THE UNIVERSITY of TENNESSEE KNOXVILLE

Lecture 21: Multistep decisions

Instructor: Sergei V. Kalinin

Multistage decisions

There is no computation without representation!

- Learn algorithms: from simple to complex ones
- Learn new concepts that emerge as a part of ML, e.g. VAE, decision making, etc.
- Learn how to represent the system in the ML setting
- Learn how to formulate the reward functions

Multistage decisions: agent

Autonomous car (or robot, or even Roomba):

- **Percepts:** inputs from sensors, cameras, GPS system, etc
- **Actions:** steering, acceleration, controls
- **Goals:** reach destination, safety, comfort, maximize profit, etc…
- **Environment:** street, highway, mountain trail, ….
- Goals specifiable by performance measure defining a numerical value for any environment history (reward)
- **Rational action:** whichever action maximizes the expecte dvalue of the performance measure given the percept sequence to date
- **Rational** does not mean omniscient, clairvoyant, or successful

Multistage decisions

	Solitaire	Backgammon	Internet shopping	Taxi
Accessible??	Yes	Yes	No	No
Deterministic??	Yes	No	Partly	No
Episodic??	No	No	No	No
Static??	Yes	Semi	Semi	No
Discrete??	Yes	Yes	Yes	No

- The environment type determines the agent design
- Real world is inaccessible, stochastic, sequential, dynamic, continuous

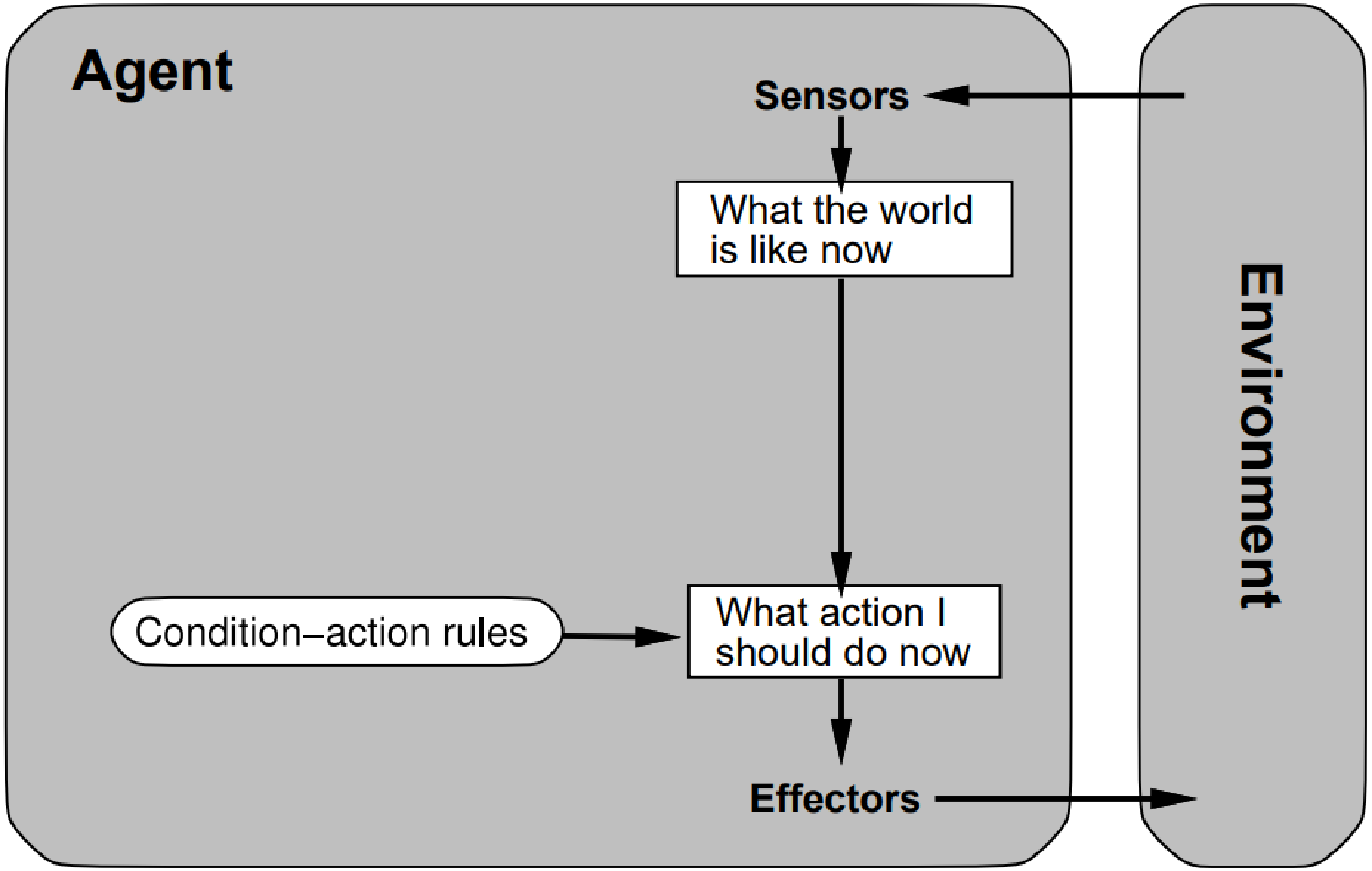
Agent functions

- Agent function is completely specified by policy (agent function) mapping percept sequence to actions
- Most primitive policy will be look-up table (i.e. simple enumeration)

