CENG 466
Artificial Intelligence

Lecture 5

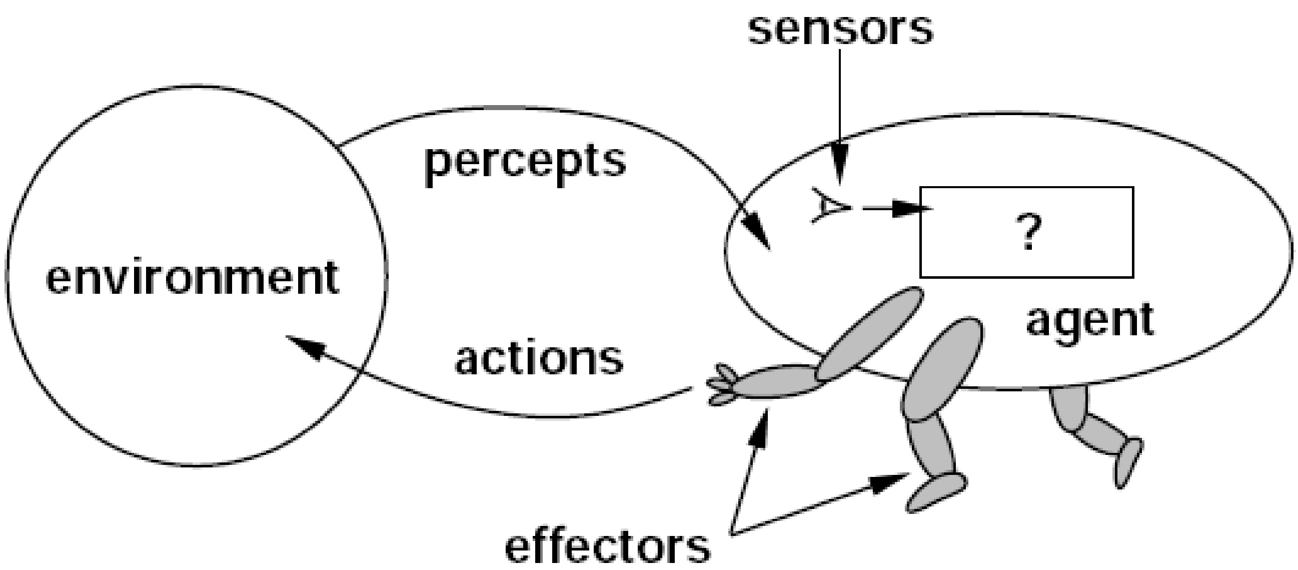
Solving Problems by Searching (III)

Topics

- Search Categories
- Informed Search Algorithms
- Best First Search
- Greedy Search
- A* Search
- Iterative Deepening A* Search
- Hill Climbing Search
- Simulated Annealing Search
- Game Playing and Min-Max Search

Intelligent Agents

- An agent is something that perceives and acts in an environment
- An ideal agent always takes actions that maximizes its performance
- An agent adopts a goal and searches the best path to reach that goal



States and State-Spaces

- State: The set of all information items that describe a system at a given time.
- State space is the set of states that an intelligent agent can be in.
- An action takes the agent from one state to another one.
- State space search is finding a sequence of states starting from the initial state to the goal

Searching

- Assuming that the agent knows:
  - how to define a problem,
  - how to recognize a solution (goal),
- finding a solution is done by a search through the state space.

Search Categories

- Un-informed Searches: If we have no extra information about the problem
- Informed Searches: If we have extra information about the problem.

Un-informed Searches

- In un-informed searches, the agent knows:
  - The initial state
  - The goal state
- But it does not know if a state is close to the goal or not
- Therefore, these searches are blind searches

Informed Searches

- Informed searches have some extra information about the problem
- At each state, we can estimate how far we are from the goal
- Using this information, we can have better search algorithms

How to Use the Information in Searches

- If the state space of a problem is large, we expand only some of the nodes.
- Choosing the next node to expand is the main difference between the search algorithms.
- An informed search algorithm uses its extra information when it decides which node should be expanded

Informed Search Algorithms

- Best First Search
- Greedy Search
- A* Search
- Hill Climbing Search
- Simulated Annealing Search

Best First Search

- Best First Search puts the nodes in order so that the node with the best value is expanded first.
- The best node is the node, that appears to be best according to the evaluation function.
- In most cases evaluation function only estimates the value of the nodes.

