16. Dynamic Path Planning

金力 Li Jin
li.jin@sjtu.edu.cn
上海交通大学密西根学院
Shanghai Jiao Tong University UM Joint Institute

上海交通大學
SHANGHAI JIAO TONG UNIVERSITY

Outline

- Problem formulation
- Markov decision processes

Static vs. dynamic path planning

- Recall SP problem.
- Static data, open-loop decision, deterministic outcome.
- However, reality may significantly deviate therefrom...

Static path planning is for long-term decisions

Dynamic path planning is for short-term decisions

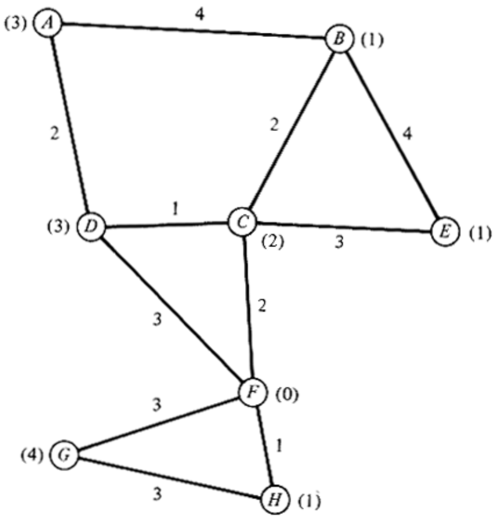
Setup

- Network $G(N, E)$
- Link parameters

$$(i, j): n_{ij}, \bar{f}_{ij}$$

- Node parameters

$$\bar{g}_i$$



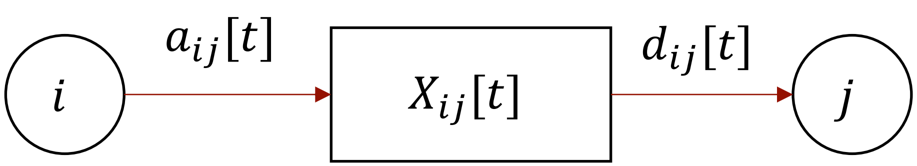
Link model

Let's begin with a simplistic model...

- Traffic state $x_{ij}[t]$ [veh]
- Arrival $a_{ij}[t]$ [veh/sec]
- Departure $d_{ij}[t]$ [veh/sec]
- Dynamical equation

$$x_{ij}[t+1] = x_{ij}[t] + a_{ij}[t] - d_{ij}[t]$$

- Is it enough?



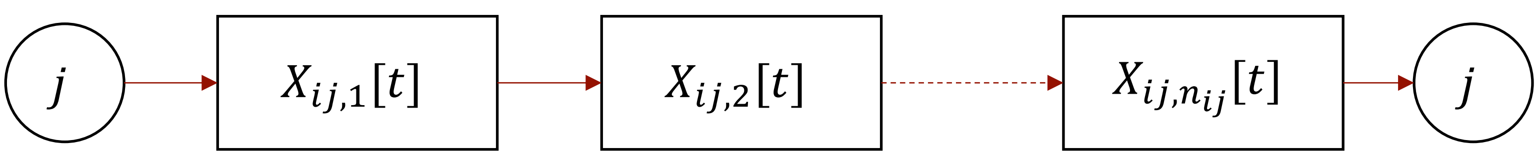
Link model

Now consider a realistic model...

- Traffic state $x_{ij}[t] = \left[x_{ij,1}[t], \ldots, x_{ij,n_{ij}}[t]\right]^T$
- Arrival $a_{ij}[t]$. Departure $d_{ij}[t]$
- Dynamical equation

$$x_{ij,1}[t+1] = a_{ij}[t]; x_{ij,k}[t+1] = x_{ij,k-1}[t], k = 2, \ldots, n_{ij} - 1;$$
$$x_{ij,n_{ij}}[t+1] = x_{ij,n_{ij}}[t] + x_{ij,n_{ij}-1}[t] - d_{ij}[t].$$

- $d_{ij}[t] = \min\left\{x_{ij,n_{ij}}[t], \bar{f}_{ij}\right\}$



Node model

External arrivals:

- Deterministic arrivals

    constant A_i

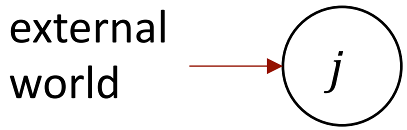
    time-varying $A_i[t]$

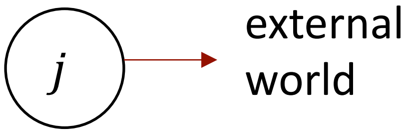
- Stochastic arrivals

    Bernoulli $p_i, p_i[t]$

    Noise $A_i[t] + w_i[t]$

External departure



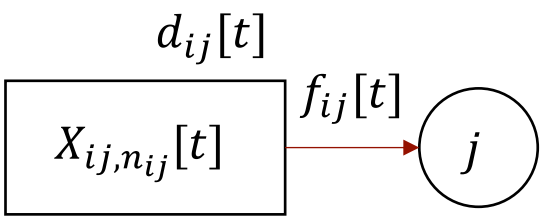


Node model

Discharge of incoming traffic

1. Sending traffic $d_{ij}[t]$

$$d_{ij}[t] = \min\left\{x_{ij,n_{ij}}[t], \bar{f}_{ij}\right\}$$



1. Intersection model

signalized

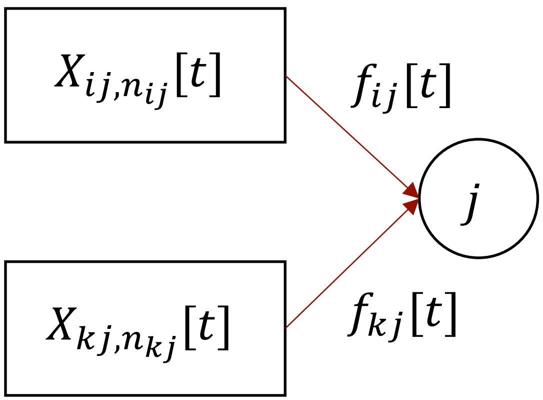
$$\alpha_{ij} + \alpha_{kj} = 1$$

$$f_{ij}[t] = \min\{d_{ij}[t], \alpha_{ij}\bar{g}_j\}$$

$$f_{kj}[t] = \min\{d_{kj}[t], \alpha_{kj}\bar{g}_j\}$$

signal-free

$$\alpha_{ij}[t] \leftarrow d_{ij}[t], d_{kj}[t]$$



Node model

Routing of outgoing traffic

- Bernoulli routing

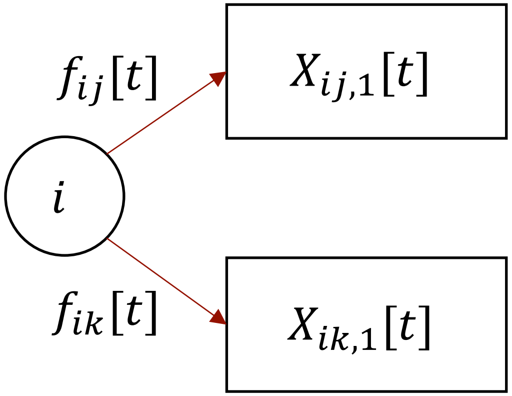
$$f_{ij}[t] = \gamma_{ij} \sum_{\ell \in In(i)} f_{\ell i}[t]$$

