Markov Decision Processes

Modeling sequential decision problems

Cathy Wu

1.041/1.200 Transportation: Foundations and Methods

References

1. Readings: Chapter 2. Grokking deep reinforcement learning.
2. Optional Readings: Chapter 1. Grokking deep reinforcement learning.
3. Bertsekas, D. P. (2005). Dynamic programming and optimal control, vol 1. *Belmont, MA: Athena Scientific*, 3rd Edition.
4. Sutton, R. S. and Barto, A. G. (2018). *Reinforcement learning: An introduction*. MIT press.
5. Some slides adapted from Alessandro Lazaric, Matteo Pirotta, Cameron Hickert.

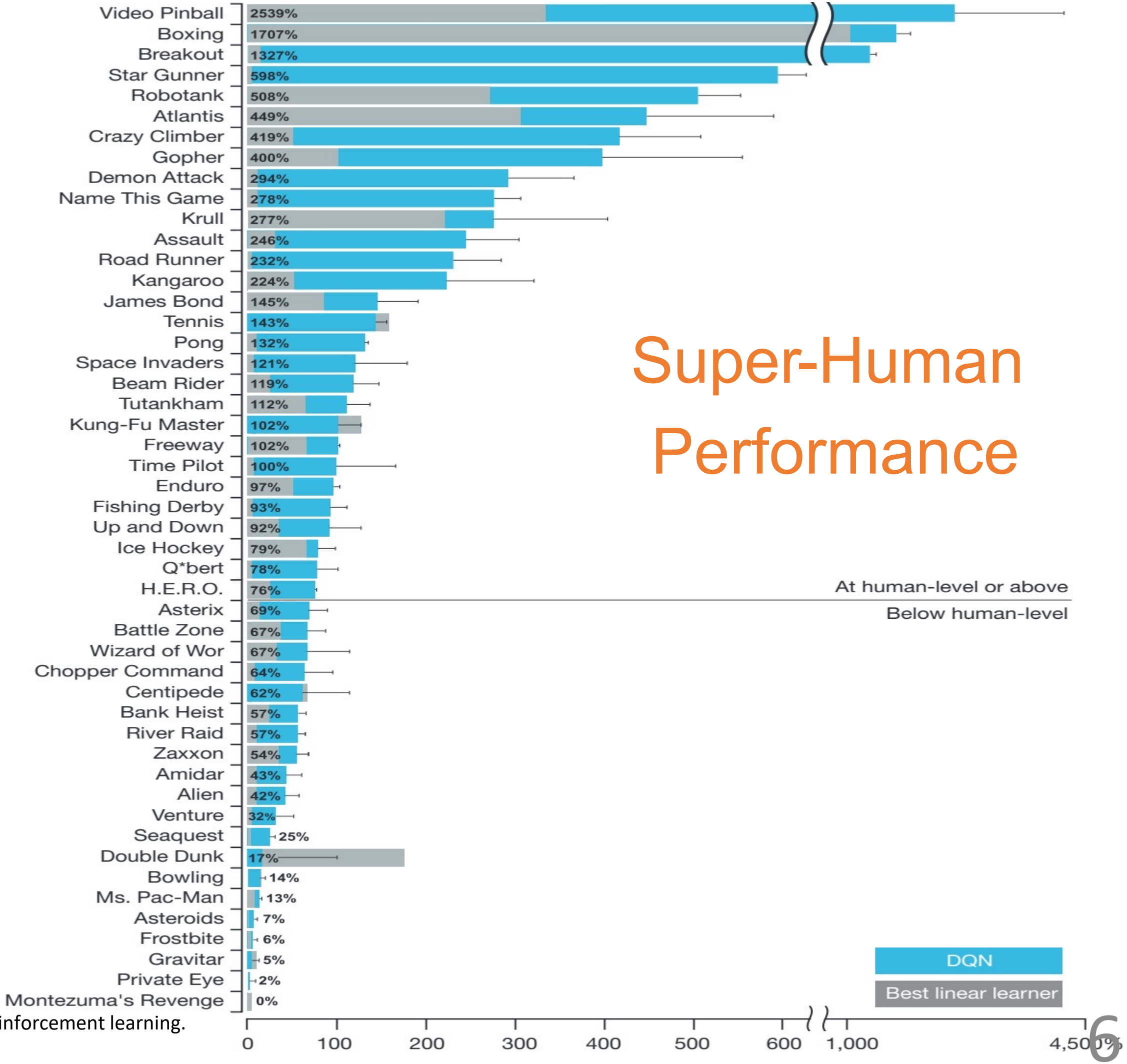
Outline

1. The main characters – the interaction loop
2. Markov Decision Process (MDP)
3. Modeling sequential decision problems as MDPs
4. Emergency medical service vehicle problem

Outline

1. **The main characters – the interaction loop**
   a. Sequential decision making in transportation
   b. Unit 3 overview
   c. A central challenge: exploration vs exploitation
2. Markov Decision Process (MDP)
3. Modeling sequential decision problems as MDPs
4. Emergency medical service vehicle problem

2015:

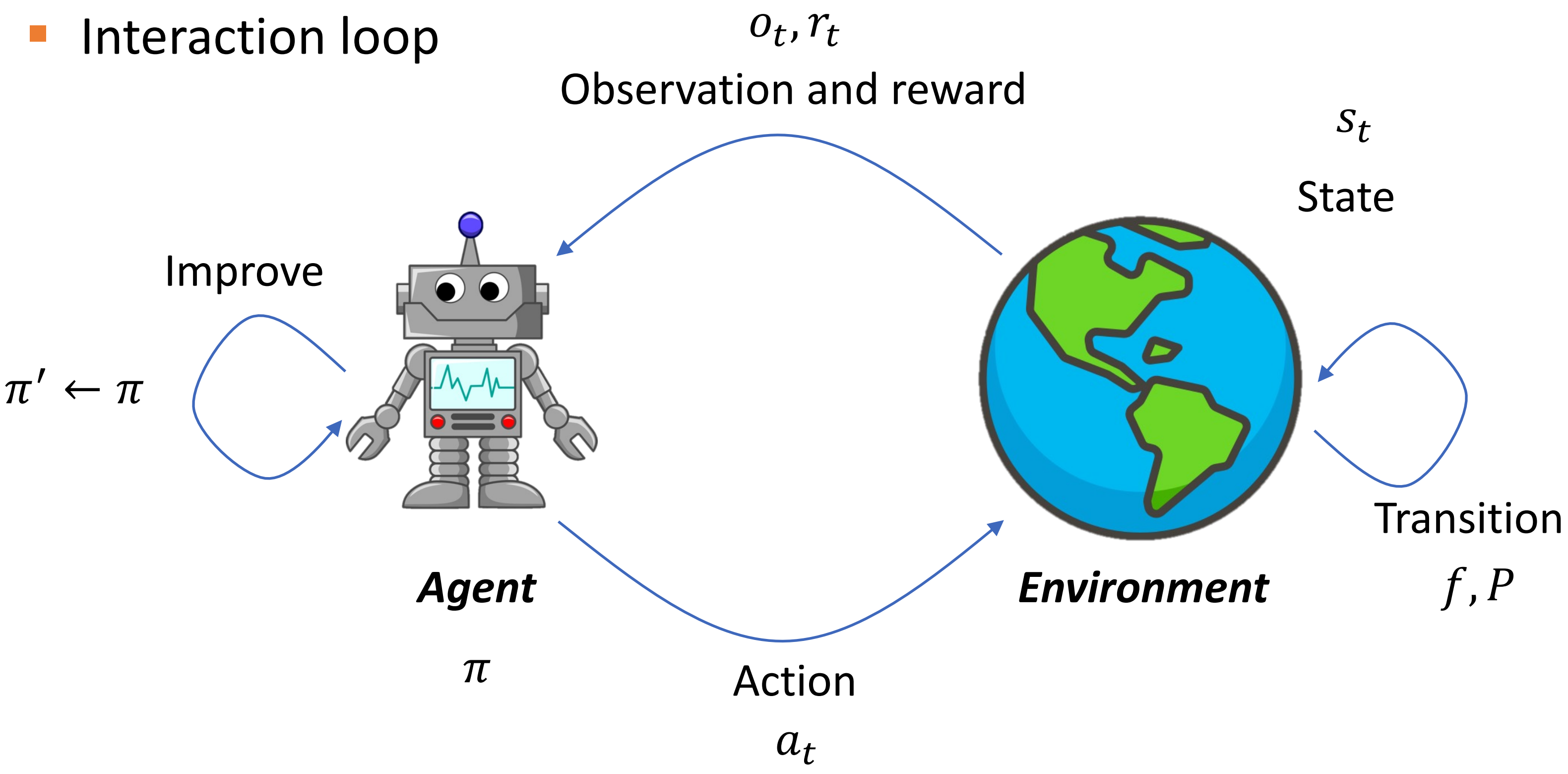


Super-Human Performance

Mnih, V., Kavukcuoglu, K., Silver, D. et al. Human-level control through deep reinforcement learning. Nature 518, 529–533 (2015). https://doi.org/10.1038/nature14236

Introduce the characters*

- Interaction loop



Goal: maximize reward over time (returns, cumulative reward)

* pun intended

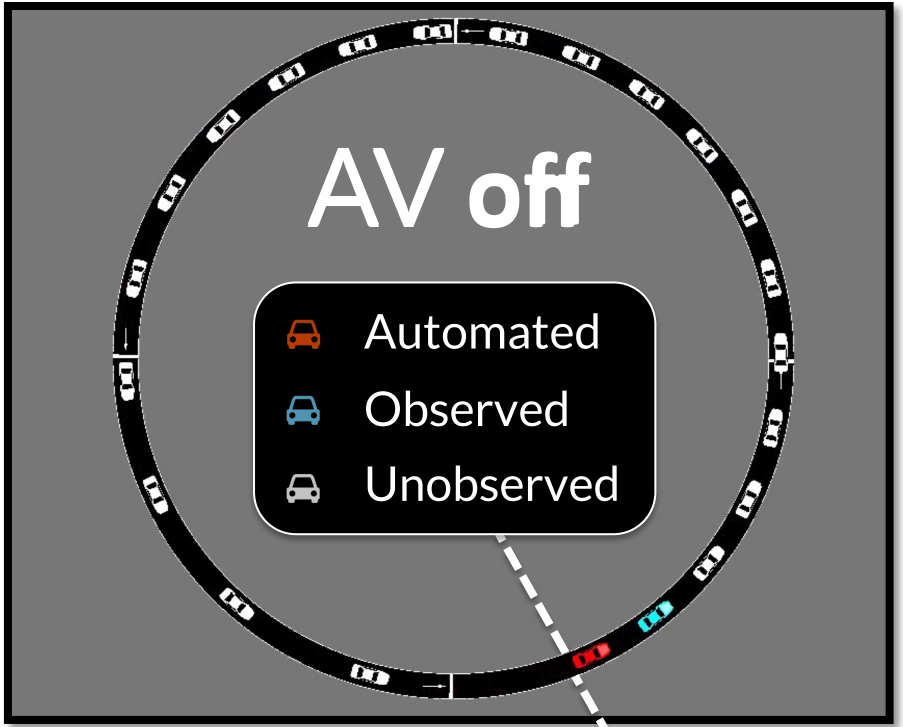
Traffic flow smoothing

Sugiyama, et al. 2008

Wu, et al. “Flow: A Modular Learning Framework for Mixed Autonomy Traffic.” T-RO, 2021.

