Problem Solving

Outline

- Problem-solving agents
- Problem types
- Problem formulation
- Example problems
- Basic search algorithms

Problem-solving agents use **atomic representations**

Atomic Agent

Input:

- Set of states
- Operators [and costs]
- Start state

Output:

- Path: start ⇒ a state satisfying goal test

- For **Shortest path problem** representation is graph and every state is node, edges are operators, actions you can apply to move from one state to another state.

- Chess playing, Robot Assembly are agent based complex problem.

- Data structure is not enough to represent the NP hard problem

Problem Representation

- Instead of defining different algorithms for different problems, a unified methodology can be used to solve a large class of problems.
- Description of the problem is input and an automated problem solving methodology i.e. systematic searching approach for different problems to generate the output.
- General representation of any problem using defined states.
- **State space representation** for path finding problem: start state, goal state, map, locations etc.

State Space Search

- State Space Search is used to model the problem.
- In the state space graph, **nodes** represent the configuration and **edges** represent the possible moves from one configuration to another.
- Any configuration can be modeled as a STATE.
- The configuration must be a VALID CONFIGURATION.
- STATE TRANSFORMATION RULES define the rules in moving from one state to another state thus producing the state space.
- However, state spaces are very large, and sometimes even infinite.

What is SEARCH?

- Suppose an agent can execute several actions immediately in a given state
- It doesn't know the utility of these actions
- Then, for each action, it can execute a sequence of actions until it reaches the goal
- The immediate action which has the best sequence (according to the performance measure) is then the solution
- Finding this sequence of actions is called search, and the agent which does this is called the problem-solver.

- NB: Its possible that some sequence might fail, e.g., getting stuck in an infinite loop, or unable to find the goal at all.

Linking Search to Trees

- You can begin to visualize the concept of a graph
- Searching along different paths of the graph until you reach the solution
- The nodes are the states
- The whole graph can be the state space
- The links are equivalent to the actions……