Agent types:

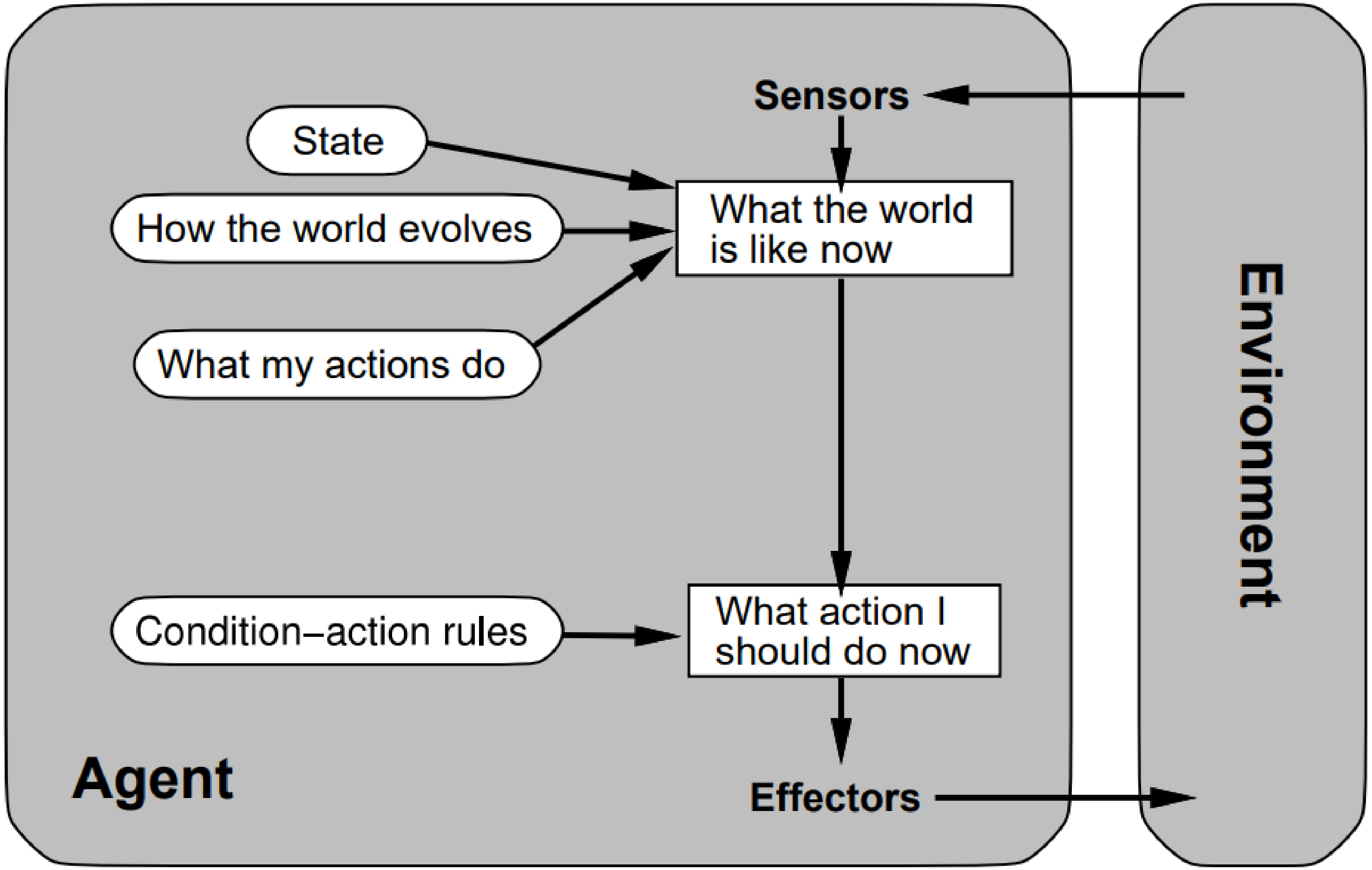
- Simple reflex agents
- Reflex agents with state
- Goal-based agents
- Utility-based agents

