Modeling Transportation Problems with MDPs

Cathy Wu

1.041/1.200 Transportation: Foundations and Methods

Readings

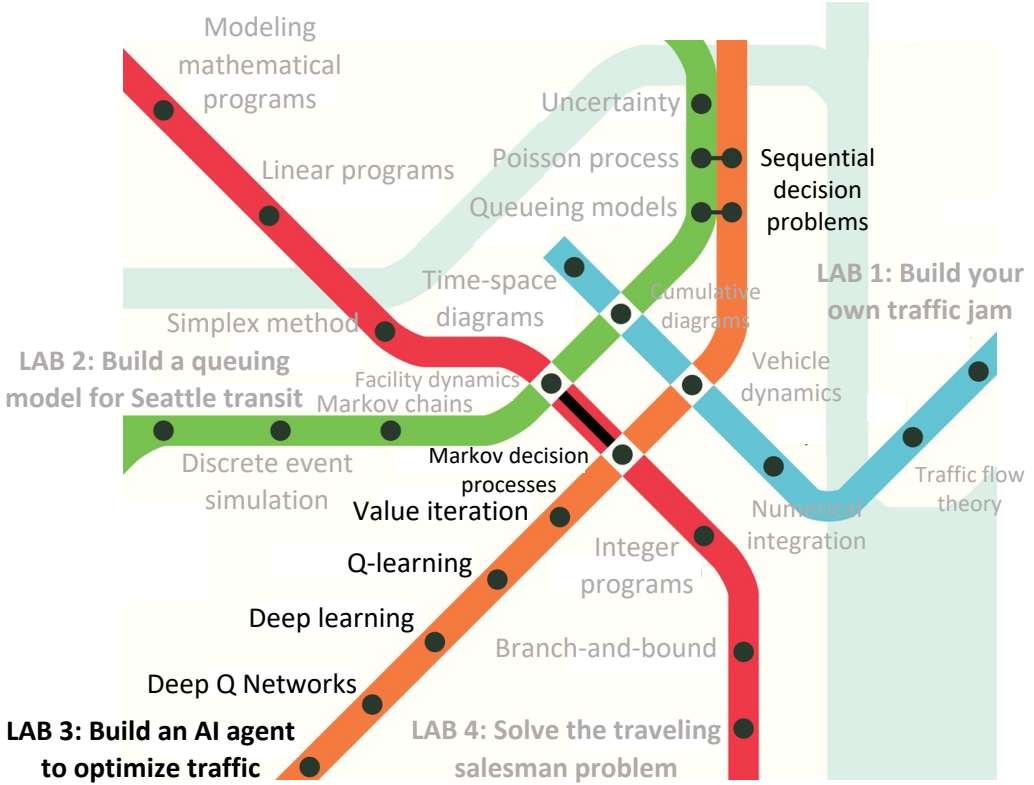
1. Gymnasium documentation
   - Basic Usage
   - Create a Custom Environment

Unit 3: Decision Making Under Uncertainty

Unit 3

Optimizing

Multi-stage



Outline

1. Software libraries for sequential decision making
2. Designing a sequential decision problem

Outline

1. **Software libraries for sequential decision making**
   a. Standardized `gym` interface for environment models
   b. Examples: taxi, train rescheduling, traffic signal control
2. Designing a sequential decision problem

Useful libraries for sequential decision making

Environments

Standardized interface for sequential decision problem models (“MDPs”)

Stable Baselines3

Algorithms

Collection of methods (reinforcement learning) for training agents (policies) to solve environments

Gymnasium: https://gymnasium.farama.org/
Stable baselines3: https://github.com/DLR-RM/stable-baselines3

Familiarizing with Gymnasium

- Introduction to Gymnasium
  - https://gymnasium.farama.org/
  - Standardized `gym` interface: https://gymnasium.farama.org/api/env/
- Toy transportation environment: Taxi pickup and drop-off
  - Documentation: https://gymnasium.farama.org/environments/toy_text/taxi/
  - Implementation: https://github.com/Farama-Foundation/Gymnasium/blob/main/gymnasium/envs/toy_text/taxi.py

Transportation environments (an incomplete list)

Routing, scheduling, facility location

- **Flatland** — efficiently manage dense traffic on complex railway networks.
- **RL4CO** — reinforcement-learning-for-combinatorial-optimization environments collected under one documented roof.

