, 2022

Solving problems by Search

Goal-based Agents

Atomic-Factored-Structured

Dr. Bilal Hoteit

Introduction to Artificial Intelligence

Table of contents

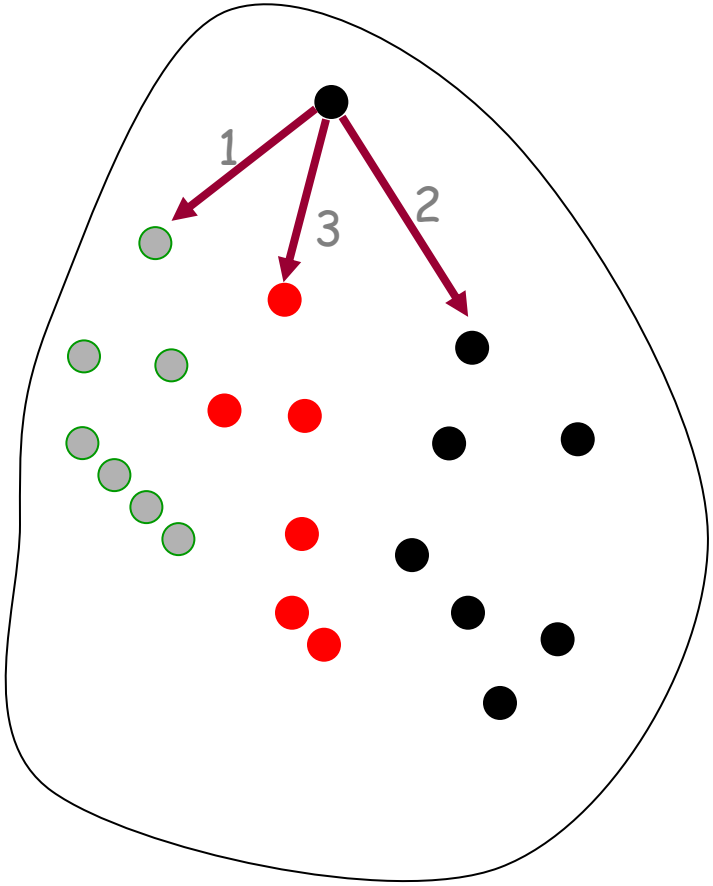
Search and AI

- Search methods are ubiquitous in AI systems. They often are the backbones of both core and peripheral modules.
- An autonomous robot uses search methods:
  a. to decide which actions to take and which sensing operations to perform
  b. to quickly anticipate and prevent collision.
  c. to plan trajectories.
  d. to interpret numerical datasets (sensors) into symbolic representations.
  e. to diagnose why something did not happen as expected.
  f. etc...

Search and AI

- Search plays a key role in many applications, e.g.:
  a. Route finding: airline travel, networks.
  b. Package/mail distribution.
  c. Pipe routing.
  d. Comparison and classification of protein folds.
  e. Pharmaceutical drug design.
  f. Design of protein-like molecules.
  g. Video games.

Stating a Problem as a Search Problem



- State space S
- Initial state s0
- Successor function:

  $x \in S \rightarrow SUCCESSORS(x) \in 2S$
- Arc cost (duration, length)
- Goal test:
- $x \in S \rightarrow GOAL?(x) = T \text{ or } F$
- A solution is a path joining the initial to a goal node

- **State Graph is defined as follows:**
  a. **Each state is represented by a distinct node**
  b. **An arc connects a node s to a node s' if s' ∈ SUCCESSORS(s)**
  c. **The state graph may contain more than one connected component**

- **Successor Function is defined as follows:**
  a. **It implicitly represents all the actions that are feasible in each state.**
  b. **Only the results of the actions (the successor states) and their costs are returned by the function.**
  c. **The successor function is a "black box": its content is unknown**

- **Path Cost** is defined as follows:
  a. An arc cost is a positive number measuring the "cost" of performing the action corresponding to the arc. The cost of a path is the sum of the edge costs along this path
  b. We will assume that for any given problem the cost c of an arc always verifies: $c \geq \varepsilon > 0$, where ε is a constant

- **Solution** to the Search Problem:
  a. A solution is a path connecting the initial to a goal node (any one)
  b. The cost of a path is the sum of the edge costs along this path
  c. An optimal solution is a solution path of minimum cost
  d. There might be no solution !

Assumptions in Basic Search

- **The world is static**
- **The world is discretizable**
- **The world is observable**
- **The actions are deterministic**

In a Real world scenario, many of these assumptions should be removed, but the search mechanism is still remained an important problem-solving tool.

- **Problem types**

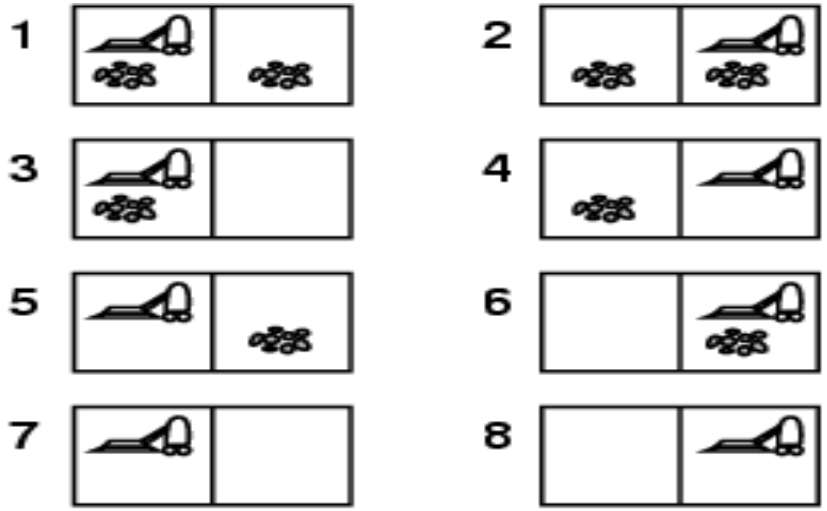
a. Deterministic, fully observable: single-state problem
   Agent knows exactly which state it will be in; solution is a sequence
b. Non-observable: sensor-less problem (conformant problem)
   Agent may have no idea where it is; solution is a sequence
c. Nondeterministic and/or partially observable (contingency problem)
   percepts provide new information about state often interleave (search, execution)
d. Unknown state space (exploration problem)

The Actual State Space?