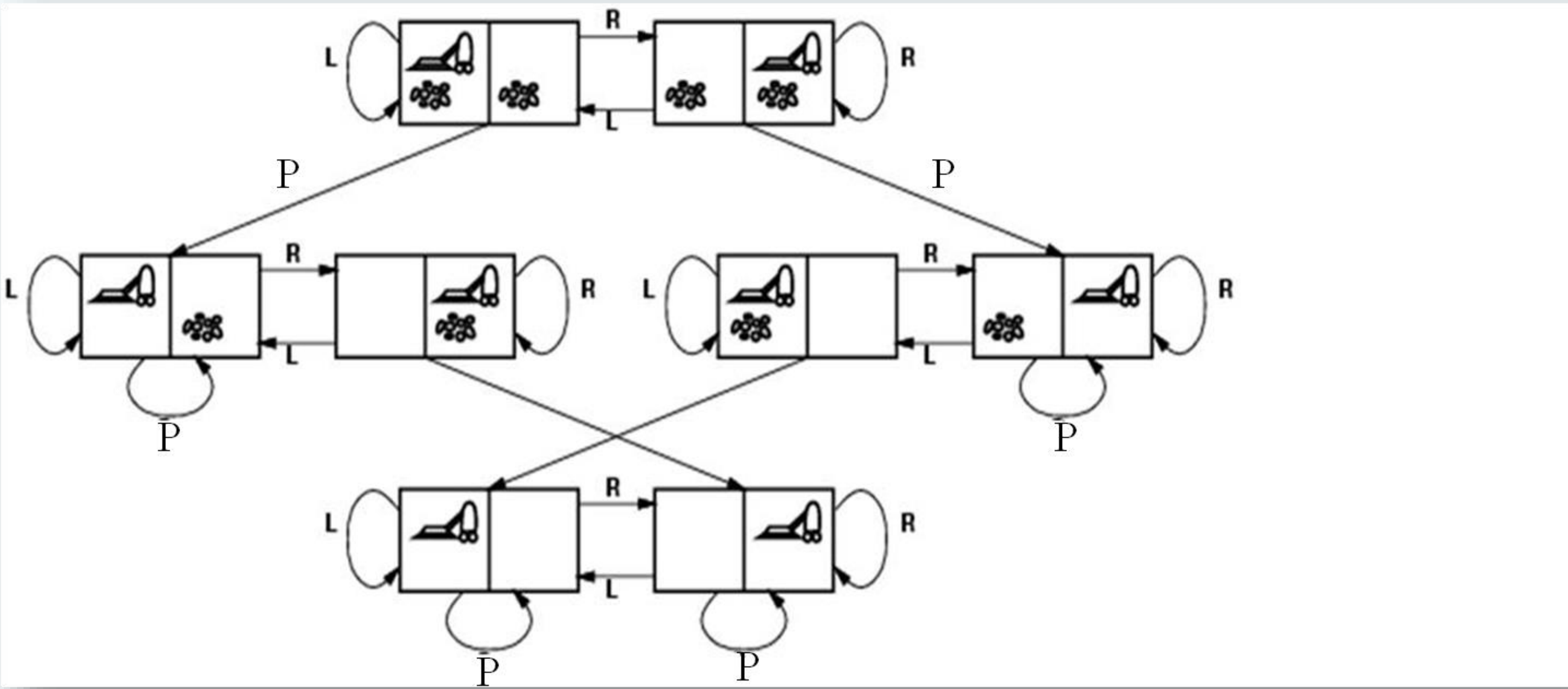
Search Problem Types

- Static: The configuration of the graph (the city map) is unlikely to change during search
- Observable: The agent knows the state (node) completely, e.g., which city I am in currently
- Discrete: Discrete number of cities and routes between them
- Deterministic: Transiting from one city (node) on one route, can lead to only one possible city
- Single-Agent: We assume only one agent searches at one time, but multiple agents can also be used.

Problem-Solver Formulation

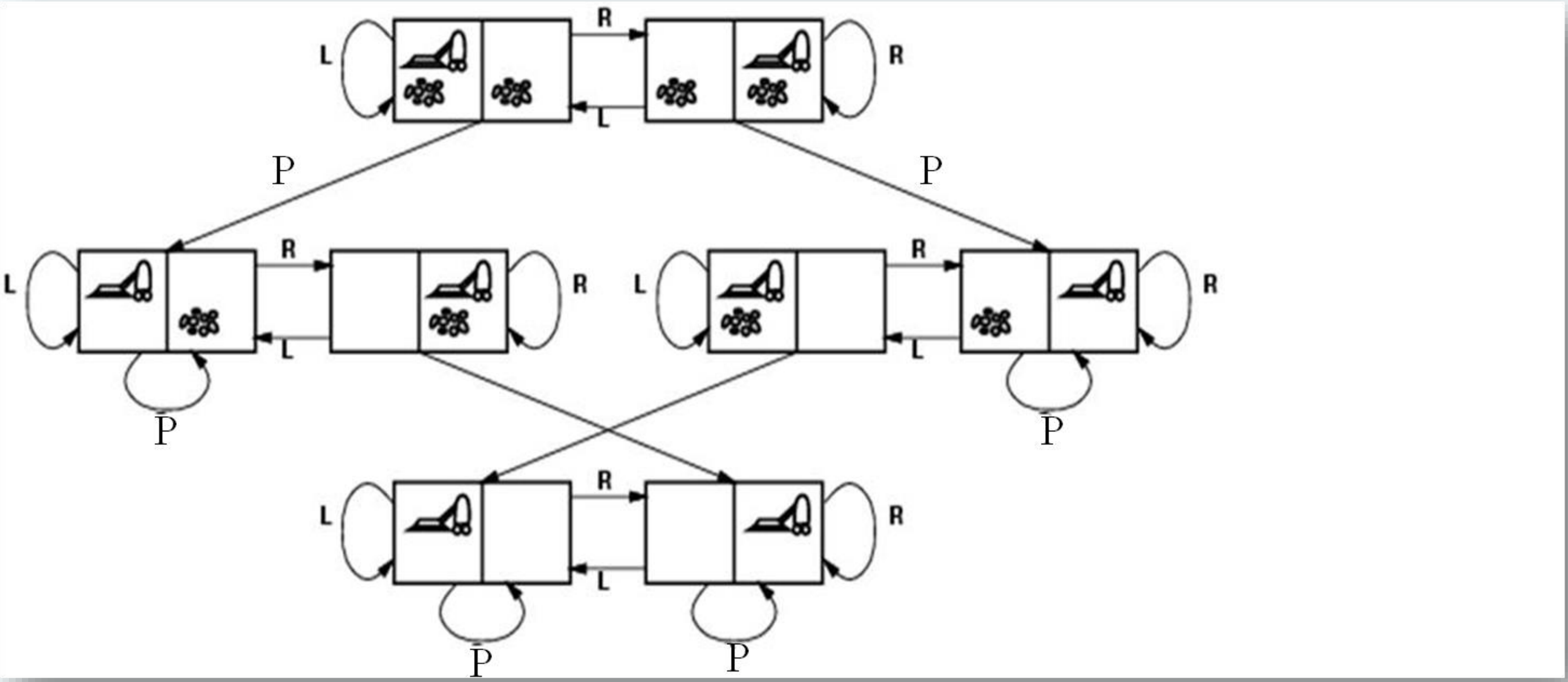
- A problem is defined by five items:
  1. An Initial state
  2. Possible actions available, ACTIONS(s) returns the set of actions that can be executed in s state.
  3. A successor function $S(x)$ = the set of all possible {Action–State} pairs from some state
  4. Goal test, can be
     - explicit, e.g., x = Bucharest
     - implicit, e.g., *Checkmate*(x)
  5. Path cost (additive)
     - e.g., sum of distances, number of actions executed, etc.
     - $c(x,a,y)$ is the step cost, assumed to be ≥ 0
- A solution is a sequence of actions leading from the initial state to a goal state.

