Reinforcement Learning in Digital Finance

General Introduction to Reinforcement Learning

Martijn Mes

University of Twente

February 3, 2025

Funded by
the European Union

Lecture agenda

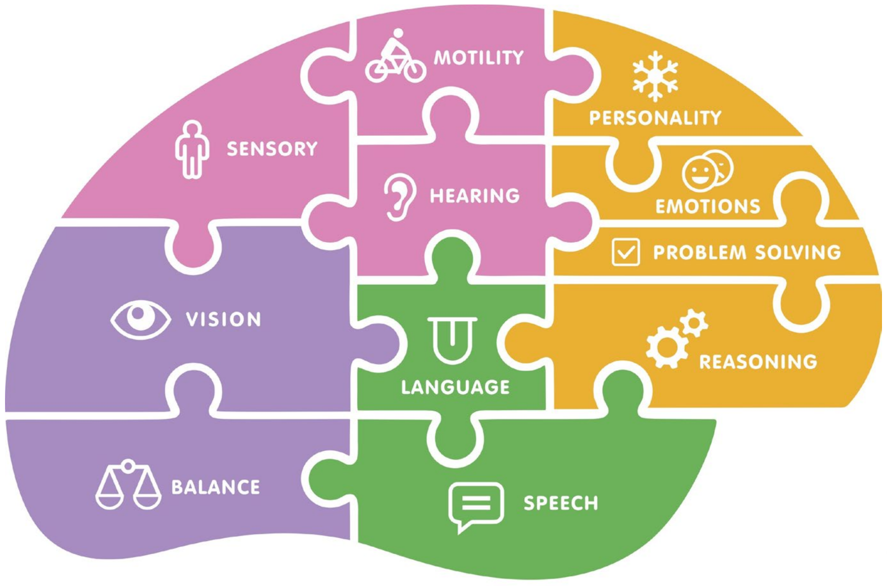
- Positioning Reinforcement Learning
- Introduction Reinforcement Learning
- Markov Decision Processes
- Preview of different forms of RL…

Positioning Reinforcement Learning

Artificial Intelligence

- “Artificial intelligence (AI) refers to systems that display intelligent behaviour by analysing their environment and taking actions – with some degree of autonomy – to achieve specific goals.”
  European Commission

- “AI […] refers to a broad class of technologies that allow a computer to perform tasks that normally require human cognition, including adaptive decision making.”
  Tambe et al. (2019)



DIGITAL

Positioning Machine Learning

Artificial intelligence
The science and engineering of making intelligent machines

AI is the broad field of developing machines that can replicate human behavior, including tasks related to perceiving, reasoning, learning, and problem-solving.

Machine learning
A major breakthrough in achieving AI

Machine learning algorithms detect patterns in large data sets and learn to make predictions by processing data, rather than by receiving explicit programming instructions.

Deep learning
An advanced branch of machine learning

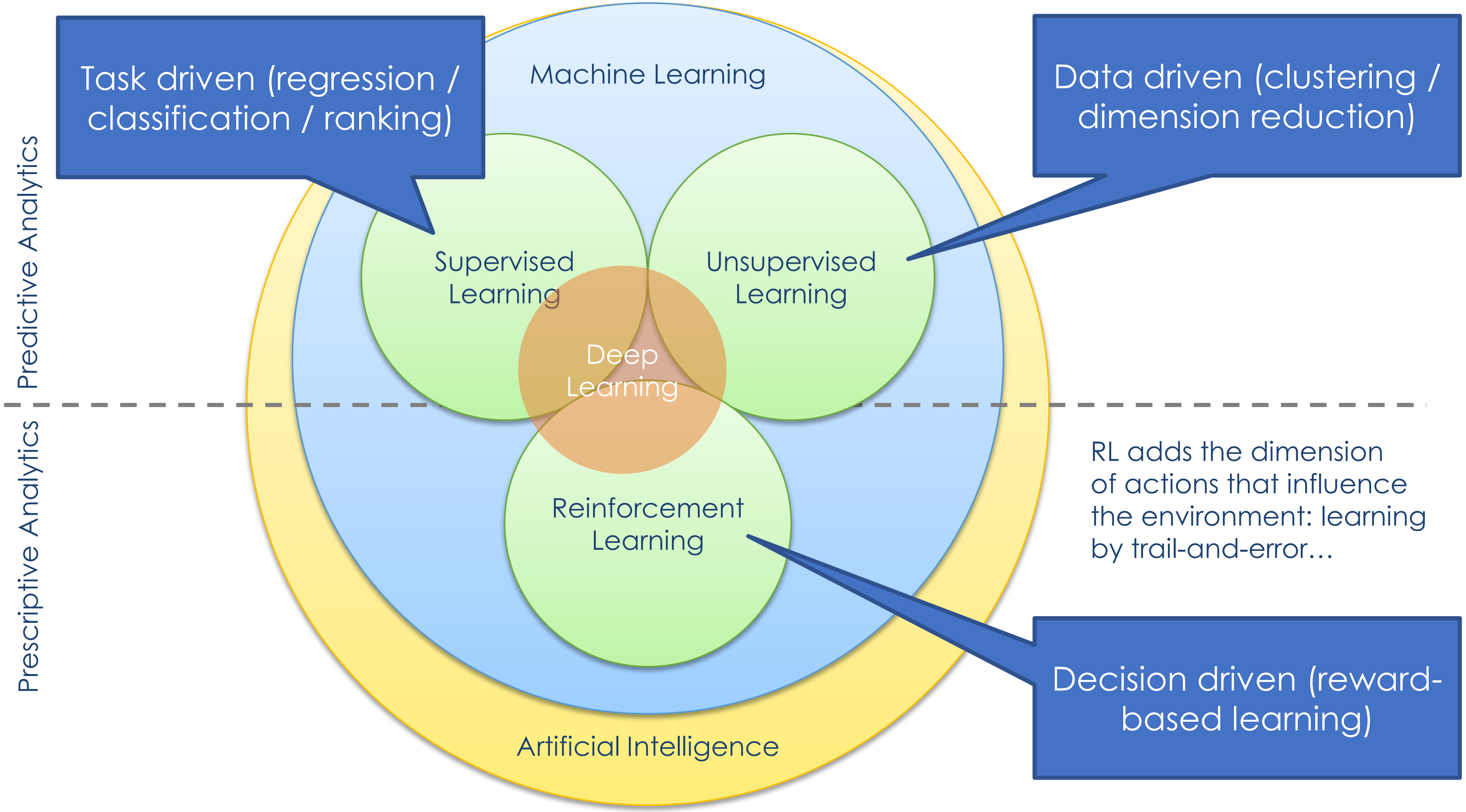
Deep learning uses neural networks, inspired by the ways neurons interact in the human brain, to ingest data and process it through multiple iterations that learn increasingly complex features of the data and make increasingly sophisticated predictions.

Generative AI
An advanced branch of deep learning

Generative AI is a branch of deep learning that uses exceptionally large neural networks called large language models (with hundreds of billions of neurons) that can learn especially abstract patterns. Language models applied to interpret and create text, video, images, and data are known as generative AI.

Source: McKinsey & Company

Positioning Reinforcement Learning



RL adds the dimension of actions that influence the environment: learning by trail-and-error…

DIGITAL

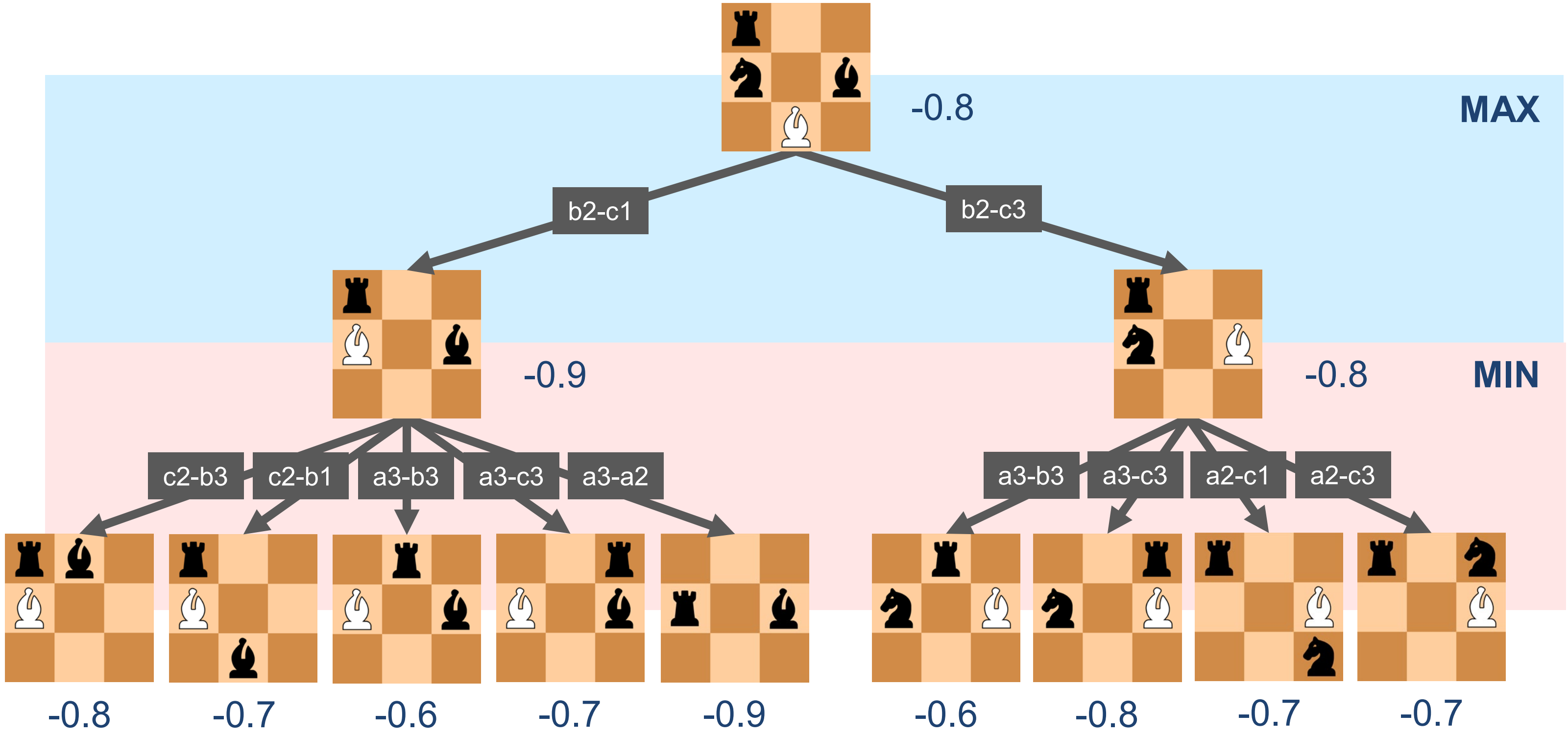
Examples of RL for game AI

- Many nice examples of computers learning to play games, e.g., from Google Deepmind:
  - AlphaGo: https://youtu.be/8tq1C8spV_g (move 37 and 78)
  - AlphaZero: “general purpose” (Shogi, Chess, Go)
  - MuZero: also learning the rules of the game
  - AlphaStar: https://youtu.be/DpRPfidTjDA (StartCraft II)

