UIT2504 Artificial Intelligence

Local Search Strategies

C. Aravindan
<AravindanC@ssn.edu.in>

Professor of Information Technology
SSN College of Engineering

August 26, 2024

Evaluating a state

- Sometimes, it may be possible to design an evaluation function $f(s)$ that evaluates the "badness" (to be minimized) or "goodness" (to be maximized) of a state s
- In such cases, the most desirable state may be chosen from the working set
- Working set is maintained as a priority queue based on the evaluation function f
- Obviously, the quality of search depends on the evaluation function f

Heuristics

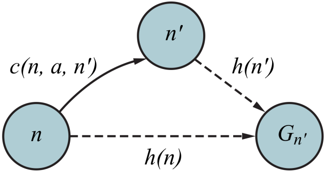
- Usually, such an evaluation function $f(s)$ is designed based on some heuristics $h(s)$ — estimation of cost of reaching a goal state from state s
- For example, can you think of a heuristics for the route finding problem in a map? — Straight line distance (SLD) from the current city to the destination city
- Heuristics should be an easy function to compute!
- $h(s^*)$ should be 0 for any goal state s^*

Admissible Heuristics

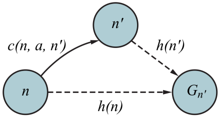
- A heuristic h is admissible if for every node n, $h(n) \leq h^*(n)$, where $h^*(n)$ is the **true** cost to reach the goal state from n.
- In other words, $f(n) = g(n) + h(n) \leq C^*$, where C^* is the optimal path cost
- An admissible heuristic **never overestimates** the cost to reach the goal, i.e., it is always **optimistic**
- The SLD heuristics and the two heuristics for sliding puzzle problem are examples of admissible heuristics

Consistent Heuristics

- A heuristics is consistent if for every node n, $h(n) \leq c(n, a, n') + h(n')$, where n' is a successor of n generated by some action a



Consistent Heuristics



- When h is consistent, we can infer the following

$$
\begin{aligned}
f(n') &= g(n') + h(n') \\
&= g(n) + c(n, a, n') + h(n') \\
&\geq g(n) + h(n) \\
&\geq f(n)
\end{aligned}
$$

- That means, evaluation function f is monotonic — it is non-decreasing along any path
- Every consistent heuristics is also admissible

Questions?

Iterative improvement algorithms

- In several problems, the path is irrelevant and a goal-state is a solution we are looking for

Iterative improvement algorithms

- In several problems, the path is irrelevant and a goal-state is a solution we are looking for
- With state space = set of "complete" configurations,
  - Find an optimal configuration (eg. TSP, maximal matching in a bipartite graph, "weights" that minimize error on the examples)

Iterative improvement algorithms

- In several problems, the path is irrelevant and a goal-state is a solution we are looking for
- With state space = set of "complete" configurations,
  - Find an optimal configuration (eg. TSP, maximal matching in a bipartite graph, "weights" that minimize error on the examples)
  - Find a configuration that satisfies some constraints (eg. Timetable generation, n-queens problem, stable matching)

Iterative improvement algorithms

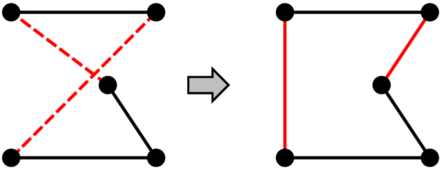
- In several problems, the path is irrelevant and a goal-state is a solution we are looking for
- With state space = set of "complete" configurations,
  - Find an optimal configuration (eg. TSP, maximal matching in a bipartite graph, "weights" that minimize error on the examples)
  - Find a configuration that satisfies some constraints (eg. Timetable generation, n-queens problem, stable matching)
- In such cases, we can use iterative improvement algorithms — "keep a single current state and try to improve it"

Example: TSP

- "Complete formulation" — any Hamiltonian circuit is a state

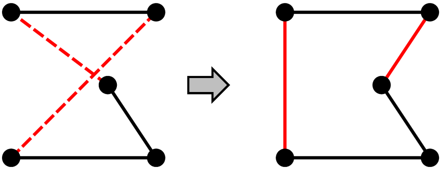
Example: TSP

- "Complete formulation" — any Hamiltonian circuit is a state
- Start with any complete state (any Hamiltonian circuit)
- Action: Pairwise exchanges — replace edges (u, v) and (u', v') with edges (u, v') and (u', v)



Example: TSP

- "Complete formulation" — any Hamiltonian circuit is a state
- Start with any complete state (any Hamiltonian circuit)
- Action: Pairwise exchanges — replace edges (u, v) and (u', v') with edges (u, v') and (u', v)



- Able to get within 1% of optimal solution very quickly, even with thousands of cities (good, as an approximation algorithm)

