Deep **Multi-Objective** RL

& its application in task-oriented dialogue systems

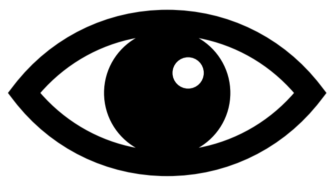
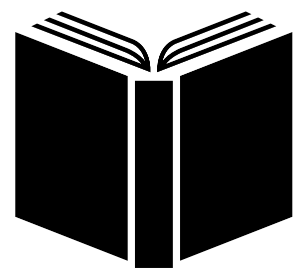
Runzhe Yang @ SJTU ACM Class

Advisor: Prof. Kai Yu

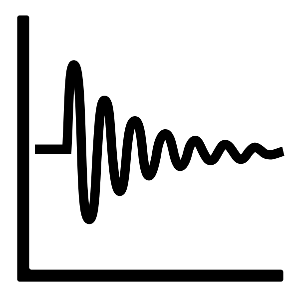
2018.06.24

https://runzhe-yang.science

“Artificial Intelligence”


Perception **Cognition** **Decision**


Intro - Motivating Scenarios

scalar reward, e.g.
max(min{speed, safety, …})

Autonomous driving as an optimization problem

Intro - Motivating Scenarios

many passengers

one pedestrian

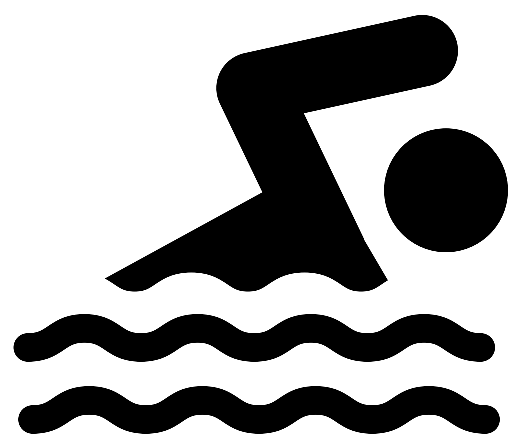
Who lives and who dies?

The autonomous car must decide between

option 1: killing one pedestrian

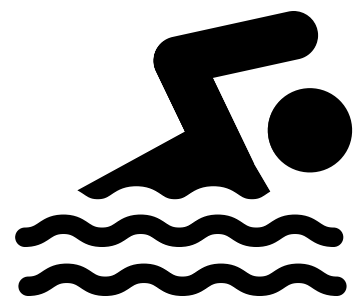
option 2: killing its own passengers

Intro - Motivating Scenarios

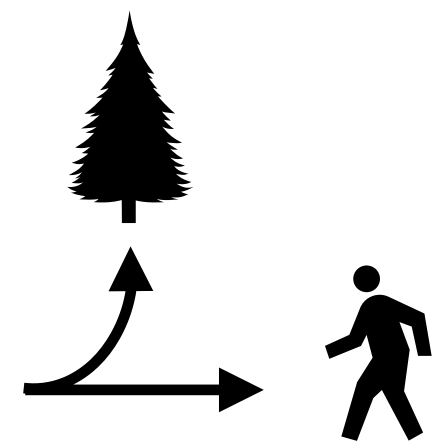


Think about how we learned to swim.
Many Objectives — speed, stability, endurance,
energy efficiency, the beauty of style…
Our coaches never teach us
relative importance weights for them.
But we swim well.

Intro - Research Questions



Multiple Competing Objectives



Human Preferences

Can we design an efficient learning algorithm, which learns **all potentially optimal policies**, and adapts optimally to any real-time specified **preference**?

Intro - Contributions & Outlines

0. Background

- Reinforcement Learning
- Problem Formulation
  - MO-MDPs
  - Optimality Concepts
  - Delayed Linear Preference Scenarios

1. Theory Contributions

- Theoretical Framework for Value-Based RL
- Two Value-Based Deep MORL Algorithms
  - Naive Version: A simple extension
  - Envelope Version

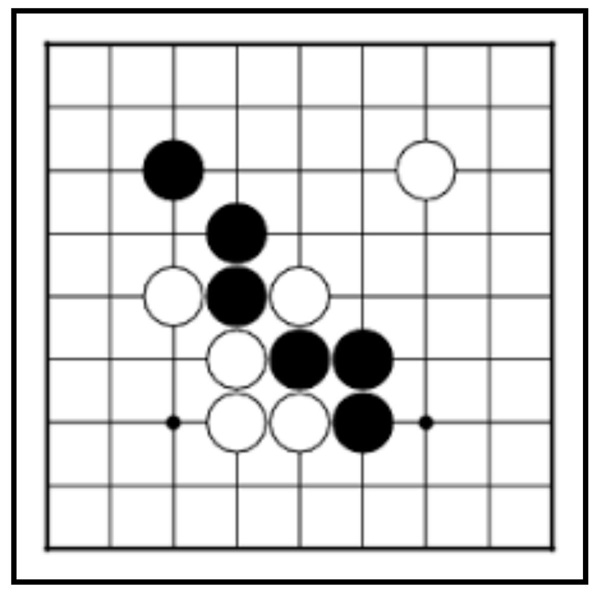
2. Evaluation Contributions

- Quantitive Evaluation Metrics
  - Coverage Ratio
  - Adaptation Quality
- Synthetic Environments

3. Application Contributions

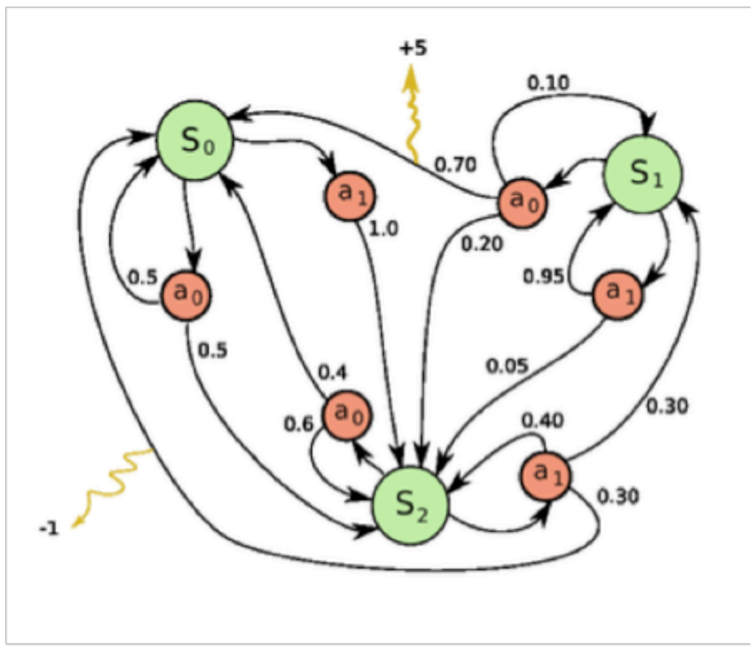
- Task-Oriented Dialogue Systems
- RL-Based Dialogue Policy Learning
  - Objectives: Brevity v.s. Success
  - User Adaptive Policies