- Let's start with a simple example: Chess

Chess: Optimization/Enumeration

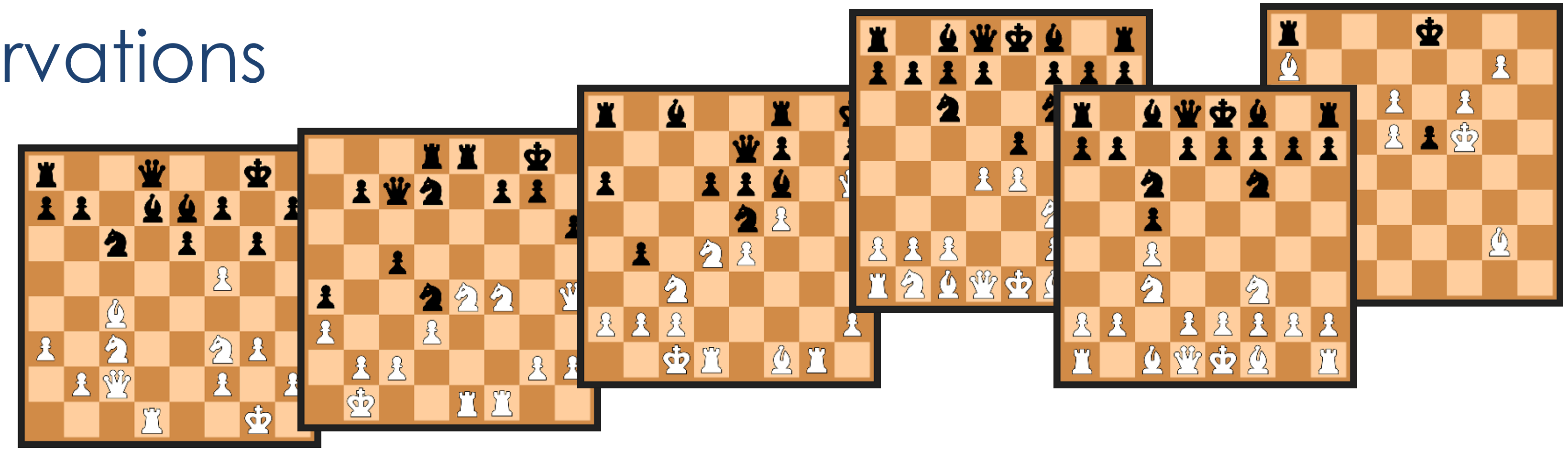
- Game tree complexity: 10^{123} (versus 10^{360} for Go)



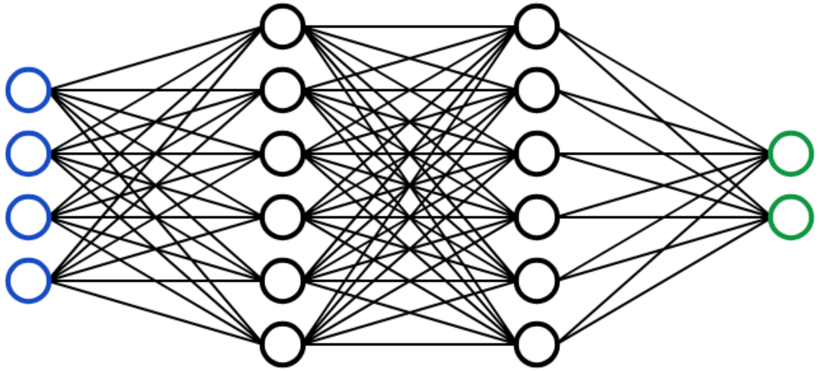
DIGITAL

Chess: Supervised Learning [1/2]

- Collect observations



Input x:
Composition of pieces on the board derived from many game plays



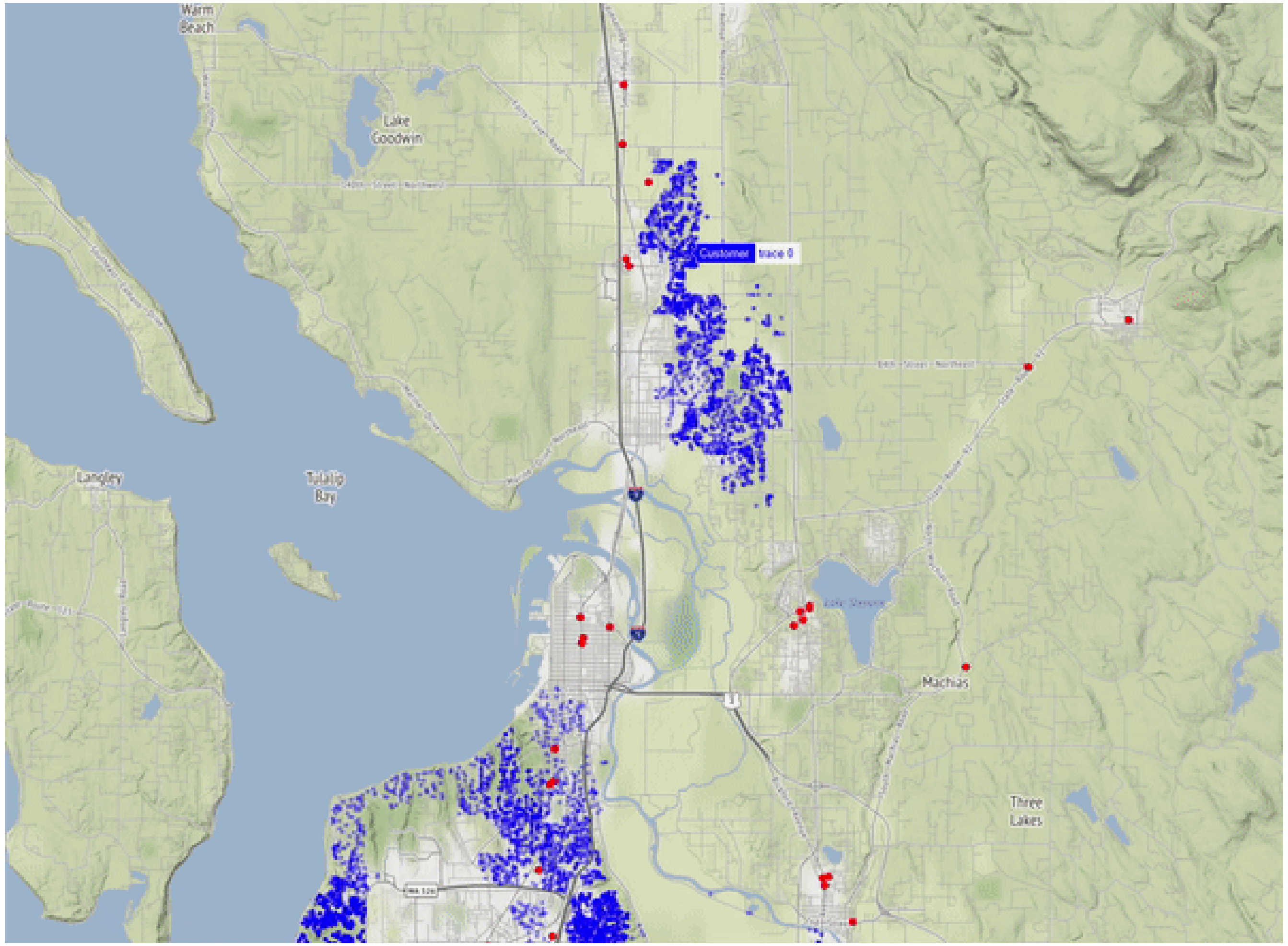
Output y:
Won or lost

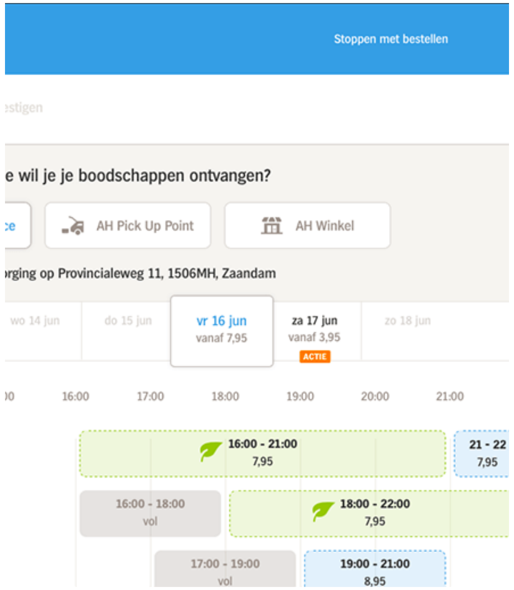
$$y = f(x)$$

Chess: Supervised Learning [2/2]

$f\Bigg($  $\Bigg)$ =[d2−d3]

Real World Problems: E-commerce

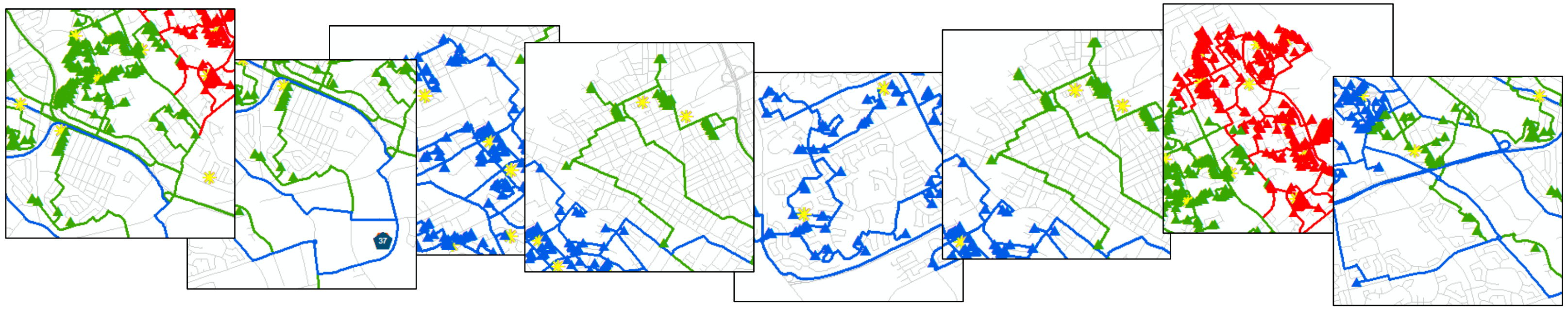




DIGITAL

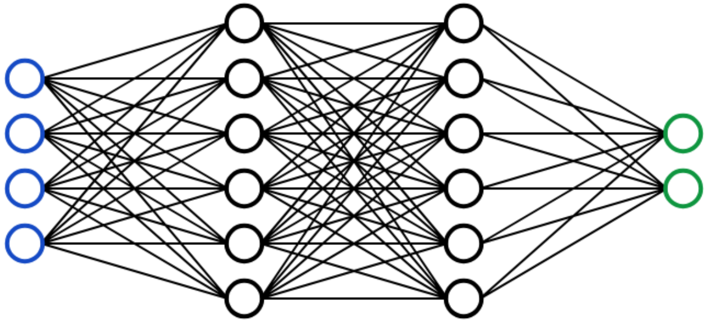
E-Commerce: Supervised Learning [1/2]