Example: n-queens problem

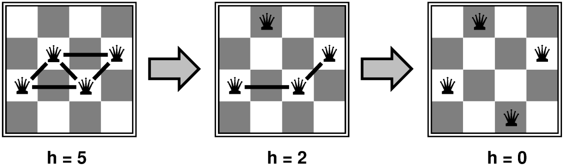
- "Complete formulation" — n queens on board, one per column

Example: n-queens problem

- "Complete formulation" — n queens on board, one per column
- Start with any state — any placement of n queens, one per column, on a chess board

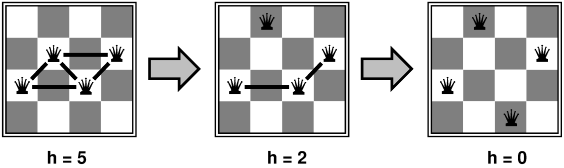
Example: *n*-queens problem

- "Complete formulation" — n queens on board, one per column
- Start with any state — any placement of n queens, one per column, on a chess board
- Action: Move a queen, within its column, to reduce the number of conflicts



Example: *n*-queens problem

- "Complete formulation" — *n* queens on board, one per column
- Start with any state — any placement of *n* queens, one per column, on a chess board
- Action: Move a queen, within its column, to reduce the number of conflicts

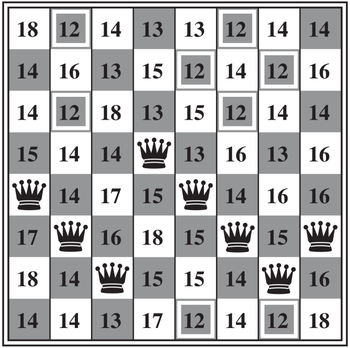


- Able to solve instantaneously even for very large *n*

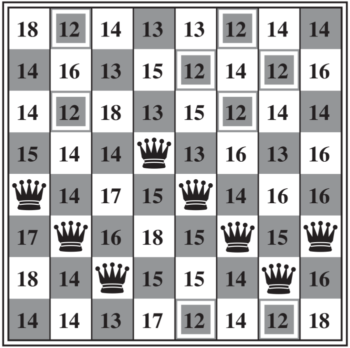
Outline of Hill Climbing Algorithm

```
def hill_climbing(problem):
    current = problem.initial()
    while True:
        neighbor = max( problem.children(current) )
        if problem.value(neighbor) <=
                                     problem.value(current):
            break
        current = neighbor
    return current
```

Example: 8-queens problem

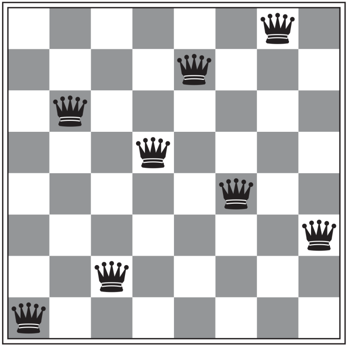


Example: 8-queens problem

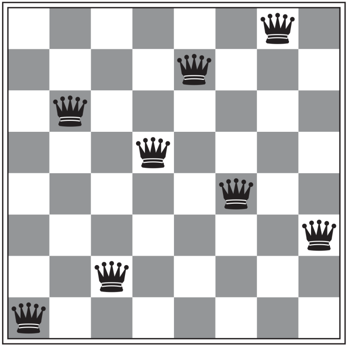


- One of the square with $h = 12$ is chosen and the corresponding queen is moved there