Simple reflex agents



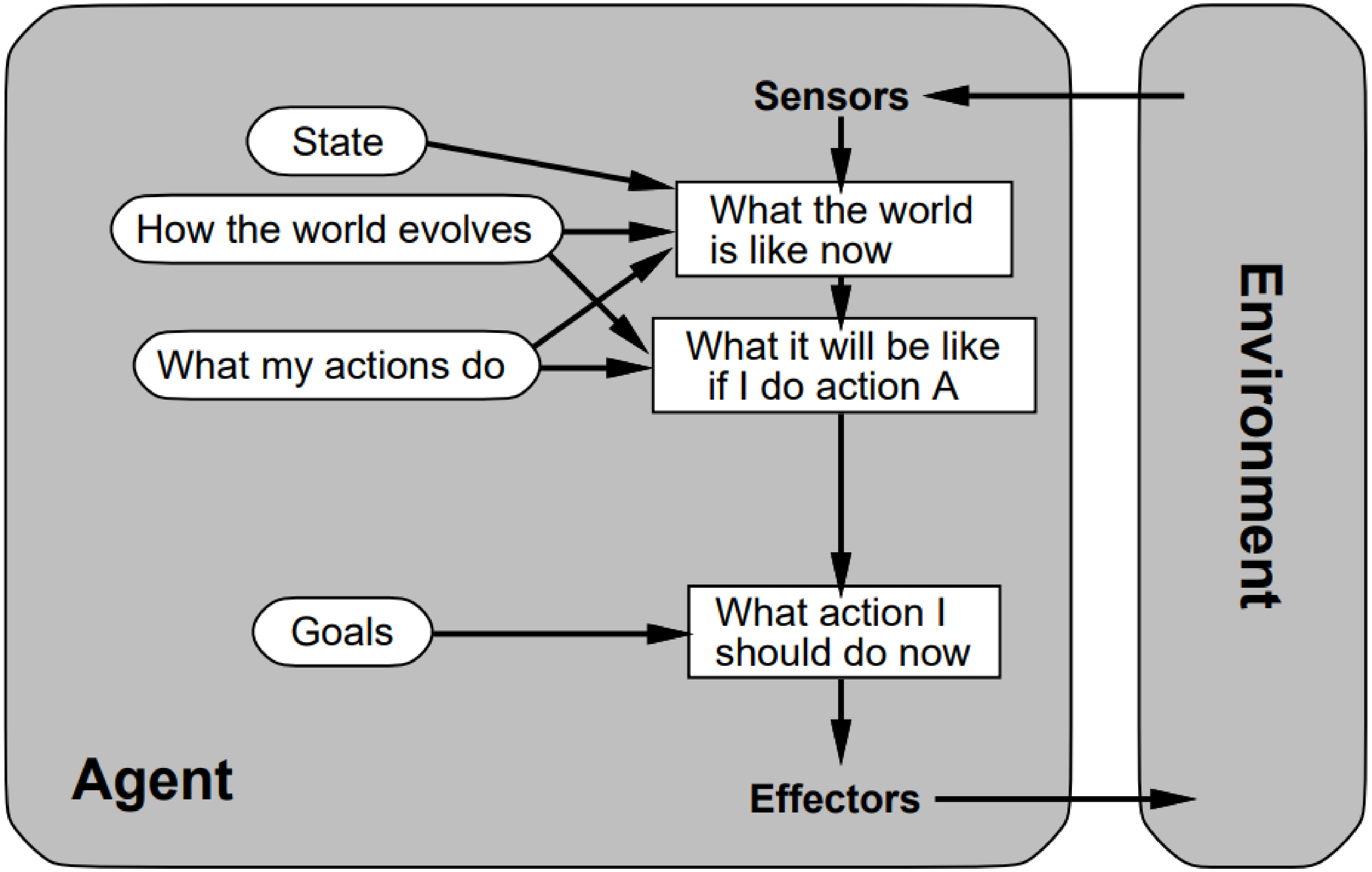
AIMA Slides ©Stuart Russell and Peter Norvig, 1998

