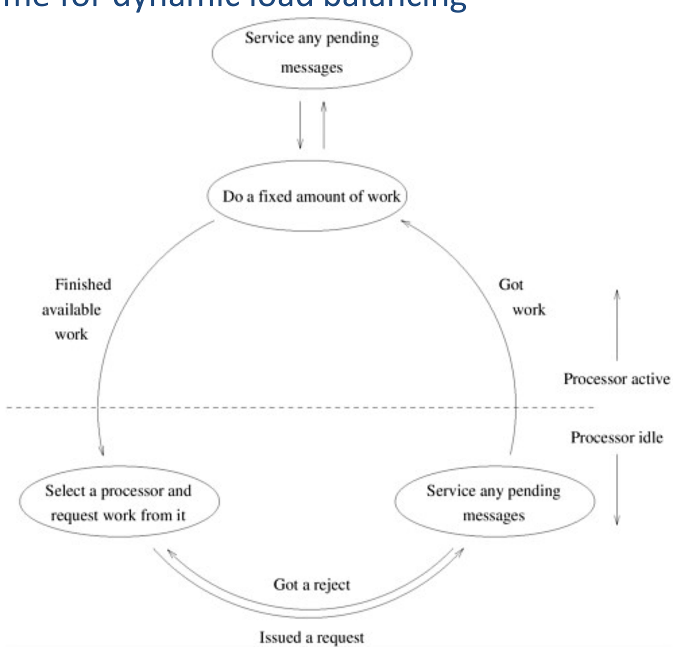
pra-sâmi

ADDITIONAL MATERIAL

pra-sâmi

55

A generic scheme for dynamic load balancing



4/11/2024

pra-sâmi

56

Parallel Depth First Search:

- Important Parameters of Parallel DFS

- Load -Balancing Schemas:
  - Asynchronous Round Robin
  - Global Round Robin
  - Random Polling

pra-sâmi

57

Parallel Best first Search:

- This algorithm contains two main components:
  - Open list , Close list

- In most parallel formulations of BFS, different processors concurrently expand different nodes from the open list

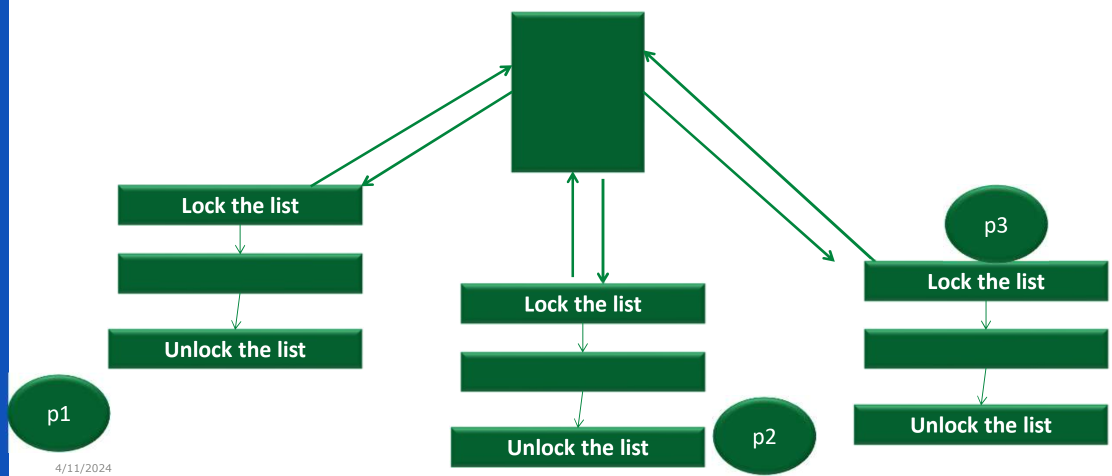
- There are two problems with this approach:
  - The termination criterion of sequential BFS fails for parallel BFS
  - Since the open list is accessed for each node expansion, it must be easily accessible to all processors, which can severely limit performance

pra-sâmi

58

Parallel Best first Search:

- A general schematic for parallel best-first search using a centralized strategy



59

Parallel Best first Search:

- One way to avoid the contention due to a centralized open list is to let each processor have a local open list.
- The processors must communicate among themselves to minimize unnecessary search

60

Parallel Best first Search:

- Communication Strategies for Parallel Best-First Tree Search
  - random communication strategy
  - ring communication strategy
  - blackboard communication strategy

61

Parallel Binary Search Algorithm:

- We have an ordered array ,we have two processors
- We part our array to P+1 parts , where p is number of processors

1	2	8	10	12	13	18	27	30