Introduction to Sequential Decision-Making: Advanced Representations

Siddharth Srivastava, Ph.D.
Assistant Professor
Arizona State University

Ira A. Fulton Schools of Engineering
Arizona State University

Suppose you have a household robot

You want it to bring you a bottle of water

Suppose you have a household robot

You want it to bring you a bottle of water

What should the robot do next?

Can we design algorithms for automatically solving such problems?

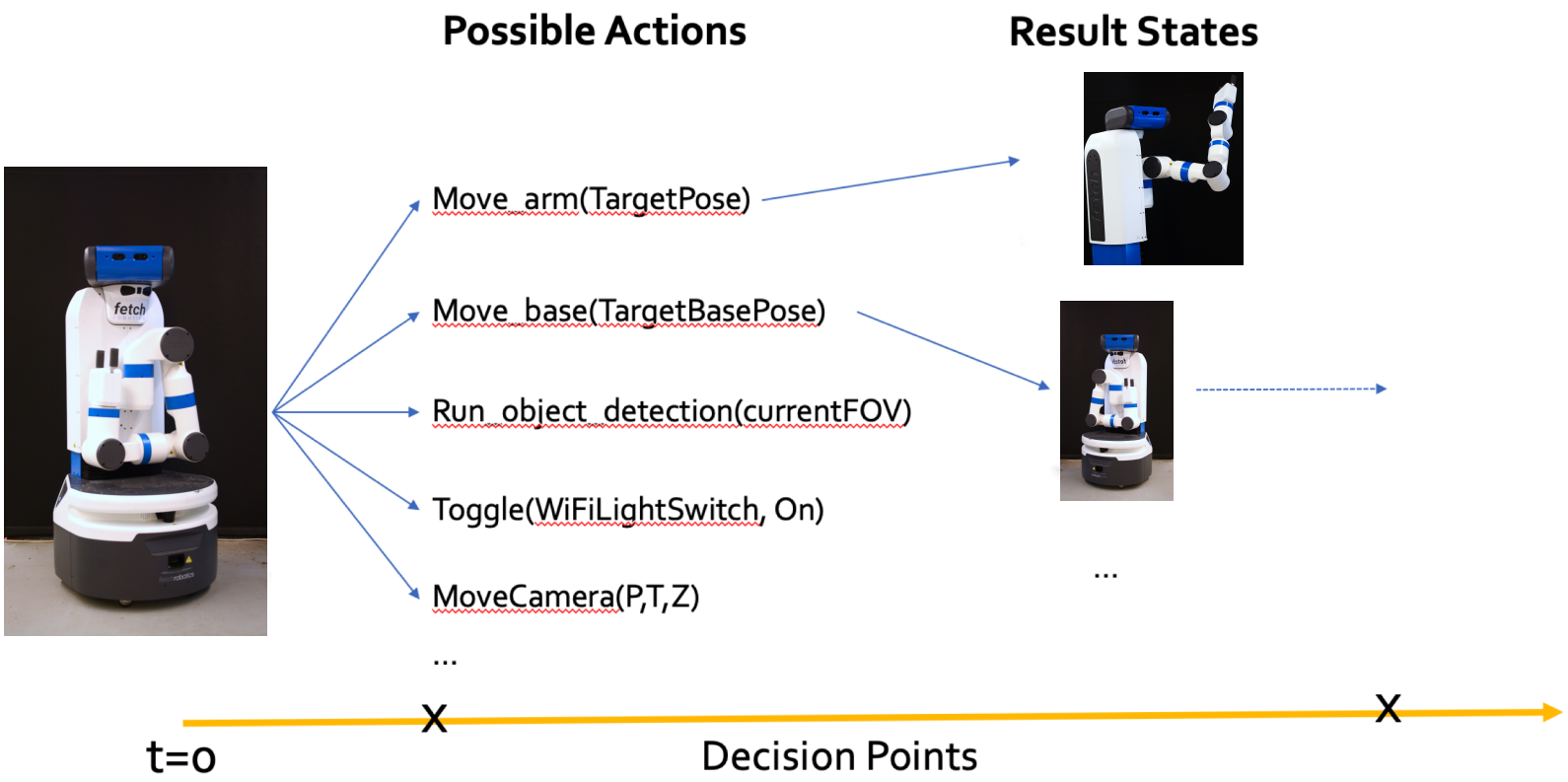
Given: high-level objective

Compute: behavior that achieves that objective

Automated Planning or Behavior Synthesis or Sequential Decision-Making

Automated Planning

- Suppose you have a household robot
- You want it to bring you a bottle of water