Traffic flow dynamics & control, autonomous driving

- **HighwayEnv** — a collection of environments for autonomous driving and tactical decision-making tasks.
- **SUMO-RL** — interface to SUMO for Traffic Signal Control.
- **Intersection Zoo** — ICLR 2025 benchmark for multi-agent eco-driving at signalized intersections (interfaces with SUMO).
- **Flow-Lite** — lightweight sibling of Flow (cooperative vehicle control, interfaces with SUMO), ideal for rapid research prototyping.
- **Waymax** — library for simulating and evaluating agents using scenarios instantiated from the Waymo Open Motion Dataset.

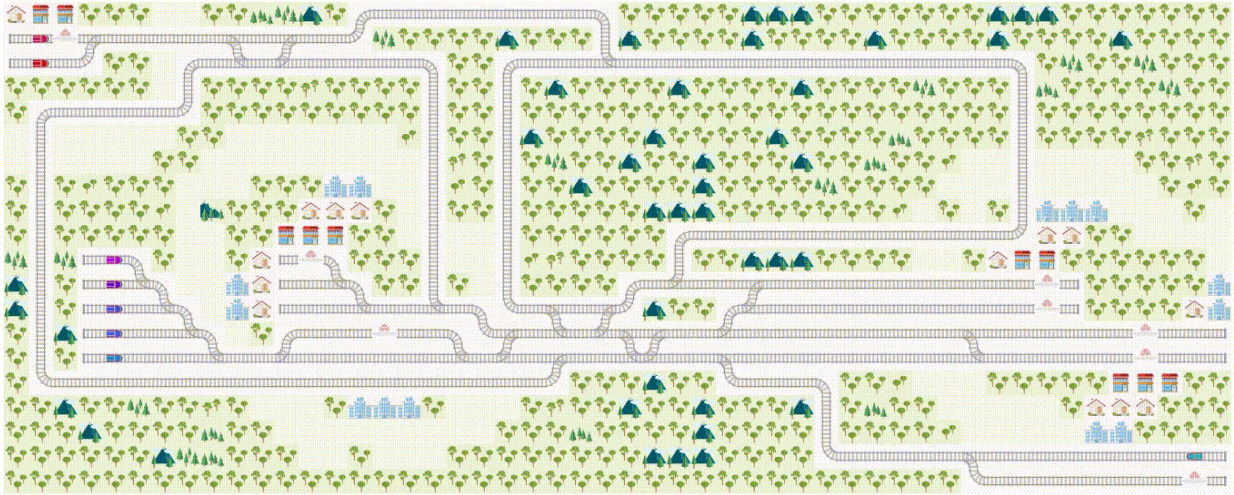
Air traffic control

- **BlueSky-Gym** — Air Traffic Management tasks

More datasets and benchmarks: https://www.rerite.org/awesome-rerite.html

Flatland challenge: Train rescheduling

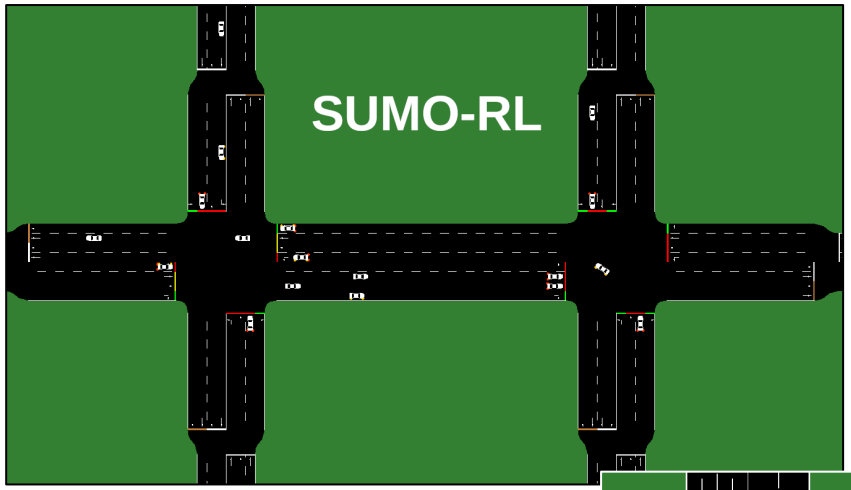
- *Example environment based on a real-world transportation problem*
- Created in collaboration with SSB, Deutsche Bahn, SNCF – the national rail operators of Switzerland, Germany, and France
- 2D grid environment modeling train rescheduling under disruptions (breakdowns, etc.)