$$f_{kj}[t] = \gamma_{kj} \sum_{\ell \in In(i)} f_{\ell i}[t]$$

- Dynamic routing

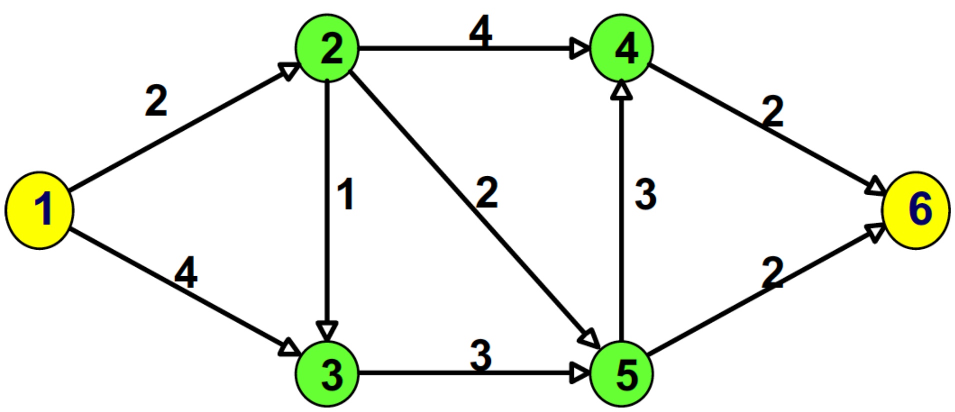
$$\gamma_{ij}[t] \leftarrow \text{traffic condition}$$

Which one is more realistic?



Path planning problem

- Suppose that a vehicle arrives at node 1 at time $t = 0$.
- Then, the vehicle needs to select between nodes 2 and 3.
- Suppose that the vehicle selects node 2 (nominally shortest path).
- It needs to make a choice again when it arrives at node 2.

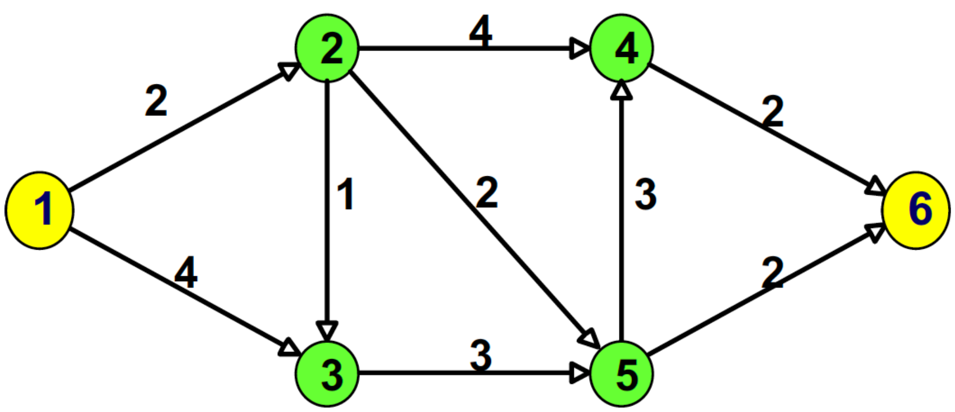


Path planning problem

- Since the vehicle only need to make choices at nodes, we can reduce the time indices:

times: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, ...

epochs: 0(0), 1(3), 2(9), ...



Path planning problem

- Therefore, we can formulate a dynamical system as follows.
- Epoch k = instants at which a node is attained; $\mathbb{Z}_{\geq 0}$.
- State $v[k]$ = node attained at time k; N.
- Action (decision/control input) $a[k]$ = the downstream node to go to; $\mathrm{Out}(v[k])$.
- Update/Dynamical equation
$$v[k+1] = a[k].$$
- How to characterize the travel cost?
- We need additional states to compute travel cost...

State update

- We also need to track $x[k] = \{x_{ij}[k]; (i,j) \in E\}$ (note that $x_{ij}[k]$ is itself an n_{ij}-vector!)
- Given $x[k], v[k], a[k]$, we can compute the time for the vehicle to cover the link $(v[k], a[k]) \in E$.
- If everything is deterministic, we can exactly compute this time $c(x[k], v[k], a[k])$.
- At the same time, we can compute the new state
$$(x[k], v[k], a[k]) \rightarrow (x[k+1], v[k+1])$$
- If there is randomness, we can characterize the distribution of this time $C(x[k], v[k], a[k])$, and distribution of new state $(x[k+1], v[k+1])$.