Vacuum World State Space Graph



- States? Actions?
- Goal test? Path cost?

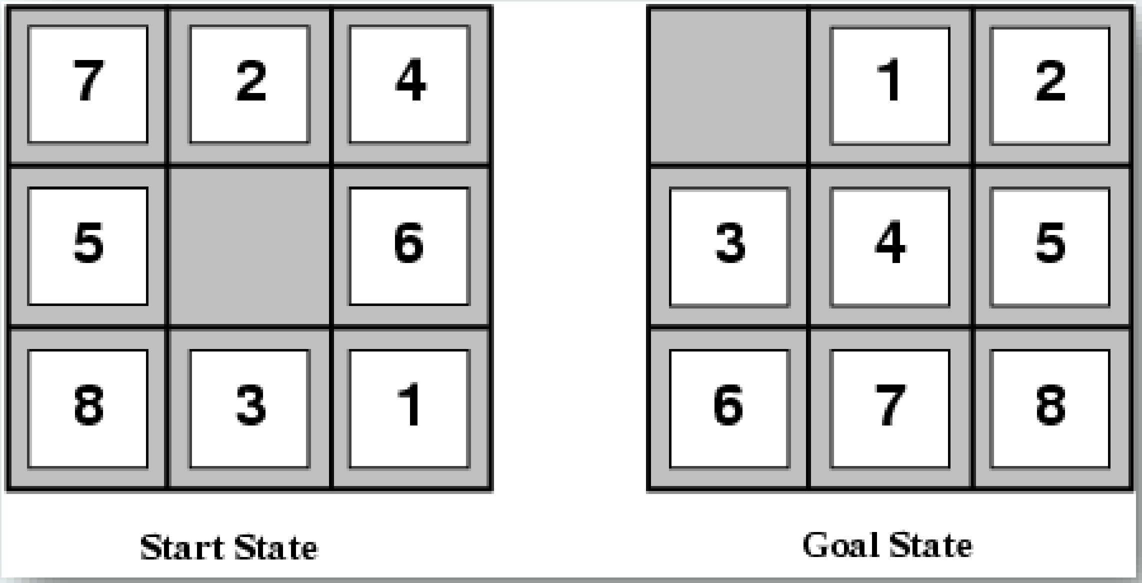
Vacuum World State Space Graph



- States? dirt and robot location
- Actions? *Left*, *Right*, *Pick*
- Goal test? no dirt at all locations
- Path cost? 1 per action

Example: The 8-puzzle

- States?
- Actions?
- Goal test?
- Path cost?

