Search

Chapter 4

(acknowledgement goes to Gillian Smith(NEU) and Hwee Tou Ng's for slides used in this lecture)

Local Search

Local Search

- For some problems, you don’t need the search path
- Just keep the current node
- Make the way to the goal

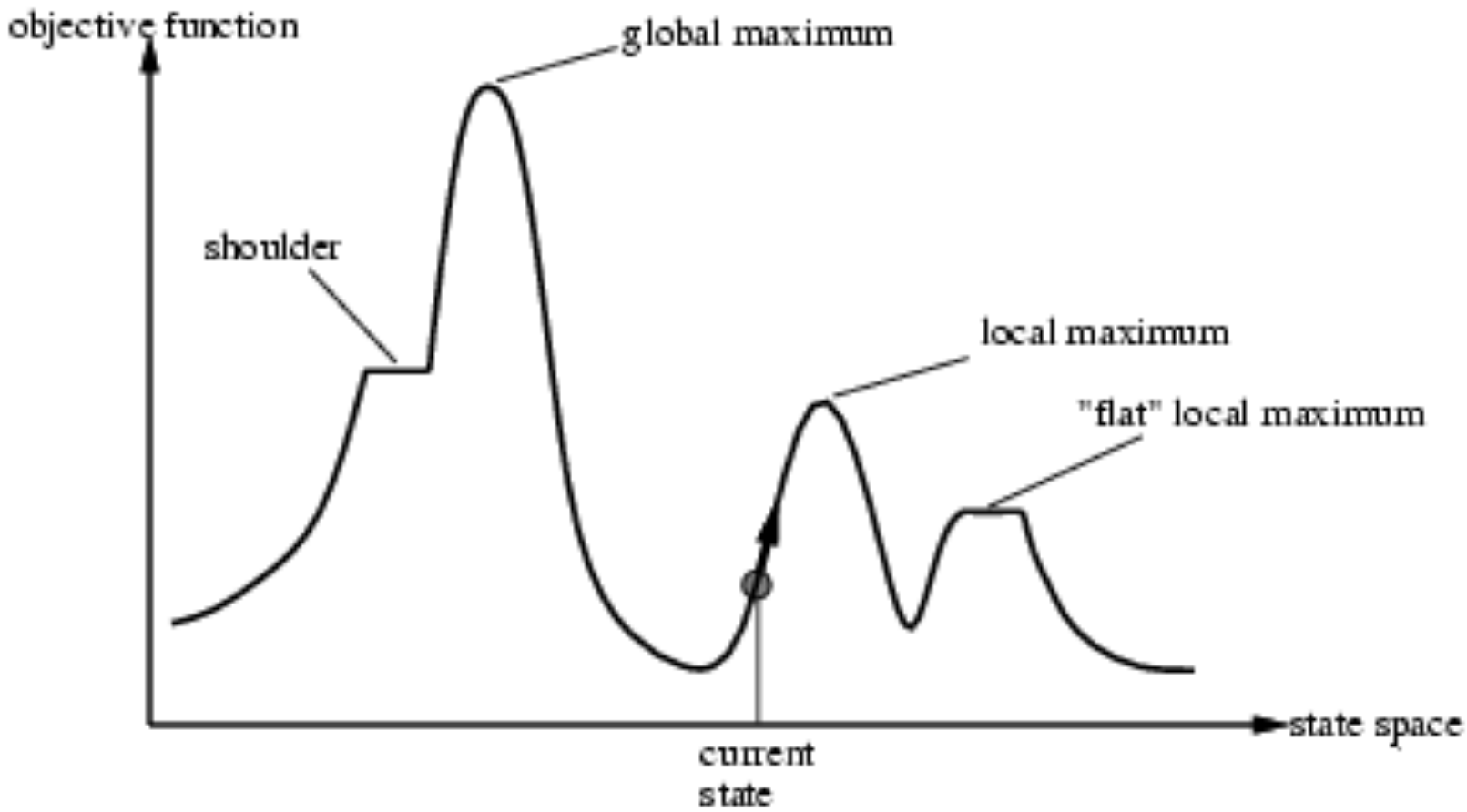
Hill-Climbing

"Like climbing Mount Everest in fog with amnesia"

```
function HILL-CLIMBING(problem) returns a state that is a local maximum
   inputs: problem, a problem
   local variables: current, a node
                    neighbor, a node

   current ← MAKE-NODE(INITIAL-STATE[problem])
   loop do
      neighbor ← a highest-valued successor of current
      if VALUE[neighbor] ≤ VALUE[current] then return STATE[current]
      current ← neighbor
```

Hill-Climbing



Simulated annealing search

Idea: escape local maxima by allowing some "bad" moves but gradually decrease their frequency

```
function SIMULATED-ANNEALING(problem, schedule) returns a solution state
   inputs: problem, a problem
           schedule, a mapping from time to "temperature"
   local variables: current, a node
                    next, a node
                    T, a "temperature" controlling prob. of downward steps

   current ← MAKE-NODE(INITIAL-STATE[problem])
   for t ← 1 to ∞ do
       T ← schedule[t]
       if T = 0 then return current
       next ← a randomly selected successor of current
       ΔE ← VALUE[next] – VALUE[current]
       if ΔE > 0 then current ← next
       else current ← next only with probability e^(Δ E/T)
```

Local beam search

- 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 any one is a goal state, stop; else select the *k* best successors from the complete list and repeat.

Navigation Problem

Use Hill Climbing to navigate the space using sonar sensors. How would you get out of local minima. How would you discretize this continuous space?

Trace the algorithm.

Genetic Algorithms

- Instead of keeping 1 state, keep a **population**
- Evolve population based on **fitness function**
  - Keep the best x% with some probability
  - Crossover the best y%
  - Mutate the remainder

Genetic algorithms

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

Genetic algorithms

- 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.