Traffic flow smoothing (2021)

- Setup
  - Circular track. Sufficient to reproduce traffic waves & jams.
  - 1 self-driving car, 21 human drivers

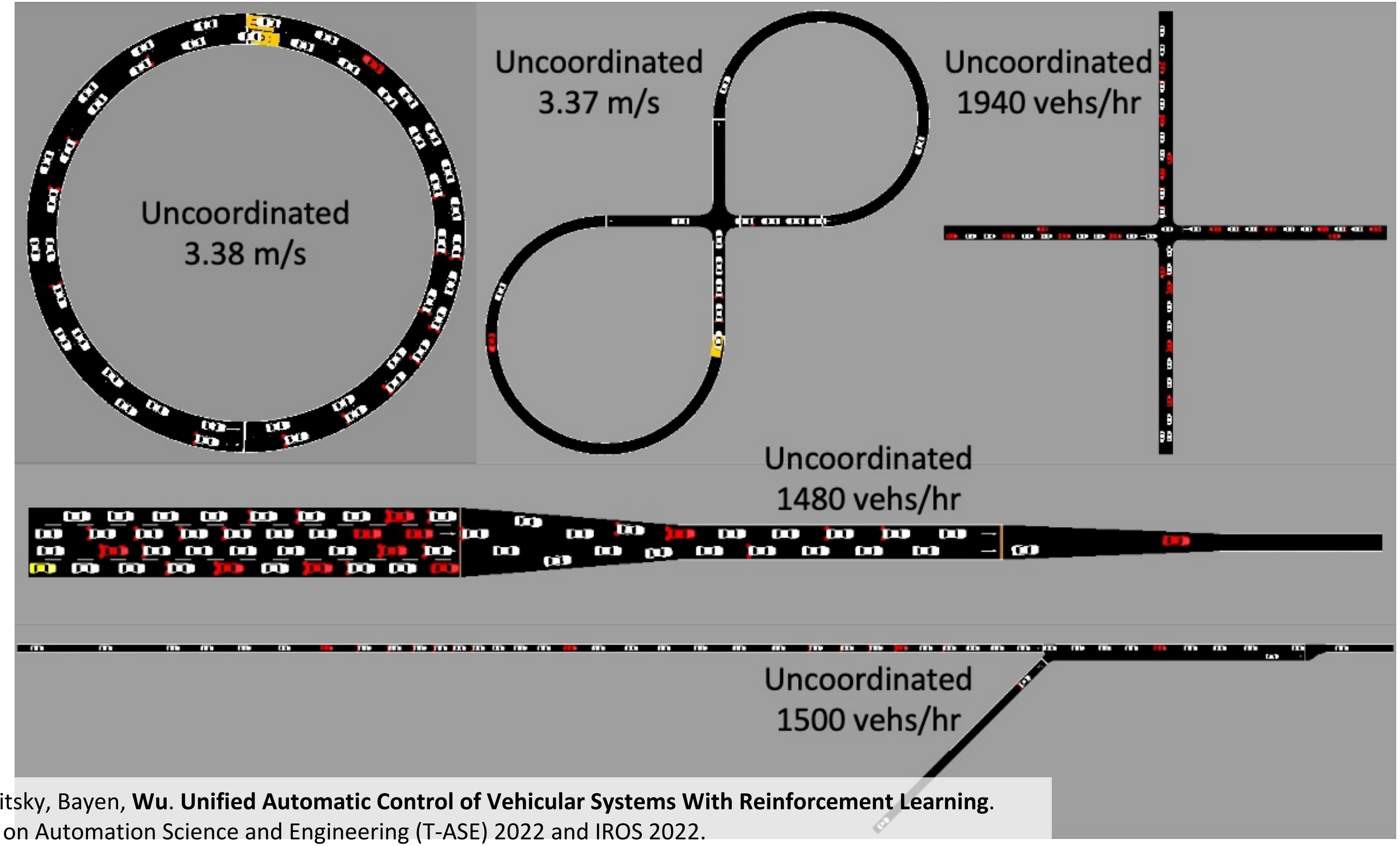


SUMO

Wu, Kreidieh, Parvate, Vinitsky, Bayen. **Flow: A Modular Learning Framework for Mixed Autonomy Traffic**. IEEE Transactions on Robotics (T-RO) 2021 and CoRL 2017.

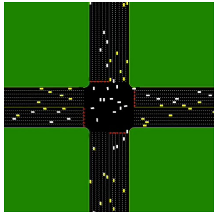
RL + traffic LEGO blocks

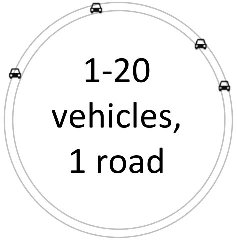
5-30% CAVs → 13-120% improvement

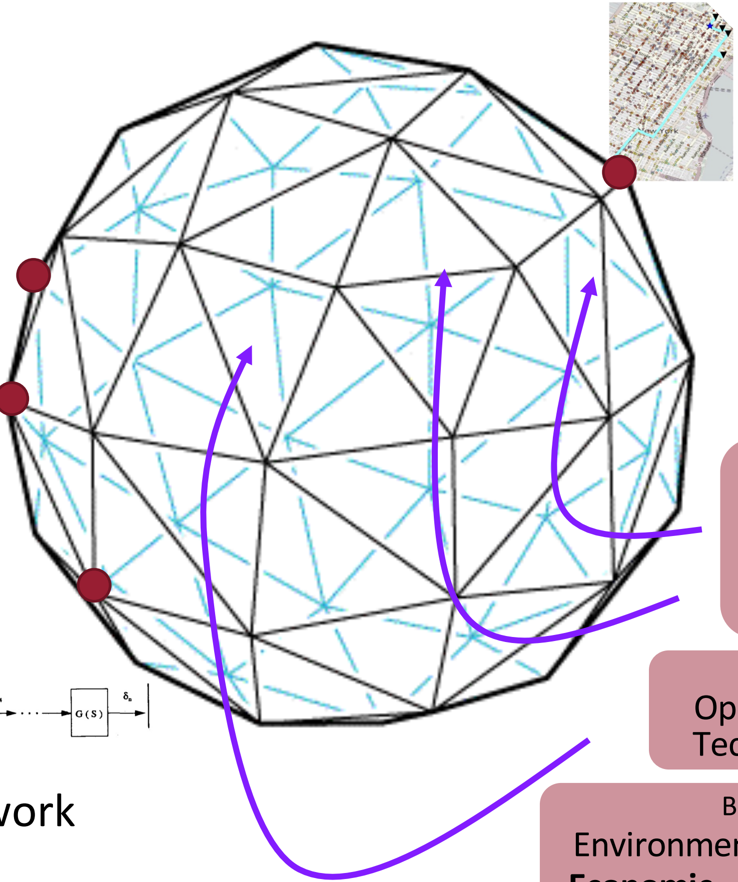


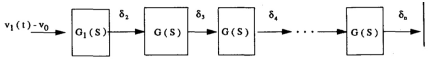
Yan, Kreidieh, Vinitsky, Bayen, **Wu**. **Unified Automatic Control of Vehicular Systems With Reinforcement Learning**.
IEEE Transactions on Automation Science and Engineering (T-ASE) 2022 and IROS 2022.

Roadway autonomy as a design space









Existing work

- **Autonomy enables control and coordination of vehicles. To what end? How effectively? At what cost?**
- **A combinatorial problem space**
  - Multiple objectives
  - Diverse scenarios
  - Spectrum of autonomy technologies
- **Multi-agent interactions**
- **Evolving design specifications**
- **We need methods that support rapid system analysis.**

NETWORK SCENARIO
Network topology
Adoption rate Scale
Levels of autonomy

TECHNOLOGY DESIGN
V2V, V2I communication
Operational design domain
Infrastructure support
Multi-modal integration

COSTS
Operation Risks Taxes
Technology Incentives

MARKET DESIGN
Private vs fleet
Shared rides
Multiple operators

BENEFITS
Environmental Land use
Economic Safety Access

HUMAN FACTORS
Public trust
Behavior drift

Sequential decision making in transportation

We are concerned with making a sequence of decisions, each of which may affect the next, leading overall to a good outcome.

- **Routing**: What is the shortest path (sequence of links) between an origin and a destination?
  - Imagine greedy strategy for routing, selecting the shortest connected edge (a bad idea). What could go wrong?
- **Ridehailing**: What is the sequence of dispatch decisions to optimize the ridehailing service?
  - Dispatching a vehicle to pick up a passenger in Region A of a city, will mean that vehicle is not available to be dispatched to Region B if a new request were to come in.
- **Scheduling**: Trains, buses, airplanes, ships. What is the sequence of trains, etc. to run to meet the travel demands and minimize costs?

Sequential decision making in transportation

We are concerned with making a sequence of decisions, each of which may affect the next, leading overall to a good outcome.

- **Transportation logistics**: What is the sequence of warehouse operations, trucking decisions, and labor decisions, in order to maximize profit for a logistics company?
  - A logistics company has a certain number of delivery trucks, warehouses, labor hours, and time-sensitive customer requests (think Amazon Prime). More customer requests may come in in the meantime.
- **Disaster planning**: What sequence of regions (or parts of a building) should be evacuated in the event of an emergency to minimize harm? What routes should they take?

Sequential decision making in transportation

We are concerned with making a sequence of decisions, each of which may affect the next, leading overall to a good outcome.

- **Traffic signal timings**: What sequence of red, green patterns (or what sequence of phase timings) results in high intersection throughput? High network throughput?
- **Controlling an automated vehicle**: What sequence of accelerations and lane changes leads of a vehicle leads to fuel efficient driving?
- **Integrating autonomy into transportation systems**: What sequence of accelerations and lane changes of fraction of controlled vehicles leads to reduced traffic congestion in the overall transportation system?

Unit 3 overview

- Deep reinforcement learning is an emerging paradigm for solving complex sequential decision problems, which are prevalent in transportation.
- The learning objective of this unit is to:
  - Learn how to model sequential decision problems.
  - Learn the foundations of deep reinforcement learning algorithms. In particular, we focus on Deep Q Networks (DQN), the basis for a class of widely used algorithms in deep reinforcement learning.
  - Learn how to apply deep reinforcement learning methods to transportation.