Evaluation in Best-First Search

- Best first search algorithms use some estimated measure of the cost of the solution and try to minimize it.
- Two basic approaches for estimating cost are:
  - The greedy search which tries to expand the node closest to the goal.
  - The A* search which tries to expand the node on the least-cost solution path.

Greedy Search

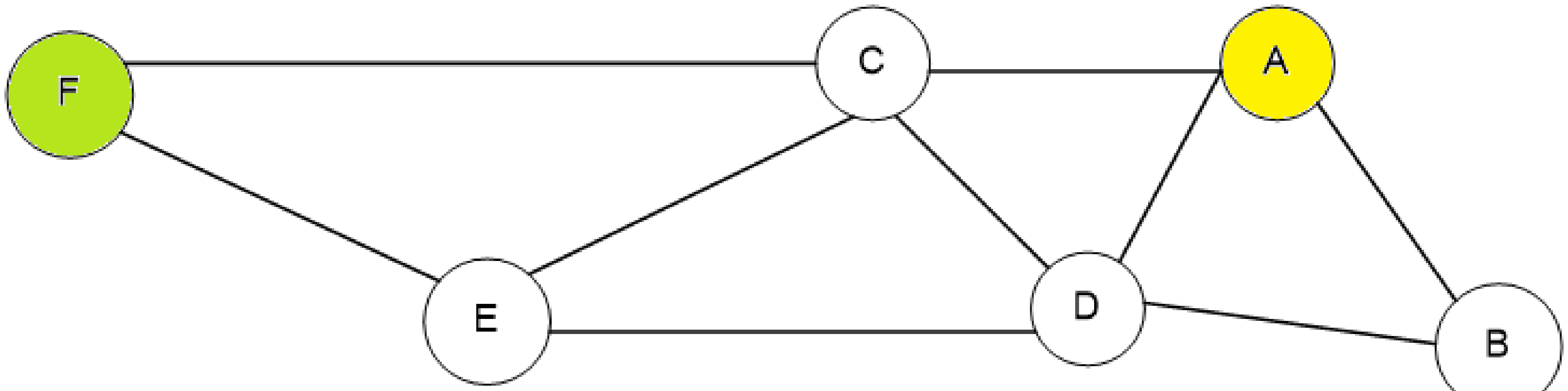
- Greedy search is one of the simplest best-first search strategies
- Greedy search minimizes the estimated cost to reach the goal.
- The cost of reaching the goal from a state can be estimated but cannot be determined exactly.
- A function that calculates such costs is called a **heuristic function**

Best First Search Example

- Assume a graph of cities and the roads connecting them is given.
- The initial city and the goal city are defined.
- The evaluation function is based on the geographical coordinates of the cities.

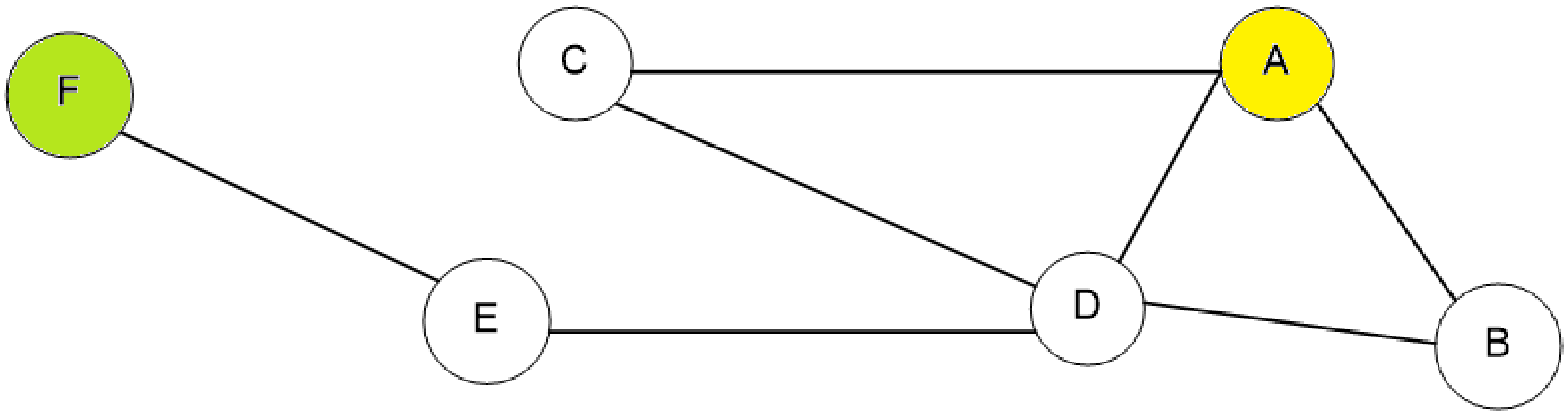
Greedy Search Solution (I)