Example: 8-queens problem

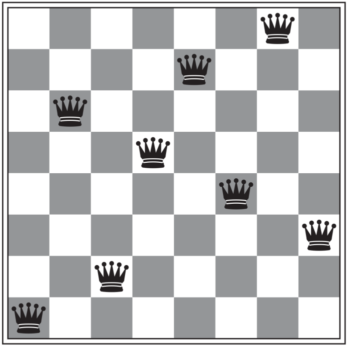


Example: 8-queens problem



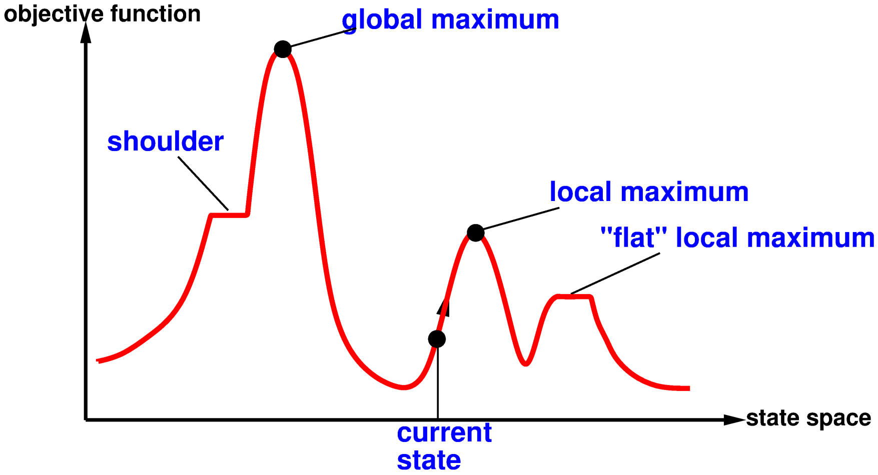
- $h(s) = 1$

Example: 8-queens problem



- $h(s) = 1$
- And, we have hit a local minimum!

Hill Climbing Search



Issues in Greedy Local Search

- Local maxima / minima

Issues in Greedy Local Search

- Local maxima / minima
- Ridges — sequence of local maxima that is very difficult for the greedy algorithm to navigate

Issues in Greedy Local Search

- Local maxima / minima
- Ridges — sequence of local maxima that is very difficult for the greedy algorithm to navigate
- Plateaux — flat local maximum or a shoulder

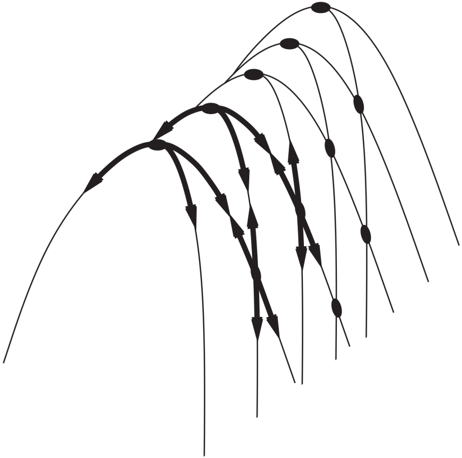
Issues in Greedy Local Search

- Local maxima / minima
- Ridges — sequence of local maxima that is very difficult for the greedy algorithm to navigate
- Plateaux — flat local maximum or a shoulder

Local maxima is a serious problem

Empirical analysis of 8-queens problem reveals that the greedy hill-climbing algorithm gets stuck 86% of the time

Ridges



Variations of Hill Climbing

- Random sideways moves

Variations of Hill Climbing

- Random sideways moves — can escape from a shoulder, but gets trapped in a local maxima

Variations of Hill Climbing

- Random sideways moves — can escape from a shoulder, but gets trapped in a local maxima — number of sideways moves may be limited

Variations of Hill Climbing

- Random sideways moves — can escape from a shoulder, but gets trapped in a local maxima — number of sideways moves may be limited — number of instances solved for the 8-queens problem increases from 14% to 94%

Variations of Hill Climbing

- Random sideways moves — can escape from a shoulder, but gets trapped in a local maxima — number of sideways moves may be limited — number of instances solved for the 8-queens problem increases from 14% to 94%
- Stochastic Hill Climbing

Variations of Hill Climbing

- Random sideways moves — can escape from a shoulder, but gets trapped in a local maxima — number of sideways moves may be limited — number of instances solved for the 8-queens problem increases from 14% to 94%
- Stochastic Hill Climbing — chooses at random, among all uphill moves

Variations of Hill Climbing

- Random sideways moves — can escape from a shoulder, but gets trapped in a local maxima — number of sideways moves may be limited — number of instances solved for the 8-queens problem increases from 14% to 94%
- Stochastic Hill Climbing — chooses at random, among all uphill moves — probability of selection can depend on the steepness of the ascent

Variations of Hill Climbing

- Random sideways moves — can escape from a shoulder, but gets trapped in a local maxima — number of sideways moves may be limited — number of instances solved for the 8-queens problem increases from 14% to 94%
- Stochastic Hill Climbing — chooses at random, among all uphill moves — probability of selection can depend on the steepness of the ascent — example of a Randomized algorithm

Variations of Hill Climbing

- Random sideways moves — can escape from a shoulder, but gets trapped in a local maxima — number of sideways moves may be limited — number of instances solved for the 8-queens problem increases from 14% to 94%
- Stochastic Hill Climbing — chooses at random, among all uphill moves — probability of selection can depend on the steepness of the ascent — example of a Randomized algorithm
- First Choice Hill Climbing — randomly generate the successors, until one better than the current is generated

Variations of Hill Climbing

- Random sideways moves — can escape from a shoulder, but gets trapped in a local maxima — number of sideways moves may be limited — number of instances solved for the 8-queens problem increases from 14% to 94%
- Stochastic Hill Climbing — chooses at random, among all uphill moves — probability of selection can depend on the steepness of the ascent — example of a Randomized algorithm
- First Choice Hill Climbing — randomly generate the successors, until one better than the current is generated
- Random Restart