Lecture-by-lecture unit overview:

- L13: Markov Decision Processes - Modeling sequential decision problems
- L14: Dynamic programming - Solving sequential decision problems
- L15: Value iteration - Solving infinite horizon problems
- L16: Reinforcement learning - Unknown transitions and rewards
- L17: Deep (reinforcement) learning - Handling very large state spaces
- L18: Advanced topics in sequential decision making and transportation

Key challenge: huge decision spaces

- Arcade Learning Environment (ALE): framework that allows researchers and hobbyists to develop AI agents for Atari 2600 games
- ALE parameters
  - 60 frames per sec
- Suppose a game is 2 minutes long
- Horizon is 2 * 60 * 60 = 7200 steps long
- Given 3 actions, the decision space is $3^{7200} \approx 10^{3435}$

$a_t = \text{left}$

t

For reference:
There are between 10^{78} to 10^{82} atoms in the observable universe.

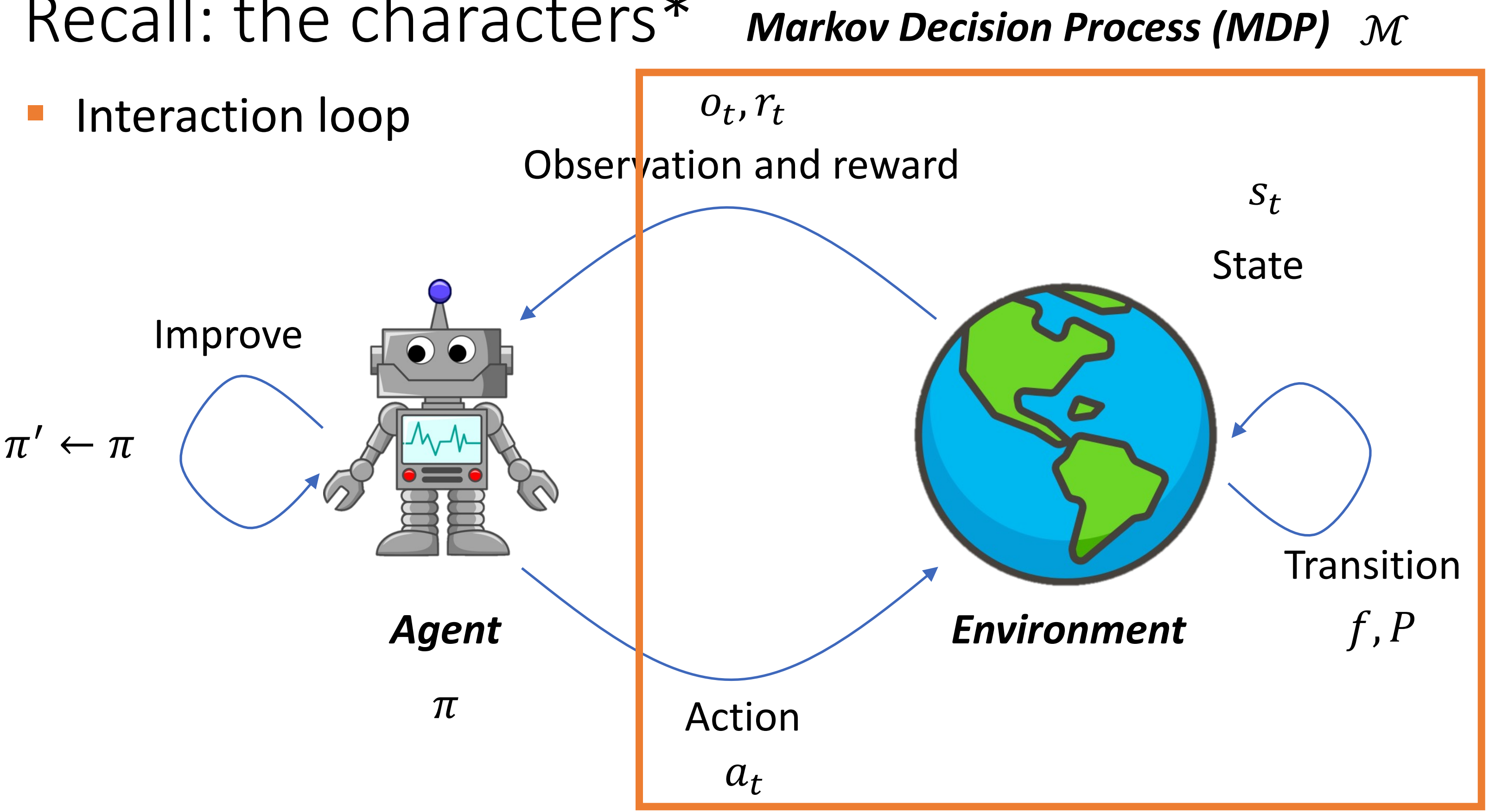
Cannot only explore. Cannot only exploit.
Must trade off exploration and exploitation.

Outline

1. The main characters – the interaction loop
2. **Markov Decision Process (MDP)**
    a. The optimization problem
    b. Examples
    c. Assumptions
    d. Policy
3. Modeling sequential decision problems as MDPs
4. Emergency medical service vehicle problem

Recall: the characters*

- Interaction loop



Goal: maximize reward over time (returns, cumulative reward)

* pun intended

Assume for now: finite horizon problems, i.e. $T < \infty$

Used when: there is an intrinsic deadline to meet.

Later: infinite horizon

The value function

Given a policy π (deterministic to simplify notation)

- Finite time horizon T: deadline at time T, the agent focuses on the sum of the rewards up to T.

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

where R is a value function for the final state.

- Shorthand: $V_t^{\pi}(s)$ or simply V_t^{π} (think: vector of size $|S|$)

Optimization Problem

- Our goal: achieve the best value
  - Max value-to-go (min cost-to-go)

Definition (Optimal policy and optimal value function)

The solution to an MDP is an optimal policy π^* satisfying

$$\pi^* \in \arg\max_{\pi \in \Pi} V_0^{\pi}$$

where Π is some policy set of interest.

The corresponding value function is the optimal value function

$$V^* = V_0^{\pi^*}$$

Expectations

- Technical note: the expectations refer to all possible stochastic trajectories.
- A (possibly non-stationary stochastic) policy π applied from state s_0 returns

$$(s_0, r_0, s_1, r_1, s_2, r_2, \dots)$$

- Where $r_t = r(s_t, a_t)$ and $s_{t+1} \sim p(\cdot \mid s_t, a_t = \pi_t(s_t))$ are random realizations.

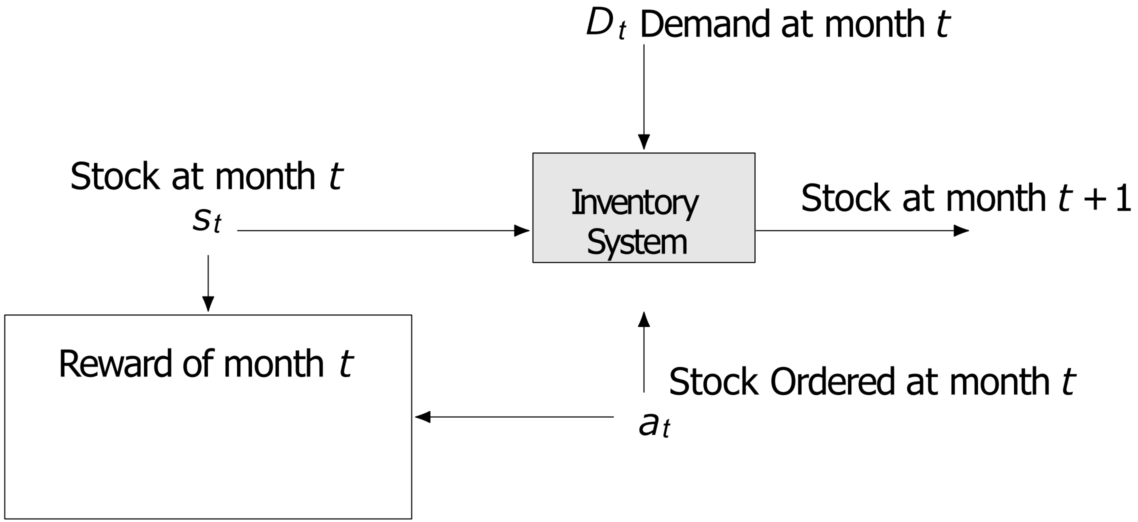
- The value function is

$$V^\pi(t, s) = \mathbb{E}_{(s_1, s_2, \dots)}\left[\sum_{\tau=t}^{T-1} r(s_\tau, \pi(s_\tau)) + R(s_T) | s_t = s; \pi\right]$$

- More generally, for stochastic policies:

$$V^\pi(t, s) = \mathbb{E}_{(a_0, s_1, a_1, s_2, \dots)}\left[\sum_{\tau=t}^{T-1} r(s_\tau, \pi(s_\tau)) + R(s_T) | s_t = s; \pi\right]$$

Example: The Amazing Goods Company Example



Example: The Amazing Goods Company Example

- *Description.* At each month t, a warehouse contains s_t *items* of a specific goods and the demand for that goods is D (stochastic). At the end of each month the manager of the warehouse can *order* a_t more items from the supplier.

- The cost of maintaining an inventory of s is $h(s)$.
- The cost to order a items is $C(a)$.
- The income for selling q items if $f(q)$.
- If the demand $d \sim D$ is bigger than the available inventory s, customers that cannot be served leave.
- The value of the remaining inventory at the end of the year is $g(s)$.
- Constraint: the store has a maximum capacity C.

Recall: Markov Chains

Definition (Markov chain)

Let the *state space S* be a subset of the Euclidean space, the discrete-time dynamic system $(s_t)_{t\in\mathbb{N}} \in S$ is a Markov chain if it satisfies the *Markov property*

$$P(s_{t+1} = s \mid s_t, s_{t-1}, \ldots, s_0) = P(s_{t+1} = s \mid s_t),$$

Given an initial state $s_0 \in S$, a Markov chain is defined by the *transition probability* p

$$p(s' \mid s) = P(s_{t+1} = s' \mid s_t = s).$$

Markov Decision Process

Definition (Markov decision process)

A **Markov decision process** (MDP) is defined as a tuple $M = (S, A, P \text{ or } f, r, H)$ where

- S is the *state* space,

Example: The Amazing Goods Company

- State space: $s \in S = \{0, 1, \ldots, C\}$.