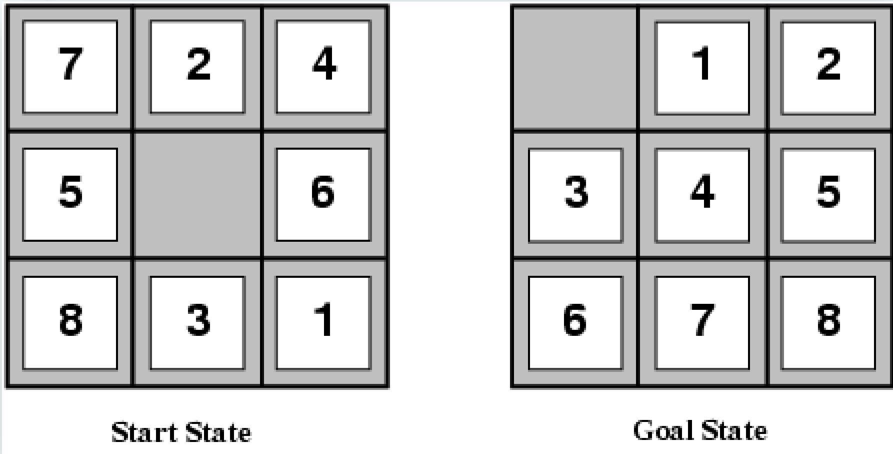


The **8-puzzle, consists of a 3×3 board with** eight numbered tiles and a blank space.

A tile adjacent to the blank space can slide into the space.

The object is to reach a specified goal state.

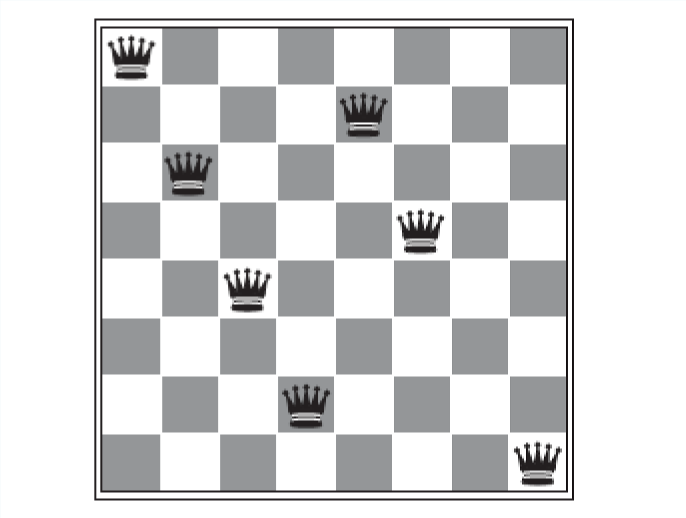
Example: The 8-puzzle



- States? locations of tiles
- Actions? move blank left, right, up, down
- Goal test? = goal state (given)
- Path cost? 1 per move

• The 8-puzzle belongs to the family SLIDING-BLOCK of **sliding-block puzzles, which are often used as** PUZZLES test problems for new search algorithms in AI.
• This family is known to be NP-complete,
• The 8-puzzle has 9!/2=181, 440 reachable states 15-puzzle (on a 4×4 board) has around 1.3 trillion states
• Random instances can be solved optimally in a few milliseconds by the best search algorithms.
• The 24-puzzle (on a 5 × 5 board) has around 1025 states, and random instances take several hours to solve optimally.