Mutation – Genetic Algorithms

- http://http://www.youtube.com/watch?v=xiRhe8mL_08

Environment types

- Static (vs. dynamic): The environment is unchanged while an agent is deliberating. (The environment is semidynamic if the environment itself does not change with the passage of time but the agent's performance score does)
- Discrete (vs. continuous): A limited number of distinct, clearly defined percepts and actions.
- Single agent (vs. multi-agent): An agent operating by itself in an environment. Does the other agent interfere with my performance measure?

Environment types

- Fully observable (vs. partially observable): An agent's sensors give it access to the complete state of the environment at each point in time.
- Deterministic (vs. stochastic): The next state of the environment is completely determined by the current state and the action executed by the agent. (If the environment is deterministic except for the actions of other agents, then the environment is strategic)
- Episodic (vs. sequential): An agent's action is divided into atomic episodes. Decisions do not depend on previous decisions/actions.

Environment types

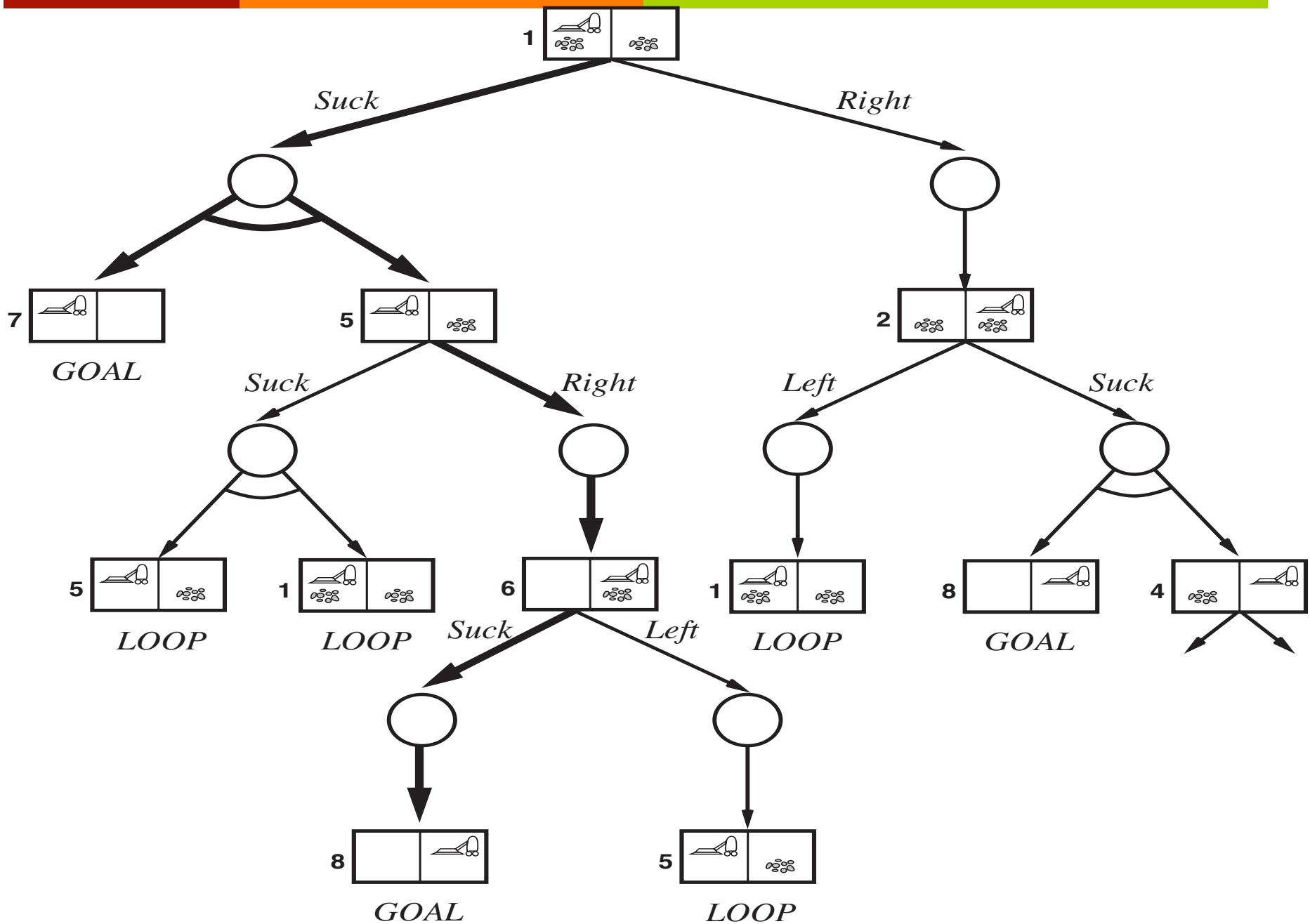
- Fully observable (vs. partially observable): An agent's sensors give it access to the complete state of the environment at each point in time.
- Deterministic (vs. stochastic): The next state of the environment is completely determined by the current state and the action executed by the agent. (If the environment is deterministic except for the actions of other agents, then the environment is strategic)
- Episodic (vs. sequential): An agent's action is divided into atomic episodes. Decisions do not depend on previous decisions/actions.

Dealing with Non-determinism

And-OR Tree

Tree alternates between two types of nodes:

- **OR nodes:** the agent chooses between actions.
- **AND nodes:** choice is induced by the environment's choice of outcome


1
Suck
Right
7
GOAL
5
2
Suck
Right
Left
Suck
5
LOOP
1
LOOP
6
1
LOOP
8
GOAL
4
Suck
Left
8
GOAL
5
LOOP