- Initial city is A. Goal is F. The evaluation function expands C because its location is closer to the location of the Goal



Greedy Search Solution (II)

- Not in all cases the evaluation function gives a good estimation.



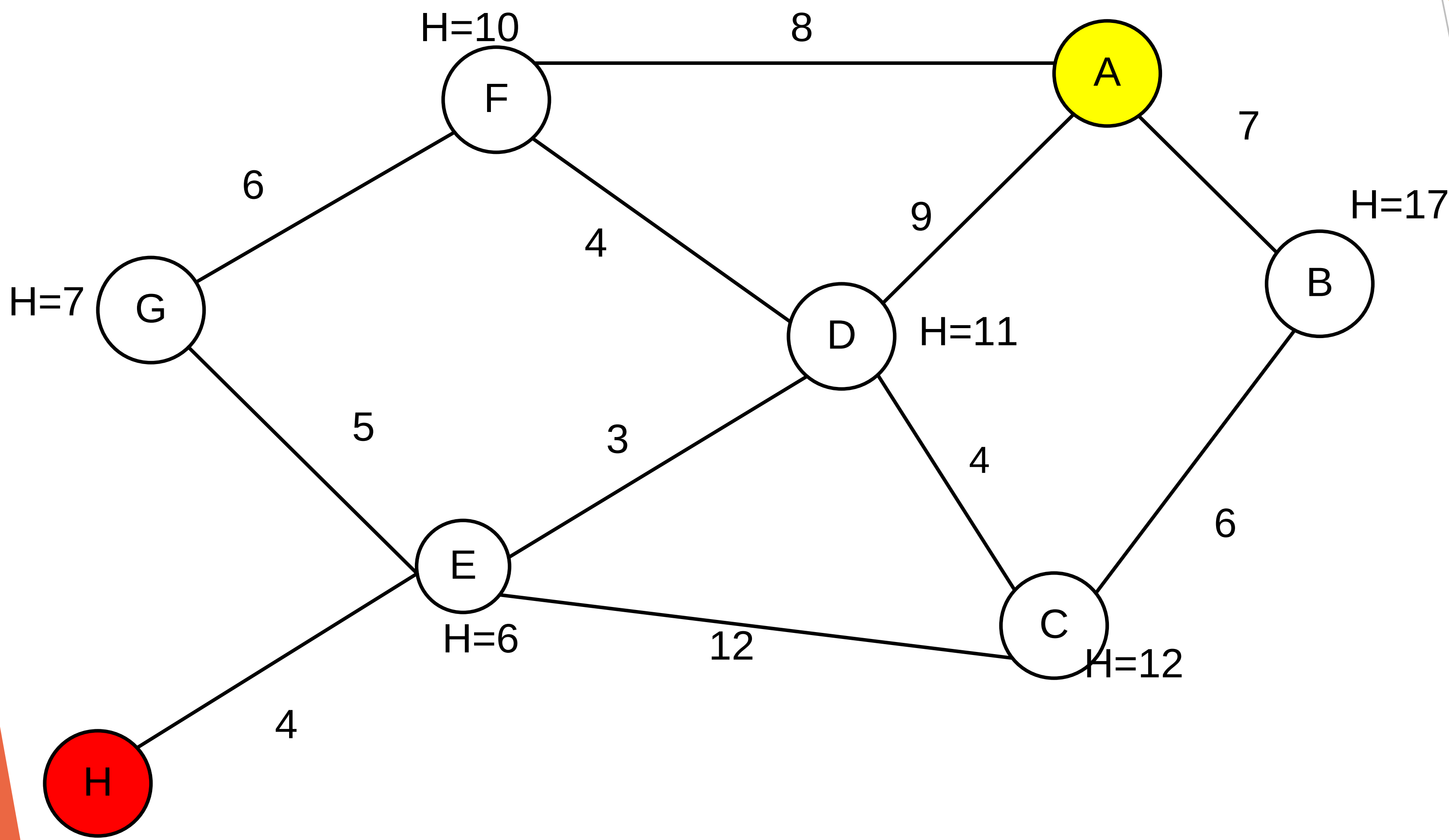
A* Search

- Greedy search minimizes the estimated cost to the goal, h(n).
- Uniform-cost search minimizes the cost of the path so far, g(n).
- A* combines these two strategies to get the advantages of both.

$$f(n) = g(n) + h(n)$$

Example: A* Search