Background - Reinforcement Learning

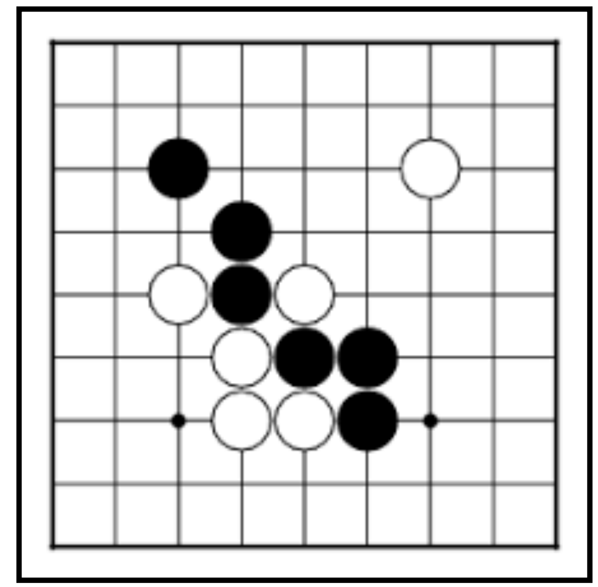


Playing Chess

Background - Reinforcement Learning

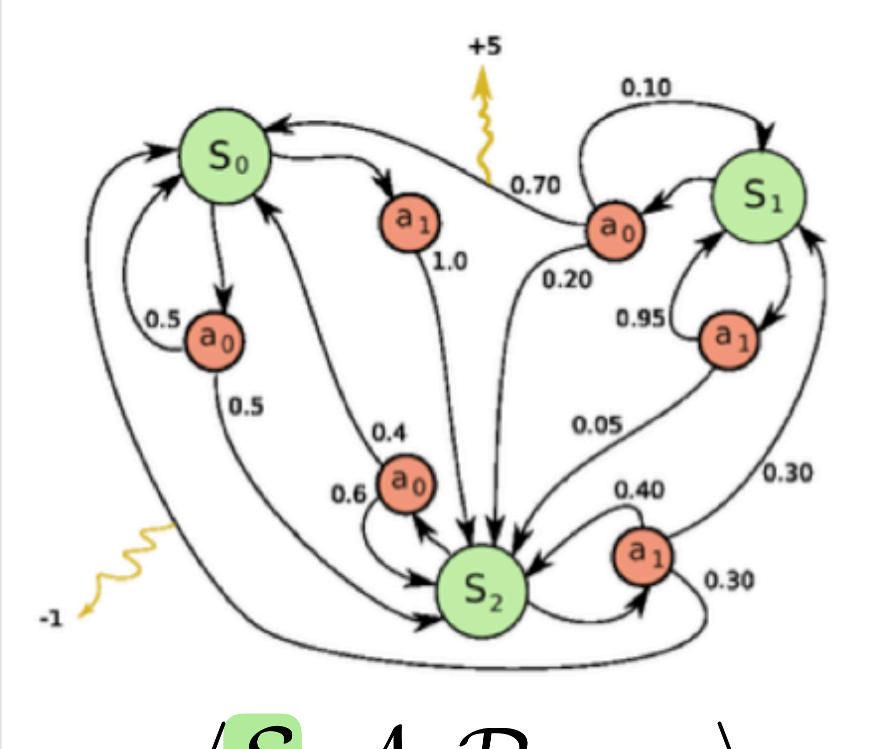


Markov Decision Process (MDP)



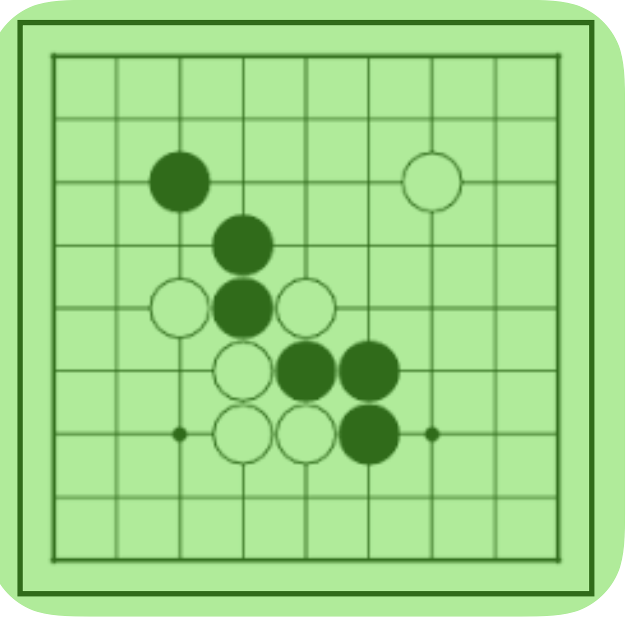
Playing Chess

Background - Reinforcement Learning



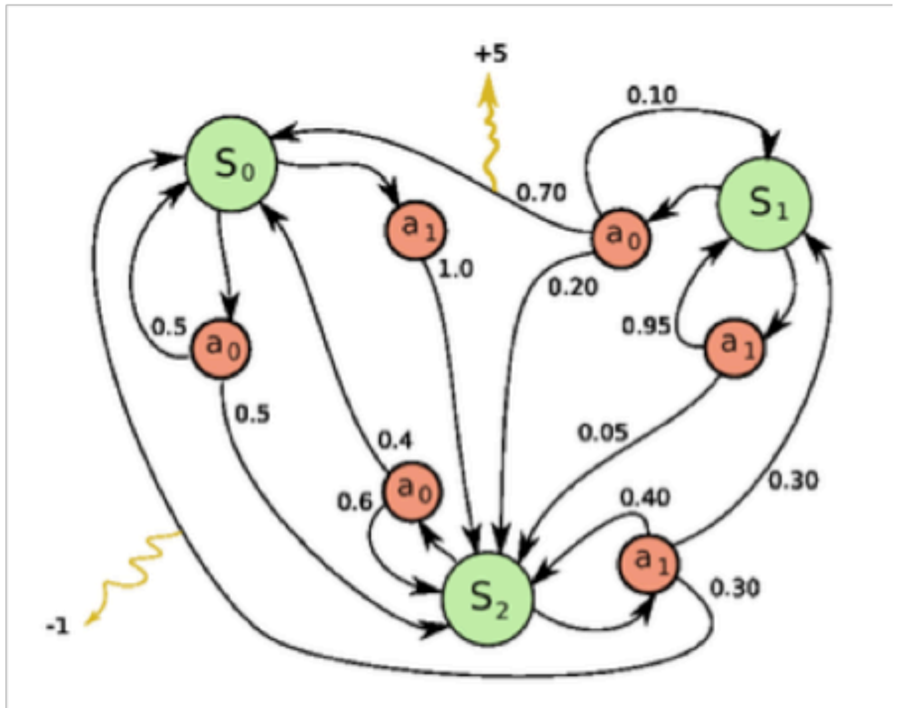
$$\langle \mathcal{S}, \mathcal{A}, \mathcal{P}, r, \gamma \rangle$$

State Space



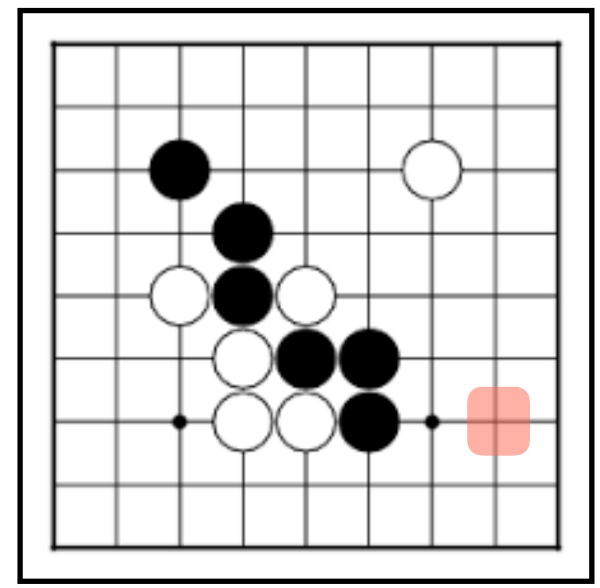
Playing Chess

Background - Reinforcement Learning



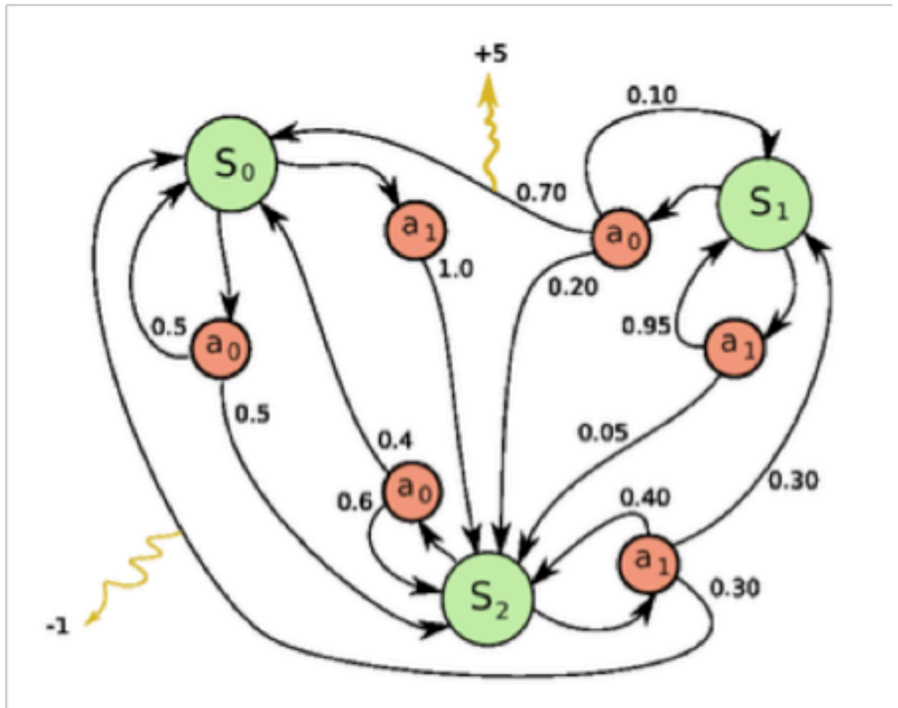
$$\langle \mathcal{S}, \mathcal{A}, \mathcal{P}, r, \gamma \rangle$$