Reflex agents with state

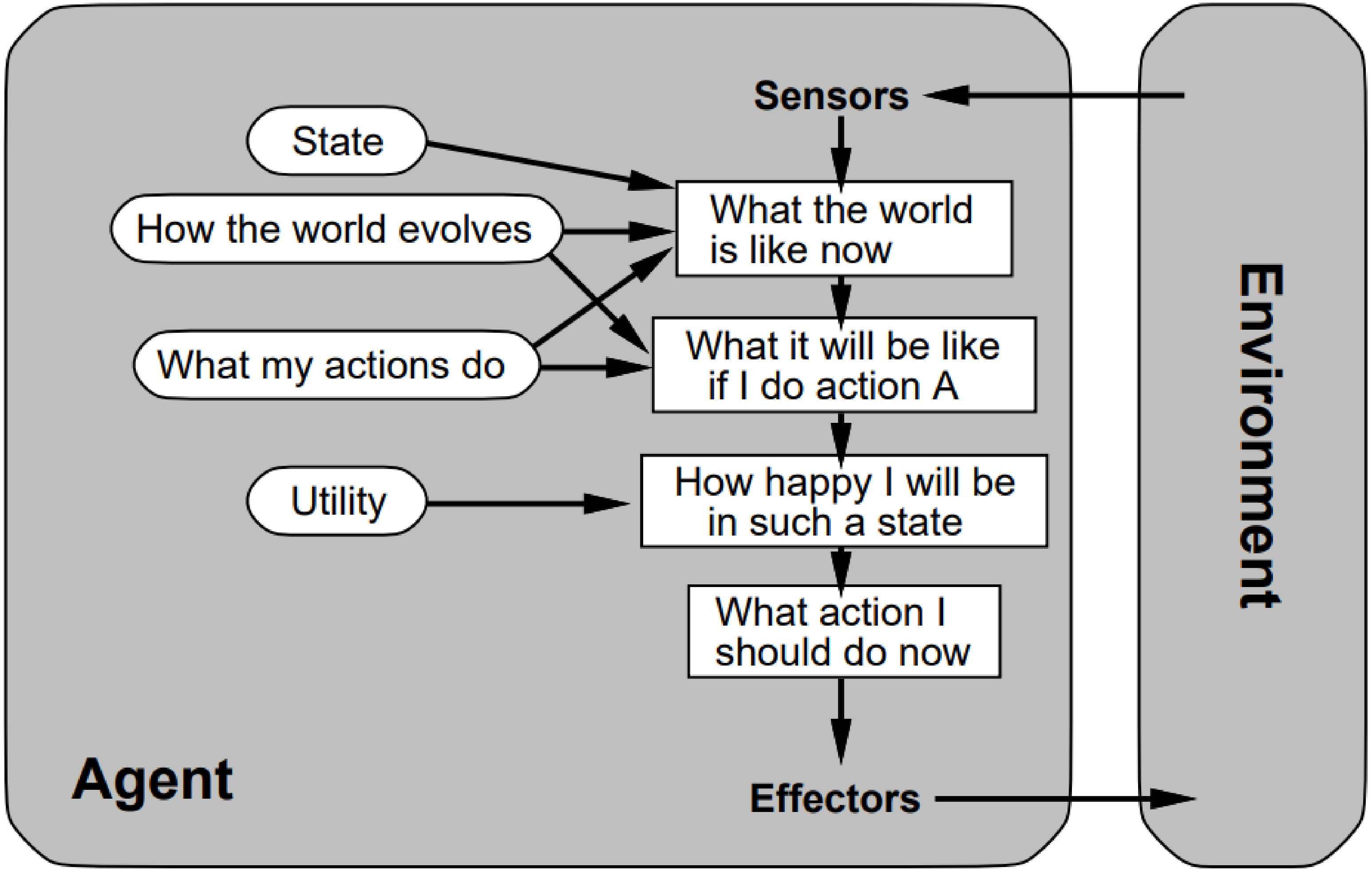


AIMA Slides ©Stuart Russell and Peter Norvig, 1998

Goal based agents



Utility based agents



From agents to search

- Rational agents need to perform sequences of actions to achieve goals
- Intelligent behavior can be generated by having a look-up table or reactive policy that tells the agent what to do in every circumstance, but:
    - Such a table or policy is difficult to build
    - All contingencies must be anticipated
- A more general approach is for the agent to have knowledge of the world and how its actions affect it and be able to simulate execution of actions in an internal model of the world in order to determine a sequence of actions that will accomplish its goals
- This is the general task of problem solving and is typically performed by searching through an internally modelled space of world states

From agents to search

- **Given:**
  - An initial state of the world
  - A set of possible actions or operators that can be performed.
  - A goal test that can be applied to a single state of the world to determine if it is a goal state.

- **Find:** A solution stated as a path of states and operators that shows how to transform the initial state into one that satisfies the goal test.

- The initial state and set of operators implicitly define a state space of states of the world and operator transitions between them. May be infinite.

From agents to search

- **Path cost:** a function that assigns a cost to a path, typically by summing the cost of the individual operators in the path. May want to find minimum cost solution.

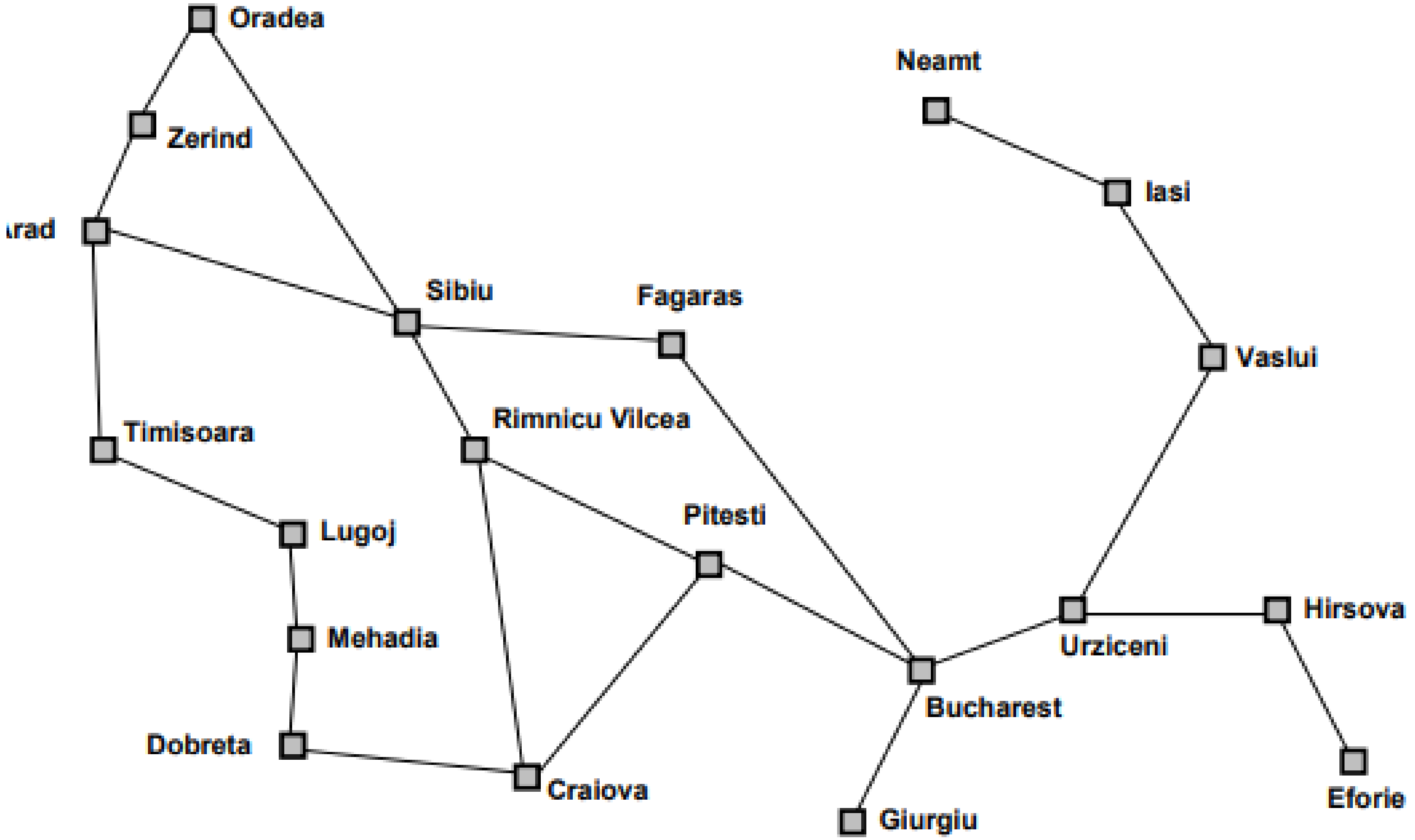
- **Search cost:** The computational time and space (memory) required to find the solution.