Markov Decision Process

Definition (Markov decision process)

A **Markov decision process** (MDP) is defined as a tuple $M = (S, A, P\ or\ f, r, H)$ where

- S is the *state* space,
- A is the *action* space,

Example: The Amazing Goods Company

- Action space: it is not possible to order more items than the capacity of the store, so the action space should depend on the current state. Formally, at state s, $a \in A(s) = \{0, 1, \dots, C - s\}$.

Markov Decision Process

Definition (Markov decision process)

A **Markov decision process** (MDP) is defined as a tuple $M = (S, A, P \text{ or } f, r, H)$ where

- S is the *state* space,
- A is the *action* space,

often simplified to finite

- $P(s'|s,a)$ is the transition probability with

$$P(s'|s,a) = \mathbb{P}(s_{t+1} = s'|s_t = s, a_t = a)$$

transition equation

$$s' = f_t(s, a, w_t)$$

where $w_t \sim W_t$

Example: The Amazing Goods Company

- Dynamics: $s_{t+1} = [s_t + a_t - d_t]^+$.
- The demand d_t is stochastic and time-independent. Formally, $d_t \overset{\text{i.i.d.}}{\sim} D$.

Markov Decision Process

Definition (Markov decision process)

A **Markov decision process** (MDP) is defined as a tuple $M = (S, A, P \text{ or } f, r, H)$ where

- S is the *state* space,
- A is the *action* space,

often simplified to finite

- $P(s'|s,a)$ is the transition probability with

$$P(s'|s,a) = \mathbb{P}(s_{t+1} = s'|s_t = s, a_t = a)$$

- $r(s,a,s')$ is the immediate reward at state s upon taking action a,

sometimes simply $r(s)$, assumed to be bounded

Example: The Amazing Goods Company

- Reward: $r_t = -C(a_t) - h(s_t + a_t) + f([s_t + a_t - s_{t+1}]^+)$. This corresponds to a purchasing cost, a cost for excess stock (storage, maintenance), and a reward for fulfilling orders.

Markov Decision Process

Definition (Markov decision process)

A **Markov decision process** (MDP) is defined as a tuple $M = (S, A, P\ or\ f, r, H)$ where

- S is the *state* space,
- A is the *action* space,

often simplified to finite

- $P(s'|s,a)$ is the transition probability with
$$P(s'|s,a) = \mathbb{P}(s_{t+1} = s'|s_t = s, a_t = a)$$
- $r(s,a,s')$ is the immediate reward at state s upon taking action a,

sometimes simply $r(s)$

- H is the horizon.

Example: The Amazing Goods Company

- The horizon of the problem is 12 (12 months in 1 year).

Markov Decision Process (infinite horizon preview)

Definition (Markov decision process)

A **Markov decision process** (MDP) is defined as a tuple $M = (S, A, P\ or\ f, r, H)$ where

- S is the *state* space,
- A is the *action* space,

often simplified to finite

- $P(s'|s,a)$ is the transition probability with
$$P(s'|s,a) = \mathbb{P}(s_{t+1} = s'|s_t = s, a_t = a)$$
- $r(s,a,s')$ is the immediate reward at state s upon taking action a,

sometimes simply $r(s)$

- $\gamma \in [0,1)$ is the discount factor.

Example: The Amazing Goods Company

- Discount: $\gamma = 0.91667$. A dollar today is worth more than a dollar tomorrow.
- The effective horizon of the problem is 12 (12 months in 1 year), i.e. $\mathrm{H} \approx \frac{1}{1-\gamma}$.

Markov Decision Process

Definition (Markov decision process)

A **Markov decision process** (MDP) is defined as a tuple $M = (S, A, P \text{ or } f, r, H)$ where

- S is the *state* space,
- A is the *action* space,

often simplified to finite

- $P(s'|s,a)$ is the transition probability with
$$P(s'|s,a) = \mathbb{P}(s_{t+1} = s'|s_t = s, a_t = a)$$
- $r(s,a,s')$ is the immediate reward at state s upon taking action a,

sometimes simply $r(s)$

- H is the horizon.

Example: The Amazing Goods Company

- Objective: $V(s_0; a_0, \ldots) = \sum_{t=0}^{H-1} r_t + r_H$, where $\mathrm{r}_{12} = \mathrm{g}(s_{12})$. This corresponds to the cumulative reward, including the value of the remaining inventory at "the end."

Markov Decision Process

Definition (Markov decision process)

A **Markov decision process** (MDP) is defined as a tuple $M = (S, A, P \text{ or } f, r, H)$ where

- S is the *state* space,
- A is the *action* space,

often simplified to finite

- $P(s'|s,a)$ is the transition probability with

$$P(s'|s,a) = \mathbb{P}(s_{t+1} = s'|s_t = s, a_t = a)$$

- $r(s,a,s')$ is the immediate reward at state s upon taking action a,

sometimes simply $r(s)$

- H is the horizon.

☞ In general, a non-Markovian decision process's transitions could depend on much more information:

$$\mathbb{P}(s_{t+1} = s'|s_t = s, a_t = a, s_{t-1}, a_{t-1}, \ldots, s_0, a_0),$$

Markov Decision Process

Definition (Markov decision process)

A **Markov decision process** (MDP) is defined as a tuple $M = (S, A, P \text{ or } f, r, H)$ where

- S is the *state* space,
- A is the *action* space,

often simplified to finite

- $P(s'|s,a)$ is the transition probability with

$$P(s'|s,a) = \mathbb{P}(s_{t+1} = s'|s_t = s, a_t = a)$$

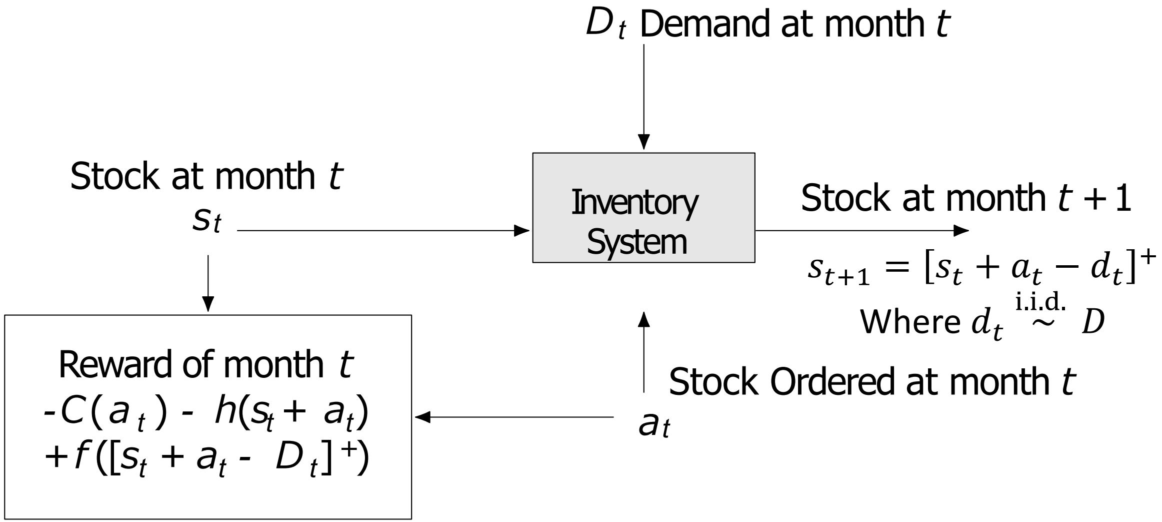
- $r(s,a,s')$ is the immediate reward at state s upon taking action a,

sometimes simply $r(s)$

- H is the horizon.

☞ The process generates trajectories $\tau_t = (s_0, a_0, \ldots, s_{t-1}, a_{t-1}, s_t)$, with $s_{t+1} \sim P(\cdot\,|s_t, a_t)$

Example: The Amazing Goods Company Example



- State space: $s \in S = \{0, 1, \ldots, C\}$.
- Action space: it is not possible to order more items than the capacity of the store, so the action space should depend on the current state. Formally, at state s, $a \in A(s) = \{0, 1, \ldots, C - s\}$.
- Objective: $V(s_0; a_0, \ldots) = \sum_{t=0}^{H-1} r_t + r_H$, where $H = 12$ and $r_{12} = g(s_{12})$

Freeway Atari game (David Crane, 1981)

FREEWAY is an Atari 2600 video game, released in 1981. In FREEWAY, the agent must navigate a chicken (think: jaywalker) across a busy road of ten lanes of incoming traffic. The top of the screen lists the score. After a successful crossing, the chicken is teleported back to the bottom of the screen. If hit by a car, a chicken is forced back either slightly, or pushed back to the bottom of the screen, depending on what difficulty the switch is set to. One or two players can play at once.

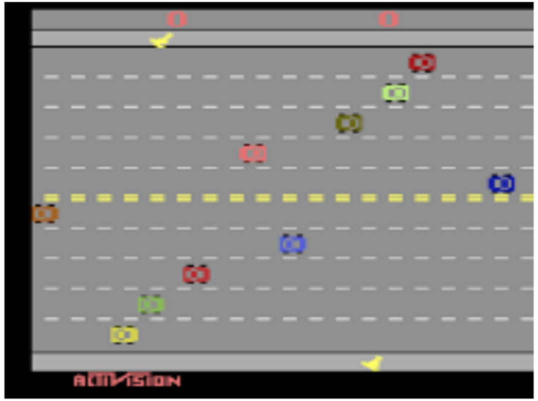


Figure: Atari 2600 video game FREEWAY.

Related applications:

- Self-driving cars (input from LIDAR, radar, cameras)
- Traffic signal control (input from cameras)
- Crowd navigation robot

Learning objective

When using MDPs to model a problem of interest, it is key to understand the underlying assumptions, properties, and generalizations of MDPs.