State Space   Action Space



Playing Chess

Background - Reinforcement Learning

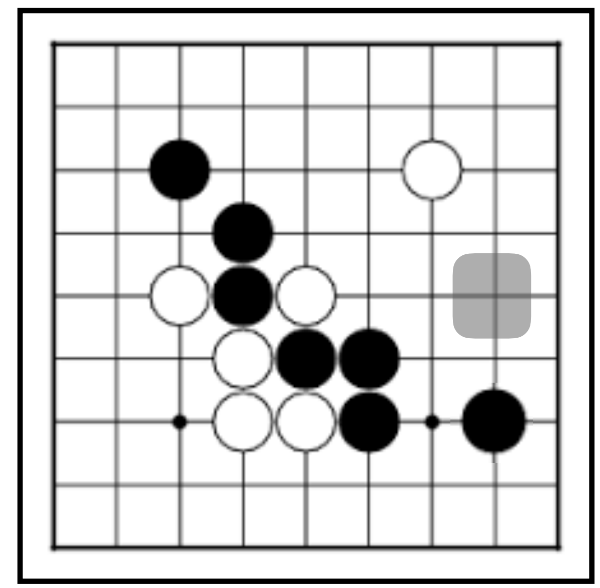


$$\langle \mathcal{S}, \mathcal{A}, \mathcal{P}, r, \gamma \rangle$$

State Space   Action Space

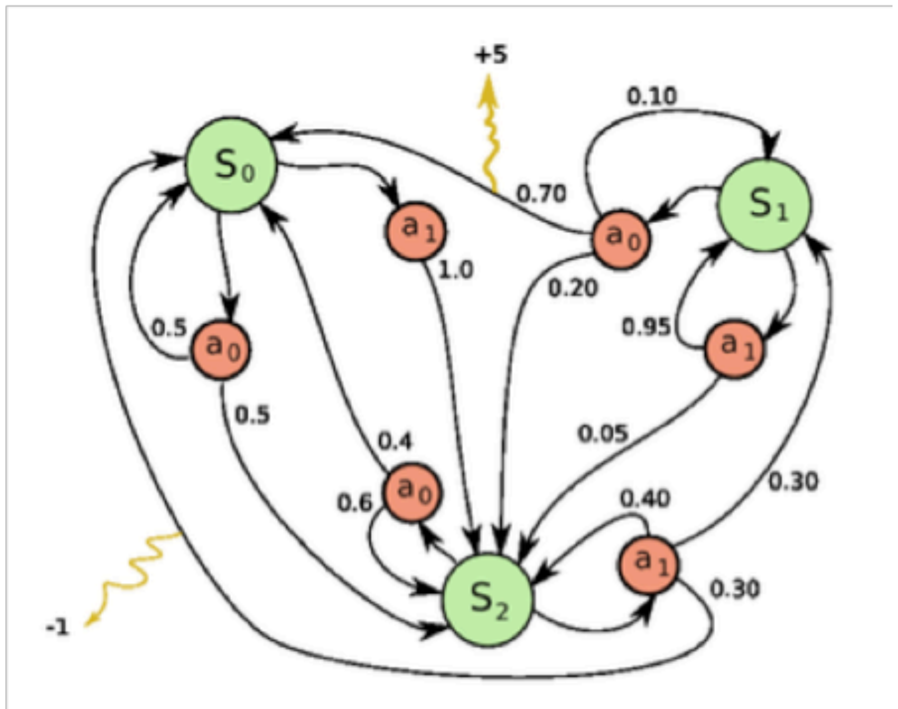
Stochastic Transition Kernel $\mathcal{P}(s'|s,a)$

e.g. $\mathcal{P}(S_0|S_1,a_0) = 0.7$



Playing Chess

Background - Reinforcement Learning



$\langle \mathcal{S}, \mathcal{A}, \mathcal{P}, r, \gamma \rangle$

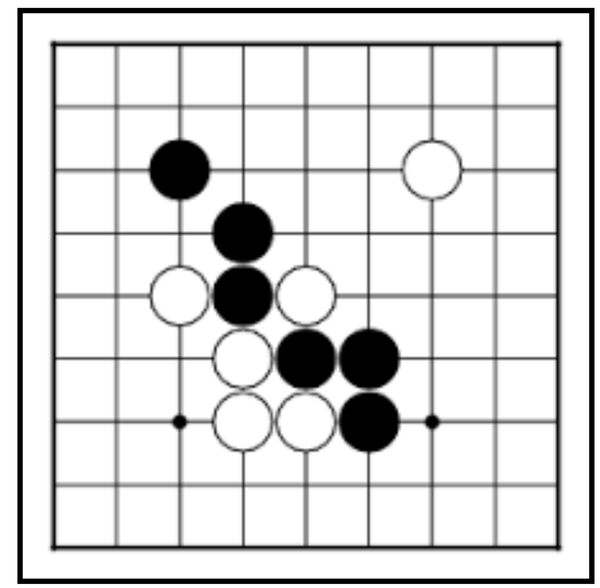
State Space   Action Space

Stochastic Transition Kernel $\mathcal{P}(s'|s, a)$

e.g. $\mathcal{P}(S_0|S_1, a_0) = 0.7$

Reward Function $r : \mathcal{S} \times \mathcal{A} \rightarrow \mathbb{R}$

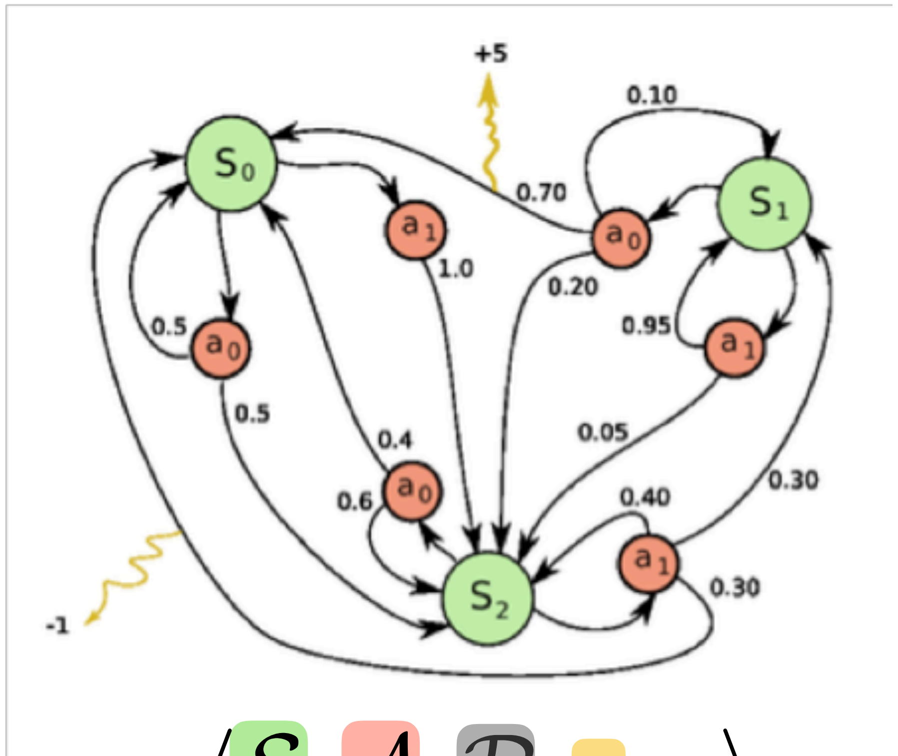
e.g. $r(S_1, a_0) = 3.5$



Playing Chess

Win - Lose

Background - Reinforcement Learning



$\langle \mathcal{S}, \mathcal{A}, \mathcal{P}, r, \gamma \rangle$

State Space Action Space

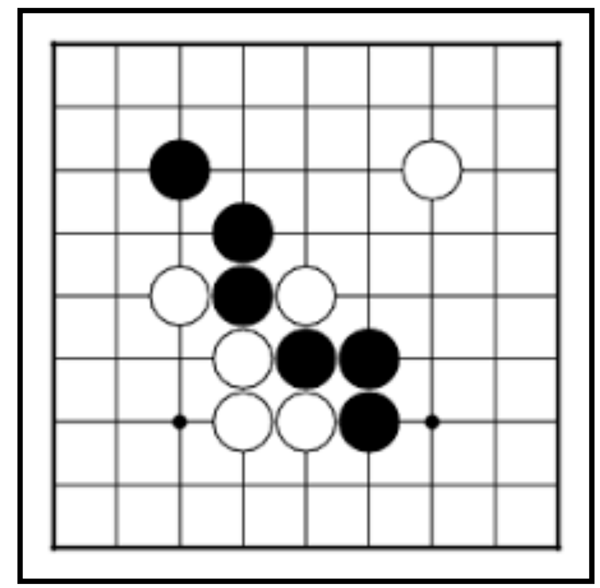
Stochastic Transition Kernel $\mathcal{P}(s'|s, a)$