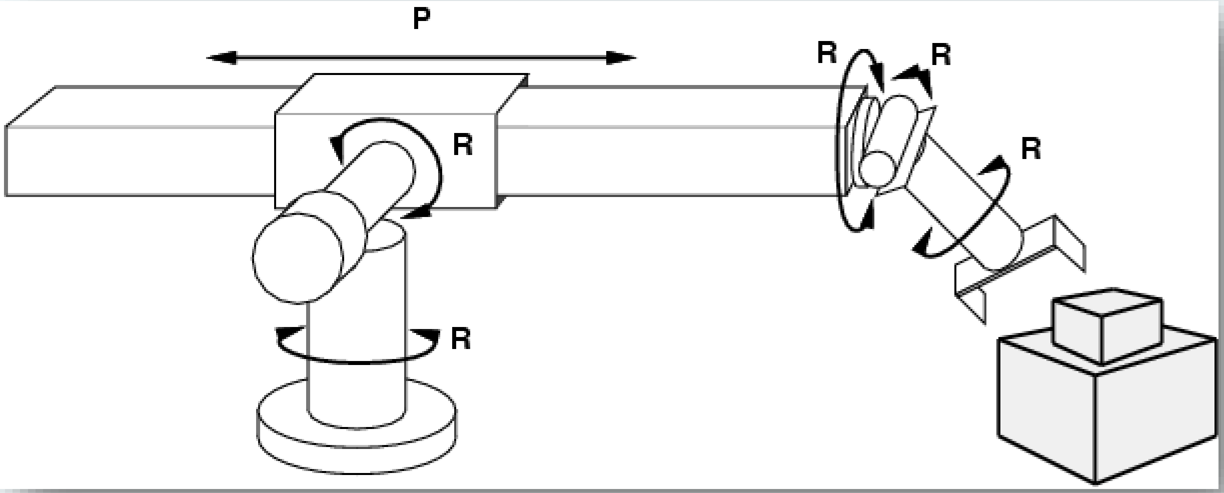
The goal of the **8-queens problem is to place eight queens on a chessboard such that** no queen attacks any other. (A queen attacks any piece in the same row, column or diagonal.)



- States: Any arrangement of 0 to 8 queens on the board is a state.
- Initial state: No queens on the board.
- Actions: Add a queen to any empty square.
- Transition model: Returns the board with a queen added to the specified square.
- Goal test: 8 queens are on the board, none attacked.

In this formulation, we have $64 \cdot 63 \cdots 57 \approx 1.8\times10^{14}$ possible sequences to investigate

Example: Robotic Assembly



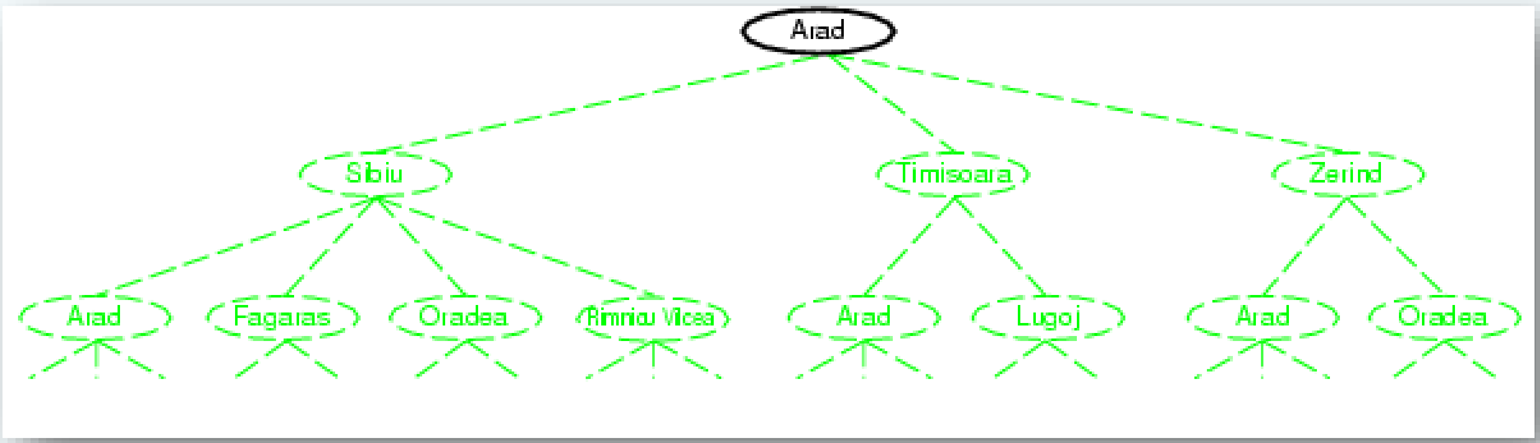
- States?: real-valued coordinates of robot joint angles, parts of the object to be assembled, current assembly
- Actions?: continuous motions of robot joints
- Goal test?: complete assembly
- Path cost?: time to execute