Markov Decision Process: the Assumptions

Stationarity assumption: the dynamics and reward do not change over time

$$p(s'|s,a) = \mathbb{P}(s_{t+1} = s'|s_t = s, a_t = a) \qquad r(s,a,s')$$

Rule of thumb: stationary → more *re-use* of dynamics/reward → easier to solve

Possible relaxations

- Identify and add/remove the non-stationary components
- Identify the time-scale of the changes
- Work on finite horizon problems

ATARI Breakout

$$\mathbb{P}\Big[\, s_{t+1} =$$

$$\Big|\, s_t =$$

$$, \text{no-move} \,\Big]$$

ATARI Breakout

$\mathbb{P}\Big[s_{t+1} =$  $\Big| s_t =$  $, \text{no-move} \Big]$

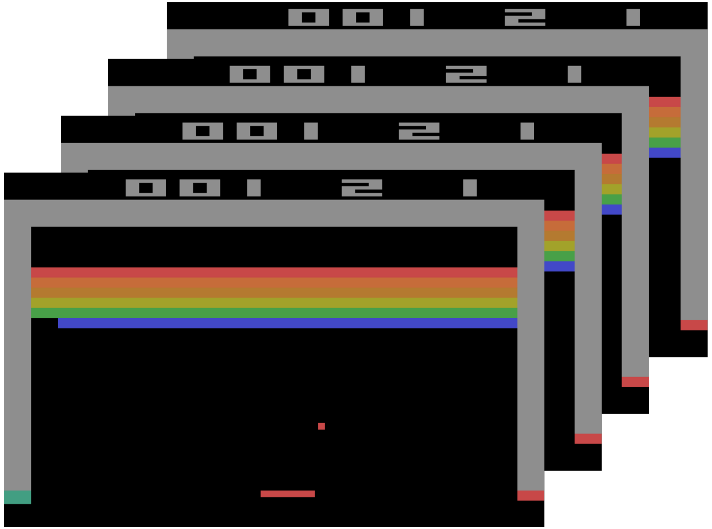
Non-Markov dynamics

Recall: An MDP satisfies the *Markovian property* if

$$\mathbb{P}(s_{t+1} = s | \tau_t, a_t) = \mathbb{P}(s_{t+1} = s | s_t, a_t, s_{t-1}, a_{t-1}, \dots, s_0, a_0) = \mathbb{P}(s_{t+1} = s | s_t, a_t)$$

i.e., the current state s_t and action a_t are sufficient for predicting the next state s.

ATARI Breakout

$$s_t = \Big\{$$  $$\Big\}$$

ATARI Breakout

$\mathbb{P}\Big[s_{t+1} =$  $\Big| \; s_t =$  $, \text{no-move} \Big]$

Non-Markov dynamics

- Non-Markovian dynamics may be unavoidable: partial observation, multi-agent settings, nonstationary dynamics
- Possible relaxation
  - *Partially observable Markov decision process* (POMDP)
  - Two more components
    - Ω, a set of observations
    - O : S×Ω → R≥0, the observation probability distribution

Markov Decision Process: the Assumptions

Time assumption: time is discrete

$$t \rightarrow t+1$$

Rule of thumb: shorter horizon → easier to solve

Possible relaxations

- Identify the proper time granularity
- Most of MDP literature extends to continuous time

ATARI Breakout

$a_t = \text{left}$

t

ATARI Breakout

$a_t = \text{left}$

$t + 1$

Too fine-grained resolution

ATARI Breakout

$a_t = \text{left}$

t

ATARI Breakout

$a_t = \text{left}$

$t + 1$

Too coarse-grained resolution

Markov Decision Process: the Assumptions

Reward assumption: the reward is uniquely defined by a transition (or part of it)

$$r(s, a, s')$$

Rule of thumb: the more informative the reward signal → easier to solve

Possible relaxations

- Distinguish between global goal and reward function
- Move to inverse reinforcement learning (IRL) to induce the reward function from desired behaviors

ATARI Breakout

Reward: score

vs

Reward: score > human baseline

Reward: win/lose

Policy

Definition (Policy)

A decision rule d can be

- Deterministic: $d: S \to A$,
- Stochastic: $d: S \to \Delta(A)$,
- History-dependent: $d: H_t \to A$,
- Markov: $d: S \to \Delta(A)$,

A policy (strategy, plan) can be

- Stationary: $\pi = (d, d, d, \dots)$,
- (More generally) Non-stationary: $\pi = (d_0, d_1, d_2, \dots)$

☞ For simplicity, we will typically write π instead of d for stationary policies, and π_t instead of d_t for non-stationary policies.

Recall: The Amazing Goods Company Example

- *Description.* At each month t, a warehouse contains s_t *items* of a specific goods and the demand for that goods is D (stochastic). At the end of each month the manager of the warehouse can *order* a_t more items from the supplier.

- The cost of maintaining an inventory of s is $h(s)$.
- The cost to order a items is $C(a)$.
- The income for selling q items if $f(q)$.
- If the demand $d \sim D$ is bigger than the available inventory s, customers that cannot be served leave.
- The value of the remaining inventory at the end of the year is $g(s)$.
- Constraint: the store has a maximum capacity C.

Recall: The Amazing Goods Company Example

- *Description.* At each month t, a warehouse contains s_t *items* of a specific goods and the demand for that goods is D (stochastic). At the end of each month the manager of the warehouse can *order* a_t more items from the supplier.

- The cost of maintaining an inventory of s is $h(s)$.
- The cost to order a items is $C(a)$.
- The income for selling q items if $f(q)$.
- If the demand $d \sim D$ is bigger than the available inventory s, customers that cannot be served leave.
- The value of the remaining inventory at the end of the year is $g(s)$.
- Constraint: the store has a maximum capacity C.

Stationary policy composed of deterministic Markov decision rules

$$\pi(s) = \begin{cases} C - s & \text{if } s < M/4 \\ 0 & \text{otherwise} \end{cases}$$

Recall: The Amazing Goods Company Example

- *Description.* At each month t, a warehouse contains s_t *items* of a specific goods and the demand for that goods is D (stochastic). At the end of each month the manager of the warehouse can *order* a_t more items from the supplier.

- The cost of maintaining an inventory of s is $h(s)$.
- The cost to order a items is $C(a)$.
- The income for selling q items if $f(q)$.
- If the demand $d \sim D$ is bigger than the available inventory s, customers that cannot be served leave.
- The value of the remaining inventory at the end of the year is $g(s)$.
- Constraint: the store has a maximum capacity C.

Stationary policy composed of stochastic history-dependent decision rules

$$\pi(s_t) = \begin{cases} U(C - s_{t-1}, C - s_{t-1} + 10) & \text{if } s_t < s_{t-1}/2 \\ 0 & \text{otherwise} \end{cases}$$

Recap

- Stochastic problems are needed to represent uncertainty in the environment and in the policy.
- Markov Decision Processes (MDPs) represent a general class of stochastic sequential decision problems, for which reinforcement learning methods are commonly designed. MDPs enable a discussion of model-free learning (later lectures).
- The Markovian property means that the next state is fully determined by the current state and action.
- Although quite general, MDPs bake in numerous assumptions. Care should be taken when modeling a problem as an MDP.
- Similarly, care should be taken to select an appropriate type of policy and value function, depending on the use case.

Outline

1. The main characters – the interaction loop
2. Markov Decision Process (MDP)
3. **Modeling sequential decision problems as MDPs**
   a. Mixed autonomy traffic (2017)
   b. Google Loon (2020)
4. Emergency medical service vehicle problem

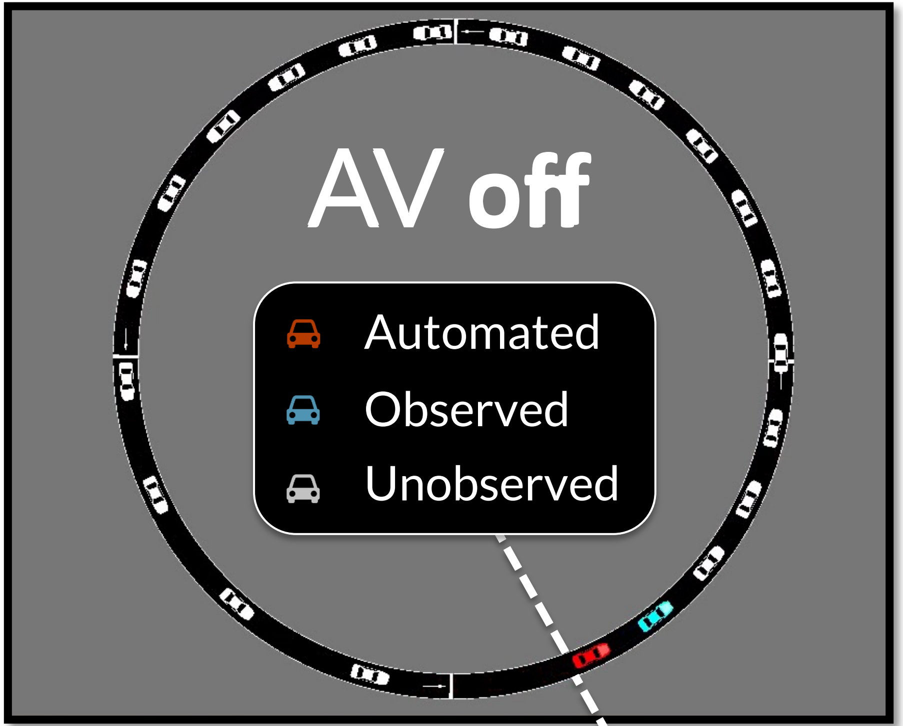
Traffic flow smoothing (2017)

- Assess the potential for self-driving cars to impact traffic flow
  - Setting: mixed autonomy (partial adoption)
- What if even one of these vehicles is not self-driving? Will we see any benefit in throughput before 100%? (2050+)
- Implications for infrastructure planning, public health & safety, equity, climate change
- Source: Wu, et al., "Flow: A Modular Learning Framework for Mixed Autonomy Traffic." IEEE T-RO, 2021. https://arxiv.org/abs/1710.05465
- Source: Wu, et al. "Emergent Behaviors in Mixed-Autonomy Traffic." CoRL, 2017. http://proceedings.mlr.press/v78/wu17a.html

Traffic flow smoothing

- Setup
  - Circular track. Sufficient to reproduce traffic waves & jams.
  - 1 self-driving car, 21 human drivers
  - State: relative velocity & headway
  - Action: acceleration
  - Reward: average velocity (for all cars)
  - Timestep: 0.1 sec
  - Horizon: 5 minutes
  - Algorithm: TRPO