- **The set of all states?**
- **For instance:** A vacuum-cleaner world with just two locations:

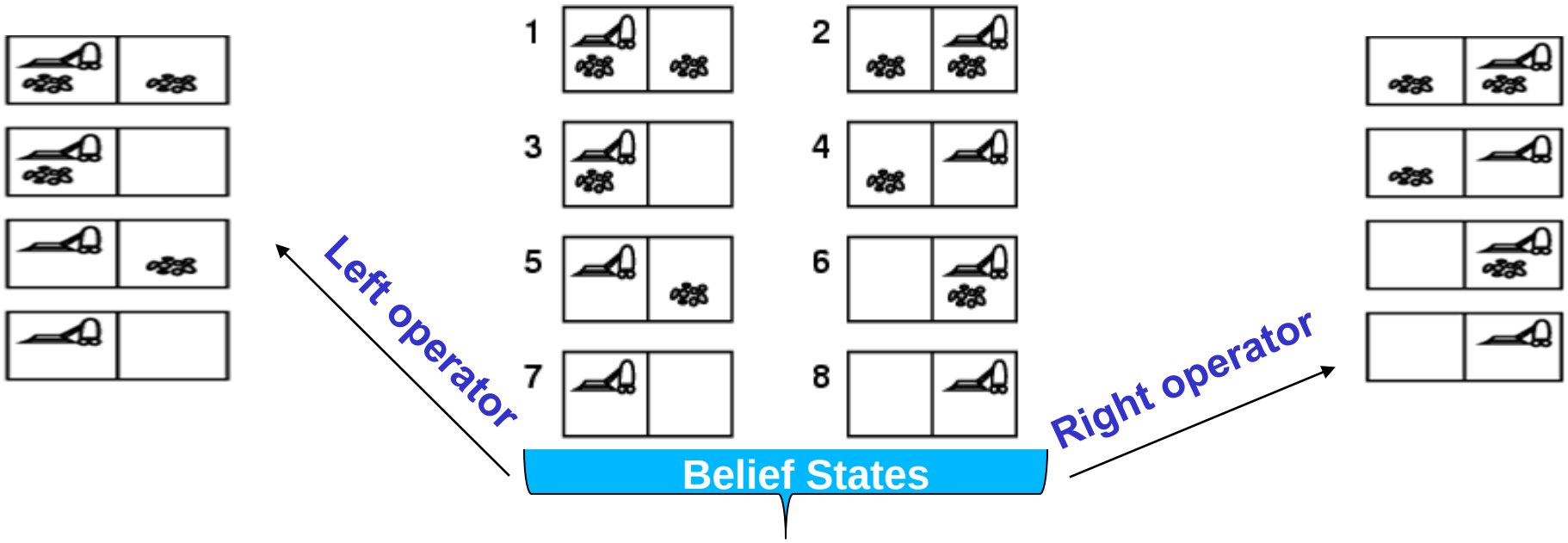
a. **Have 8 states in the state space!**

b. **Successor function:** the successors of a state correspond to trying 3 actions: **Right, Left, Suck**



- **In real world scenario, belief states should be considered (conformat problem)?**
- **Contingency problem can be used to consider conditions about the states**

Re-Formulation with “Belief States”?



Re-Formulation with “Contingencies”?

Nondeterministic: *Suck* may dirty a clean location.
Partially observable: location, dirt at current location.
Percept: *[L, Clean],* i.e., start in #5 or #7.
Solution? *[Right, **if** dirt **then** Suck]*

Simple Problem-Solving-Agent Algorithm

- **initial state**
- **actions**
- **transition model**
- **goal test**
- **path cost function**

Solution: a sequence of actions that leads from the initial state to a goal state