Objective function

- Hence, we are interested in minimizing the total time for the vehicle to go through the network:

$$\min \sum_{k} c(x[k], v[k], a[k]).$$

- $c(x[k], v[k], a[k])$ refers to either the deterministic travel time or the expectation of the random travel time.
- The decision variable here is $a[k]$.

Outline

- Problem formulation
- Markov decision processes

Markov process

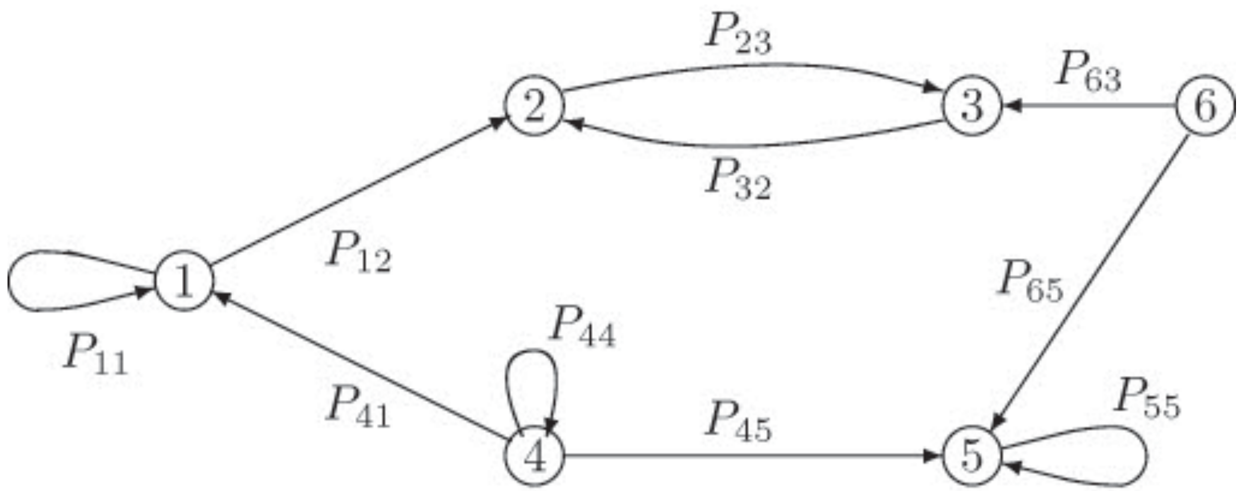
- Consider a set $S \subseteq \mathbb{Z}$.
- The rv X_n is called the **state** of the chain at time n.
- S is thus called the **state space**.
- Notational convention:
  - We use x to denote elements of the set S.
  - We use X_n to denote the state at time n.
  - Hence, $\Pr\{X_n = x\}$ and $\Pr\{X_n = X_{n+1}\}$ are meaningful, while $\Pr\{x = x'\}$ is not.
- **Transition probability**

$$P_{ij} = \Pr\{X_n = j | X_{n-1} = i\}, \qquad i, j \in S.$$

- From a control-theoretic perspective, $\{P_{ij}; i, j \in S\}$ specifies the **dynamics** of the system.

Graphical representation

- Node = state
- Arc = transition with strictly positive probability
- Intuitive, but not applicable to complex Markov chains



Markov decision process

- Suppose that at each state $i \in S$, we can take a set of **actions** $\mathcal{A}_i$.
- The reward is $r_i(a)$, where $a \in \mathcal{A}_i$ is the action.
- The transition probabilities are $P_{ij}(a)$, i.e. action-dependent.
- Now, the cumulative reward will depend on the sequence of actions that we select.
- This is a **Markov decision process (MDP)**.
- Objective of an MDP: find the sequence of actions that maximizes the cumulative reward.

Markov decision process

- Objective: given initial condition $x \in S$, find $a = \{a_0, a_1, \ldots\}$ that
$$\max_a \mathrm{E}[\textstyle\sum_n R_n \,|X_0 = x]\,.$$
- Since we are considering Markov processes, our actions can be written as a function of states, i.e. a **control policy** $\alpha: S \to \mathcal{A}$
$$a_n = \alpha(X_n).$$
- Given a control policy, we can define the **state value function** $v_\alpha(x)$ as follows:
$$v_\alpha(x) = \mathrm{E}_\alpha[\textstyle\sum_n R_n \,|X_0 = x].$$
- Note: $v_\alpha(x)$ must depend on α.