Sugiyama, et al. 2008

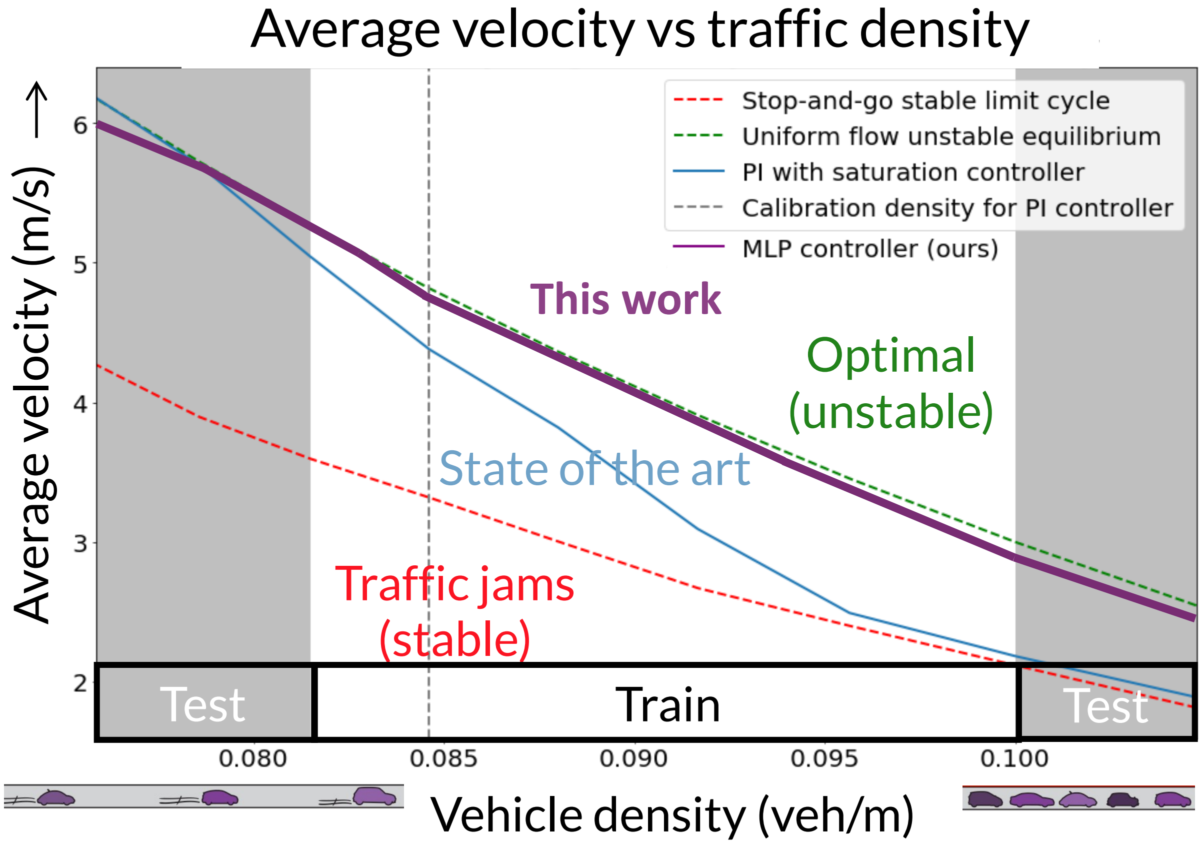


SUMO

Wu

Traffic flow smoothing

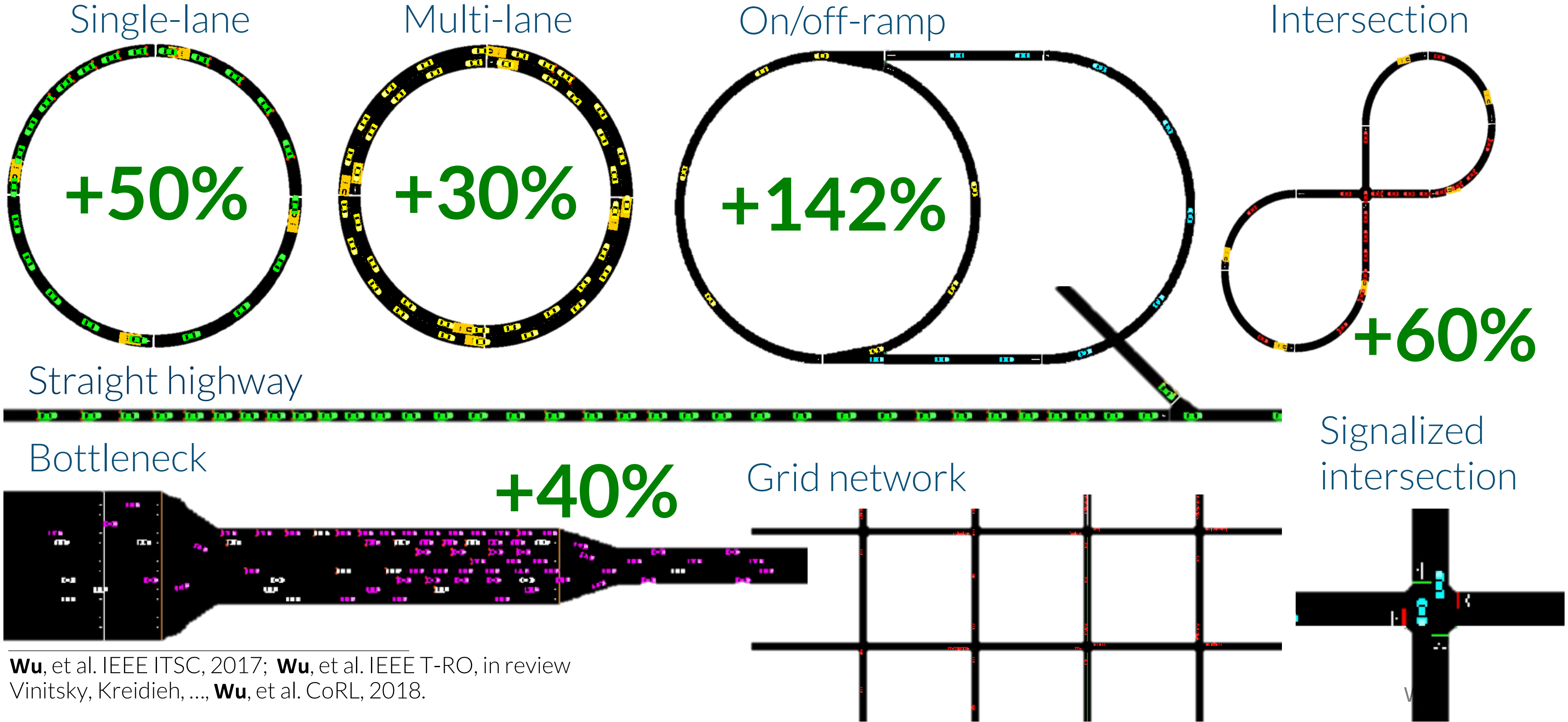
- 5% AVs → 50% improvement in velocity for <u>all</u> cars
- Near-optimal
- Robust
- Training time: a few hours on 1 CPU
- Tweaks that made it work
  - Partial observation sufficient → fast training
  - "Sufficient": Control theory → optimal performance



Traffic LEGO blocks

Benchmarks for autonomy in transportation

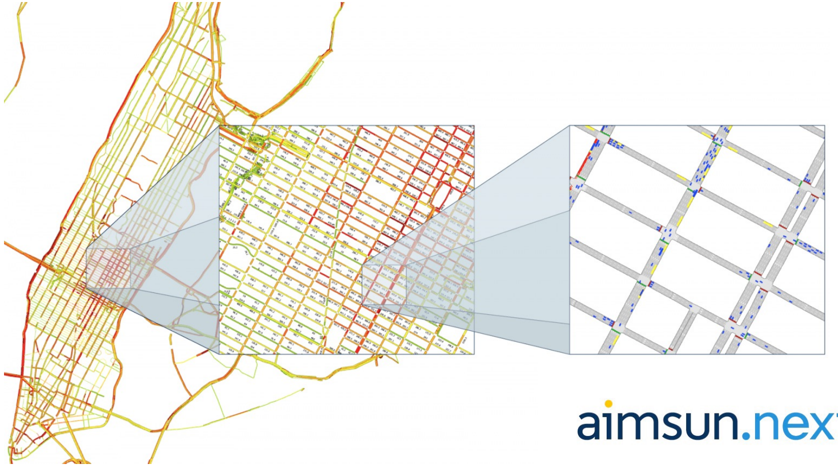
5-10% AVs



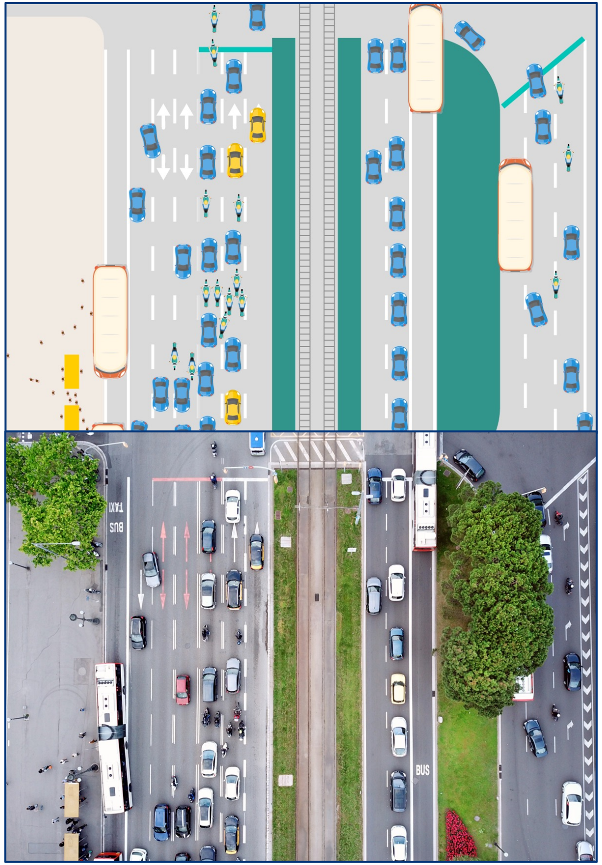
Wu, et al. IEEE ITSC, 2017; **Wu**, et al. IEEE T-RO, in review
Vinitsky, Kreidieh, ..., **Wu**, et al. CoRL, 2018.