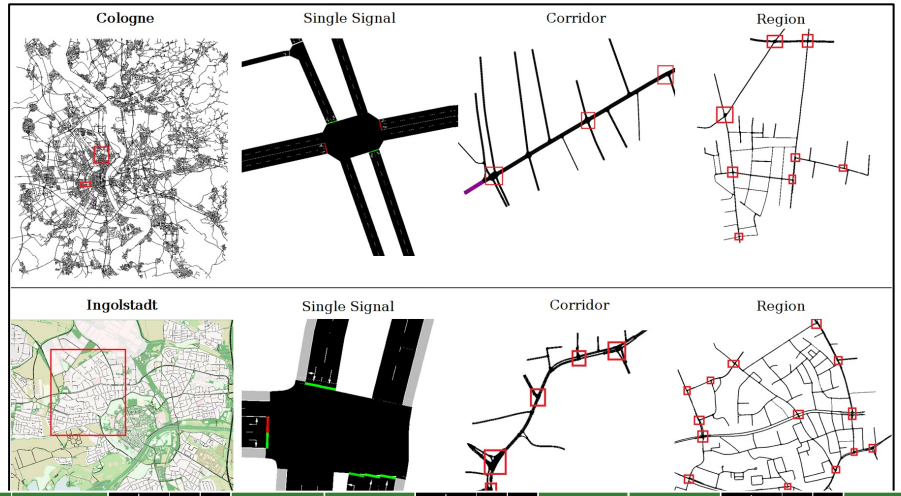


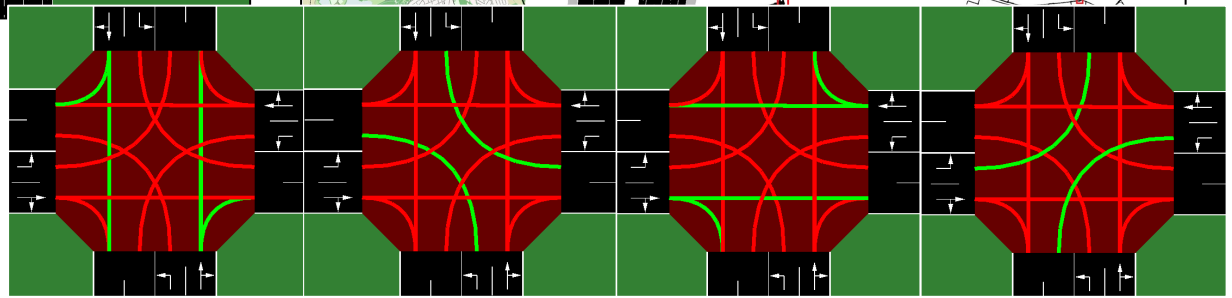
Main page: https://flatland.aicrowd.com/intro.html
Environment description: https://gitlab.aicrowd.com/flatland/flatland/-/blob/master/docs/specifications/railway.md
Rail environment: https://gitlab.aicrowd.com/flatland/flatland/-/blob/master/flatland/envs/rail_env.py
Colab tutorial notebook: https://colab.research.google.com/drive/1RH0OSRTfZjGj2reG2a0xvo0kQYdjeJ-f

Traffic signal control

- *Example environment which interfaces with a high-fidelity simulator*







Main page: https://github.com/LucasAlegre/sumo-rl
Interface with SUMO traffic simulator: https://github.com/LucasAlegre/sumo-rl/blob/main/sumo_rl/environment/env.py