e.g. $\mathcal{P}(S_0|S_1, a_0) = 0.7$

Reward Function $r : \mathcal{S} \times \mathcal{A} \rightarrow \mathbb{R}$

e.g. $r(S_1, a_0) = 3.5$

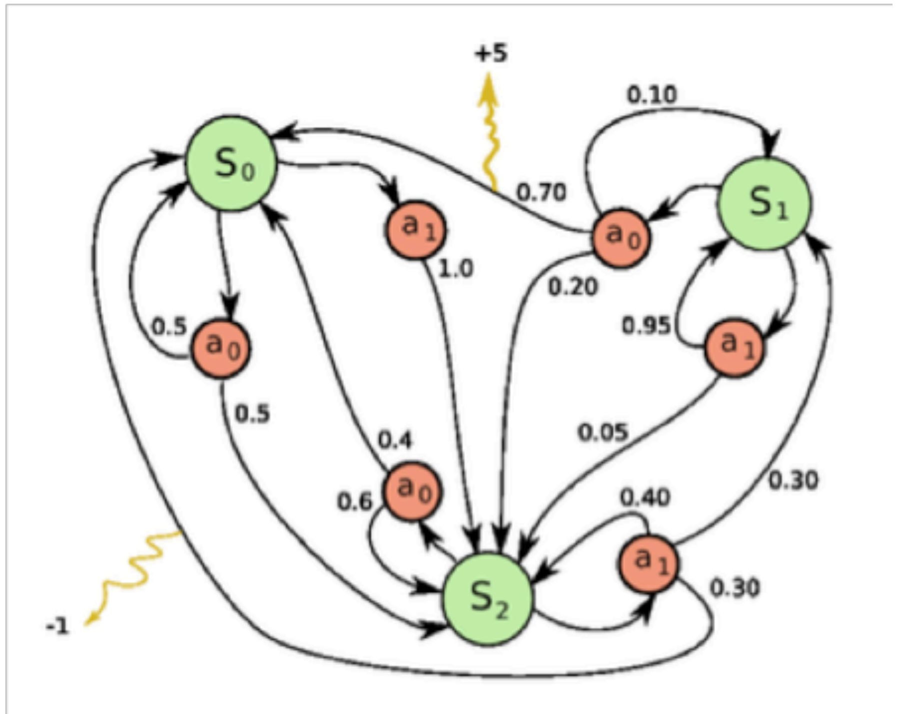
$\gamma \in [0, 1)$ is a discount factor



Playing Chess

Win - Lose

Background - Reinforcement Learning



$\langle \mathcal{S}, \mathcal{A}, \mathcal{P}, r, \gamma \rangle$

State Space

Action Space

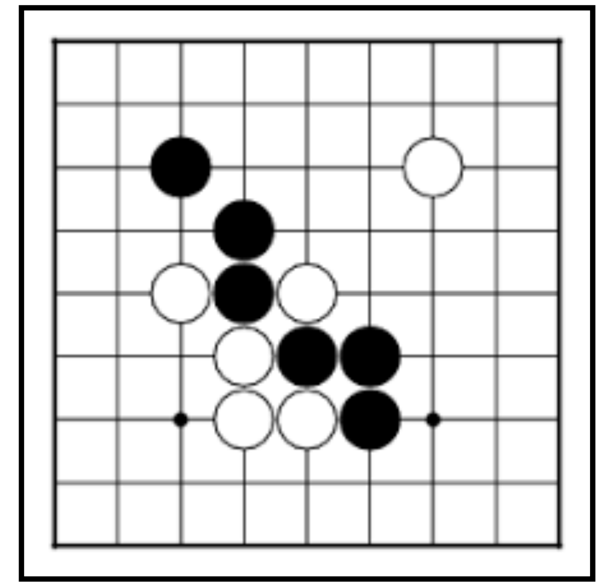
Stochastic Transition Kernel $\mathcal{P}(s'|s,a)$

e.g. $\mathcal{P}(S_0|S_1,a_0) = 0.7$

Reward Function $r : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$

e.g. $r(S_1,a_0) = 3.5$

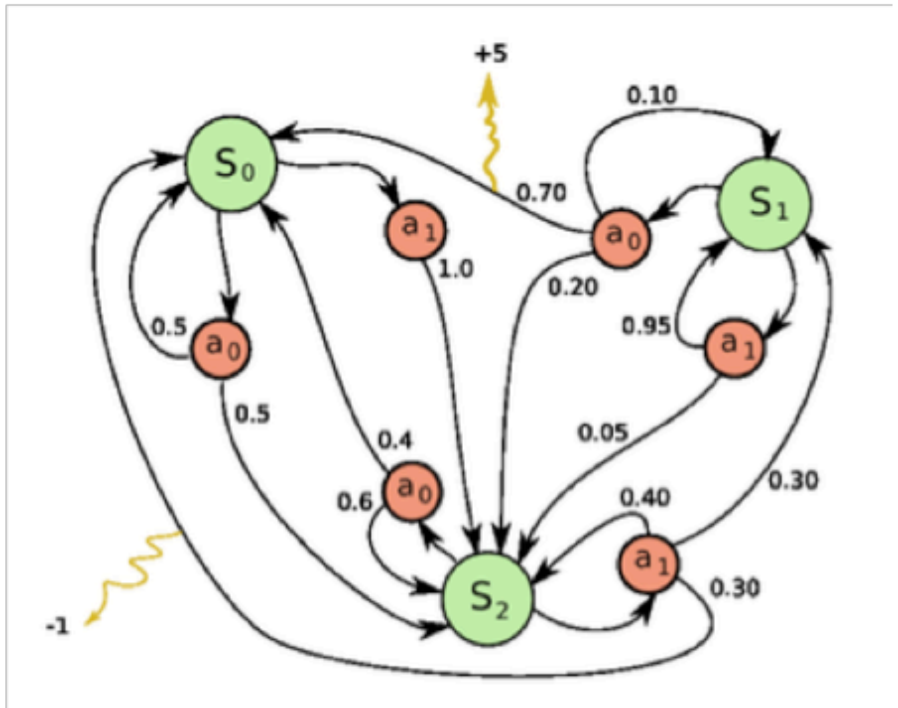
$\gamma \in [0,1)$ is a discount factor



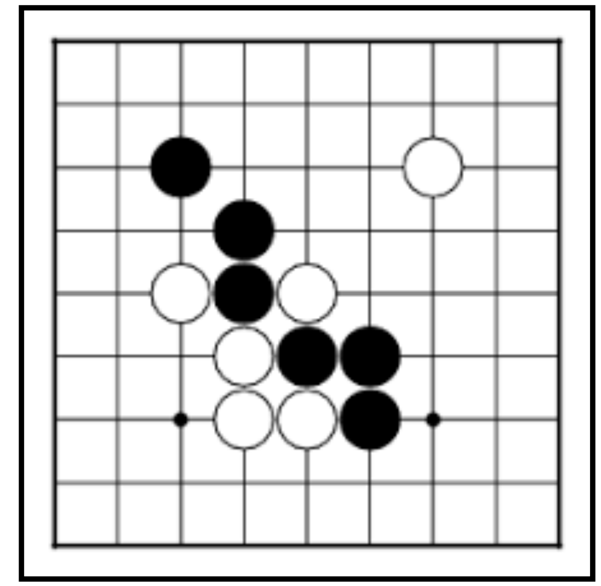
Playing Chess

Stationary policy: $\pi(a|s)$ is a function that maps each state to a probability distribution over the action space.