Historical optimized routing plans:



Input x:

Sets of customers with their characteristics (location, size, time-window)



$$y = f(x)$$

Output x:

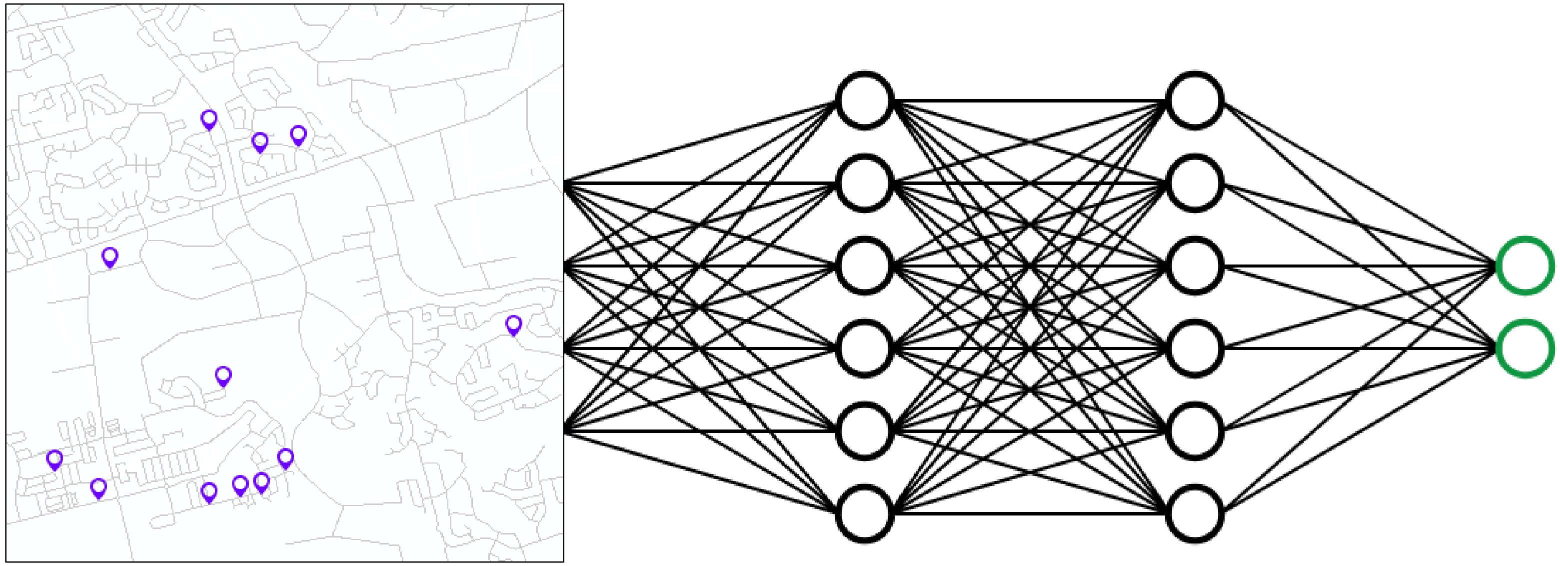
For each set, aspects like total distance, total costs, number of vehicles, costs per customer, lateness, etc.

DIGITAL

E-Commerce: Supervised Learning [2/2]

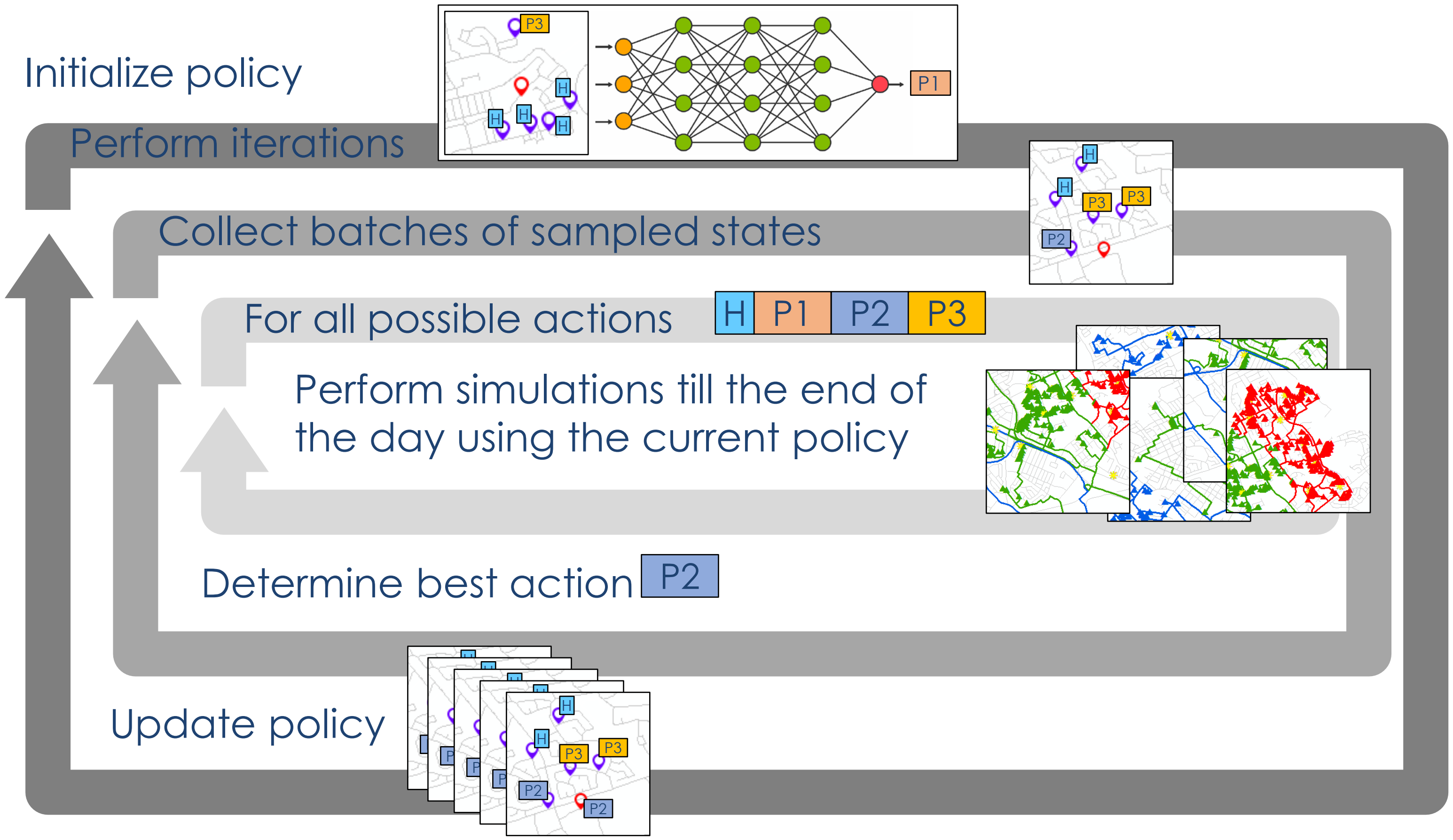
Input to trained neural network: set of specific customers

Expected in yet unknown final routing plan



We can use this NN to decide whether to accept a customer, what time window to offer, or perhaps whether to nudge a customer to be delivered at a parcel locker. Possible weakness of this approach?

Preview Deep Reinforcement Learning

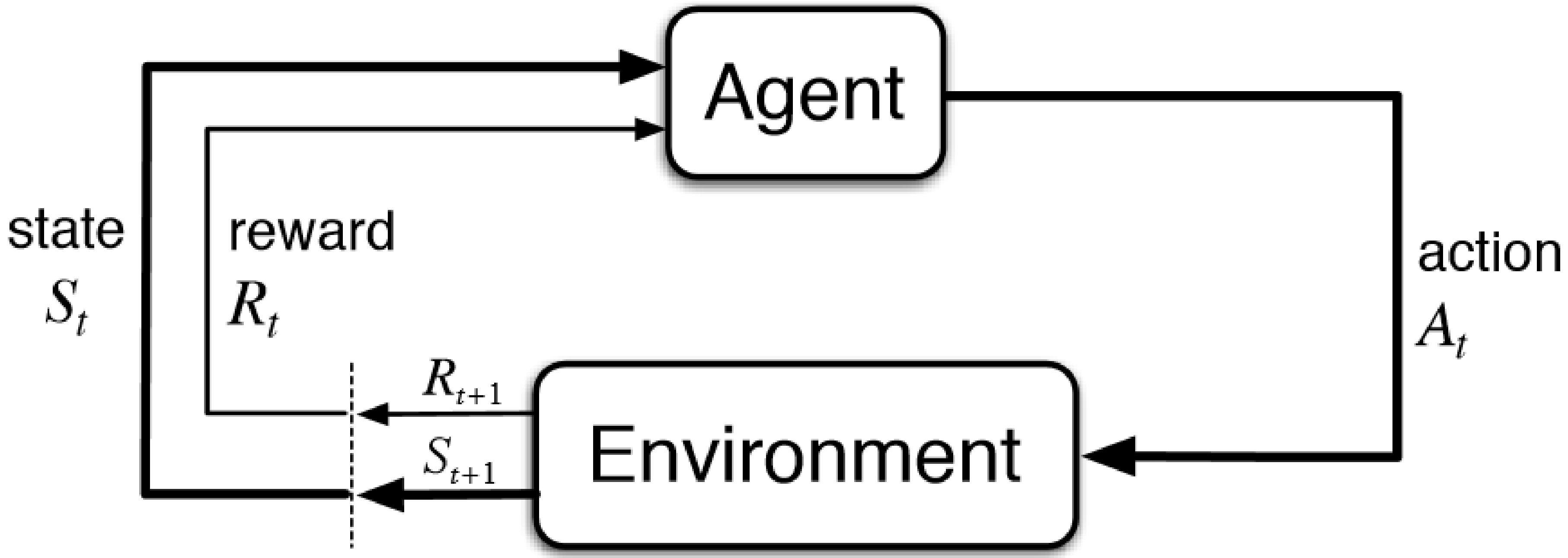


DIGITAL

Introduction Reinforcement Learning

Basic Idea Reinforcement Learning

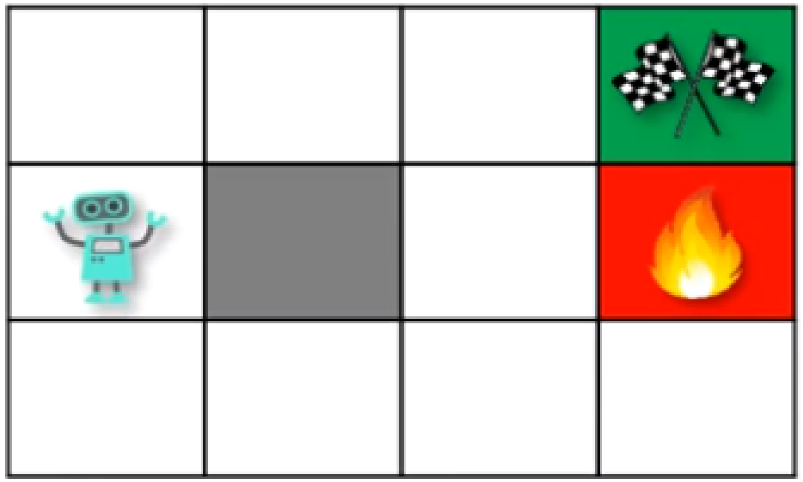
- With RL, we study the interaction with the environment:



DIGITAL

The Idea of Value-Based RL

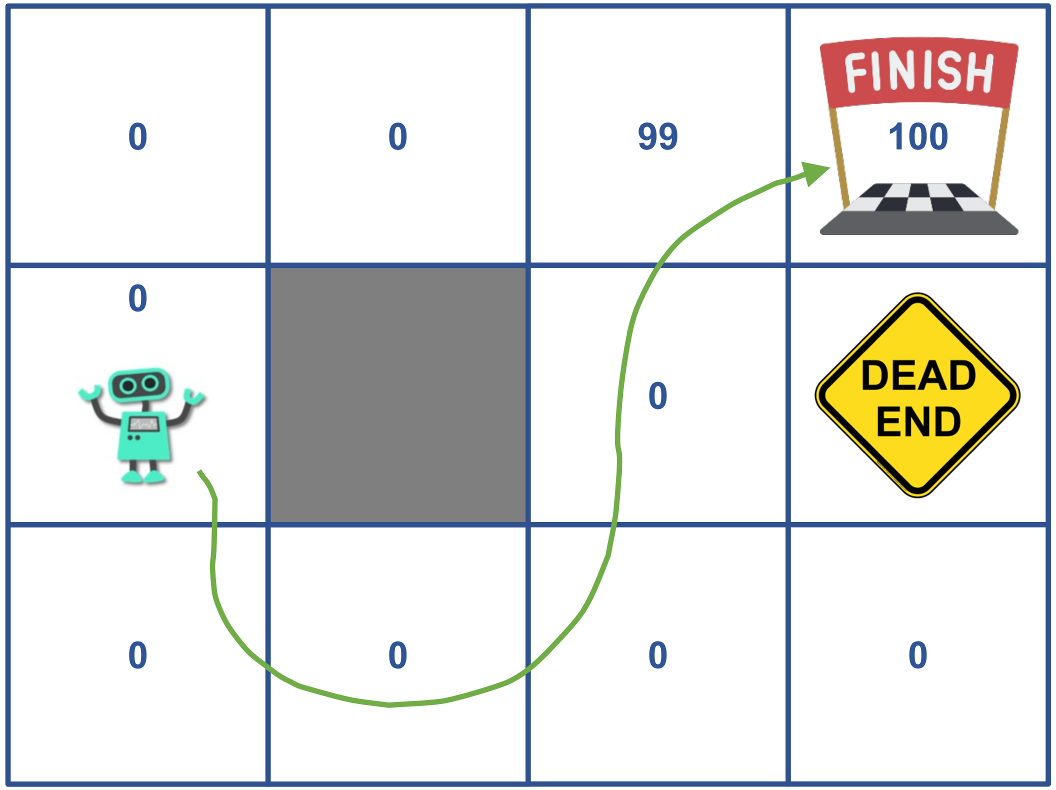
- We have to learn the consequences of our actions (rewards + reachable states):
  - $V(S)$ = (discounted) (in)finite future rewards from state S onwards given an optimal policy
    $Q(S,a)$ = previous + direct reward of a in S



- Ways of learning:
  - Look-ahead one step, take action, update $V(S)/Q(S,a) \rightarrow TD(0)$
  - Play out an entire episode with a "given policy" (probably using Monte Carlo simulation) and propagate values to update the Q-values for observed (S,a) combinations or $V(S)$ for observed $S \rightarrow TD(1)$

Learning the Value of States

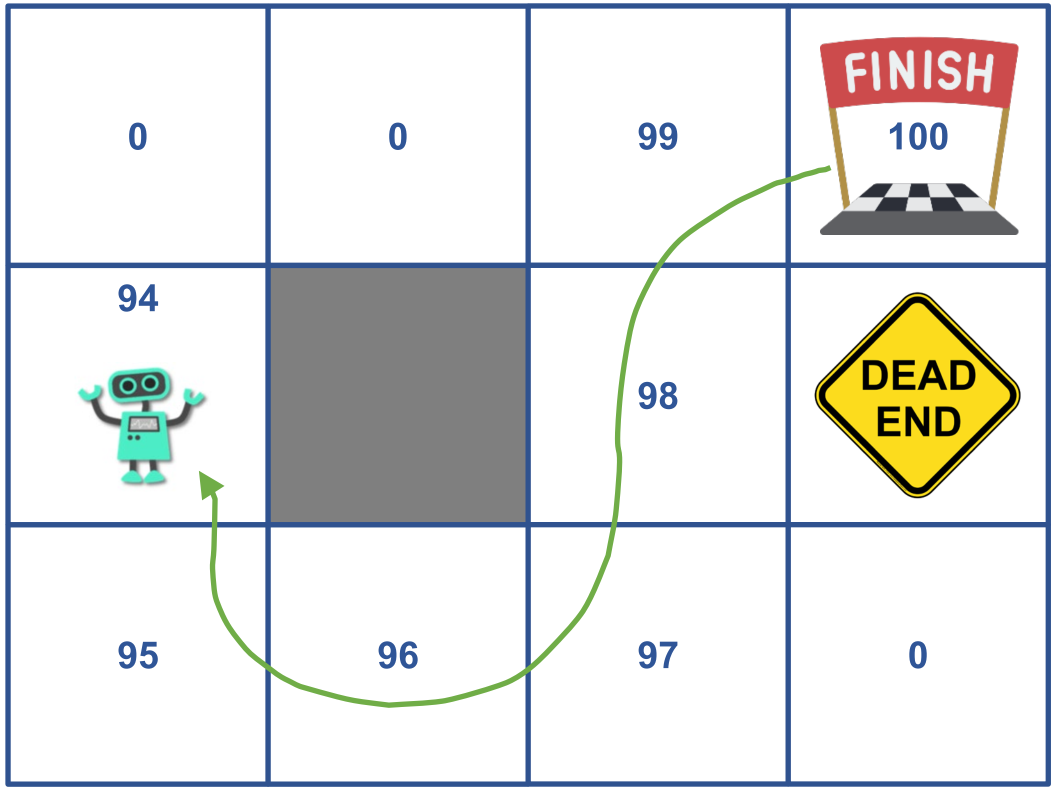
- *V(S)* = value of *S*
- Ways of learning:
  1. Look-ahead one step, take action, update *V(S)*
  2. Play out entire episode and propagate values to update *V(S)* for all encountered *S*



DIGITAL

Learning the Value of States

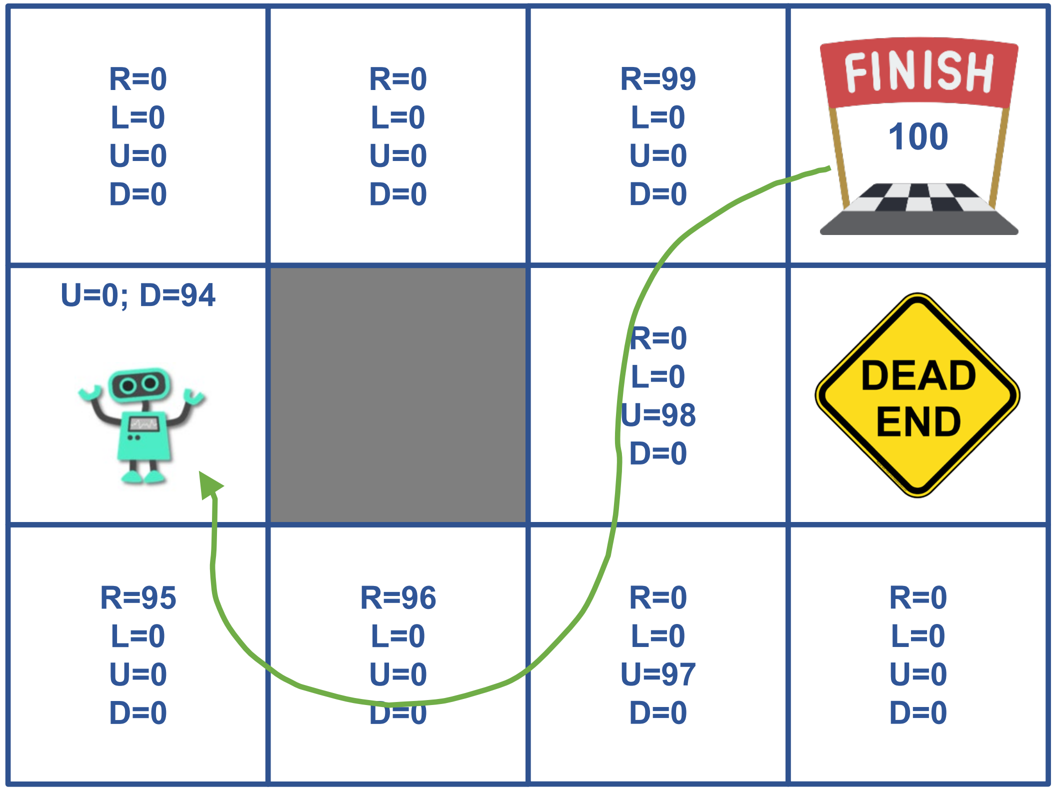
- $V(S)$ = value of S
- Ways of learning:
  1. Look-ahead one step, take action, update $V(S)$
  2. Play out entire episode and propagate values to update $V(S)$ for all encountered S