More transportation environment examples

- Air traffic management
  - https://github.com/TUDelft-CNS-ATM/bluesky-gym
  - Vertical control: https://github.com/TUDelft-CNS-ATM/bluesky-gym/blob/main/bluesky_gym/envs/vertical_cr_env.py

- Tactical decision-making in driving
  - Roundabout: https://highway-env.farama.org/environments/roundabout/
  - Implementation: https://github.com/Farama-Foundation/HighwayEnv/blob/master/highway_env/envs/roundabout_env.py

Stable Baselines3 (SB3)

- A set of reliable implementations of reinforcement learning algorithms in PyTorch for solving sequential decision problems.
- Compatible with `gym` environments.

Name
ARS[1]
A2C
CrossQ[1]
DDPG
DQN
HER
PPO
QR-DQN[1]
RecurrentPPO[1]
SAC
TD3
TQC[1]
TRPO[1]
Maskable PPO[1]

https://github.com/DLR-RM/stable-baselines3

Outline

1. Software libraries for sequential decision making
2. **Designing a sequential decision problem**
   a. Design objectives
   b. Algorithmic implications of modeling choices

Key objectives for designing an environment

1. **Decision relevance**: Does the model capture the *real system well enough* to produce meaningful decisions?
   - Garbage in, garbage out

2. **Data & implementation feasibility**: Can we *build and run* the model in practice?
   - Data availability, Sensing requirements, Engineering / monetary cost

3. **Computational tractability**: Can we *solve* the model?
   - Training cost, sample complexity, convergence

1) Decision relevance

- Does the model capture the *real system well enough* to produce meaningful decisions?
- Vast diversity in decisions ←→ Vast diversity in model design needs
- Example: Traffic signal control
  - Operations vs Design vs Planning vs Policy / Strategy

2) Data & implementation feasibility

- Can we *build and run* the model in practice?
- How to implement the various components of an environment: State, Action, Reward, Transitions, Horizon / discount factor
- For each, consider the costs of implementation: monetary cost, time, labor, ethical considerations
- Example: Data
  - Purchase data?
  - Collect data?
  - Simulate data?
  - Some mixture?