Background - Reinforcement Learning



$\langle \mathcal{S}, \mathcal{A}, \mathcal{P}, r, \gamma \rangle$



Playing Chess

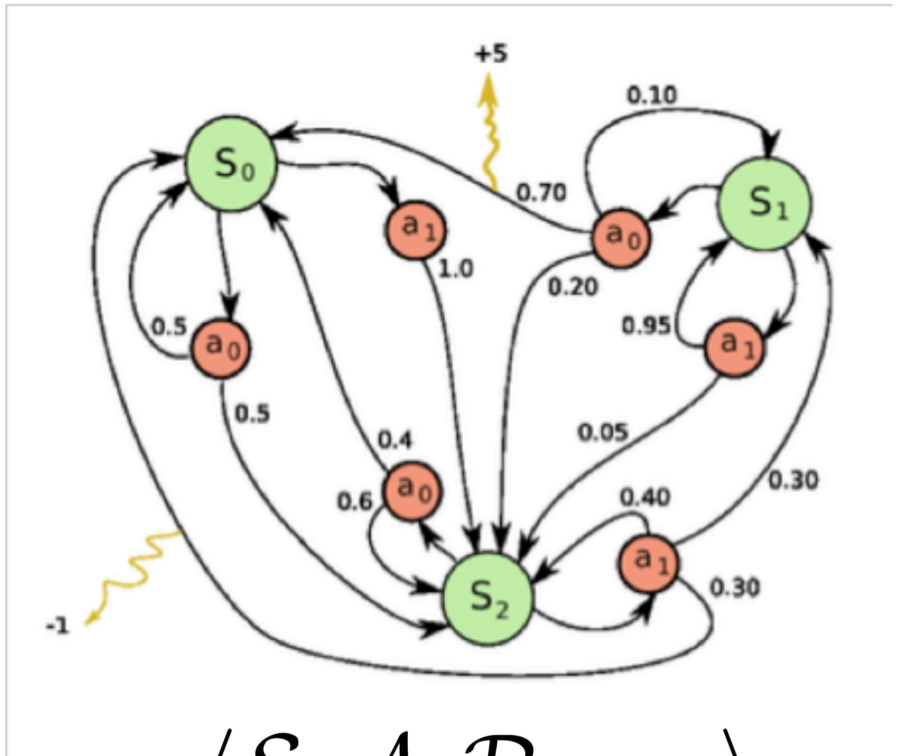
Total Reward: $\hat{r}_\tau := \sum_{t=0}^{\infty} \gamma^t r(s_t, a_t)$

if the agent executes a trajectory

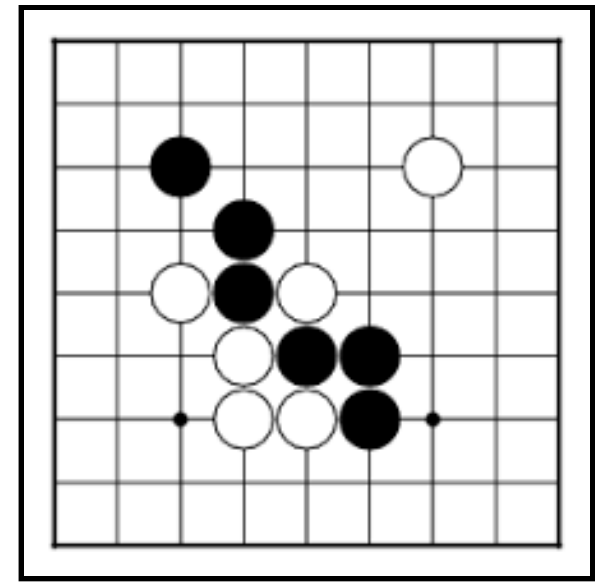
$\tau = \{(s_t, a_t)\}_{t=0}^{\infty}$.

Stationary policy: $\pi(a|s)$ is a function that maps each state to a probability distribution over the action space.

Background - Reinforcement Learning



$$\langle \mathcal{S}, \mathcal{A}, \mathcal{P}, r, \gamma \rangle$$



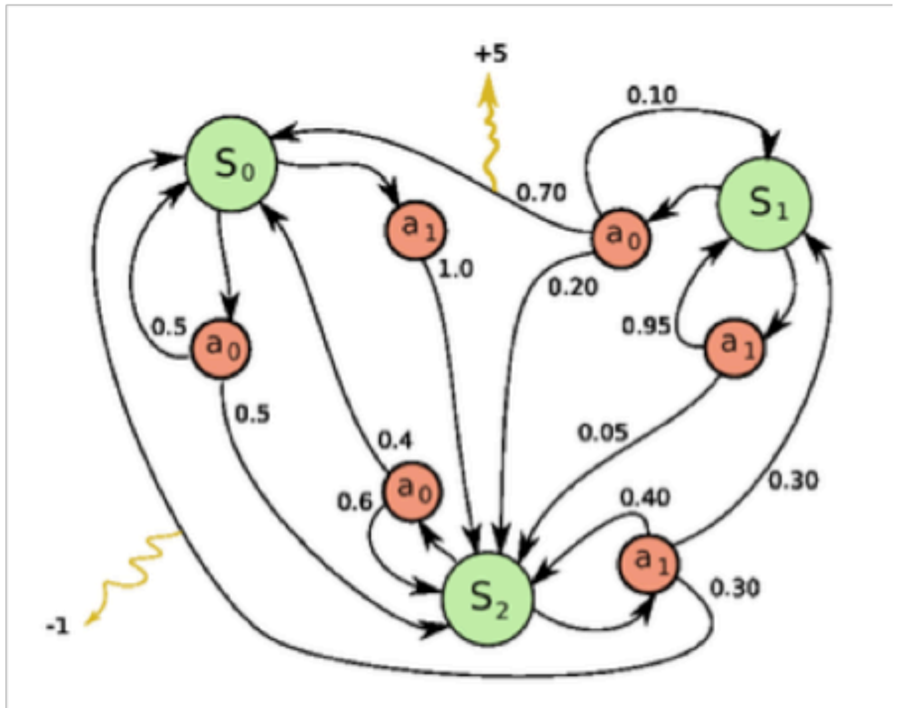
Playing Chess

Goal: find optimal π such that

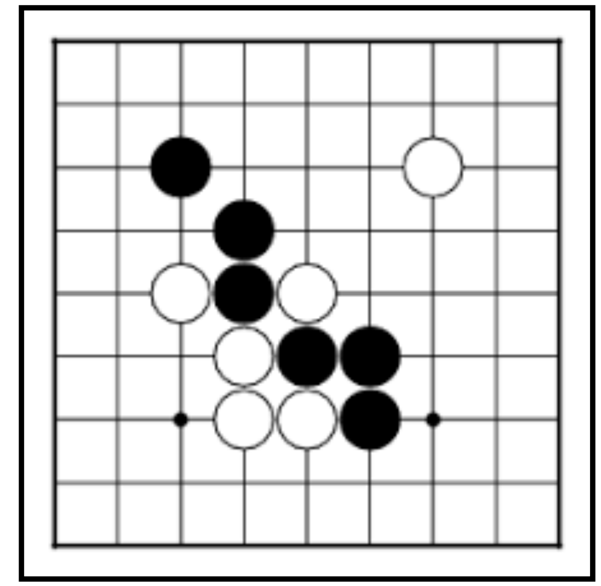
$$V^{\pi}(s) := \mathbb{E}_{\tau \sim (\mathcal{P}, \pi) | s_0 = s} \left[\hat{r}_{\tau} \right]$$

is maximized.

Background - Reinforcement Learning



$$\langle \mathcal{S}, \mathcal{A}, \mathcal{P}, r, \gamma \rangle$$



Playing Chess

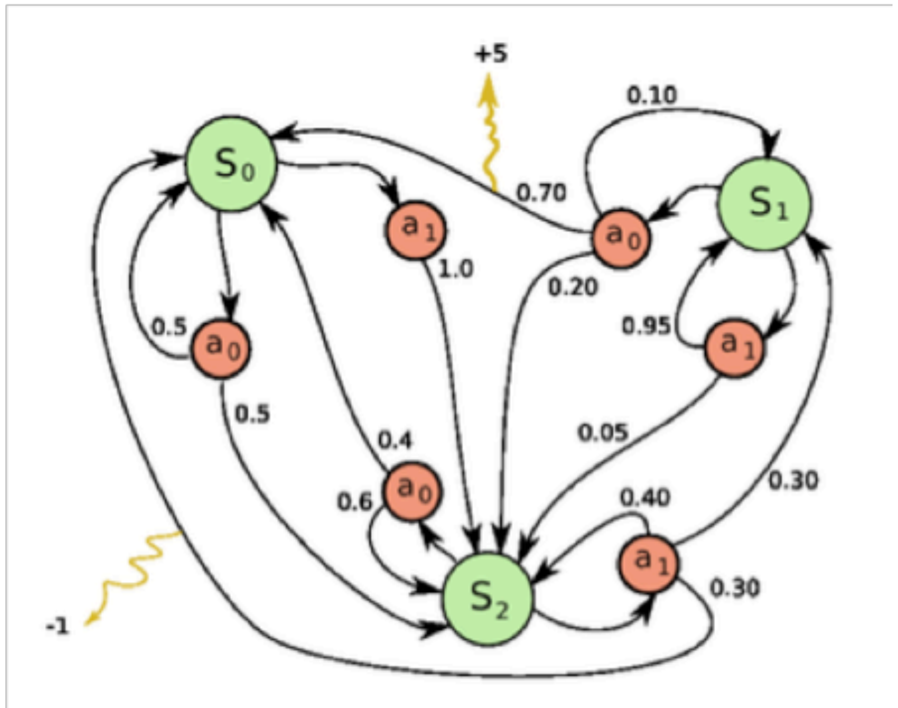
Goal: find optimal π such that

$$V^{\pi}(s) := \mathbb{E}_{\tau \sim (\mathcal{P}, \pi) | s_0 = s} \left[\hat{r}_{\tau} \right]$$