DIGITAL

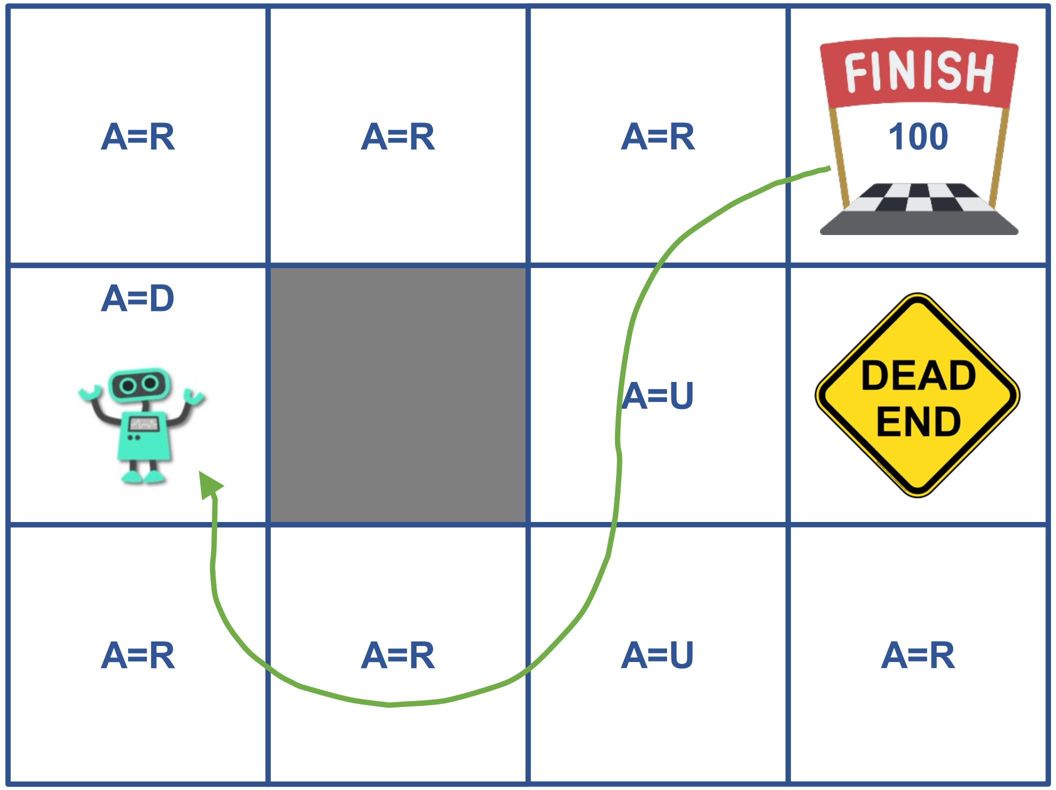
Learning the Value of State-Action Pairs

- $Q(S,a)$ = value of action a in S
- Ways of learning:
  - Look-ahead one step, take action, update $Q(S,a)$
  - Play out entire episode with a "given policy" and propagate values to update $Q(S,a)$ for all encountered (S,a) combinations



Learning the Policy Directly

- $f(S) = a$
- Ways of learning:
  - Evaluate long term impact of all decisions in a state
  - Update policy function approximation $f(S)$



DIGITAL

LEARNING DIMENSIONS

- Model-free or model-based: do we have a model of the world, i.e., of the rewards and transition probabilities?
- Real-world (online) or simulator (offline): can we train offline in a simulator before implementing our decision/policy in the real world?
- Active or passive learning: do we simultaneously need to learn the value functions and the policy (active) or is the policy already given (passive)?
- On-policy or off-policy: do we learn the optimal policy independently on the agent's actions (off) or does the agent learn the value of the policy followed including the exploration steps (on)? The latter constrains our learning process, as we need an exploration strategy that is built into the policy itself.

Markov Decision Processes (MDPs)

MODELING THE DYNAMICS

- MDPs are used to model and solve sequential decision problems under uncertainty
- Characterized by:
- Decision epochs t (not necessarily time)
- States $S_t \in \mathcal{S} = \{1,2,\dots\}$
- Decisions/actions $x_t \in \mathcal{X}_t(S_t)$: follow from a decision function X_t^{π}, where $\pi \in \Pi$ is a family of policies
- Immediate contributions/rewards (payment or costs): $C(S_t, x_t)$
- Exogenous information W_{t+1}: information arriving between t and $t+1$ (transition probabilities)
- Choosing decision x_t in the current state S_t with exogenous information W_{t+1}, results in a transition $S_{t+1} = S^M(S_t, x_t, W_{t+1})$ with contribution $C(S_t, x_t)$

DIGITAL

OBJECTIVE

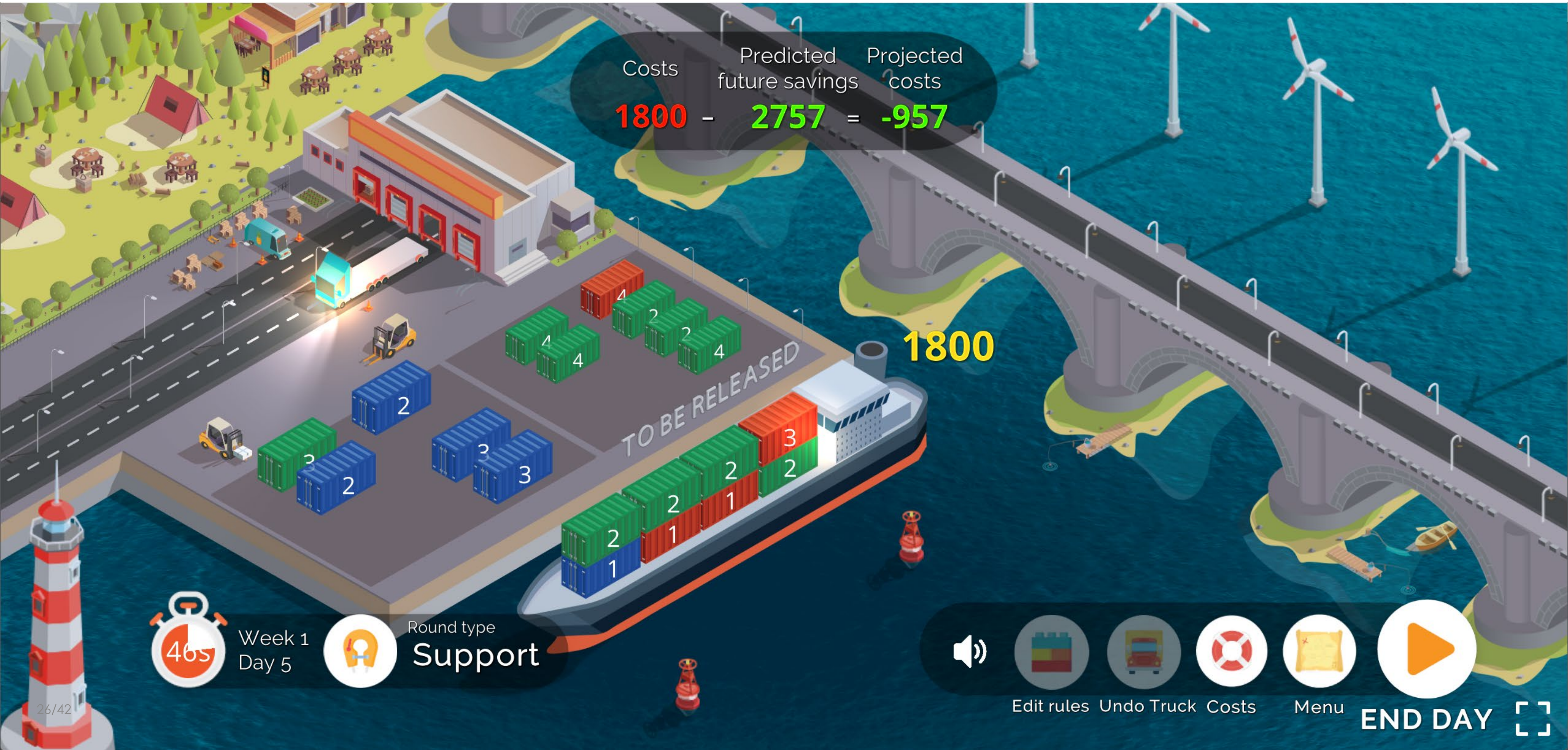
- Objective is to find the policy π that maximizes the expected sum of discounted contributions over all time periods (where T could be infinite):

$$\sup_{\pi \in \Pi} \mathbb{E} \left\{ \sum_{t=0}^{T} \gamma^t C(S_t, X_t^\pi) \middle| S_0 \right\}$$

DIGITAL

TRUCKS & BARGES GAME:

Decision epochs / States / Decisions / Rewards / Transition



MDP FORMULATION TRUCKS & BARGES [1/3]

- Decision epochs: days $t \in \mathcal{T} = \{0, \dots, T\}$
- States: $S_t = [S_{t,d,r,k}]_{\forall [d,r,k] \in \mathcal{D} \times \mathcal{R} \times \mathcal{K}}$
- Decisions:

m = modality
d = destination
r = release day
k = due day

$$x_t = [x_{t,m,d,k}]_{\forall [m,d,k] \in \mathcal{M} \times \mathcal{D} \times \mathcal{K}},$$

s.t.

$$\begin{aligned}
x_{t,m,d,k} &\le S_{t,d,0,k} && \forall [m,d,k] \in \mathcal{M} \times \mathcal{D} \times \mathcal{K}, \\
\sum_{m \in \mathcal{M}} x_{t,m,d,0} &= S_{t,d,0,0} && \forall d \in \mathcal{D}, \\
\sum_{[d,k] \in \mathcal{D} \times \mathcal{K}} x_{t,m,d,k} &\le Q^m && \forall m \in \mathcal{M}, \\
x_{t,m,d,k} &\in \mathbb{N} && \forall [m,d,k] \in \mathcal{M} \times \mathcal{D} \times \mathcal{K}.
\end{aligned}$$

MDP FORMULATION TRUCKS & BARGES [2/3]

- Rewards: $$C(S_t, x_t) = \sum_{m \in \mathcal{M}} \left(\sum_{\mathcal{D}' \subseteq \mathcal{D}} I_{m,\mathcal{D}'} \cdot c^f_{m,\mathcal{D}'} \right) + \sum_{[m,d,k] \in \mathcal{M} \times \mathcal{D} \times \mathcal{K}} c^v_{m,d} \cdot x_{t,m,d,k} + I_t C^T(S_t, x_t),$$

s.t.

$$I_{m,\mathcal{D}'} = \begin{cases} 1 & \text{if} \quad \left(\prod_{d \in \mathcal{D}'} \left(\sum_{k \in \mathcal{K}} x_{t,m,d,k} \right) > 0 \right) \wedge \left(\sum_{[d,k] \in \mathcal{D} \setminus \mathcal{D}' \times \mathcal{K}} x_{t,m,d,k} = 0 \right) \\ 0 & \text{otherwise} \end{cases},$$

$$I_t = \begin{cases} 1 & \text{if} \quad t = T \\ 0 & \text{otherwise} \end{cases}.$$