- Assume the estimation to goal [h(.)] and the distance between cities are as shown in the figure.
- Initial city is A
- Goal is H


H=10
8
F
A
7
6
H=17
4
9
B
H=7
G
D
H=11
5
3
4
6
E
C
H=6
12
H=12
4
H

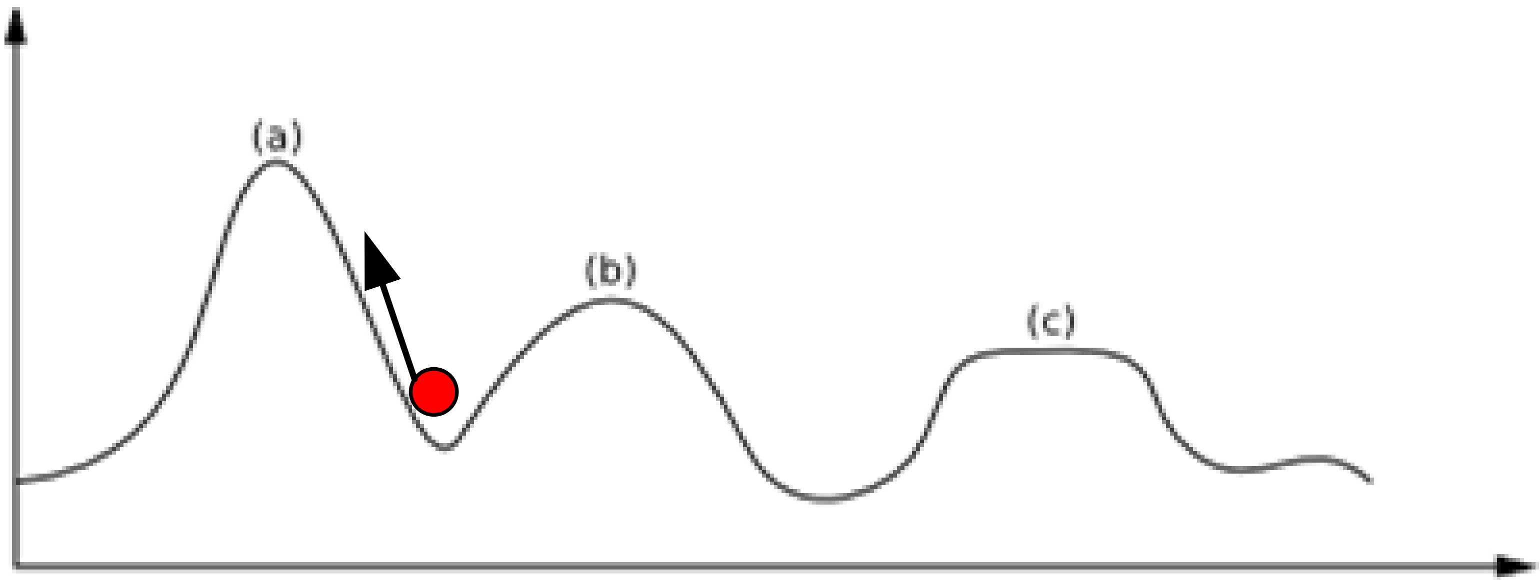
Iterative Deepening A* Search

- In un-informed searches iterative deepening is used as a useful technique for reducing memory requirements.
- We can use the same method with A* search
- Iterative A* search works like depth-first search, however, the maximum depth is increased at each step (iteration).