Markov decision process

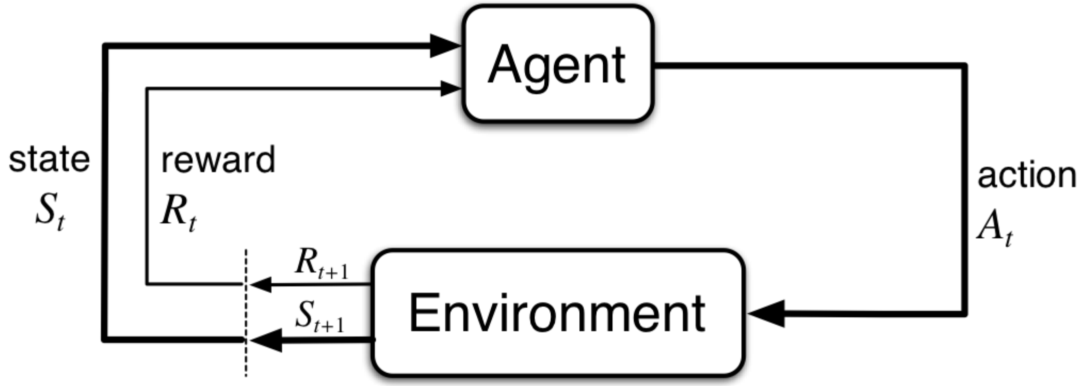
- Let $\mathcal{A}(x)$ be the set of allowable actions at state x.
- A policy is **admissible** if $\alpha(x) \in \mathcal{A}(x)$ for each $x \in S$.
- Let $\mathbb{A}$ be the set of admissible policies.
- In terms of control policies, we can reformulate the MDP as the search for the **optimal policy**, i.e. a policy $\alpha^* \in \mathbb{A}$ such that
$$v_{\alpha^*}(x) \geq v_\alpha(x), \qquad \forall x \in S, \forall \alpha \in \mathbb{A}.$$
- Note that $v_\alpha(x)$ is always well-defined for finite-time MDPs.
- For infinite-time MDPs, we can use a discounted return
$$G(x) = \mathrm{E}[\textstyle\sum_n \gamma^n R_n \,|X_0 = x], \qquad \gamma \in (0,1).$$

Markov decision process

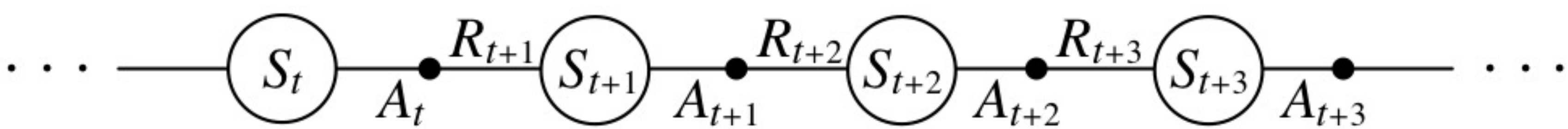
The task of finding good control policies leads to the following techniques:

- Analytically determine a satisfactory control policy ≈ **stochastic control**
- Numerically compute the optimal policy ≈ **dynamic programming**
- Learn the model behavior and make decisions simultaneously ≈ **reinforcement learning**
- Use neural networks to approximate value function or control policy ≈ **approximate dynamic programming** or **learning-based control**