- Exogenous information: $W_t = [\tilde{S}_{t,d,r,k}]_{\forall [d,r,k] \in \mathcal{D} \times \mathcal{R} \times \mathcal{K}}$

$$\mathbb{P}^\Omega(W_t = \omega_t) = \beta \mathbb{P}^F(f = F) \prod_{[d,r,k] \in \mathcal{D} \times \mathcal{R} \times \mathcal{K}} \left(\mathbb{P}^J([d,r,k] = [D,R,K]) \right)^{\tilde{S}_{d,r,k}},$$

s.t.

$$f = \sum_{[d,r,k] \in \mathcal{D} \times \mathcal{R} \times \mathcal{K}} \tilde{S}_{d,r,k},$$

$$\beta = \frac{s!}{\prod_{[d,r,k] \in \mathcal{D} \times \mathcal{R} \times \mathcal{K}} \tilde{S}_{d,r,k}!}.$$

DIGITAL

MDP FORMULATION TRUCKS & BARGES [3/3]

- Transition: $S_t = S^M(S_{t-1}, x_{t-1}, W_t)$,

$$
\begin{aligned}
&\text{s.t.}\\
S_{t,d,0,k} &= S_{t-1,d,0,k+1} - \sum_{m\in\mathcal{M}} x_{t-1,m,d,k+1} + S_{t-1,d,1,k} + \tilde{S}_{t,d,0,k} \\
&\qquad \forall [d,k] \in \mathcal{D}\times\mathcal{K}\setminus\{K^{max}\} \ , \\
S_{t,d,0,K^{max}} &= S_{t-1,d,1,K^{max}} + \tilde{S}_{t,d,0,K^{max}} \\
&\qquad \forall d \in \mathcal{D} \ , \\
S_{t,d,r,k} &= S_{t-1,d,r+1,k} + \tilde{S}_{t,d,r,k} \\
&\qquad \forall [d,r,k] \in \mathcal{D}\times\mathcal{R}\setminus\{0,R^{max}\}\times\mathcal{K} \ , \\
S_{t,d,R^{max},k} &= \tilde{S}_{t,d,R^{max},k} \\
&\qquad \forall [d,k] \in \mathcal{D}\times\mathcal{K} \ .
\end{aligned}
$$

- Bellman optimality equation:

$$V_t(S_t) = \min_{x_t\in\mathcal{X}(S_t)} \left(C(S_t,x_t) + \sum_{\omega\in\Omega} \mathbb{P}^{\Omega}(W_{t+1}=\omega) V_{t+1}\left(S^M(S_t,x_t,\omega)\right)\right) \forall S_t \in \mathcal{S}_t$$

SOLVING AN MDP

- Value iteration → Iterations of 1-step value function updates (or backwards induction for finite horizon problems)
- Policy iteration → Evaluate value function for given policy and greedily update policy
- Linear programming → Solve the Bellman optimality equation as used in value iteration for all decision epochs at once

DIGITAL

THREE CURSUS OF DIMENSIONALITY

- Three possible cursus of dimensionality:
  - State space S_t might be too large to evaluate $V_t(S_t)$ for all states:
  - Decision space $\mathcal{X}_t(S_t)$ might be too large to evaluate the impact of every decision.
  - Outcome space (possible states for the next time period) might be too large to compute the expectation of cost-to-go.
- Value iteration might become intractable
- Similar arguments apply to the other two methods (policy iteration and linear programming)

⇨ Solve approximately…

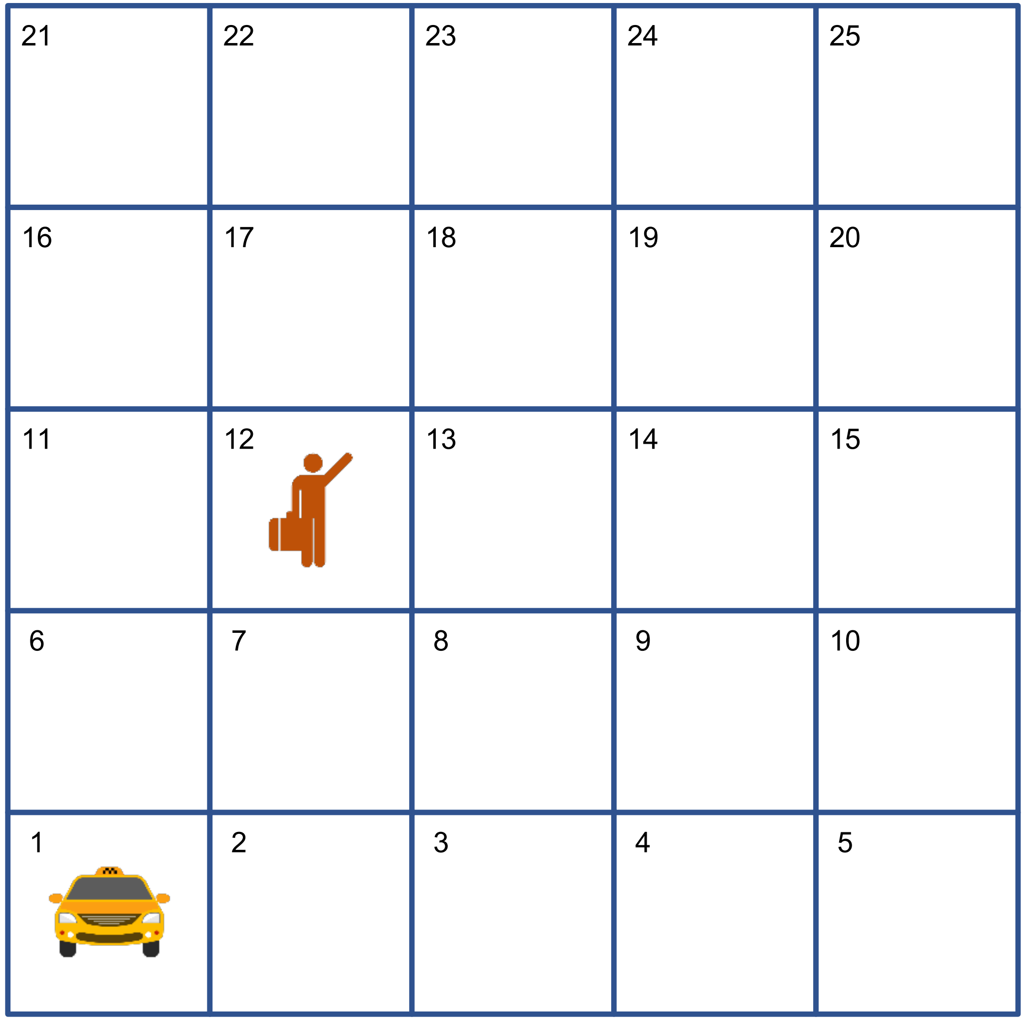
SOLVE MDPs APPROXIMATELY

- Reinforcement Learning: class of methods to solve MDPs approximately… (and more)
- Solve approximately:
  - Approximate value iteration $\rightarrow$ Typical approach used in Approximate Dynamic Programming (ADP)
  - Approximate policy iteration $\rightarrow$ The basis of, e.g., AlphaZero
  - Linear programming based ADP

DIGITAL

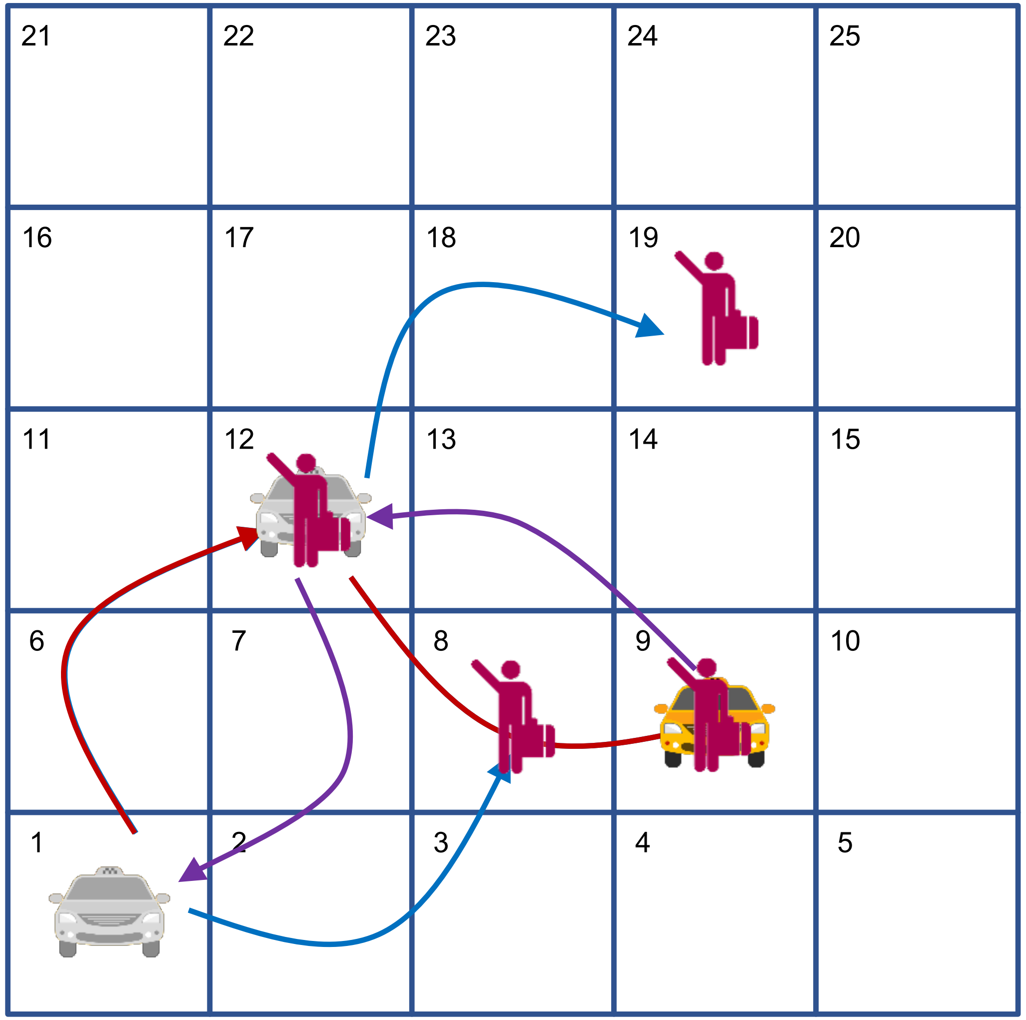
Preview of different forms of RL…

ILLUSTRATION RL OPTIONS



- Nomadic trucker problem
- 1 taxi
- Imbalanced network
- Objective: maximize infinite horizon discounted rewards
- Learn which trips to accept and when/where to move empty to
- Here simplified to just visiting a customer