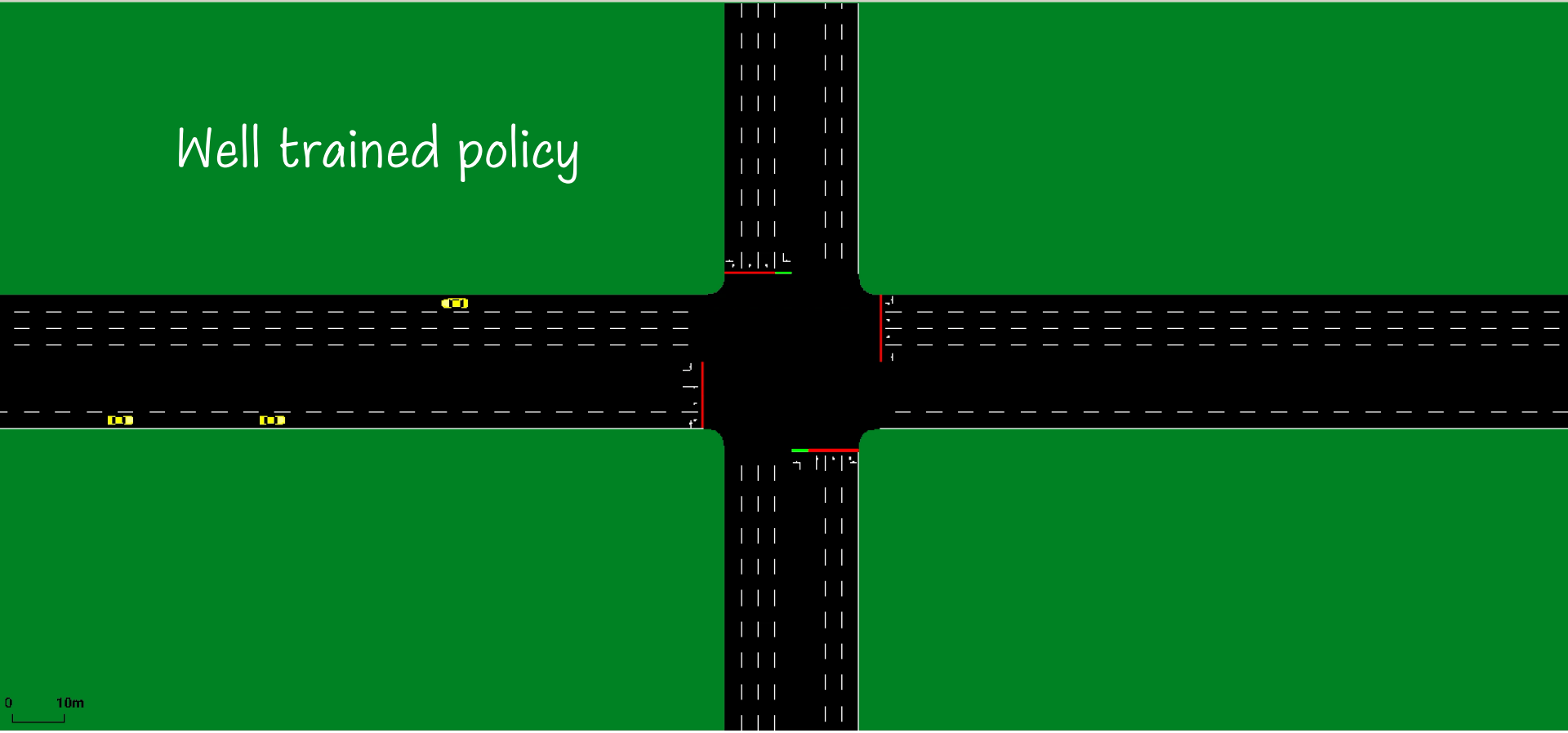
3) Computational tractability

- Can we *solve* the model?

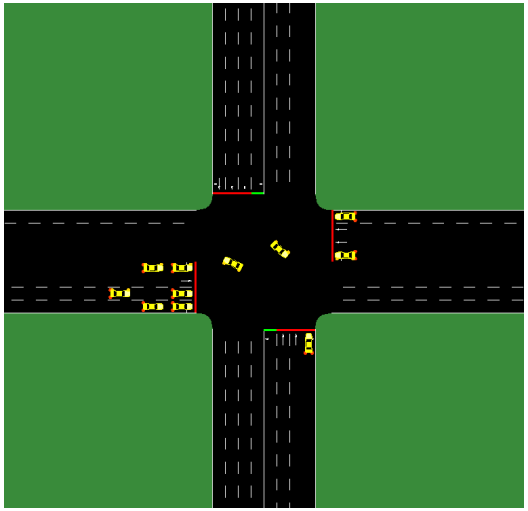
- Recall: Dynamic programming algorithm (stochastic version)
  - Algorithmic complexity: O(|S|^2|A|T)
  - Deterministic shortest path routing: O(|S||A|T)