Hill-Climbing Search

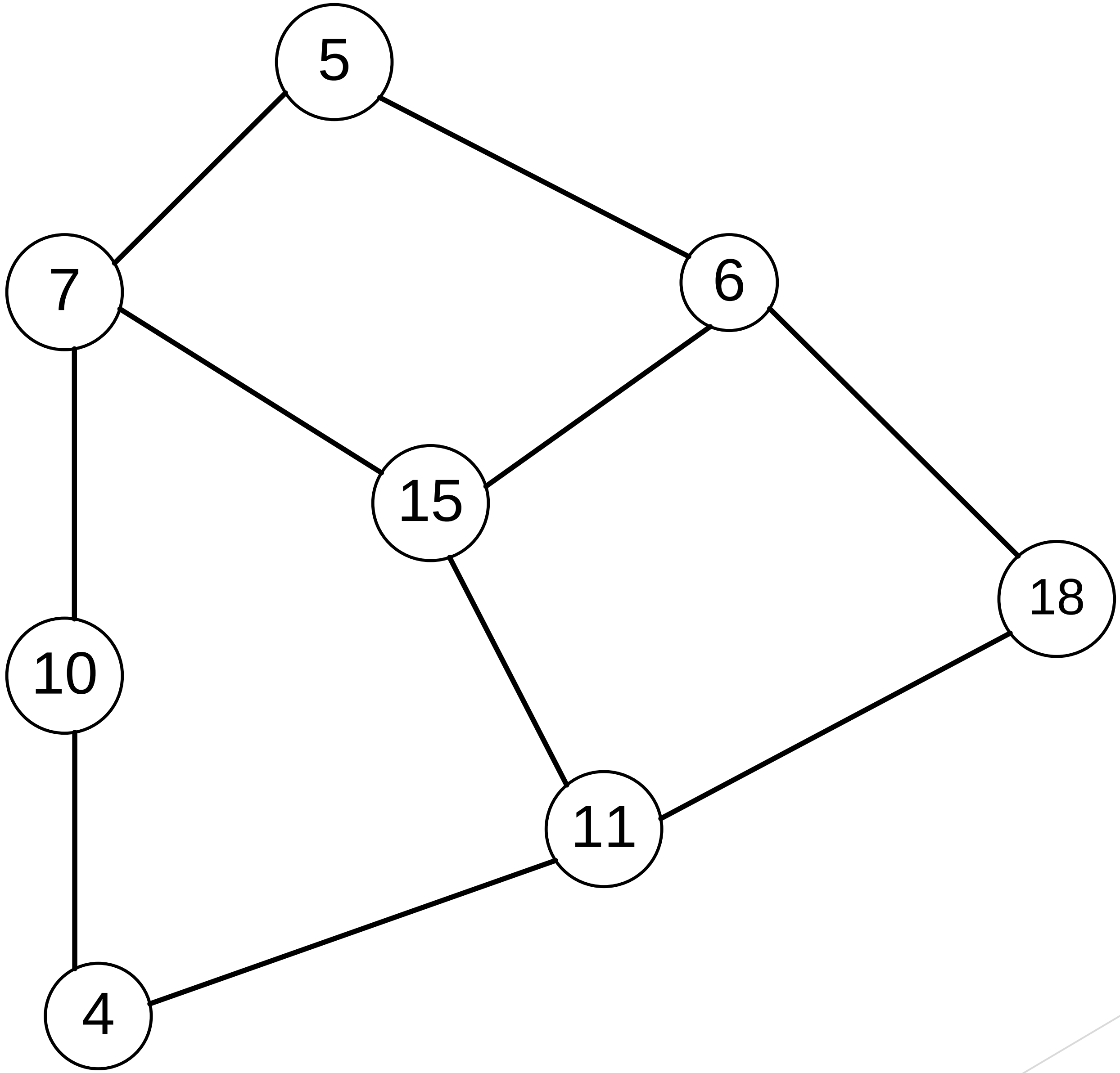
- The hill-climbing search algorithm moves in the direction of increasing values.
- The algorithm only follows the neighbors having larger values.
- The algorithm may stop at local maximum nodes.