node a data structure that keeps track of - a state

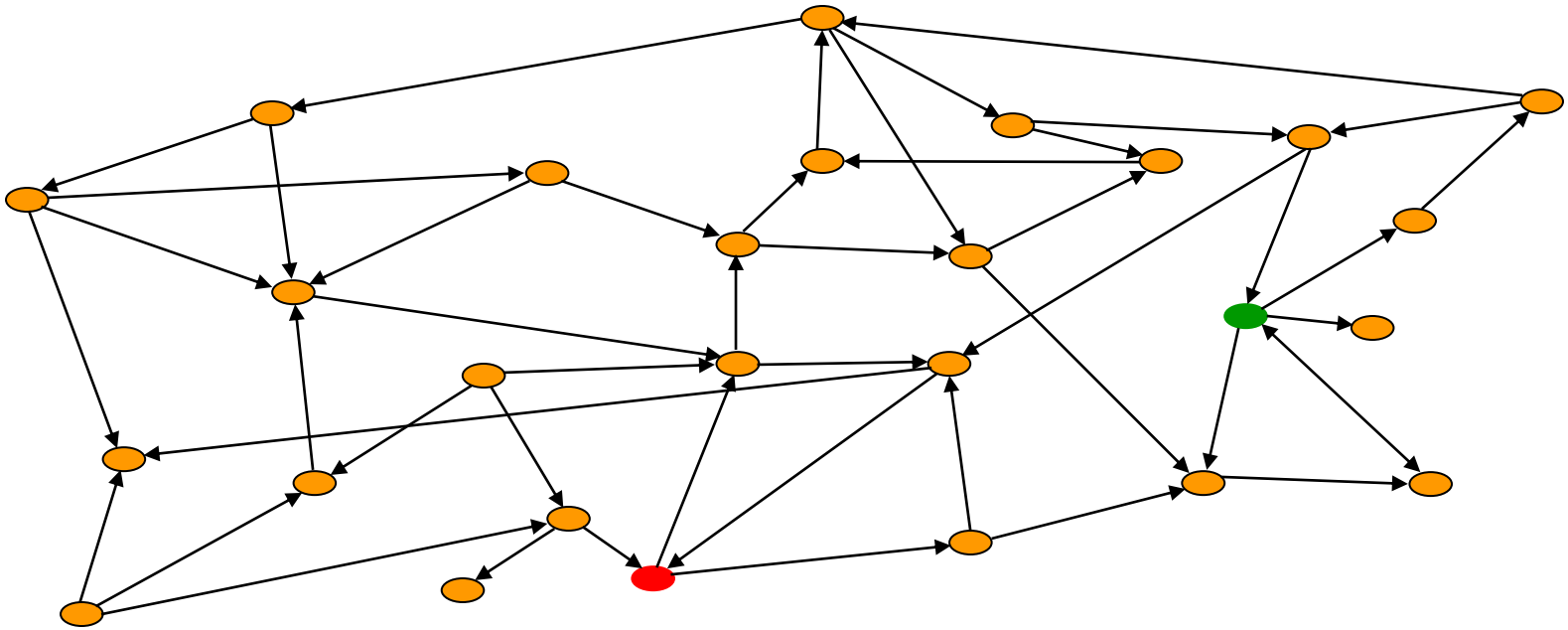
- a parent (node that generated this node)
- an action (action applied to parent to get node)
- a path cost (from initial state to node)

Simple Problem-Solving-Agent Algorithm

- Graph (State space): Construct the structure
  a. List of states ⟸ Nodes to represents the states
  b. Succ ⟸ select/read successor function
- Problem formulation : Construct the Problem
  a. S0 ⟸ sense/read initial state
  b. GOAL? ⟸ select/read goal test
  c. Succ ⟸ select/read successor function (operators- Static Graph)
- Solution ⟸ search(Graph, Problem formulation)