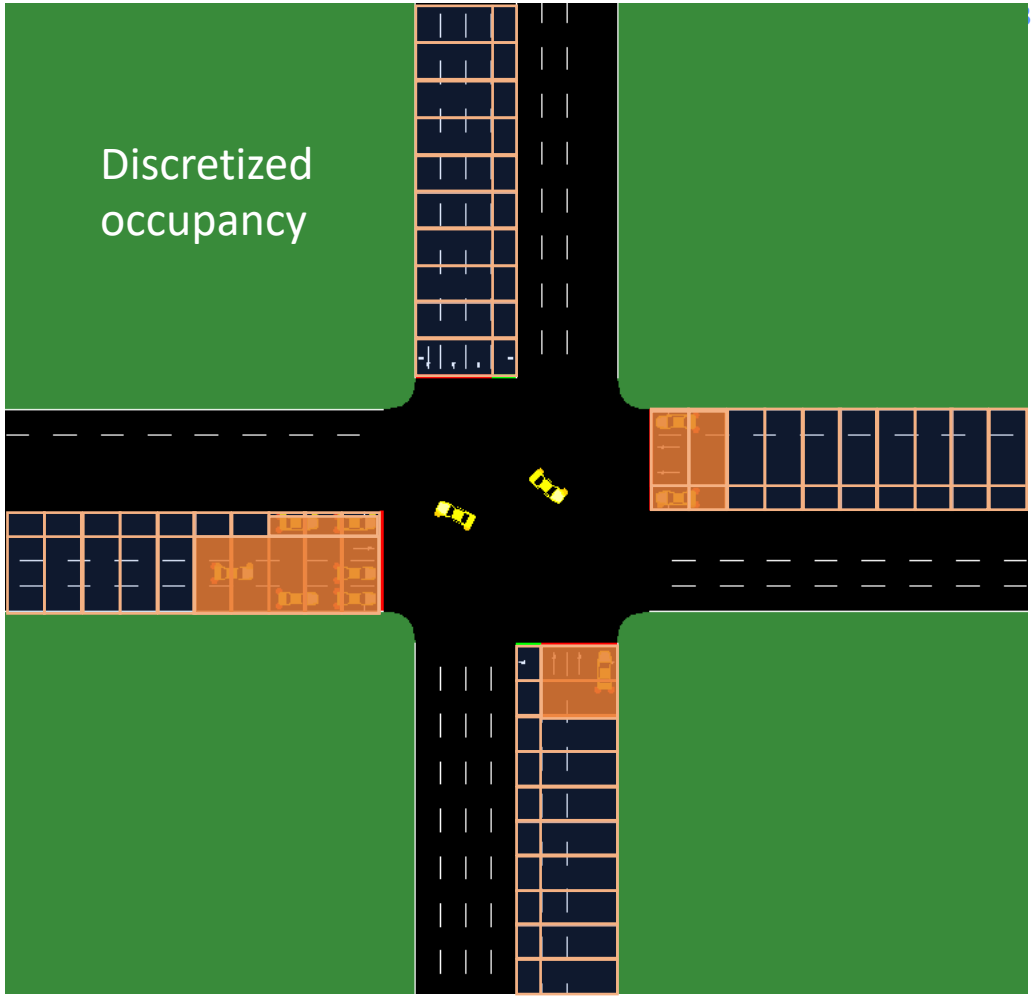
$V_T(s_T) = r_T(s_T)$
for $t = T-1, \ldots, 0$ **do**
  for $s_t \in \mathcal{S}_t$ **do**
    $V_t(s_t) = \max_{a_t \in \mathcal{A}_t(s_t)} r_t(s_t, a_t) + \mathbb{E}_{s_{t+1} \sim P(\cdot|s_t, a_t)}[V_{t+1}(s_{t+1})]$
end for

Example: Traffic signal control



State representation





$O(|S|^2|A|T) = 2^{80\times2}\times4\times5400$

Choosing the right time step

Example: Traffic signal control

- $O(|S|^2|A|T) = 2^{80\times2}\times4\times5400$
- 5400 steps = 1 step/sec x 60 sec/min x 90 min
- A typical traffic cycle is 90 sec
  - 0.1 sec/step → too granular
  - 1 min/step → too coarse

Choosing the right discount factor

- Discounted finite horizon value function

$$V^{\pi}(t,s) = \mathbb{E}\left[\sum_{\tau=t}^{T-1} \gamma^{\tau-t} r(s_\tau, \pi(s_\tau)) + R(s_T) | s_t = s; \pi\right]$$

  - with discount factor $0 \leq \gamma < 1$
  - Note: For any $\gamma \in [0, 1)$ the series always converges (for bounded rewards)

- **Decision relevance**: choose an appropriate γ that suits the problem
  - Small = short-term rewards, big = long-term rewards
- Common choices $\gamma \in \{0.9, 0.95, 0.99, 0.995\}$