Hill-Climbing Search



Example: Hill-Climbing Search

- Assume a graph of nodes with different values is given.
- Starting from an initial node, find the node with the maximum value.


5
7
6
15
18
10
11
4

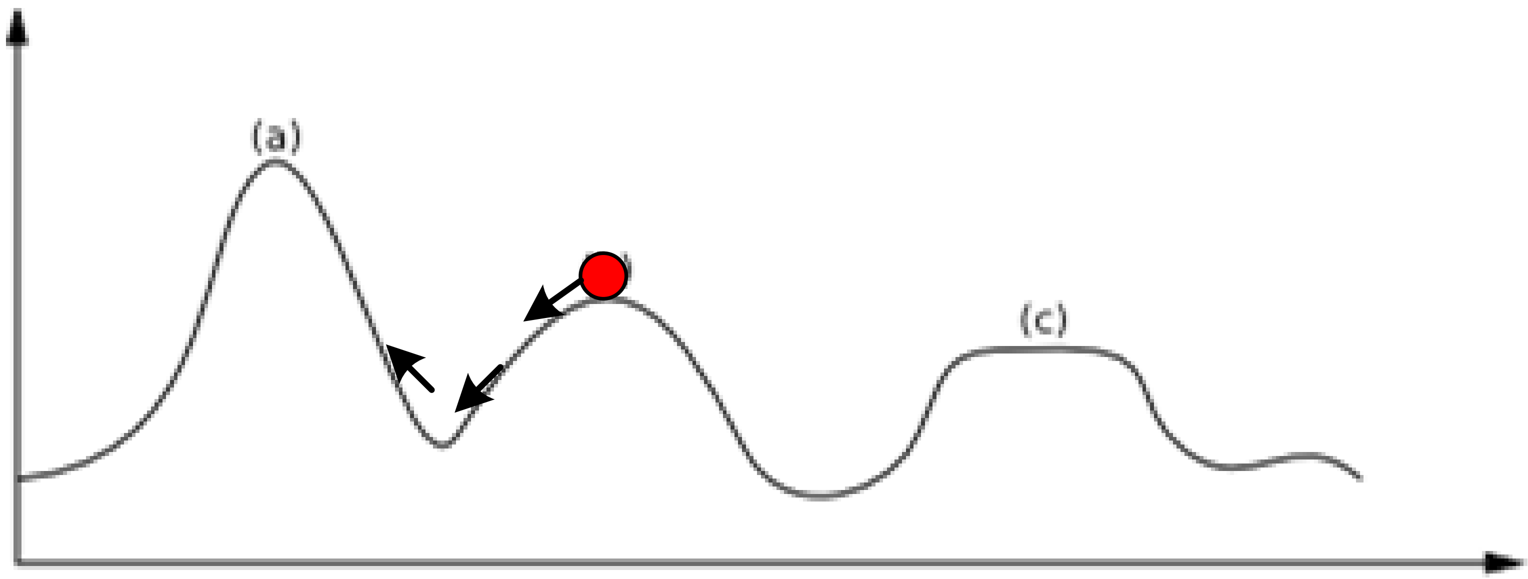
Problem with Hill-Climbing Search

- Hill-Climbing stops at local maximums.
- In the previous example it finds 15 as the maximum value. But the node with 18 has the maximum value.
- As a solution when the hill-climbing stops at a maximum point we re-start it from a random point.
- If the algorithm goes to the same point, that point is probably a global maximum.

Simulated Annealing

- Simulated Annealing solves the local maximum problem in hill-climbing algorithm by allowing it to follow down-hill paths in a limited range.

Example: Simulated Annealing



Game Playing

- Game playing can be considered as a graph search problem.
- Each node of the graph is a state in the game.
- The goal nodes are the states when we win the game.

The Difference between Games and Search Problems

- In the normal search problems, the search algorithm tries to find the best path to the goal state.
- In a game, the opponents make moves in turns
- Therefore, in one step we want to maximize a value, while in the next step we want to minimize it.
- An algorithm named min-max algorithm is used for game playing.

Questions?