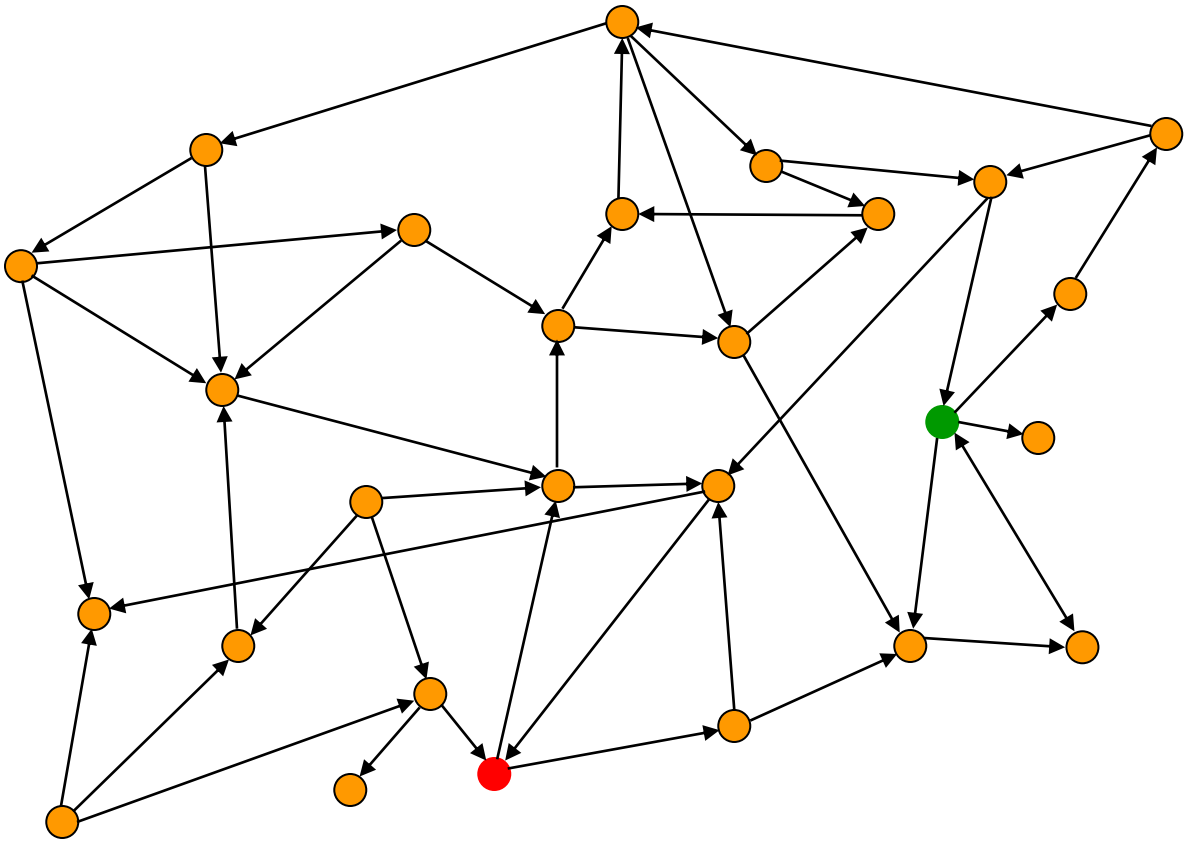
Searching the State Space

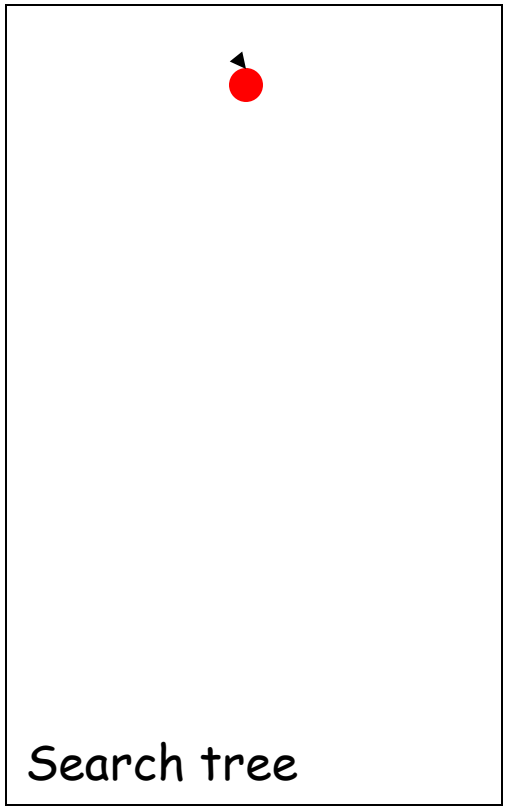
- Often it is not feasible to build a complete representation of the state graph



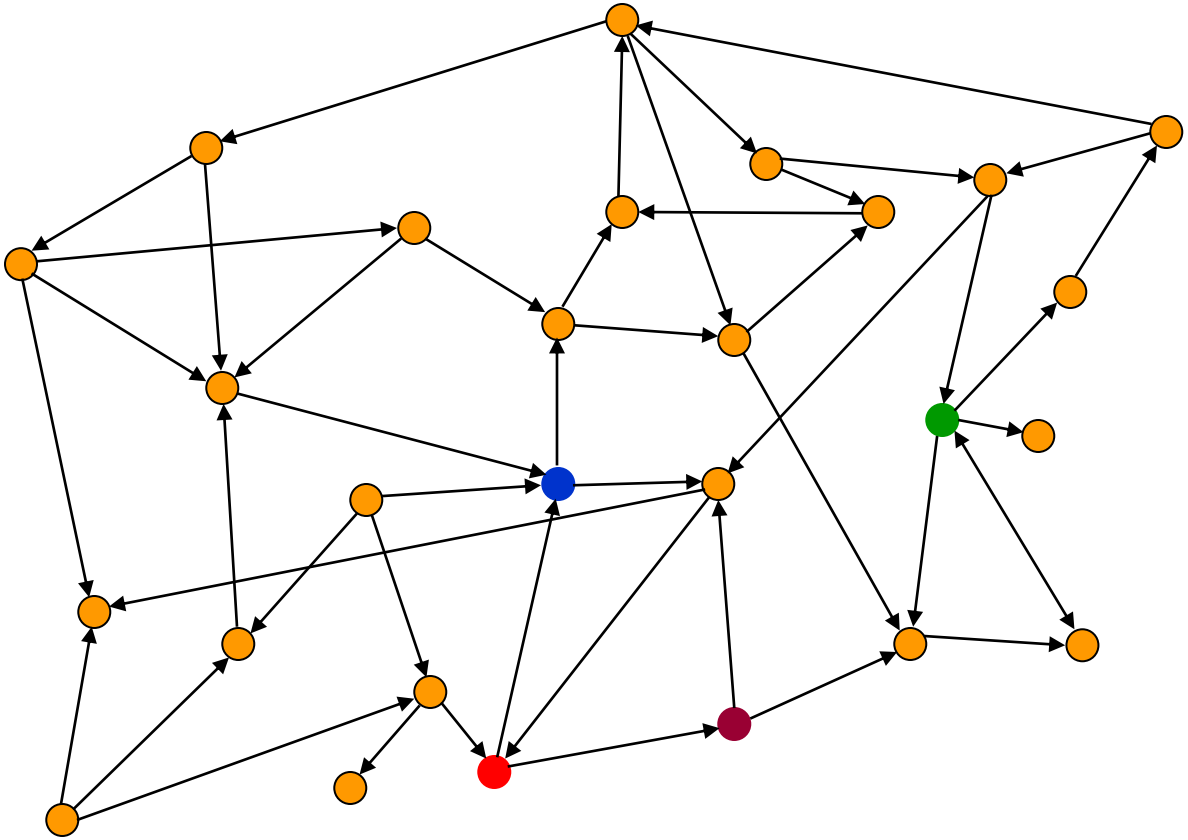
- A problem solver must construct a solution by exploring a small portion of the graph