ILLUSTRATION RL OPTIONS



- Approximate Value iteration
- 1-step lookahead & update
- ADP, SARSA, TD(0)
- Post-decision state

$$\tilde{a} = \arg \max_{a \in \{8,12\}} \{R(1,a) + \bar{V}^n(a)\}$$

$$\bar{V}^{n+1}(1) \leftarrow R(1,12) + \bar{V}^n(12)$$

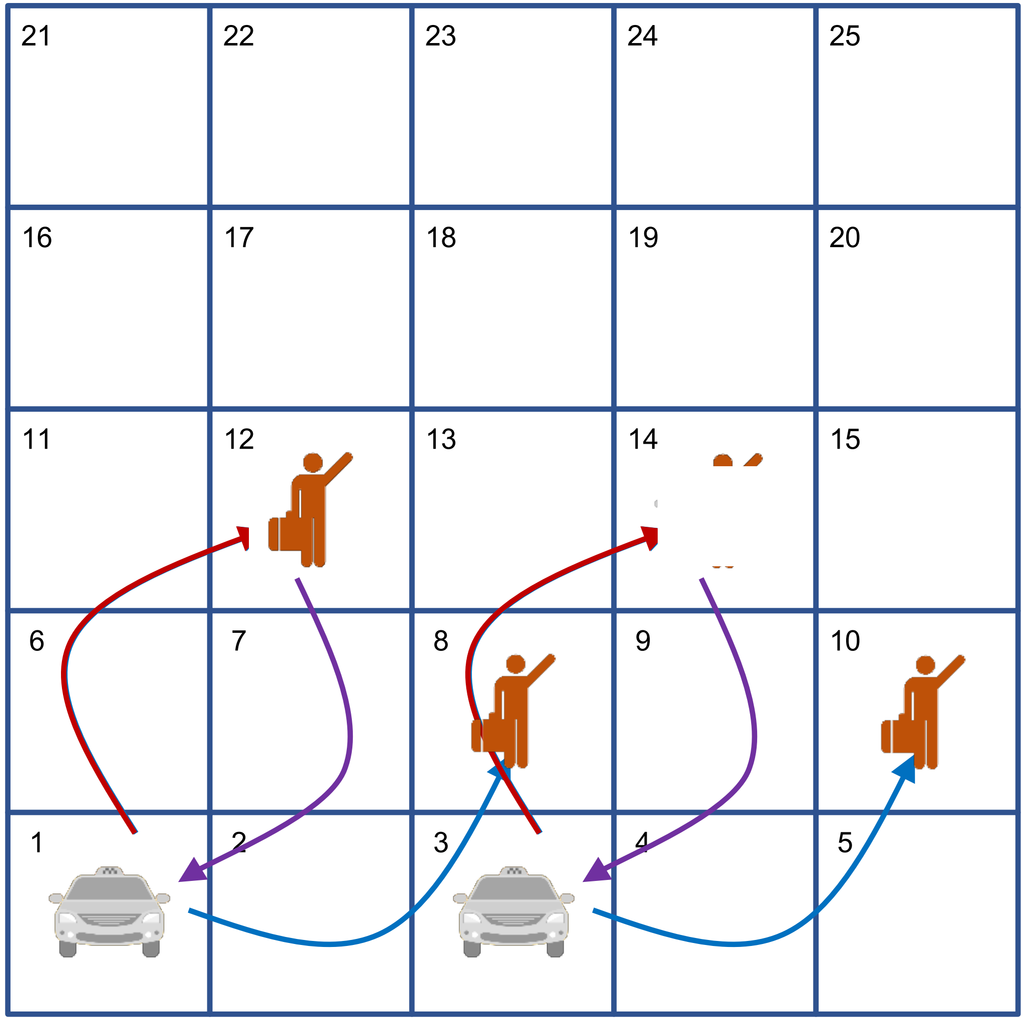
- Alternatively (Q-learning)

$$\tilde{a} = \arg \max_{a \in \{8,12\}} \{\bar{Q}^n((1,8,12), a)\}$$

$$\bar{Q}^n((1,8,12), 12)) \leftarrow R(1,12) + \arg \max_{a \in \{?,?\}} \{\bar{Q}^n((12, ?, ?), a)\}$$

DIGITAL

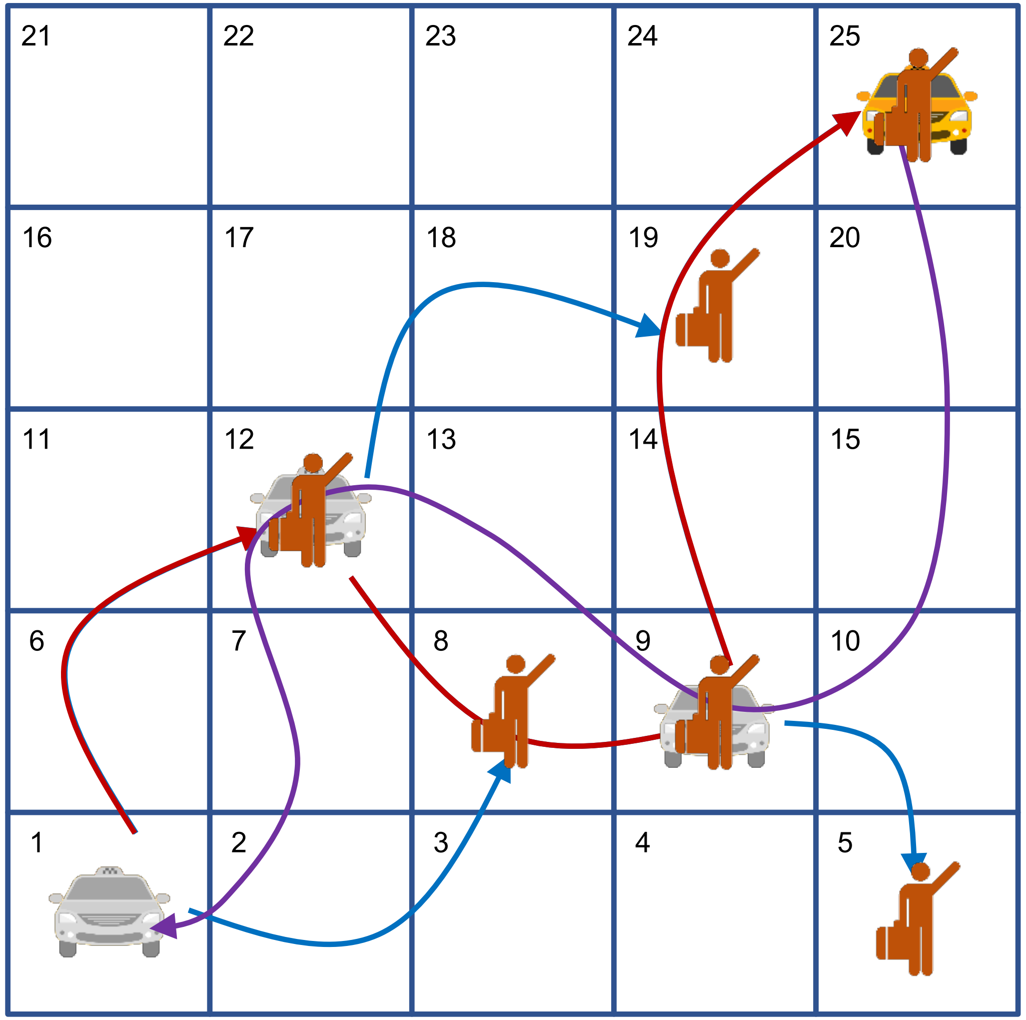
ILLUSTRATION RL OPTIONS



- Approximate Value iteration
- 1-step lookahead & update
- Explore, off-policy learning

DIGITAL

ILLUSTRATION RL OPTIONS



- Approximate Value iteration
- n-step lookahead & update
- n-SARSA, MC learning, TD(1)

$$a_1^* = \text{arg} \max_{a \in \{8,12\}} \{R(1,a) + \bar{V}^n(a)\}$$

$$a_2^* = \text{arg} \max_{a \in \{9,19\}} \{R(12,a) + \bar{V}^n(a)\}$$

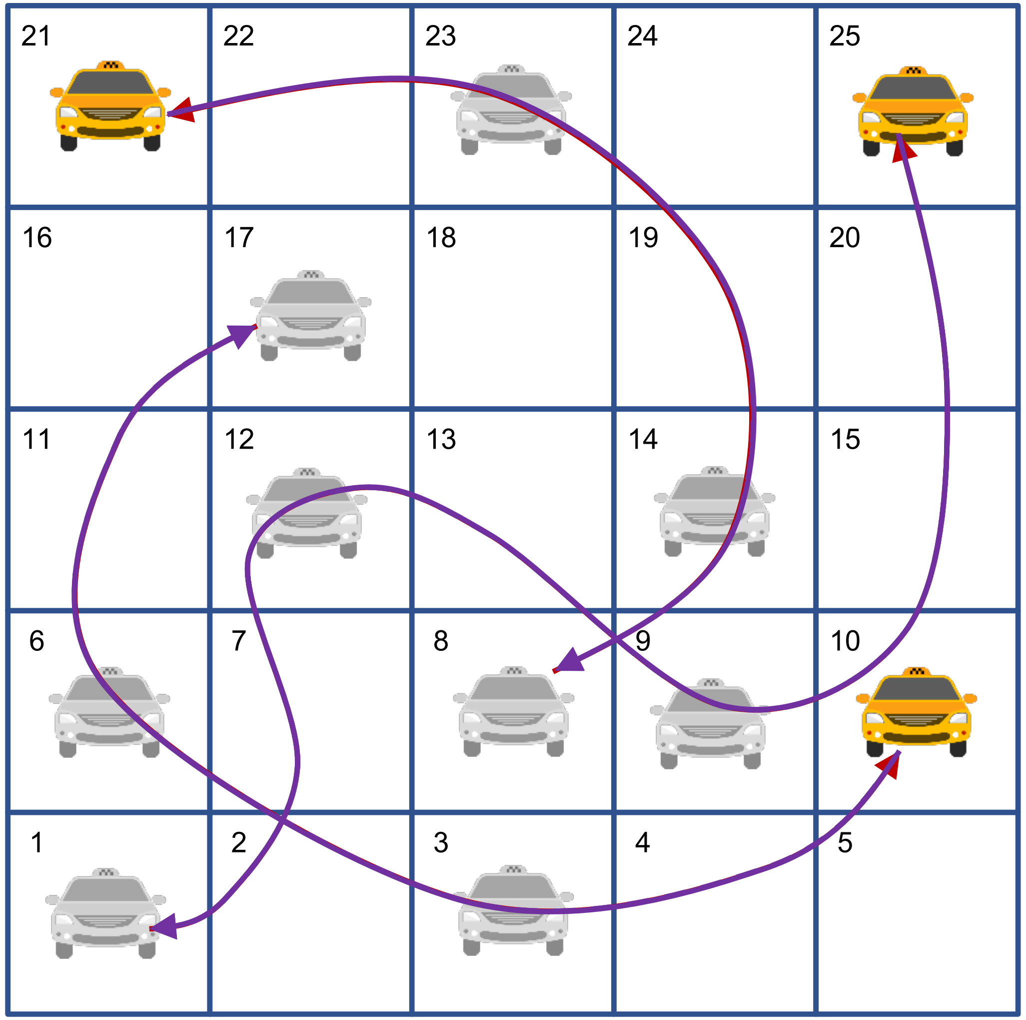
$$a_3^* = \text{arg} \max_{a \in \{5,25\}} \{R(9,a) + \bar{V}^n(a)\}$$