- Generally there is a trade-off between path cost and search cost and one must satisfice and find the best solution in the time that is available

From agents to search

- Rational agents need to perform sequences of actions to achieve goals
- Intelligent behavior can be generated by having a look-up table or reactive policy that tells the agent what to do in every circumstance, but:
    - Such a table or policy is difficult to build
    - All contingencies must be anticipated
- A more general approach is for the agent to have knowledge of the world and how its actions affect it and be able to simulate execution of actions in an internal model of the world in order to determine a sequence of actions that will accomplish its goals
- This is the general task of problem solving and is typically performed by searching through an internally modelled space of world states

Route finding

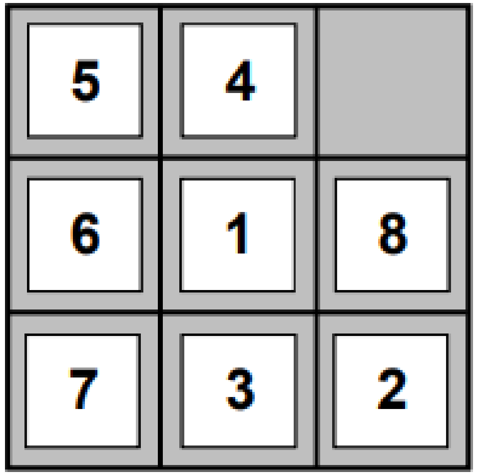


- Initial state: Arad
- Goal state: Bucharest
- Path cost: Number of intermediate cities, distance traveled, expected travel tin

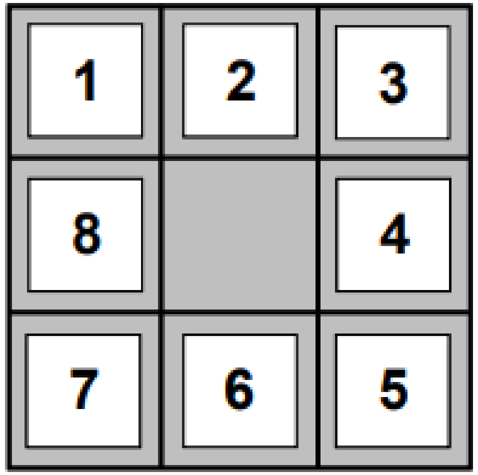
From agents to search

- Constraints and constraint-satisfaction problems

- 8-puzzle (sliding tile puzzle)

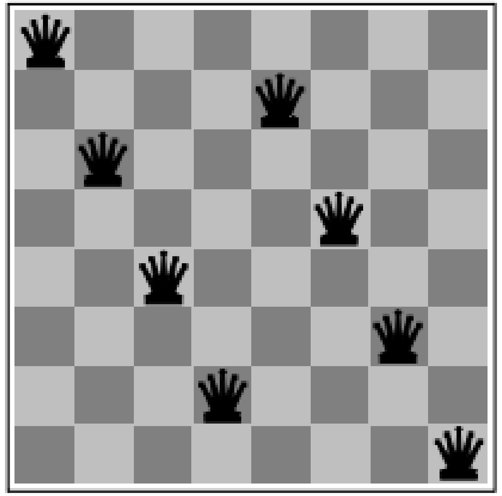


Start State



Goal State

- 8-queens problem (N-queens problem)