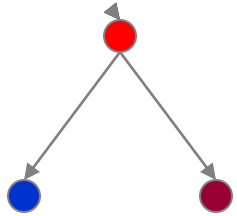
Searching the State Space





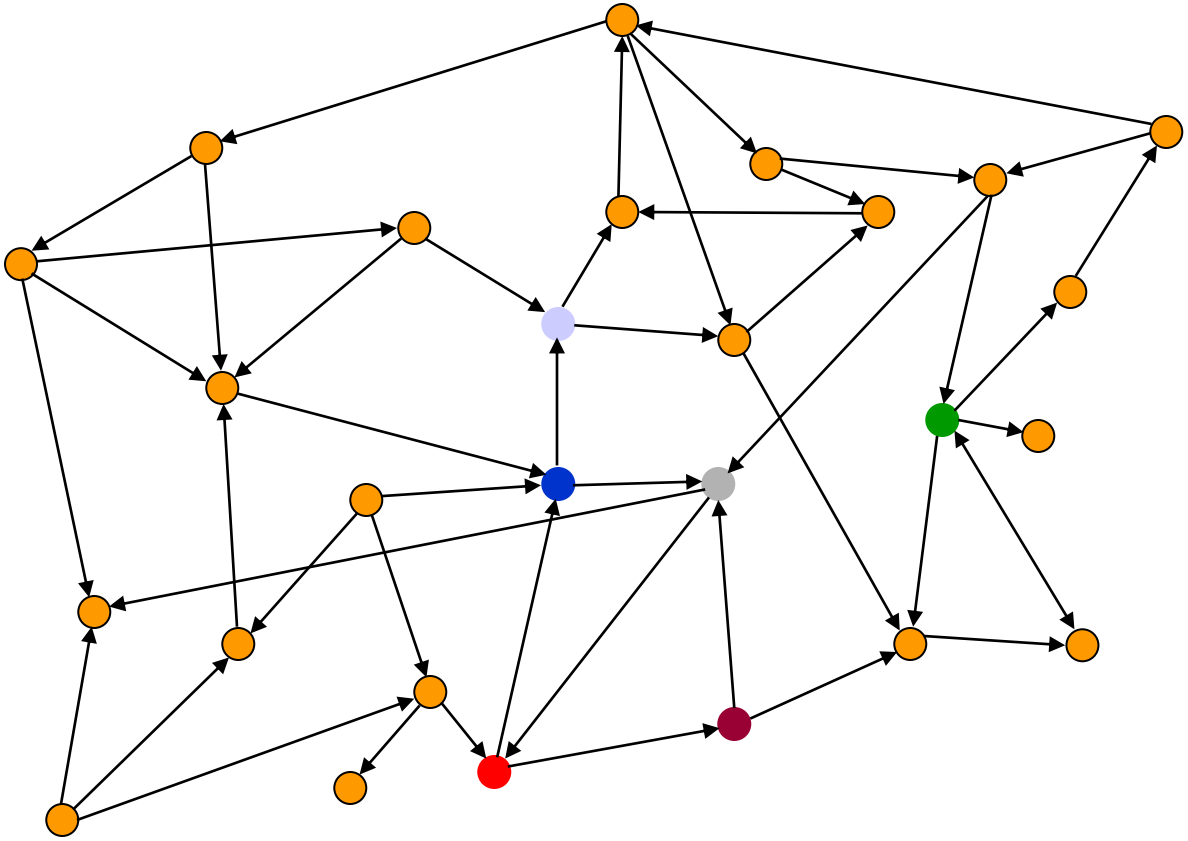
Searching the State Space

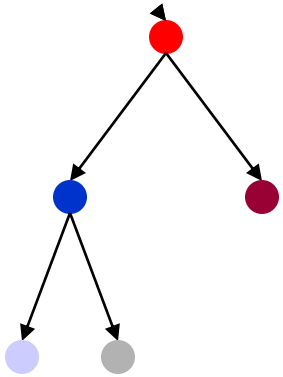




Search tree

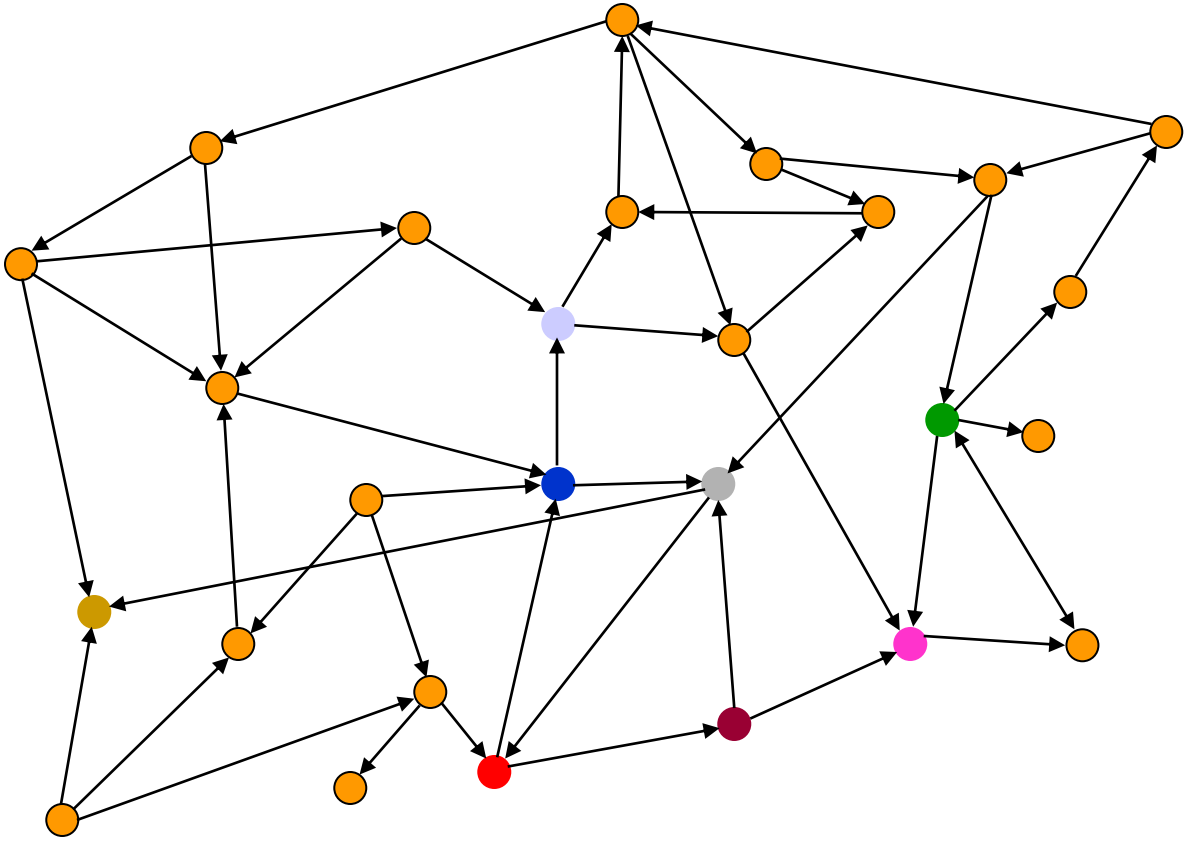