Tree Search Algorithms

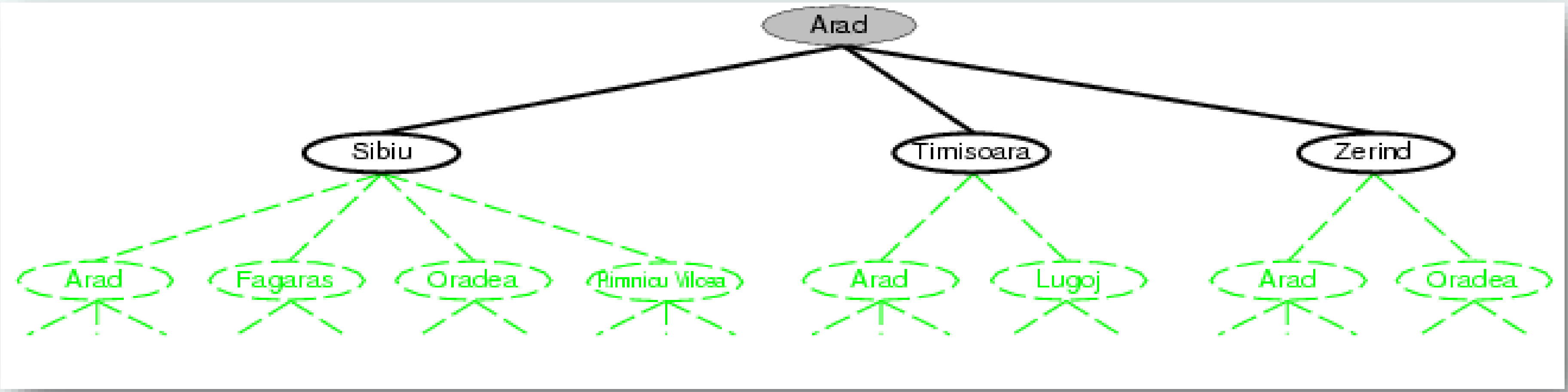
- Basic idea:
  - Offline (not dynamic), simulated exploration of state space by generating successors of already-explored states (a.k.a. expanding the states)
  - The expansion strategy defines the different search algorithms.

```
function TREE-SEARCH( problem, strategy) returns a solution, or failure
  initialize the search tree using the initial state of problem
  loop do
      if there are no candidates for expansion then return failure
      choose a leaf node for expansion according to strategy
      if the node contains a goal state then return the corresponding solution
      else expand the node and add the resulting nodes to the search tree
```