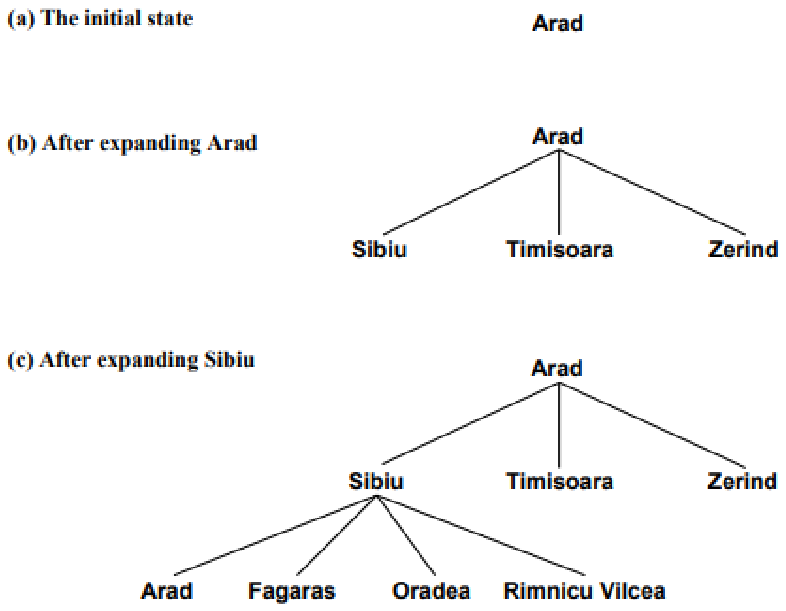


More realistic problems:

- Route finding (Google Maps)
- Travelling salesman problem
- Supply chain planning
- Integrated circuit design
- Experiment planning

Search concepts

- A state can be expanded by generating all states that can be reached by applying a legal operator to the state
- State space can also be defined by a successor function that returns all states produced by applying a single legal operator
- A search tree is generated by generating search nodes by successively expanding states starting from the initial state as the root



- A search node in the tree can contain
  - Corresponding state
  - Parent node
  - Operator applied to reach this node
  - Length of path from root to node (depth)
  - Path cost of path from initial state to node

Search strategies

- Easiest way to implement various search strategies is to maintain a queue of unexpanded search nodes

- Different strategies result from different methods for inserting new nodes in the queue

Search strategies

Properties of search strategies

- Completeness: does it always find solution if one exists?
- Time Complexity: number of nodes generated/expanded
- Space Complexity: maximum number of nodes in memory
- Optimality: does it always find least cost solution?

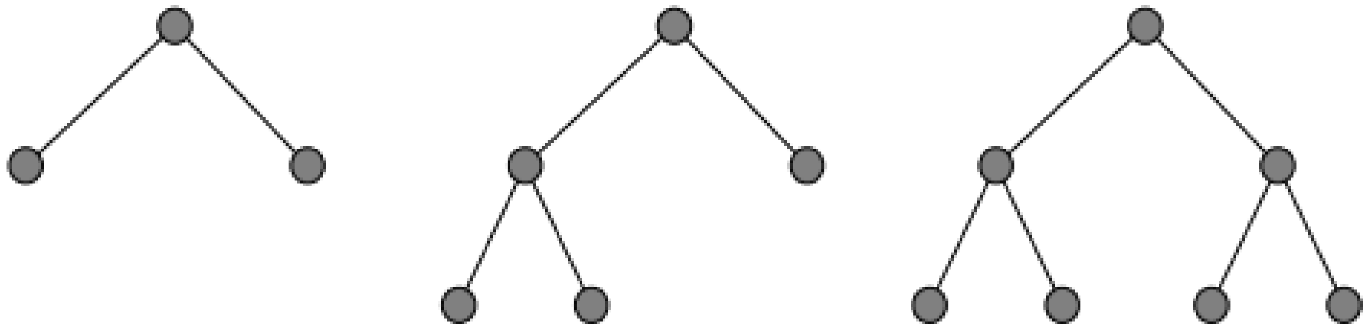
Informed vs. uninformed:

- Uninformed search strategies (blind, exhaustive, brute force) do not guide the search with any additional information about the problem.

- Informed search strategies (heuristic, intelligent) use information about the problem (estimated distance from a state to the goal) to guide the search

Breadth-first search

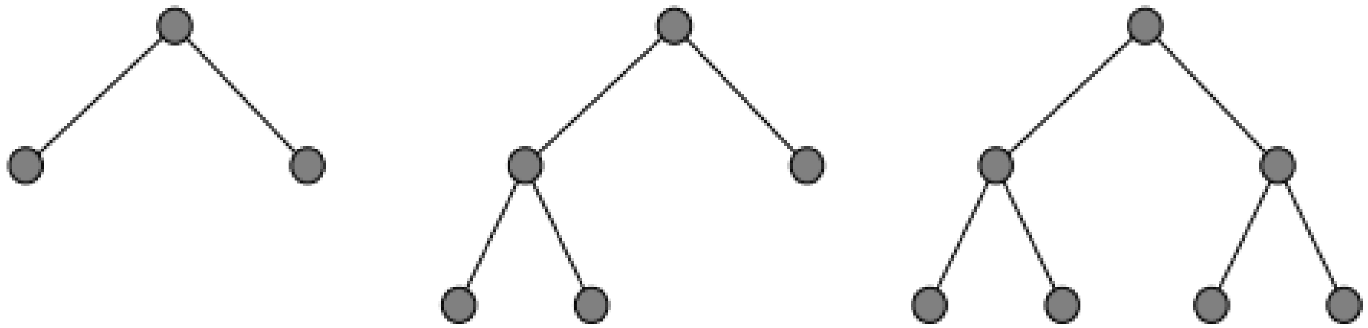
- Expands search nodes level by level, all nodes at level *d* are expanded before expanding nodes at level *d*+1