Searching the State Space

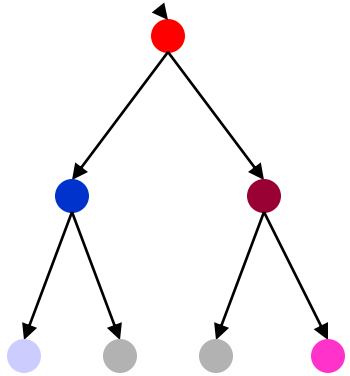




Search tree

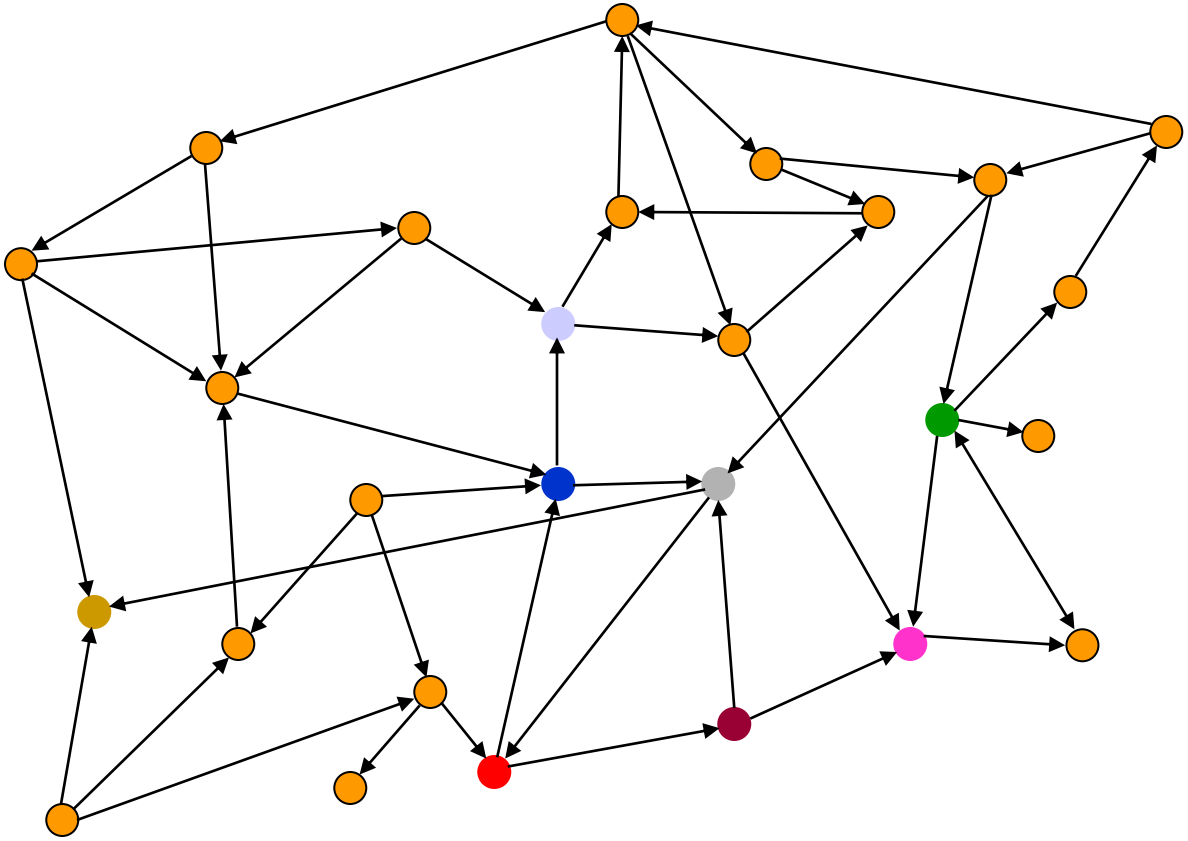
Searching the State Space

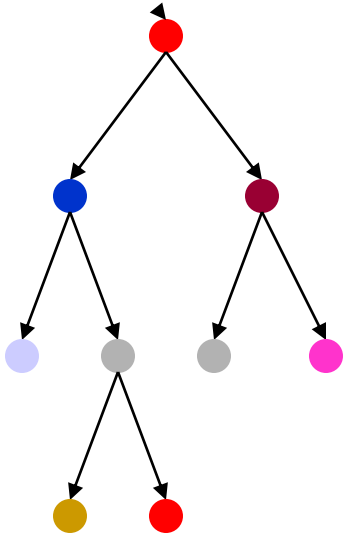




Search tree