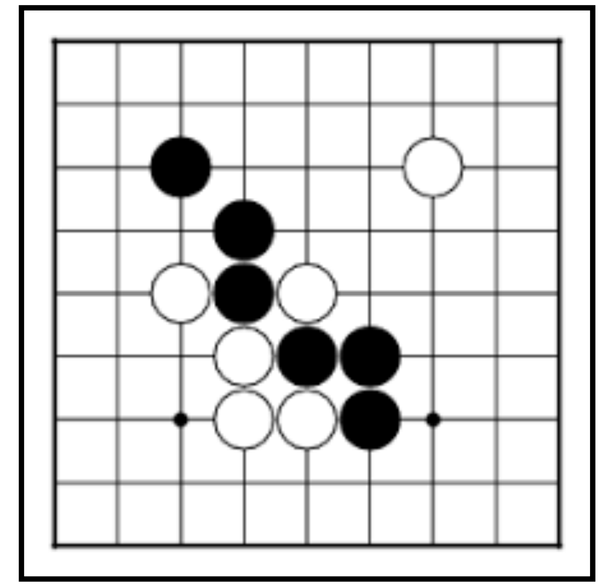
is maximized.

Value Function

Background - Reinforcement Learning



$$\langle \mathcal{S}, \mathcal{A}, \mathcal{P}, r, \gamma \rangle$$



Playing Chess

Goal: find optimal π such that

unknown: model-free

$$V^{\pi}(s) := \mathbb{E}_{\tau \sim (\mathcal{P}, \pi) | s_0 = s} \left[\hat{r}_{\tau} \right]$$

is maximized.

Value Function

Background - Reinforcement Learning

Goal: find optimal π such that

$$V^\pi(s) := \mathbb{E}_{\tau \sim (\mathcal{P}, \pi) | s_0 = s} \left[\hat{r}_\tau \right]$$

is maximized.

How? - (1) Evaluation & (2) Control

Background - Reinforcement Learning

How? - (1) Evaluation & (2) Control

(1)

$$\tilde{V}^{\pi}(s_0) = \frac{1}{N}\sum_{i=1}^{N} \hat{r}_{\tau_i},$$

(2)

$$\begin{cases} \boldsymbol{\eta} \propto \nabla_\theta \mathbb{E}_{\pi_\theta}\left[\tilde{V}^{\pi_\theta}(s_0)\right] = \mathbb{E}_{\pi_\theta}\left[\nabla_\theta \log \pi_\theta \cdot \tilde{V}^{\pi_\theta}(s_0)\right] \\ \theta \leftarrow \theta + \boldsymbol{\eta} \end{cases}$$

Policy-Based Methods: Large Variance, On-Policy

Background - Reinforcement Learning

How? - (1) Evaluation & (2) Control

(1)

$$\tilde{V}^{\pi}(s_0) = \frac{1}{N}\sum_{i=1}^{N} \hat{r}_{\tau_i},$$

(2)

$$\begin{cases} \boldsymbol{\eta} \propto \nabla_\theta \mathbb{E}_{\pi_\theta}\left[\tilde{V}^{\pi_\theta}(s_0)\right] = \mathbb{E}_{\pi_\theta}\left[\nabla_\theta \log \pi_\theta \cdot \tilde{V}^{\pi_\theta}(s_0)\right] \\ \theta \leftarrow \theta + \boldsymbol{\eta} \end{cases}$$

Policy-Based Methods: Large Variance, On-Policy

$$Q^{\pi}(s,a) := \mathbb{E}_{\tau\sim(\mathcal{P},\pi)|s_0=s,a_0=a}\left[\hat{r}_\tau\right]$$

(1)

$$Q^{\pi}(s,a) = r(s,a) + \gamma \mathbb{E}_{\mathcal{P},\pi} Q^{\pi}(s',a')$$

Bellman Expectation Equation

(2)

Bellman Optimality Equation

$$Q^{*}(s,a) = r(s,a) + \gamma \mathbb{E}_{\mathcal{P}} \max_{a'\in\mathcal{A}} Q^{*}(s',a')$$

Value-Based Methods

Background - Reinforcement Learning

How? - (1) Evaluation & (2) Control

Value-Based Methods

(1)

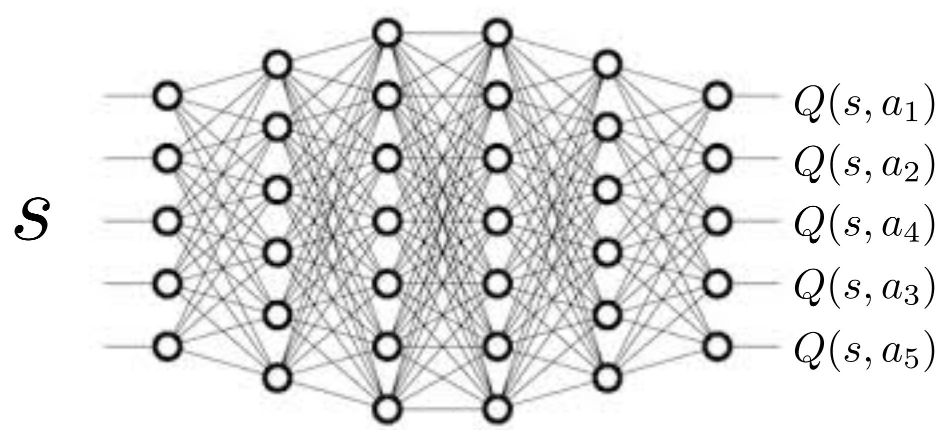
$$Q^{\pi}(s,a) = r(s,a) + \gamma \mathbb{E}_{\mathcal{P},\pi} Q^{\pi}(s',a')$$

Bellman Expectation Equation

(2)

Bellman Optimality Equation

$$Q^{*}(s,a) = r(s,a) + \gamma \mathbb{E}_{\mathcal{P}} \max_{a' \in \mathcal{A}} Q^{*}(s',a')$$



Deep Q-Network

Loss Functions: $L_k(\theta) = \mathbb{E}_{s,a}\left[(y_k - Q(s,a;\theta))^2\right]$

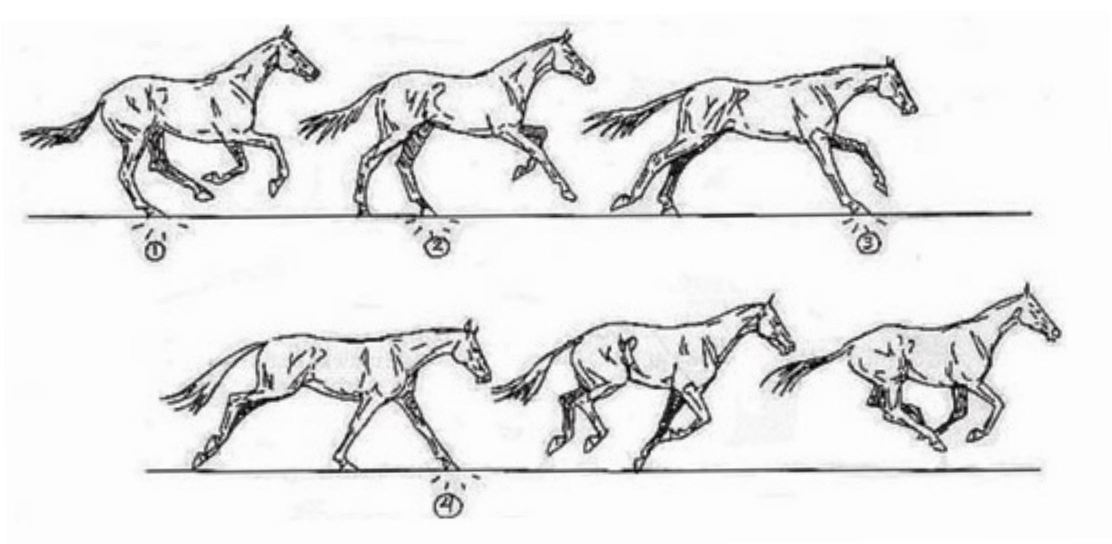
$$y_k = \mathbb{E}_{s'}\left[r(s,a) + \gamma \max_{a'} Q(s',a';\theta_k)\right]$$

Background - Problem Formulation

Reward Function $r : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$
e.g. $r(S_1, a_0) = 3.5$

The scalarized reward function design is often infeasible in practice.

e.g. Empirical hypothesis may be wrong.