- Implemented by adding new nodes to the end of the queue
- Since eventually visits every node to a given depth, guaranteed to be complete
- Optimal provided path cost is a nondecreasing function of the depth of the node (e.g. all operators of equal cost) since nodes explored in depth order

Breadth-first search

- Expands search nodes level by level, all nodes at level *d* are expanded before expanding nodes at level *d*+1



- Implemented by adding new nodes to the end of the queue
- Since eventually visits every node to a given depth, guaranteed to be complete
- Optimal provided path cost is a nondecreasing function of the depth of the node (e.g. all operators of equal cost) since nodes explored in depth order

Complexity of breadth-first search

- Assume there are an average of b successors to each node, called the branching factor
- Therefore, to find a solution path of length d must explore

$$1 + b + b^2 + b^3 + \ldots + b^d$$

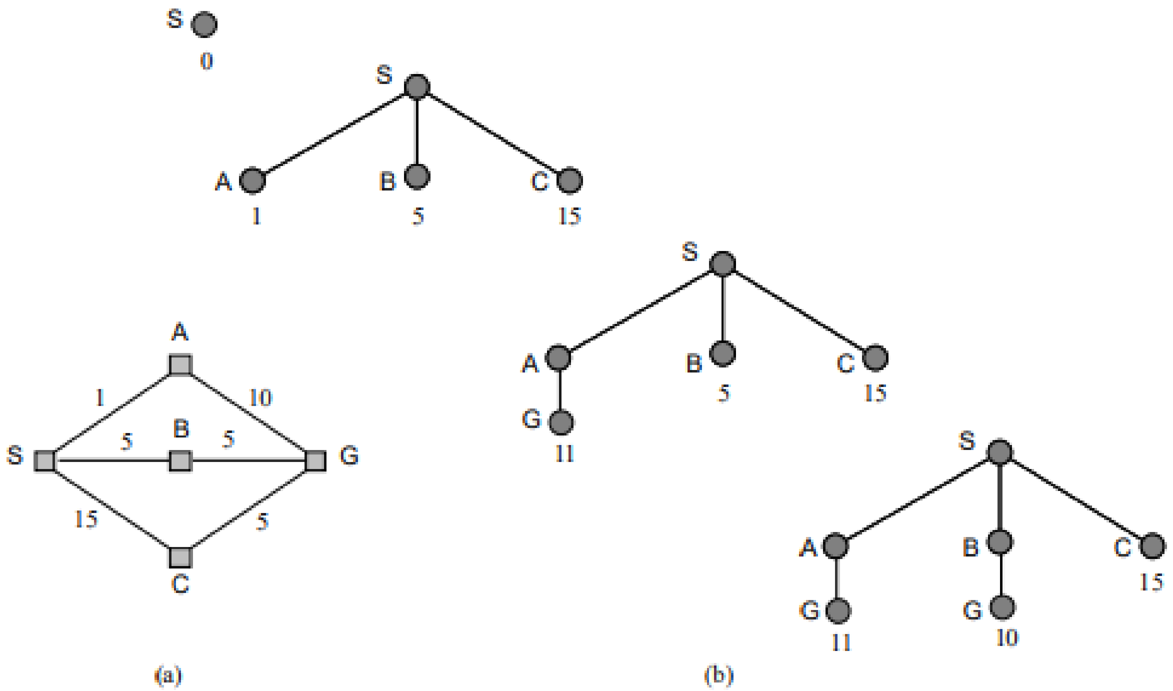
- Plus need b^d nodes in memory to store leaves in queue
- Assuming can expand and check 1000 nodes/sec and need 100 bytes/node storage , b=10

Depth	Nodes	Time	Memory
0	1	1 millisecond	100 bytes
2	111	.1 seconds	11 kilobytes
4	11,111	11 seconds	1 megabyte
6	10^6	18 minutes	111 megabytes
8	10^8	31 hours	11 gigabytes
10	10^{10}	128 days	1 terabyte
12	10^{12}	35 years	111 terabytes
14	10^{14}	3500 years	11,111 terabytes

Note memory is a bigger problem than time.