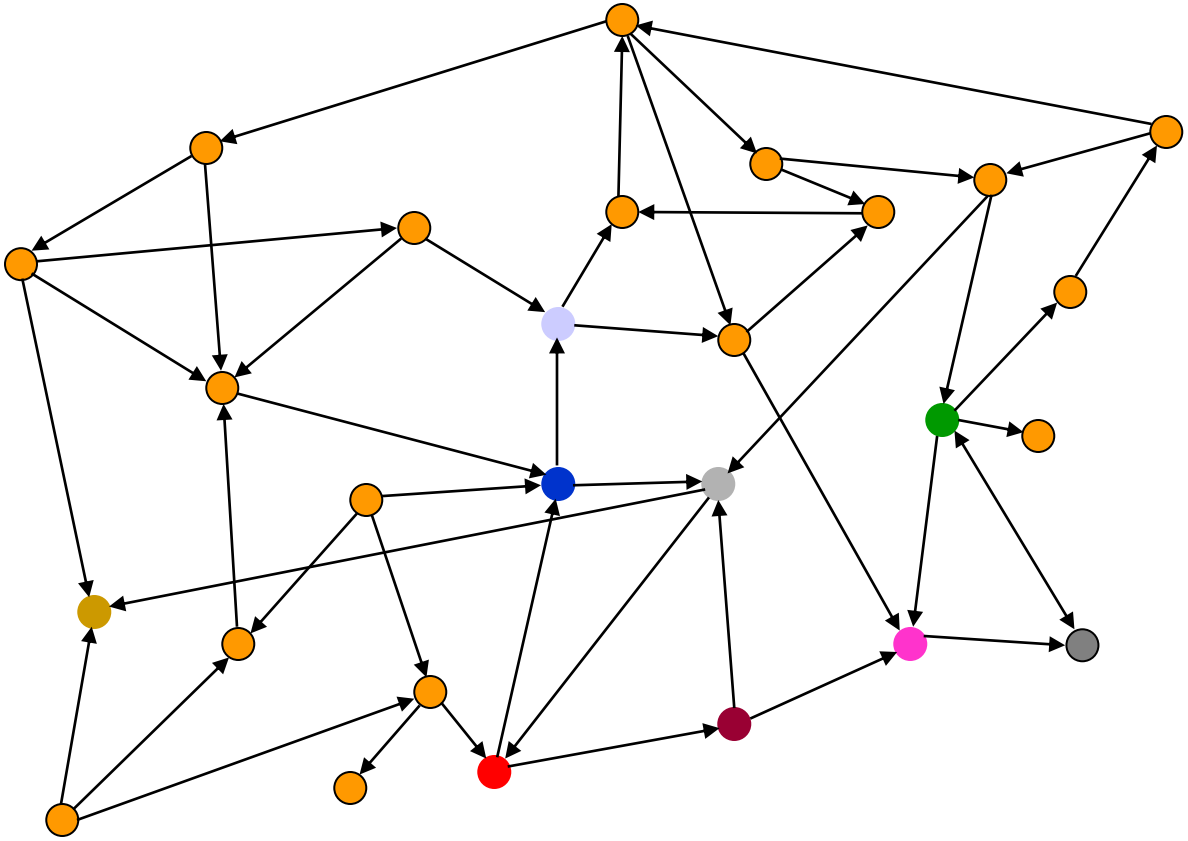
Searching the State Space

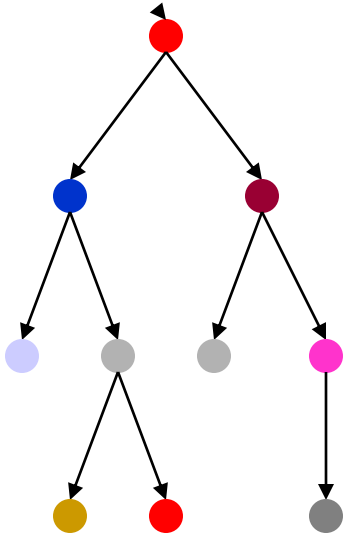




Search tree

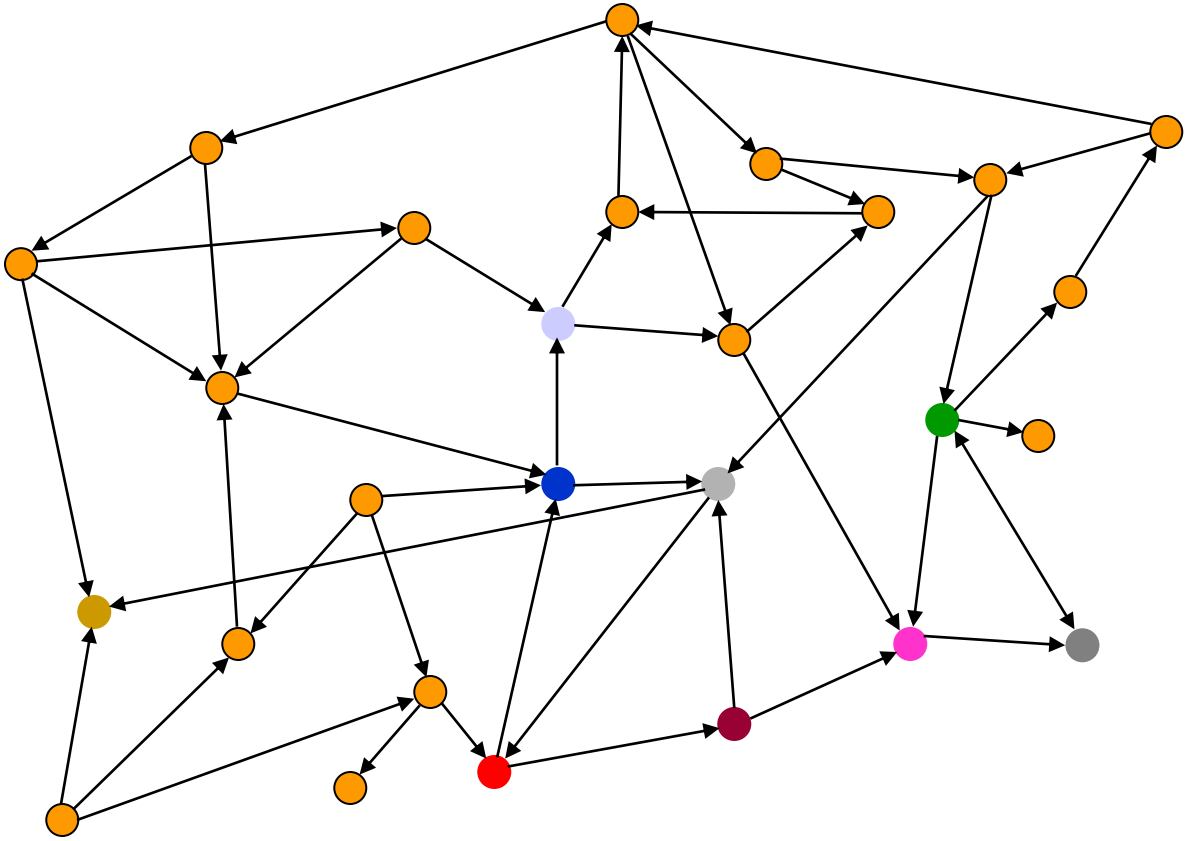
Searching the State Space

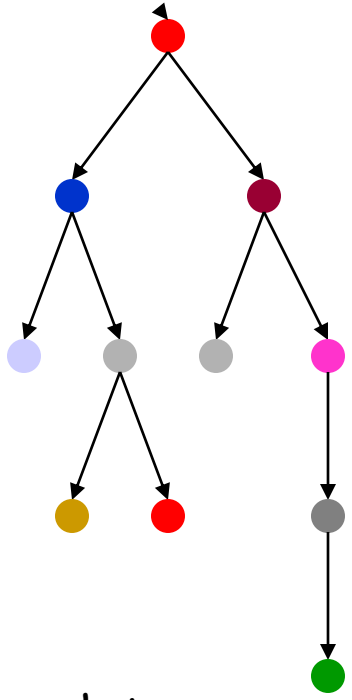