Agent-Environment Interface



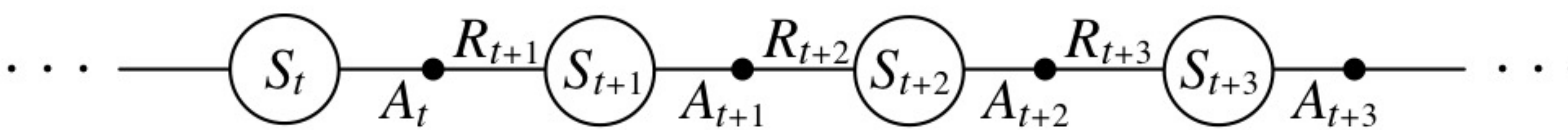
- Agent & environment interact at discrete times $t \in \mathbb{Z}_{\geq 0}$.
- Agent observes state at step t: $S_t \in \mathcal{S}$,
- Implements an action at step t: $A_t \in \mathcal{A}(S_t)$,
- Gets resulting reward $R_{t+1} \in \mathcal{R}$.
- Environment goes to next state $S_{t+1} \in \mathcal{S}$.



Markov Decision Processes

- The process of selecting actions A_t is called a **Markov Decision Process (MDP)**.
- If state and action sets are finite, it is a finite MDP.
- To define a finite MDP, you need to give:
  - state and action sets $\mathcal{S}, \mathcal{A} = \cup_{s \in \mathcal{S}} \mathcal{A}(s)$
  - one-step “dynamics” (“four-argument probability”) $p(s', r | s, a) = \Pr\{S_{t+1} = s', R_{t+1} = r | S_t = s, A_t = a\}$;
  - there is also “state-transition probability”

$$p(s' | s, a) = \Pr\{S_{t+1} = s' \, | S_t = s, A_t = a\} = \sum_{r \in \mathcal{R}} p(s', r | s, a) \, .$$



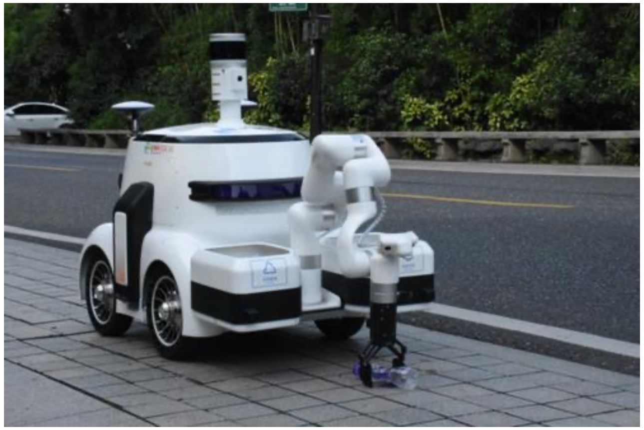
Policy

- A **policy** π is a function defined in either of the following ways:
  - $\pi: \mathcal{S} \rightarrow [0,1]^{|\mathcal{A}|}$; $\pi(a|s) = \Pr\{A_t = a | S_t = s\}$ (probabilistic policy).
  - $\pi: \mathcal{S} \rightarrow \mathcal{A}$; $\pi(s) \in \mathcal{A}$ (deterministic policy or control law).
- Roughly, the agent's goal is to get as much reward as it can over the long run.
- For finite MDPs, π is essentially a table.
- Hence, the associated decision-making methods are called **tabular methods**.
- Reinforcement learning methods specify how the agent changes its policy as a result of experience.