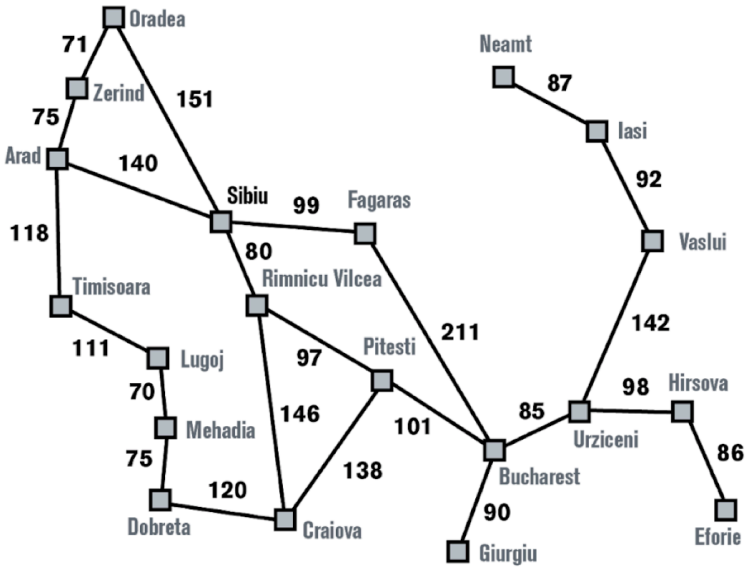


First Principles: Search Problems

Recall the search problem formulation

Given:

- Graph $\langle V, E, A, \ell \rangle$
  - V set of vertices, representing states
  - E set of directed edges (actions)
  - A set of actions
  - $\ell: E \rightarrow \langle A, \mathbb{R}^+ \rangle$
- $v_i \in V$ start vertex
- G goal vertex set



Find a path from v_i to G

Example: Logistics

Vertices

Edges (directed)

Delivery service

- Trucks, drivers, packages

Actions:

- Get a driver to drive a truck to a location
- Load a package on to a truck
- Unload a package from a truck
- Refuel a truck

Example: Logistics

Vertices

Edges (directed)

Delivery service

- Trucks, drivers, packages

Actions:

- Get a driver to drive a truck to a location
- Load a package on to a truck
- Unload a package from a truck
- Refuel a truck

How Should We Express This Problem?

Option 1: Naïve representation: draw the explicit state space graph

- Each state is a configuration of trucks, packages, drivers
- n cities, p packages, t trucks, d drivers
- n^{p+t+d} vertices
  - Small problem: 100 cities, 1000 packages: $> 100^{1000}$ vertices
  - Number of atoms in the universe ~10^{82}
- But that is not all… Actions correspond to edges in the graph
  - Need to work out edges corresponding to each action instance (load p1 onto t1 when t1 is in PHX)
  - Amount of fuel used needs to be calculated separately for each edge

Problems with Naïve Representation

Option 1: Naïve representation: draw the explicit state space graph

- n cities, p packages, t trucks, d drivers
- n^{p+t+d} vertices
- Edges for actions

Creating or maintaining such a representation is a problem

- New requirement: some packages requires refrigerated trucks (any of $\{t_1, t_{12}, t_{1232}, \dots\}$)
- Iteratively prune each edge that violates this
- Changes in fuel efficiency → redo the fuel usage for each edge

A Step Up: Factored Representations

States are factored into variables

$\langle \mathrm{loc_{p1}}, \mathrm{loc_{p2}}, \ldots, \mathrm{loc_{t1}}, \ldots \mathrm{loc_{d2}}, \ldots, \mathrm{fuel_{t1}}, \mathrm{fuel_{t2}}, \ldots \rangle$

Successor function can "focus" on only the relevant factors

- The action "drive truck t1 from Phoenix to Ontario" changes loc_{t1} :
  - loc_{t1}= Ontario
- Regardless of where the other trucks, drivers, packages are
- One variable assignment equivalent to specifying edges from $O(ptd)$ vertices (states)

Problems with Factored Representations

Repetition

- Successor function has separate rules for driving each truck from Phoenix to Ontario
- Also for unloading/loading each package

This is fine until a new package, truck, or driver shows up

Then, need to redo:

- State vector representation
- Successor functions for load, unload
- (But the actions do the same thing they did to every other package!!)

Is there a Better Way?

Can we avoid spelling out each rule for each tuple of objects?

Can we express the problem in terms of properties of objects and their relationships,

rather than in terms of their names?

Relational Representations

A state is defined using properties and relationships among objects

Given a domain, fix a set of relations R and a set of functions F as it's vocabulary

- A binary relation between sets A and B defines a subset of A x B

Relational Representations

- A state is defined using properties and relationships among objects
- Given a domain, fix a set of relations R and a set of functions F as it's vocabulary
- Express the successor function for each action in terms of these functions and relations
- Logistics domain:

R = {In(Package, Truck), On(Package, Package)}
F = {Fuel(Truck), Location(Truck), DriverOf(Truck)}

Successor Functions for Relational Representations

R = {In(Package, Truck), On(Package, Package)}

F = {Fuel(Truck), Location(Truck), DriverOf(Truck)}

Relations and functions specify the properties that can be used while writing successor functions or constraints

Example: the effect of "load package p onto truck t" results in the following changes:

- If p and t are at the same location,
  - In(p, t) becomes true
  - At(p, l) becomes false for all l
- Regardless of
  - object names
  - values of other properties and relationships between other objects

A Comparison of the Three Representational Choices

Atomic	Factored	Relational
At_t_1Phoenix_At_t_2Ontario_...	$loc_{t_1} = Phoenix$ $loc_{t_2} = Ontario$	$loc(t_1) = Phoenix$ $loc(t_2) = Ontario$
A state is a black-box String representations are used to alleviate the situation	A state is a valuation of a fixed set of "state variables" Successor functions for each action can depend on a small set of variables Problem: dependent on object names, numbers	A state is defined using properties and relationships among objects Specifying successor functions seems to be easier through the use of variables, quantifiers