Search tree

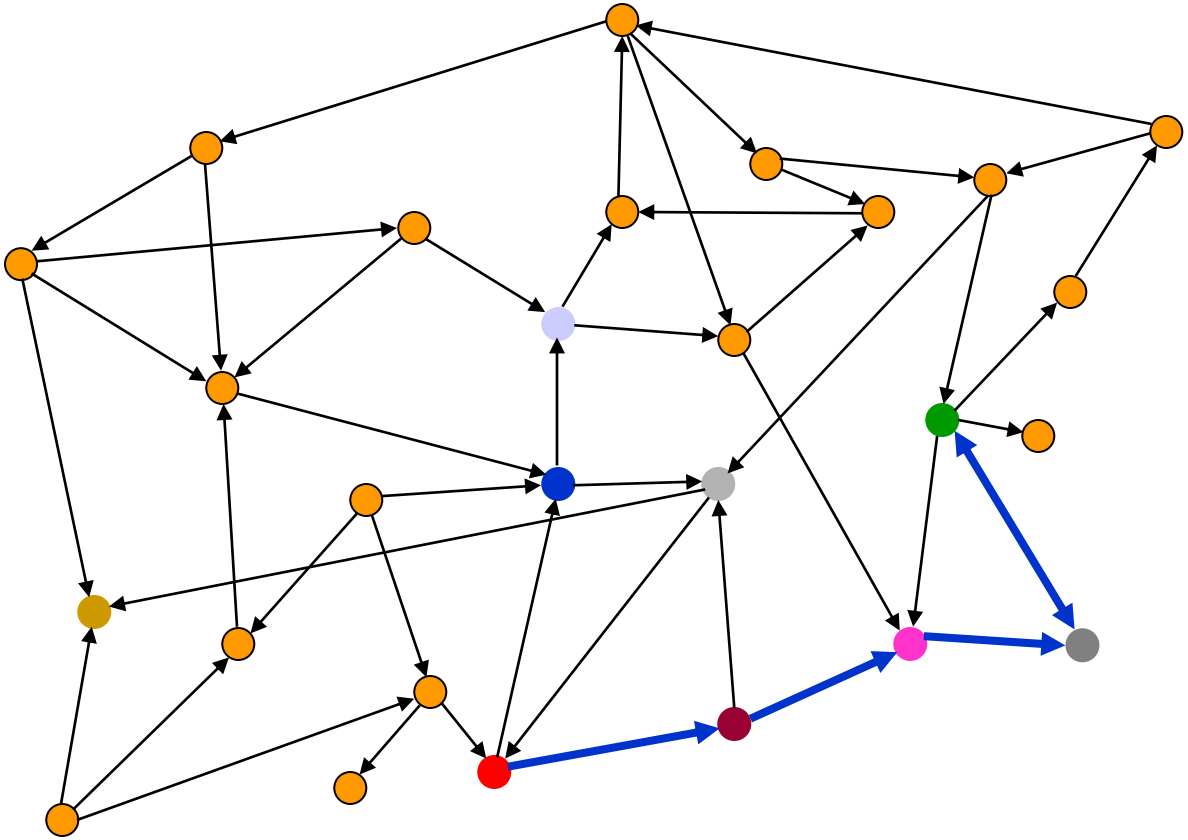
Searching the State Space

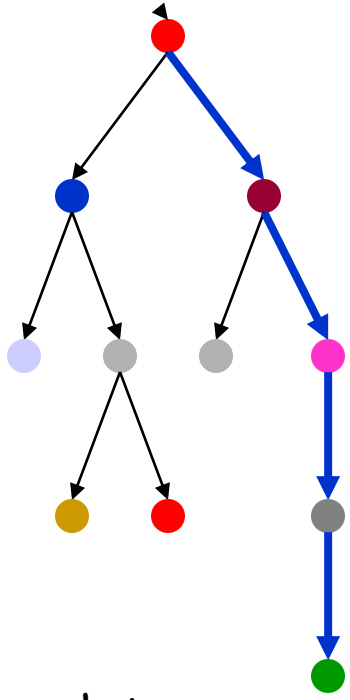




Search tree

Searching the State Space





Search tree

Thank you for your attention!

Questions?