An Example Finite MDP

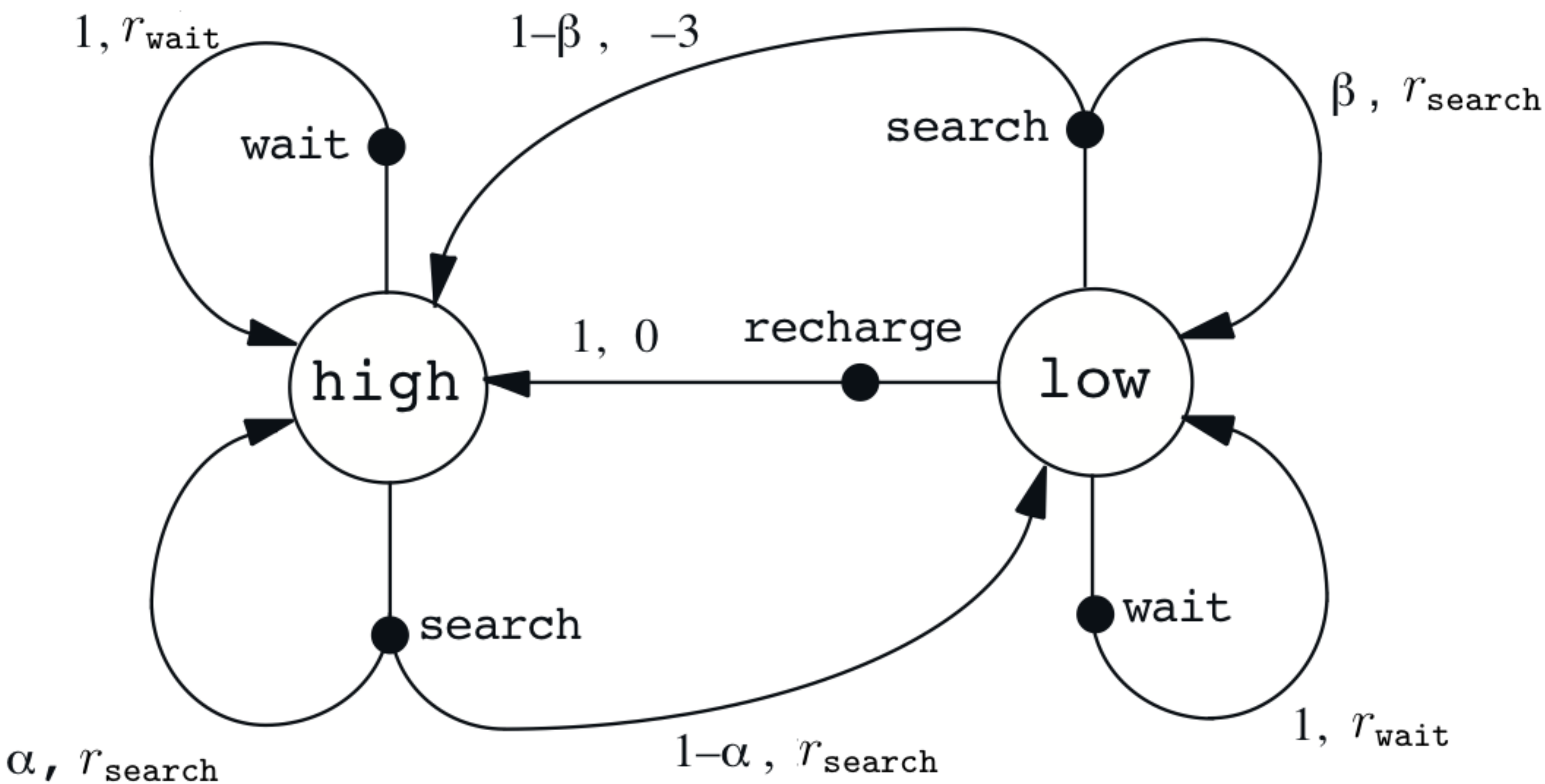
Recycling Robot

- At each step, robot has to decide whether it should (1) actively search for a can, (2) wait for someone to bring it a can, or (3) go to home base and recharge.
- Searching is better but runs down the battery; if runs out of power while searching, has to be rescued (which is bad).
- Decisions made on basis of current energy level: high, low.
- Reward = number of cans collected



$\mathcal{S} = \{\text{high}, \text{low}\}$, $\mathcal{A}(\text{high}) = \{\text{search}, \text{wait}\}$, $\mathcal{A}(low) = \{\text{search}, \text{wait}, \text{recharge}\}$, r = expected # of cans collected

An Example Finite MDP



Stochastic control & MDP

- MDPs are essentially control problems.
- A primary objective: ensure that the system is functional.
  - State should be bounded on average. (Foster-Lyapunov)
  - State should stay in a safe set. (Reachability analysis)
- A secondary objective: approach the control target with minimal cost.
  - Time-cumulative cost is minimized. (Dynamic programming)
  - Equilibrium is optimized. (Optimization)
- In terms of optimization, we first find a set of feasible solutions and then find the optimal one out of it.