Choosing the right discount factor

- Discounted finite horizon value function

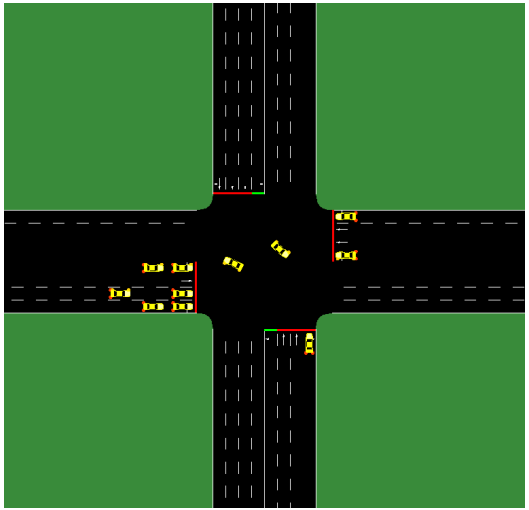
$$V^{\pi}(t,s) = \mathbb{E}\left[\sum_{\tau=t}^{T-1} \gamma^{\tau-t} r(s_\tau, \pi(s_\tau)) + R(s_T) | s_t = s; \pi\right]$$

  - with discount factor $0 \leq \gamma < 1$
  - Note: For any $\gamma \in [0, 1)$ the series always converges (for bounded rewards)

- **Computational tractability**: smaller γ shortens the "effective horizon" of the DP
  - Each individual action only affects future rewards
  - With a discount, an action loses its influence after $O\left(\frac{1}{1-\gamma}\right)$ steps (infinite geometric series)
  - After $\frac{K}{1-\gamma}$ steps, the return (value) has accumulated $1 - \left(\frac{1}{e}\right)^K$ of the possible return (value)
  - For K=1, this is 63%; For K=2, this is 86%; for K=3, this is 95%
- Revised DP complexity: $O\left(|S|^2 |A| \min\left\{T, \frac{1}{1-\gamma}\right\}\right)$

Choosing the right reward function

- How many steps an action influences rewards is also a function of the reward definition



Total wait time among all vehicles (over 90 minutes)

Instantaneous throughput

Instantaneous queue length

sparse

dense

effective horizon

Final note: What if my problem isn't Markovian?

- Common non-Markovian settings: partial observations, multi-agent problems, nonstationary dynamics

- Partial observation example: Traffic signal control
  - Vehicle positions is not enough to define a Markovian process; also need vehicle velocities
  - The current traffic signal phase is not enough; also need time elapsed in the phase to comply with minimum green/red times

- Alternatively, include some recent history
  - This is known as **state augmentation**

Appendix: Modeling a problem of your choice

Modeling a sequential decision problem: 4 stages

Take-home exercise: **Test your understanding** of DP/RL by formulating a sequential decision problem.

1. Select a candidate sequential decision problem.
   - What is the problem, roughly?
2. Is your problem suitable for DP/RL?
   - How would you model the problem?
3. Is DP/RL suitable for your problem?
   - Sequential decision problems are hard. DP/RL has limitations, but does it have fewer limitations than other approaches?
4. If not: Can we make your problem more suitable for DP/RL?
   - Keep iterating!

What is the problem?

- Describe the domain and the problem
- What makes the problem sequential?
  - What about the problem makes *planning* necessary?
- What's the goal?
- What's the challenge?

How would you model the problem?

- How would you define a Markov Decision Process?
  - State
  - Action
  - Reward
  - Transitions
  - Horizon / discount factor
- Defining the reward can be tricky
  - Is there a single measure of success or multiple?
  - Can the reward be measured?
  - Is a reward a function of the states and actions?
- Can the state be measured?
- Can the action be applied? Considerations: safety.
- What data is needed? For informing: transitions (for example, to inform a simulator)

Whether to use DP/RL?

- What are existing approaches to the problem? What are its shortcomings?
- What data is available? Are there data access issues (privacy, cost)? Are there proxies for data?
- What algorithm would you use?
- Are there any “complications”?
  - Is the problem nonstationary? (not an MDP)
  - Are there safety concerns at training or deployment time? (need constraints or risk-sensitive objectives)