Traffic flow smoothing

- Near misses
  - Traffic is notoriously difficult to analyze (cascaded nonlinear dynamics, delayed effects, multi-agent, partially observed)
  - Human driving is fairly predictable in aggregate → can simulate data
  - Traffic phenomena can be reproduced with minimal system complexity → cheap simulation

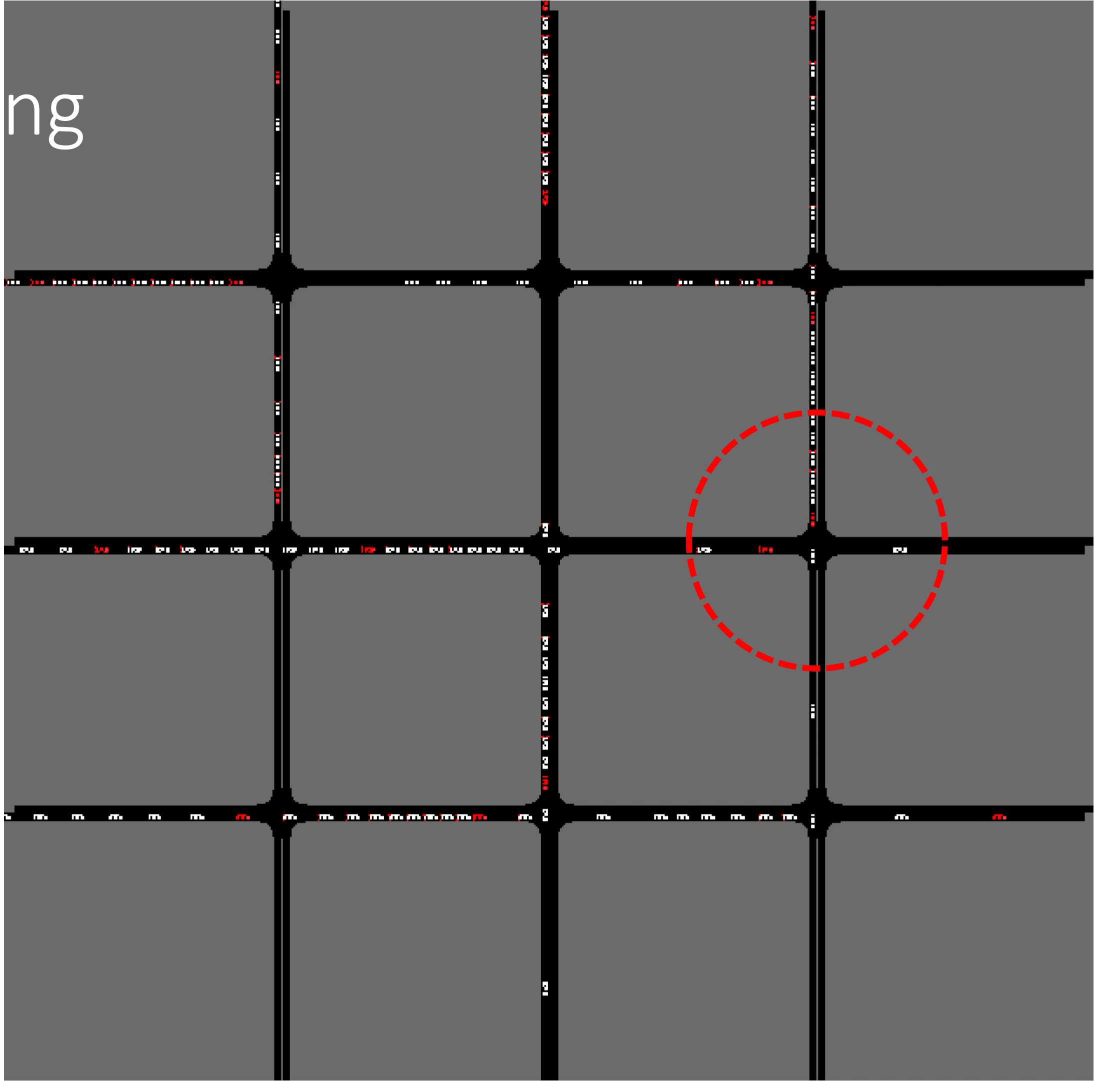


aimsun.next



Traffic flow smoothing

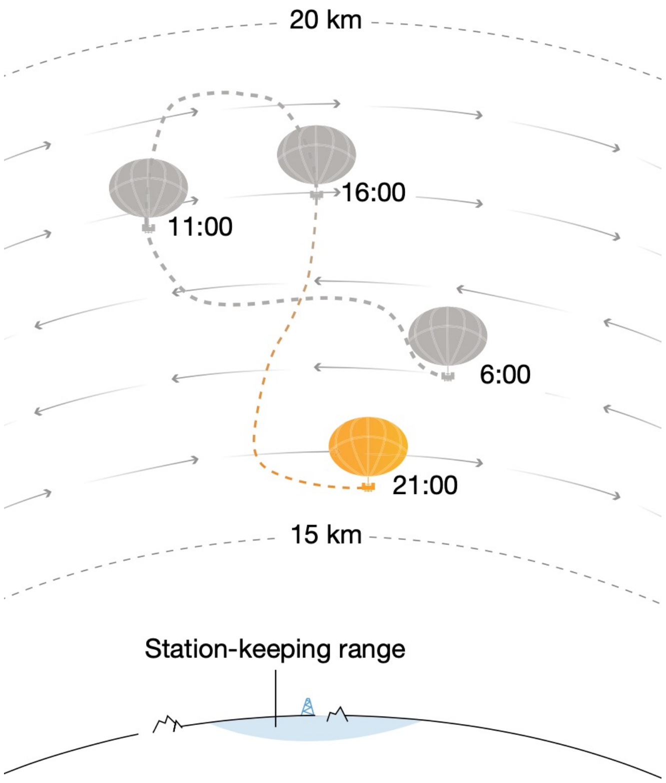
- Challenge: grid network
  - Long-horizon multi-agent coordination & control
- Result: 10% AVs → 26% improvement over human driving baseline
- Tweaks that make it work
  - Shared parameter (homogenous) multi-agent training
  - Restricted observation space
  - (Zero-shot) transfer learning



Yan, Z. and Wu, C., 2021. In preparation.

Google Loon (2020)

- High-altitude balloon navigation (“station keeping”) over a desired area, in order to provide internet connectivity to remote areas.
  - After success in simulation, deployed successfully for ~3,000 flight hours for 2 months over the Pacific Ocean.
- Why a good use case for RL?
  - “Stratospheric equivalent of giving intense attention to watching paint dry”:
    - Requires minute-to-minute attention on boring task for weeks on end.
    - Near-continuous inflow of significant amounts of data.
  - Partial observability difficult for conventional control techniques.
- Source: Bellemare et al., “Autonomous navigation of stratospheric balloons using reinforcement learning” https://www.nature.com/articles/s41586-020-2939-8



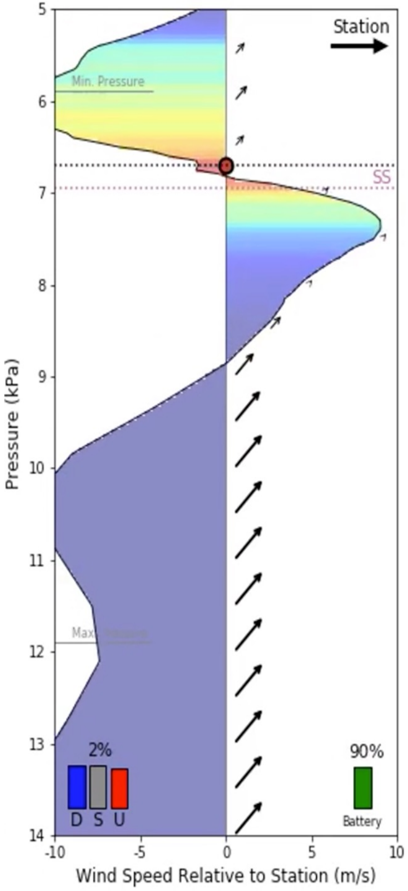
Google Loon

- Setup:
  - Action:
    - Stay (no power cost)
    - Ascend (no power cost)
    - Descend (power cost)
      - Has to pump ambient air into second chamber
  - Timestep: 3 minutes
  - Horizon: 2 days (~1000 steps)
  - Learning algorithm: DQN variant
    - Distributional quantile regression (QR-)DQN
    - Estimates value *distribution*, rather than value mean
    - Quantile regression → use more robust statistics → stabilize training w/ function approximation

Google Loon

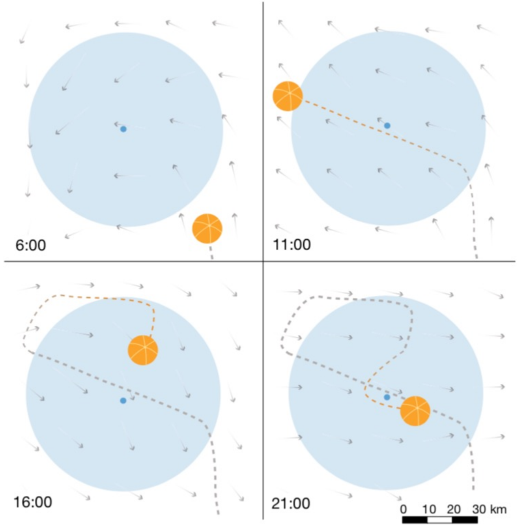
- Setup:
  - State:

Feature	Range	Normalized Range	Notes
Altitude of balloon	5 – 14 kPa	[0, 1]	
Battery charge	0 – 100%	[0, 1]	
Solar elevation	-90 – 90°	[-1, 1]	
Distance to station	$0 - \infty$ km	[0, 1]	Normalized: $f(x) = \frac{x}{x+250}$
Relative bearing	0 – 180°	[-1, 1], [0, 1]	Normalized: $f(\theta) = (\cos\theta, \sin\theta)$
Time from sunrise	0 – 360°	[-1, 1], [-1, 1]	Normalized: $f(\theta) = (\cos\theta, \sin\theta)$
Navigation enabled	Boolean	-	Unary encoding
Has excess energy	Boolean	-	
Descent cost	0 – 300 W	[0, 1]	
Internal pressure ratio	1 – 2	-	
Last command	0, 1, 2	-	Ascend, descend, stay
Wind column ($\times 361$)			
Magnitude	$0 - \infty$ m/s	[0, 1]	Normalized: $f(x) = \frac{x}{x+30}$
Relative bearing	0 – 180°	[0, 1]	
Uncertainty	0 – 100%	[0, 1]	



Wu

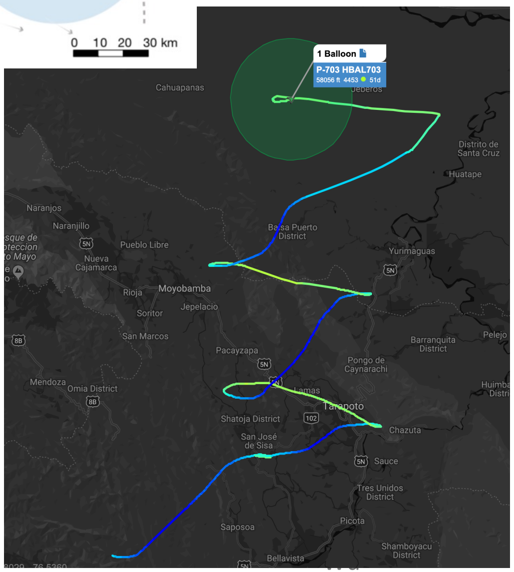
Google Loon



- Setup:
  - Reward: $r(s,a) = r(\Delta,\omega) = f_\omega r_{dist}(\Delta)$

$$r_{\text{dist}}(\Delta) = \begin{cases} 1.0 & \text{if } \Delta < \rho \\ c_{\text{cliff}} 2^{-(\Delta-\rho)/\tau} & \text{otherwise} \end{cases},$$

$$f_\omega = \begin{cases} 0.95 - 0.3\omega & \text{if } \omega > 0 \\ 1.0 & \text{otherwise} \end{cases}.$$



- $c_{cliff} = 0.4$ at 50km and decays by half every $\tau = 100$km
- Δ is the balloon's distance to the station
- ρ is the target radius.
- $\omega \in [0,1]$ is a normalized measure of power consumption during the timestep

Google Loon

- Tweak that made it work:
  - Data augmentation for simulations
    - Issue: Wind is notoriously difficult to model (need HPC, still active research)
    - Solution: Use data from previous balloon flights
    - Issue: woefully insufficient, data cost ~200x of Atari & can't be used to evaluate case in which agent deviates significantly.
    - Solution: Used historical wind data modified with procedural noise to generate arbitrary number of high-resolution wind fields.
- **Discuss**: Why is this sufficient to serve as a "simulator"?