Variations of Hill Climbing

- Random sideways moves — can escape from a shoulder, but gets trapped in a local maxima — number of sideways moves may be limited — number of instances solved for the 8-queens problem increases from 14% to 94%
- Stochastic Hill Climbing — chooses at random, among all uphill moves — probability of selection can depend on the steepness of the ascent — example of a Randomized algorithm
- First Choice Hill Climbing — randomly generate the successors, until one better than the current is generated
- Random Restart — enough restarts may make this algorithm complete

Variations of Hill Climbing

- Random sideways moves — can escape from a shoulder, but gets trapped in a local maxima — number of sideways moves may be limited — number of instances solved for the 8-queens problem increases from 14% to 94%
- Stochastic Hill Climbing — chooses at random, among all uphill moves — probability of selection can depend on the steepness of the ascent — example of a Randomized algorithm
- First Choice Hill Climbing — randomly generate the successors, until one better than the current is generated
- Random Restart — enough restarts may make this algorithm complete — if each hill-climbing has a probability p of success, then $1/p$ restarts are expected

Variations of Hill Climbing

- Random sideways moves — can escape from a shoulder, but gets trapped in a local maxima — number of sideways moves may be limited — number of instances solved for the 8-queens problem increases from 14% to 94%
- Stochastic Hill Climbing — chooses at random, among all uphill moves — probability of selection can depend on the steepness of the ascent — example of a Randomized algorithm
- First Choice Hill Climbing — randomly generate the successors, until one better than the current is generated
- Random Restart — enough restarts may make this algorithm complete — if each hill-climbing has a probability p of success, then $1/p$ restarts are expected — for 8-queens, $p \approx 0.14$, and so roughly 7 restarts are expected

Questions?

Simulated Annealing

- One interesting variation of hill climbing is to adopt the concept of simulated annealing

Simulated Annealing

- One interesting variation of hill climbing is to adopt the concept of simulated annealing
- For example, consider a ball set to roll on a state landscape

Simulated Annealing

- One interesting variation of hill climbing is to adopt the concept of simulated annealing
- For example, consider a ball set to roll on a state landscape
- The ball simply follows the rules of gravity and moves towards nearby valley

Simulated Annealing

- One interesting variation of hill climbing is to adopt the concept of simulated annealing
- For example, consider a ball set to roll on a state landscape
- The ball simply follows the rules of gravity and moves towards nearby valley
- The ball needs to make some uphill moves to escape from local minima!

Simulated Annealing

- One interesting variation of hill climbing is to adopt the concept of simulated annealing
- For example, consider a ball set to roll on a state landscape
- The ball simply follows the rules of gravity and moves towards nearby valley
- The ball needs to make some uphill moves to escape from local minima!
- Imagine applying just enough force for it to escape from all local minima but not from global minima

Simulated Annealing

- One interesting variation of hill climbing is to adopt the concept of simulated annealing
- For example, consider a ball set to roll on a state landscape
- The ball simply follows the rules of gravity and moves towards nearby valley
- The ball needs to make some uphill moves to escape from local minima!
- Imagine applying just enough force for it to escape from all local minima but not from global minima
- How to find that “just enough force”?

Simulated Annealing

- Similar to the metallurgical process of annealing

Simulated Annealing

- Similar to the metallurgical process of annealing
- Start with a high "temperature"

Simulated Annealing

- Similar to the metallurgical process of annealing
- Start with a high "temperature" — probability of selecting a bad move is high

Simulated Annealing

- Similar to the metallurgical process of annealing
- Start with a high "temperature" — probability of selecting a bad move is high
- Slowly reduce the "temperature"

Simulated Annealing

- Similar to the metallurgical process of annealing
- Start with a high "temperature" — probability of selecting a bad move is high
- Slowly reduce the "temperature" — probability of selecting a bad move reduces slowly

Simulated Annealing

- Similar to the metallurgical process of annealing
- Start with a high "temperature" — probability of selecting a bad move is high
- Slowly reduce the "temperature" — probability of selecting a bad move reduces slowly
- "Schedule" of reducing the temperature is very critical

Simulated Annealing

function SIMULATED-ANNEALING($problem$, $schedule$) **returns** a solution state
  $current \leftarrow problem$.INITIAL
  for t = 1 **to** ∞ **do**
    $T \leftarrow schedule(t)$
    if T = 0 **then return** $current$
    $next \leftarrow$ a randomly selected successor of $current$
    $\Delta E \leftarrow$ VALUE($current$) – VALUE($next$)
    if $\Delta E > 0$ **then** $current \leftarrow next$
    else $current \leftarrow next$ only with probability $e^{-\Delta E/T}$

Questions?