$$\bar{V}^{n+1}(1) \leftarrow R(1,12) + R(12,9) + R(9,25) + \bar{V}^n(25)$$

- Possibly use replications…

DIGITAL

ILLUSTRATION RL OPTIONS



- Approximate Value iteration
- n-step lookahead with batch updating

$$\bar{V}^{n+1}(1) \leftarrow R(1,12) + R(12,9) + R(9,25) + \bar{V}^{n}(25)$$

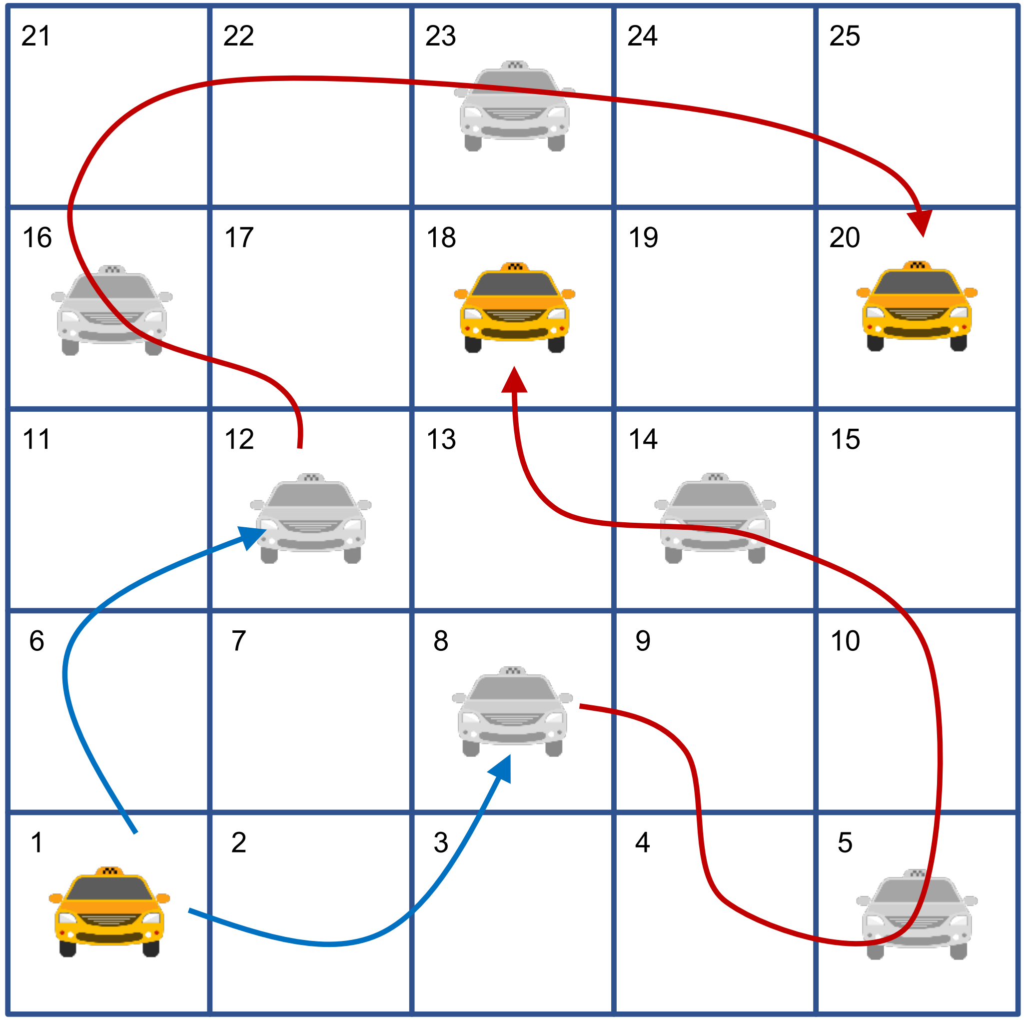
$$\bar{V}^{n+1}(8) \leftarrow R(8,14) + R(14,23) + R(23,21) + \bar{V}^{n}(21)$$

$$\bar{V}^{n+1}(17) \leftarrow R(17,6) + R(6,3) + R(3,10) + \bar{V}^{n}(10)$$

- Ideal for NN-based VFA
- Or Deep Q-Learning

DIGITAL

ILLUSTRATION RL OPTIONS

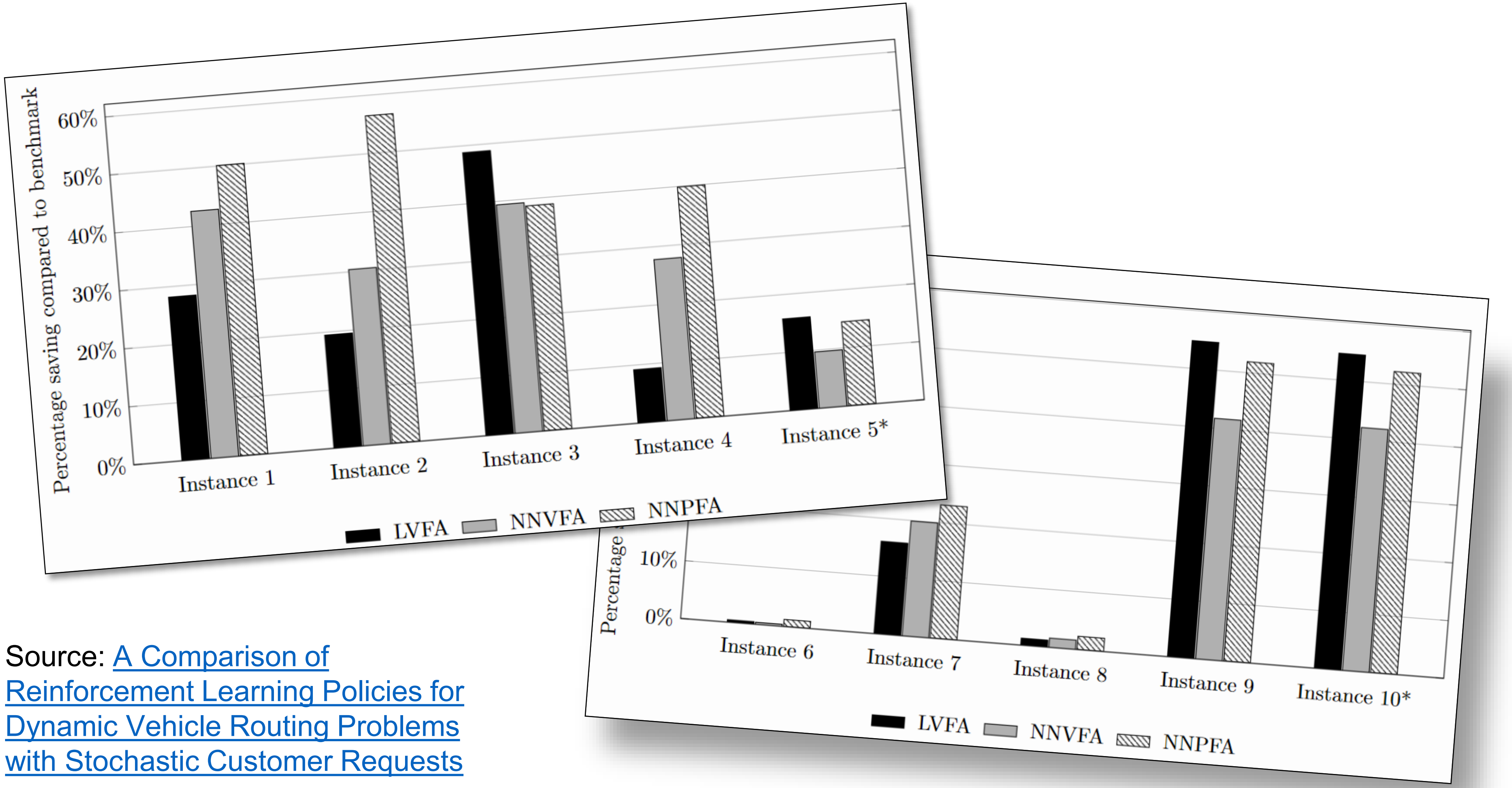


- Approximate Policy iteration
- n-step lookahead (rollout) with batch updating

1. Sample a state
2. Evaluate all possible actions
3. After the action, run multiple long simulations following the current policy
4. Evaluate best action
5. Repeat the above for many states
6. Update NN mapping states to actions

DIGITAL

NOT CLEAR UP FRONT WHAT FORM IS BEST



Source: [A Comparison of Reinforcement Learning Policies for Dynamic Vehicle Routing Problems with Stochastic Customer Requests]()

DIGITAL

TRUCKS & BARGES GAME:
FEATURES OF THE REGRESSION MODEL OF THE RL AGENT

State variable [R1]
State variable [R2]
State variable [R3]
State variable [R1-nr]
State variable [R2-nr]
State variable [R3-nr]
State variable [G1]
State variable [G2]
State variable [G3]
State variable [G1-nr]
State variable [G2-nr]
State variable [G3-nr]
State variable [B1]
State variable [B2]
State variable [B3]
State variable [B1-nr]
State variable [B2-nr]
State variable [B3-nr]
Number destinations MUST
Sum freights MUST
Number destinations MAY
Sum freights MAY
Number destinations FUTURE
Sum freights FUTURE
Sum All Freights
Constant
Destination Must Future [No]
Destination Must Future [B]
Destination Must Future [G]
Destination Must Future [R]
Destination Must Future [GB]
Destination Must Future [RB]
Destination Must Future [RG]
Destination Must Future [RGB]
Destination May Future [No]
Destination May Future [B]
Destination May Future [G]
Destination May Future [R]
Destination May Future [GB]
Destination May Future [RB]
Destination May Future [RG]
Destination May Future [RGB]

DISCUSSION

- RL is the method of choice for sequential decision problems where current decisions have a long-term, uncertain effect.
- Hence, RL is not the method of choice when:
  - Not dealing with uncertainty
  - Not dealing with a decision-problem
  - One-shot decision-making (no need for a policy)
  - Decisions don’t affect the environment
  - Decisions don’t have a longer-term impact
- In these cases, use (un)supervised learning, classical optimization approaches, heuristics, or a predict-then-optimize method.

DIGITAL

Funded by
the European Union

Funded by the European Union. Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union or Horizon Europe: Marie Skłodowska-Curie Actions. Neither the European Union nor the granting authority can be held responsible for them.

This project has received funding from the European Union's Horizon Europe research and innovation programme under the Marie Skłodowska-Curie grant agreement No. 101119635

DIGITAL