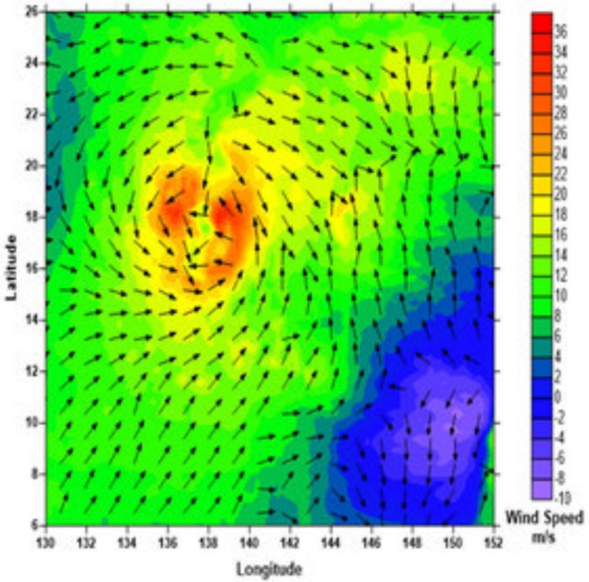


Image Source: Tangao Hu, https://www.researchgate.net/figure/Regional-wind-field-map-a-Wind-direction-map-b-Wind-speed-map-c-Wind-field-map_fig2_320886446

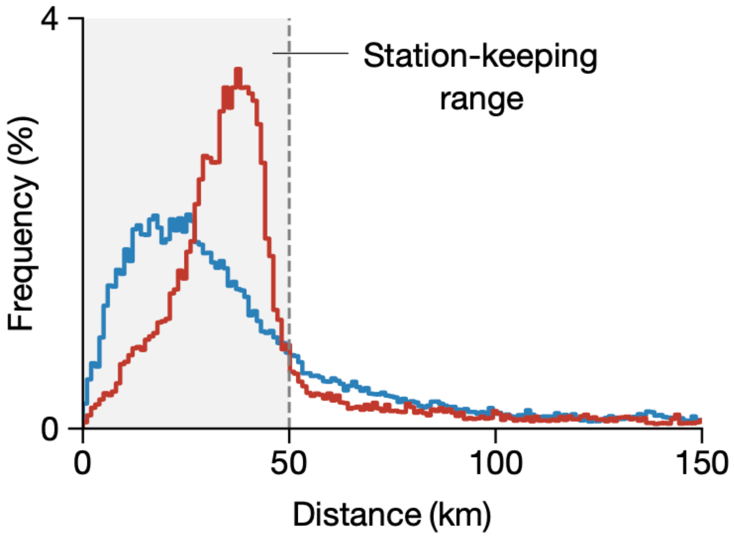
Google Loon

- Near misses:
  - Fortunate to have "simulator"
    - No historical data → No simulator
    - No effective mechanism for augmentation of data → No simulator
  - Balloon actions do not significantly affect wind currents
    - Minnow swimming in ocean → agent has minor effect on system dynamics
    - Big shark swimming in little tank → agent has major effect on dynamics; model must account for them

Image Source: https://movies.disney.com/finding-nemo

Google Loon

- Simulation evaluation: "[T]he difference amounts to a <u>substantial 3.5h per day average improvement in time spent near the station.</u>"

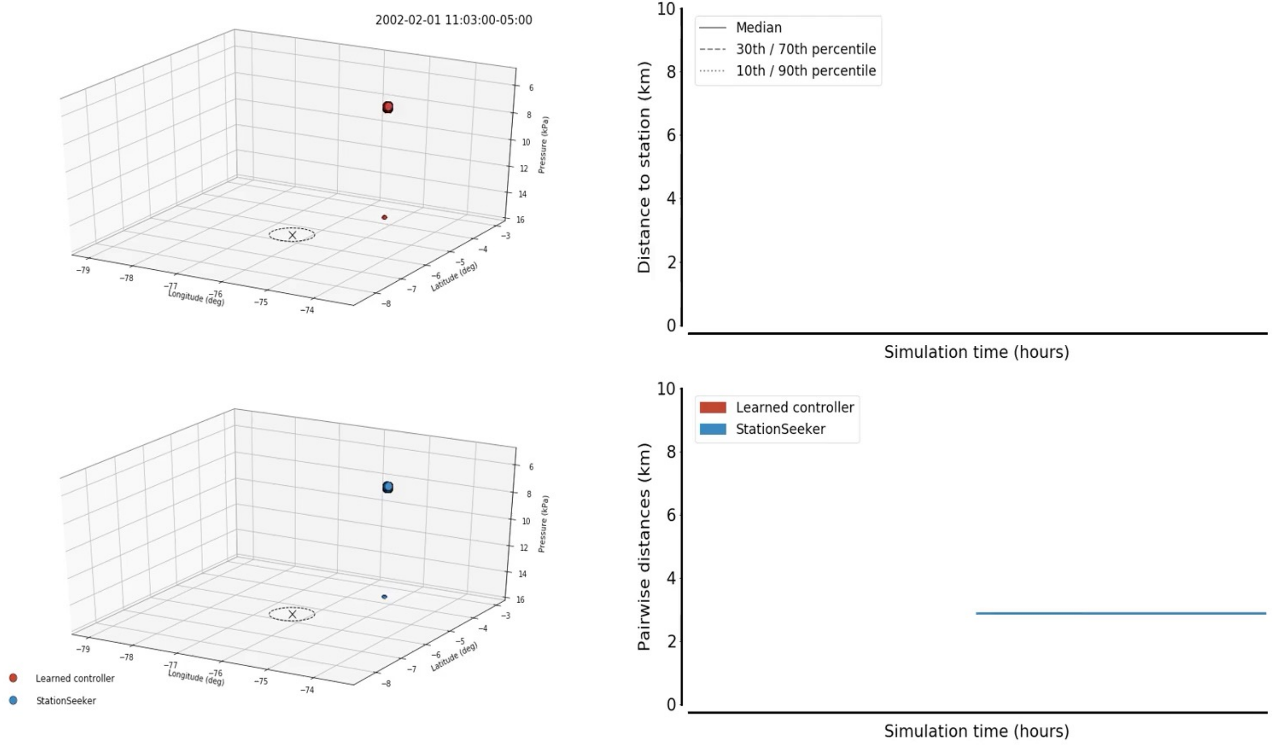


Red = RL controller

Blue = Previous SOTA

Red = RL controller

Blue = Previous SOTA



Outline

1. The main characters – the interaction loop
2. Markov Decision Process (MDP)
3. Modeling sequential decision problems as MDPs
4. **Emergency medical service vehicle problem**

EMS maneuver under mixed autonomy

- Emergency medical service (EMS) vehicle
- Scenario:
  - EMS may stop or travel at low speeds on congested roads (e.g., signalized intersections)

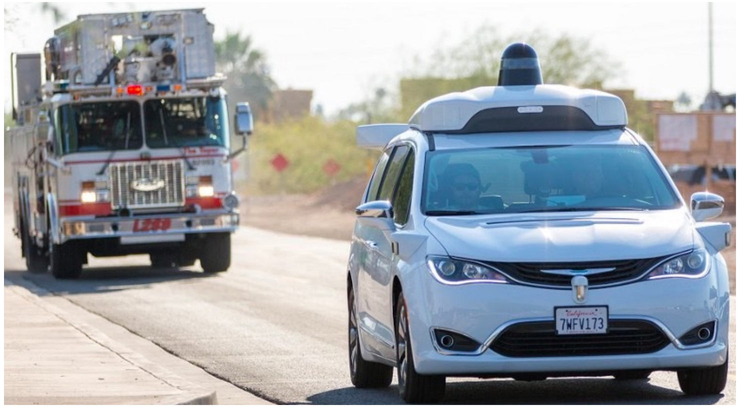
- Motivation: **Reduce** emergency service vehicle (EMS) **travel times** to reduce mortality rate [OBENAUF et al. 2019]

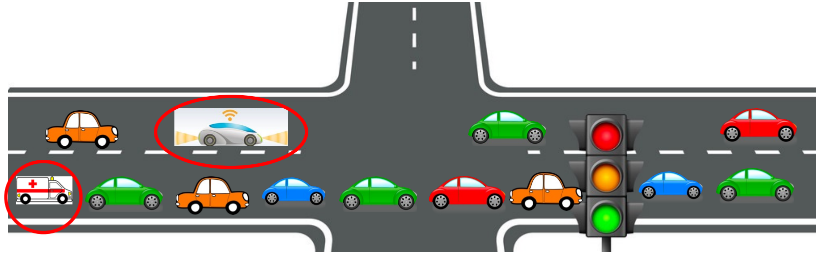
Originated as 1.200 class project!

Based on: Suo, Jayawardana, **Wu**. "Model-free Learning of Multi-objective Corridor Clearance in Mixed Autonomy," 2023. Under review.

EMS maneuver under mixed autonomy (Suo et al., 2023)

- Problem: How should the AV take maneuvers to assist EMS in crossing intersections?
- A specific scenario:
  - Right lane (where the EMS currently locates) fully congested
  - An autonomous vehicle can receive inputs from onboard sensors (e.g., lidar, radar, camera)
  - The AV can communicate with traffic infrastructure and EMS for non-line-of-sight conditions
- The goal of the AV is to assist EMS maneuvers to reduce its travel time crossing the intersection





Exercise: Define a Markov Decision Process to model the problem, including the state space, action space, transitions, reward, and objective function.

Based on: Suo, Jayawardana, **Wu**. "Model-free Learning of Multi-objective Corridor Clearance in Mixed Autonomy," 2023. Under review.

EMS maneuver under mixed autonomy

What **observations** available to the AV?

How to define **reward** for the AV?

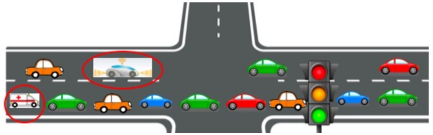
o_t, r_t

s_t

State?

Improve

$\pi' \leftarrow \pi$



Transition

P

Agent

π

Environment

$\mathcal{M}$

Actions available to the AV?

a_t

Based on: Suo, Jayawardana, **Wu**. “Model-free Learning of Multi-objective Corridor Clearance in Mixed Autonomy,” 2023. Under review.