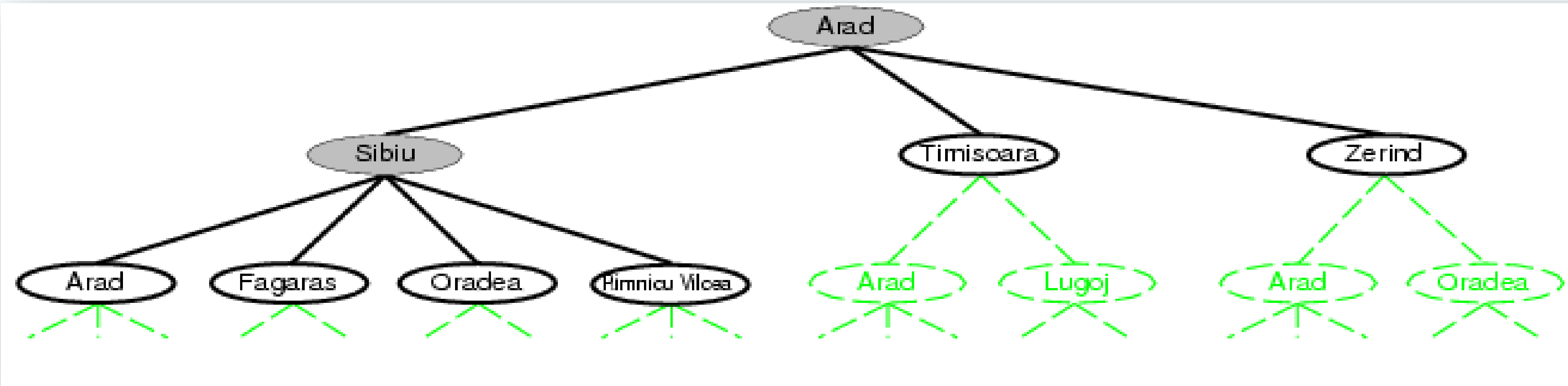
Tree search example



Tree search example

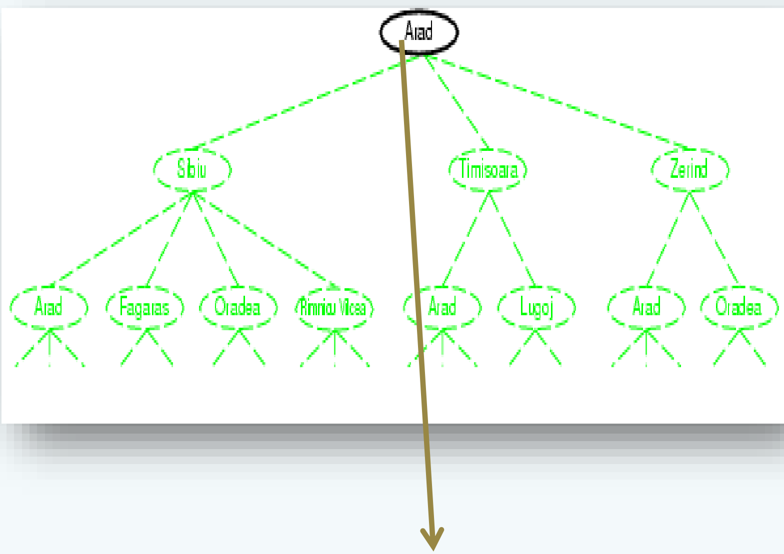


Tree search example

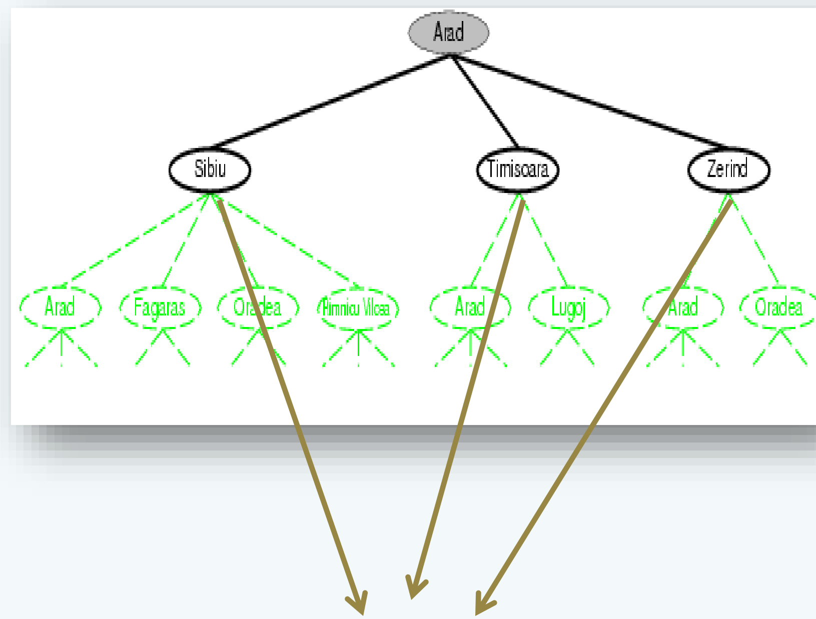


Fringe

- Fringe: The collection of nodes that have been generated but not yet expanded
- Each element of the fringe is a leaf node, with (currently) no successors in the tree
- The search strategy defines which element to choose from the fringe



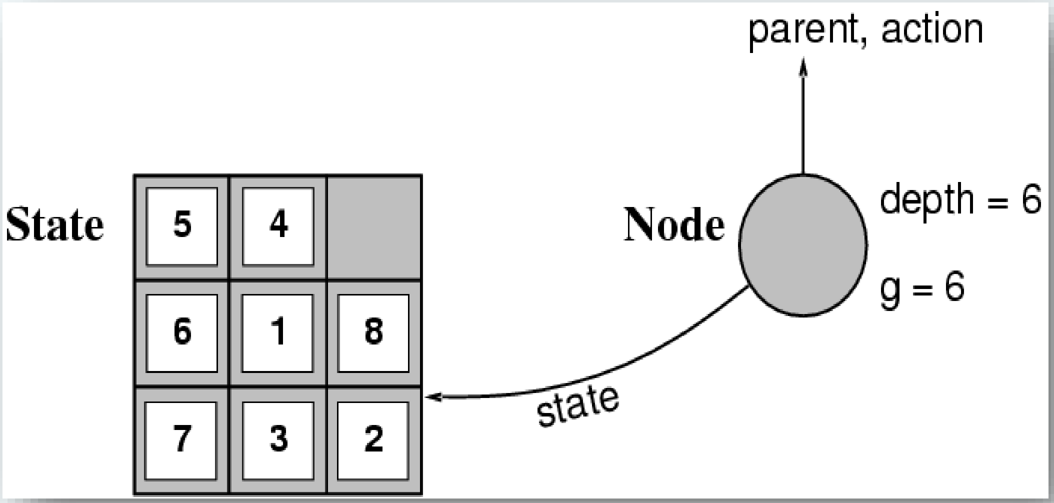
fringe



fringe

Implementation: States vs. Nodes

- A state is a representation of a physical configuration
- A node is a data structure constituting part of a search tree includes state, parent node, action, path cost *g(x)*, depth
- The Expand function creates new nodes, filling in the various fields and using the SuccessorFn of the problem to create the corresponding states.



Search Strategies

- A search strategy is defined by picking the order of node expansion
- Strategies are evaluated along the following dimensions:
  - Completeness: Does it always find a solution if one exists?
  - Time complexity: Number of nodes generated
  - Space complexity: Maximum number of nodes in memory
  - Optimality: Does it always find a least-cost solution?
- Time and space complexity are measured in terms of
  - *b*: maximum no. of successors of any node
  - *d*: depth of the shallowest goal node
  - *m*: maximum length of any path in the state space.

Types of search algorithms

Based on the search problems we can classify the search algorithms into uninformed (Blind search) search and informed search (Heuristic search) algorithms.

Uninformed/Blind Search:

The uninformed search does not contain any domain knowledge such as closeness, the location of the goal.

- It operates in a brute-force way as it only includes information about how to traverse the tree and how to identify leaf and goal nodes.

Breadth-first search
Uniform cost search
Depth-first search
Iterative deepening depth-first search
Bidirectional Search

Informed Search

- Informed search algorithms use domain knowledge.
- In an informed search, problem information is available which can guide the search.
- Informed search strategies can find a solution more efficiently than an uninformed search strategy.
- Informed search is also called a Heuristic search.
- A heuristic is a way which might not always be guaranteed for best solutions but guaranteed to find a good solution in reasonable time.

An example of informed search algorithms is a traveling salesman problem.
Greedy Search
A* Search