Uniform cost search

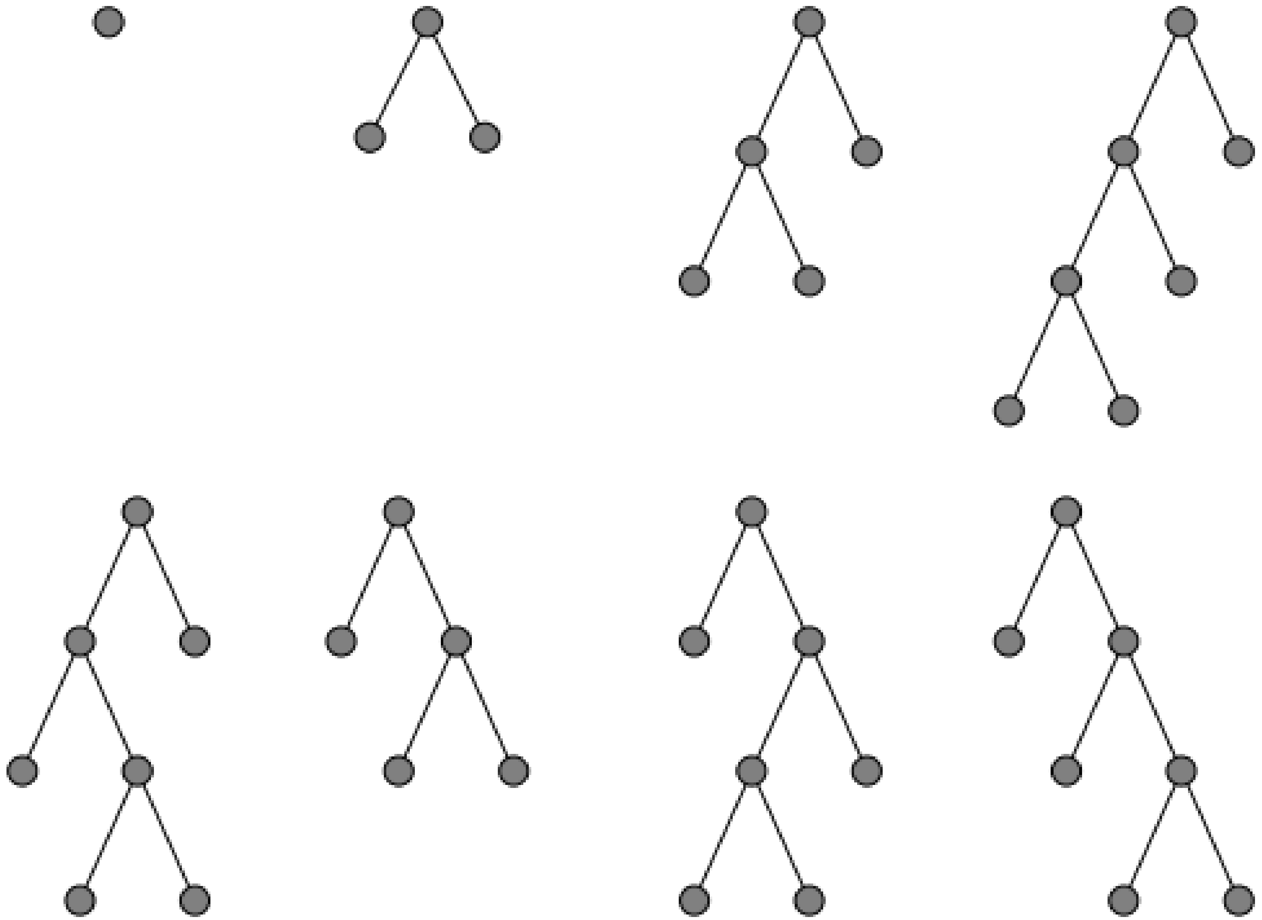
- Like breadth-first except always expand node of least cost instead of least depth (i.e. sort new queue by path cost)



- Do not recognize goal until it is the least cost node on the queue and removed for goal testing
- Therefore, guarantees optimality as long as path cost never decreases as a path increases (non-negative operator costs)

Depth-first search

- Always expand node at deepest level of the tree, i.e. one of the most recently generated nodes. When hit a dead-end, backtrack to last choice



Breadth-first search

- Not guaranteed to be complete since might get lost following infinite path

- Not guaranteed optimal since can find deeper solution before shallower ones explored

- Time complexity in worst case is still O(b^d) since need to explore entire tree. But if many solutions exist may find one quickly before exploring all of the space

- Space complexity is only O(b^m) where m is maximum depth of the tree since queue just contains a single path from the root to a leaf node along with remaining sibling nodes for each node along the path

- Can impose a depth limit, l, to prevent exploring nodes beyond a given depth. Prevents infinite regress, but incomplete if no solution within depth limit.

Iterative deepening

- Conduct a series of depth-limited searches, increasing depth-limit each time.
- Seems wasteful since work is repeated, but most work is at the leaves at each iteration and is not repeated.

Depth-first:

$$1 + b + b^2 + \ldots + b^{d-2} + b^{d-1} + b^d$$

Iterative deepening:

$$(d+1)1 + db + (d-1)b^2 + \ldots + 3b^{d-2} + 2b^{d-1} + 1b^d$$

Heuristic Search

- Heuristic or informed search exploits additional knowledge about the problem that helps direct search to more promising paths.
- A **heuristic function**, $h(n)$, provides an estimate of the cost of the path from a given node to the closest goal state
- Must be zero if node represents a goal state.
  - Example: Straight-line distance from current location to the goal location in a road navigation problem
- Many search problems are NP-complete so in the worst case still have exponential time complexity; however a good heuristic can:
  - Find a solution for an average problem efficiently.
  - Find a reasonably good but not optimal solution efficiently.