$\langle \mathcal{S}, \mathcal{A}, \mathcal{P}, r, \gamma \rangle$

State Space Action Space

Stochastic Transition Kernel $\mathcal{P}(s'|s, a)$
e.g. $\mathcal{P}(S_0|S_1, a_0) = 0.7$

Reward Function $r : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$
e.g. $r(S_1, a_0) = 3.5$

$\gamma \in [0, 1)$ is a discount factor

Background - Problem Formulation

Reward Function $r : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$

e.g. $r(S_1, a_0) = 3.5$

The scalarized reward function design is often infeasible in practice.

e.g. Multi-attribute reward function.

Speed $\times$ Efficiency

$\times$ Stability

$\times$ Wear and tear on muscles

$\mapsto \mathbb{R}$?

$\langle \mathcal{S}, \mathcal{A}, \mathcal{P}, r, \gamma \rangle$

State Space

Action Space

Stochastic Transition Kernel $\mathcal{P}(s'|s, a)$

e.g. $\mathcal{P}(S_0|S_1, a_0) = 0.7$

Reward Function $r : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$

e.g. $r(S_1, a_0) = 3.5$

$\gamma \in [0, 1)$ is a discount factor

Background - Problem Formulation

Multi-Objective Markov Decision Processes(MOMDPs):

Use *Vectorial Rewards* to encode many possibly competing objectives.

e.g. $\boldsymbol{r} = \left[\text{Speed}, \text{Efficiency}, \text{Stability}, \text{Wear and Tear}\right]^\mathsf{T}$

e.g. Multi-attribute reward function.

Speed $\times$ Efficiency

$\times$ Stability

$\times$ Wear and tear on muscles

$\mapsto \mathbb{R}$?

$\langle \mathcal{S}, \mathcal{A}, \mathcal{P}, \boldsymbol{r}, \gamma \rangle$

State Space — Action Space

Stochastic Transition Kernel: $\mathcal{P}(s'|s,a)$, e.g. $\mathcal{P}(S_0|S_1,a_0) = 0.7$

Reward Function: $\boldsymbol{r} : \mathcal{S} \times \mathcal{A} \to \mathbb{R}^m$, e.g. $\boldsymbol{r}(s,a) = [0.1\ \ 2.0]^\mathsf{T}$

$\gamma \in [0,1)$ is a discount factor

Background - Problem Formulation

Multi-Objective Markov Decision Processes(MOMDPs):

Use *Vectorial Rewards* to encode many possibly competing objectives.

e.g. $\boldsymbol{r} = [\text{Speed}, \text{Efficiency}, \text{Stability}, \text{Wear and Tear}]^\mathsf{T}$

Goal: find all optimal π's such that

$$V^\pi(s) := \mathbb{E}_{\tau\sim(\mathcal{P},\pi)}[\hat{\boldsymbol{r}}_\tau]$$

$$:= \mathbb{E}_{\tau\sim(\mathcal{P},\pi)}\left[\sum_{t=0}^{\infty}\gamma^t \boldsymbol{r}(s_t, a_t)\right]$$

are ~~maximized~~ optimal ?

$\langle \mathcal{S}, \mathcal{A}, \mathcal{P}, \boldsymbol{r}, \gamma \rangle$

State Space

Action Space

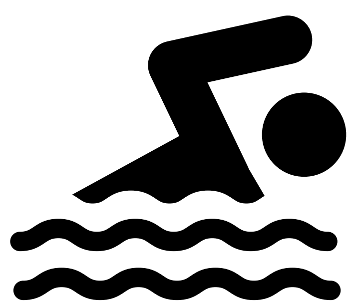
Stochastic Transition Kernel $\mathcal{P}(s'|s, a)$ e.g. $\mathcal{P}(S_0|S_1, a_0) = 0.7$

Reward Function $\boldsymbol{r} : \mathcal{S} \times \mathcal{A} \rightarrow \mathbb{R}^m$ e.g. $\boldsymbol{r}(s, a) = [0.1 \;\; 2.0]^\mathsf{T}$

$\gamma \in [0, 1)$ is a discount factor

Background - Problem Formulation

Optimality Concepts



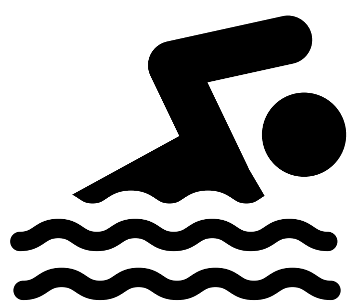
Multiple Competing Objectives

1. Policy dominance:

$$\pi' \succ \pi \Leftrightarrow \forall i \in [m], V_i^{\pi'}(s_0) > V_i^{\pi}(s_0)$$

Background - Problem Formulation

Optimality Concepts



Multiple Competing Objectives

1. Policy dominance:

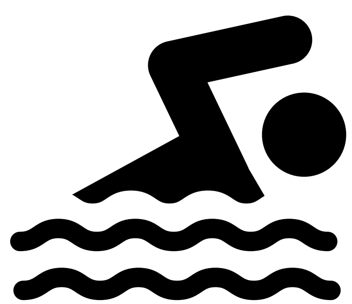
$$\pi' \succ \pi \Leftrightarrow \forall i \in [m], V_i^{\pi'}(s_0) > V_i^{\pi}(s_0)$$

2. Pareto optimal policies:

$$\Pi^* := \{\pi \mid \nexists \pi' \in \Pi, \pi' \succ \pi\}$$

Background - Problem Formulation

Optimality Concepts



Multiple Competing Objectives

1. Policy dominance:

$$\pi' \succ \pi \Leftrightarrow \forall i \in [m], V_i^{\pi'}(s_0) > V_i^{\pi}(s_0)$$

2. Pareto optimal policies:

$$\Pi^* := \{\pi \mid \nexists \pi' \in \Pi, \pi' \succ \pi\}$$

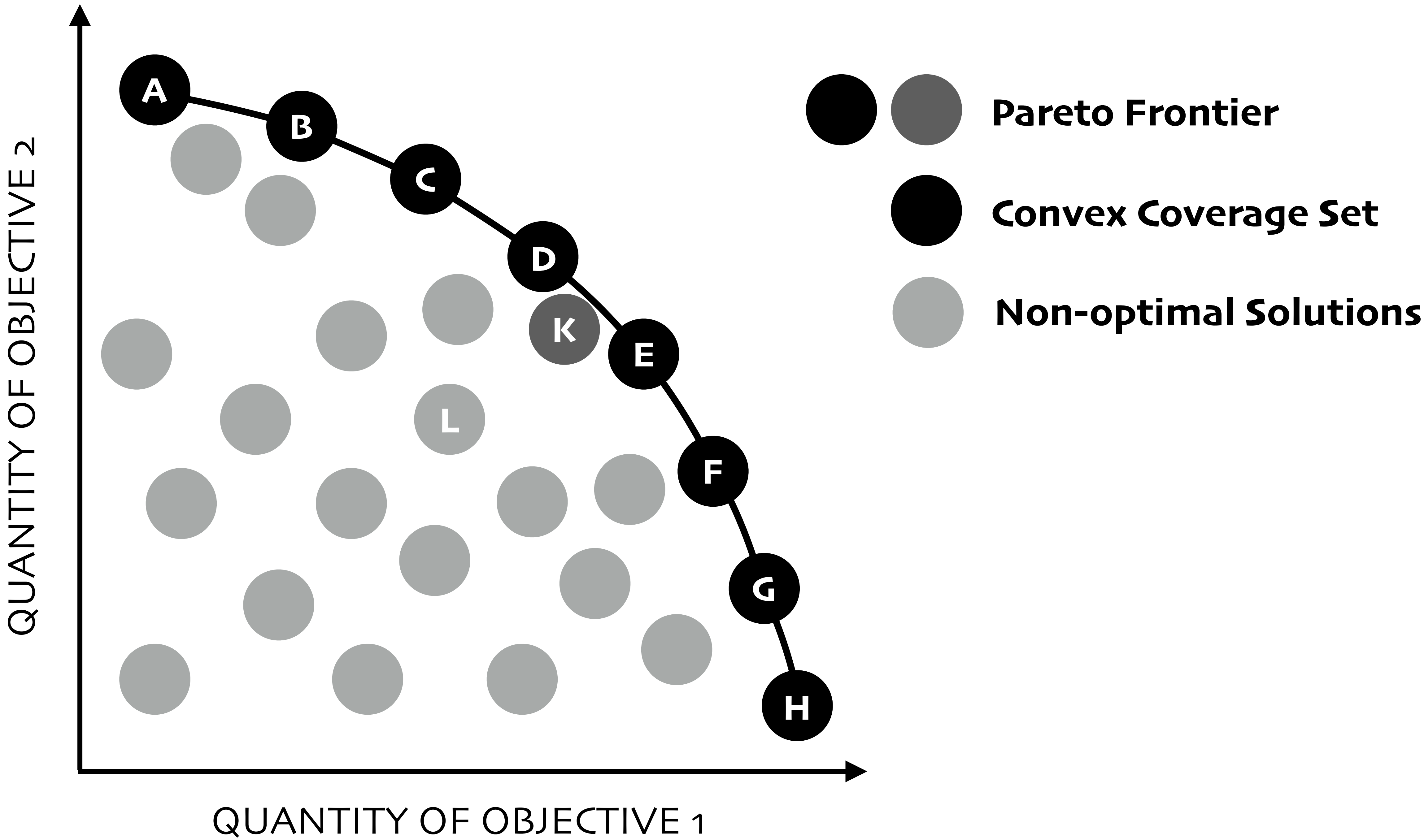
3. Pareto (optimal solutions) frontier:

$$\mathcal{F}^* := \{\boldsymbol{V}^{\pi}(s_0) \mid \pi \in \Pi^*\}$$ or

$$\mathcal{F}^* := \{\hat{\boldsymbol{r}}_{\tau} \mid \tau \sim (\mathcal{P}, \pi), \pi \in \Pi^*\}$$

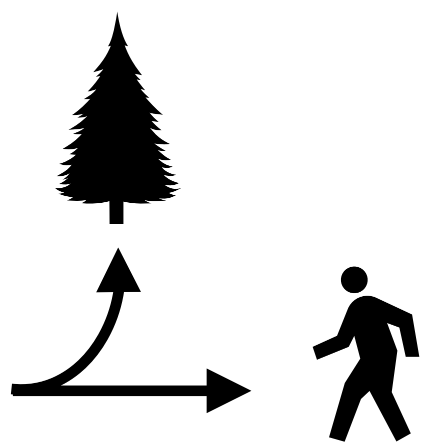
Background - Problem Formulation

Optimality Concepts



Background - Problem Formulation

Optimality Concepts



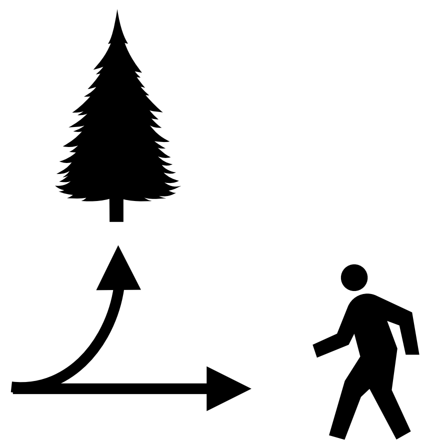
Human Preferences

A preference function $f : \mathbb{R}^m \to \mathbb{R}$

maps the value or reward consisting of quantity of m objectives in to one real scalar. Given value function $\boldsymbol{V}^{\pi}(s)$ or discounted total rewards $\hat{\boldsymbol{r}}_{\tau}$, we name the real value $f \circ \boldsymbol{V}^{\pi}(s)$ or $f(\hat{\boldsymbol{r}}_{\tau})$ the **policy's utility** under preference f.

Background - Problem Formulation

Optimality Concepts



Human Preferences

A preference function $f : \mathbb{R}^m \to \mathbb{R}$

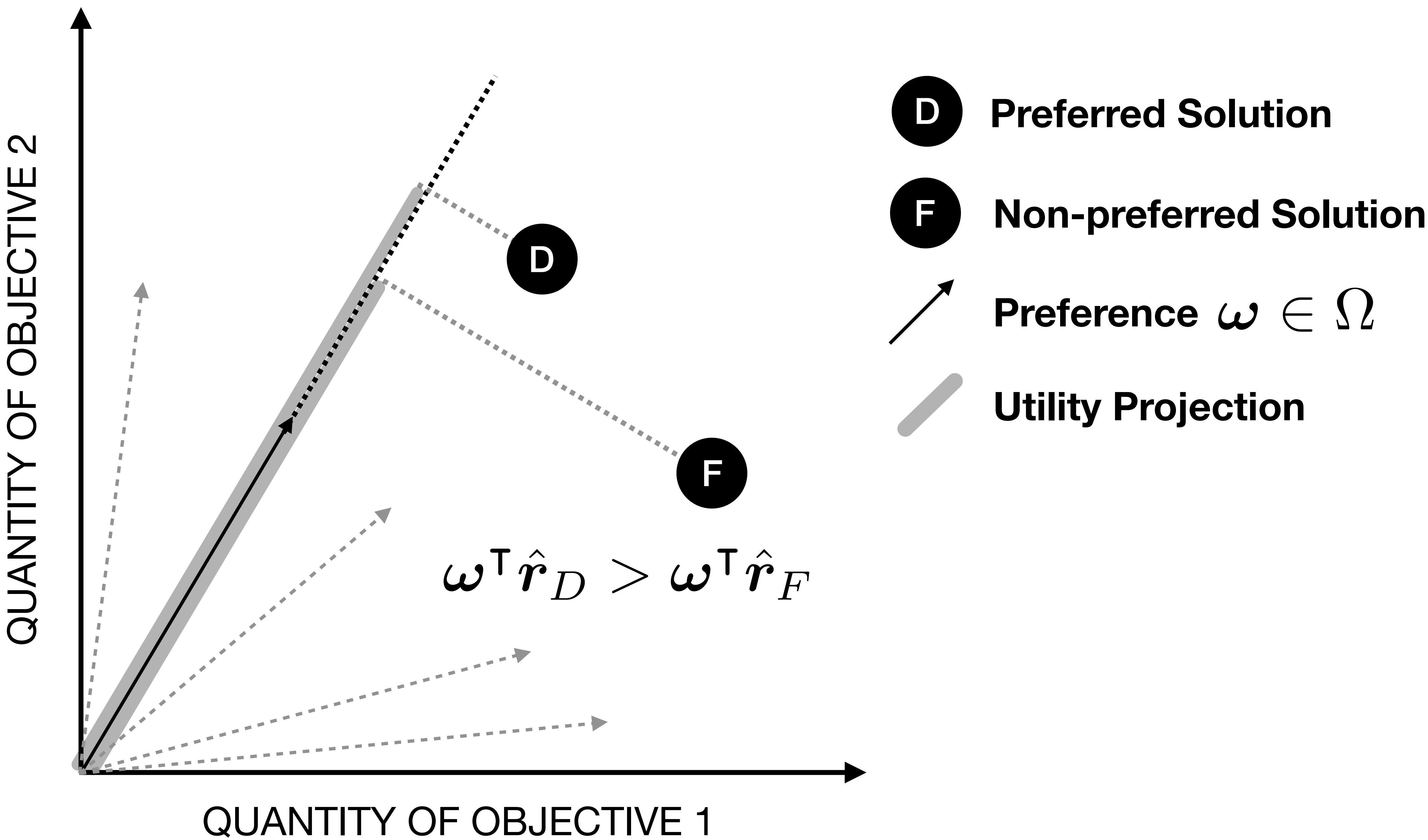
maps the value or reward consisting of quantity of m objectives in to one real scalar. Given value function $\boldsymbol{V}^{\pi}(s)$ or discounted total rewards $\hat{\boldsymbol{r}}_{\tau}$, we name the real value $f \circ \boldsymbol{V}^{\pi}(s)$ or $f(\hat{\boldsymbol{r}}_{\tau})$ the **policy's utility** under preference f.

Linear preference: $f_{\omega}(\boldsymbol{r}) = \boldsymbol{\omega}^{\intercal}\boldsymbol{r}$

Relative Importance Weights

Background - Problem Formulation

Optimality Concepts



Background - Problem Formulation

Delayed Linear Preference Scenarios

Learning Phase:

Unknown Linear Preference / Abundant Resources / Learn all policies.

$$\pi \in \Pi_{\mathcal{L}} \Rightarrow \exists\ \boldsymbol{\omega} \in \Omega, \text{s.t.}\ \ \forall \pi' \in \Pi, \boldsymbol{\omega}^{\intercal} \boldsymbol{v}^{\pi}(s_0) \geq \boldsymbol{\omega}^{\intercal} \boldsymbol{v}^{\pi'}(s_0)$$

Analysis Phase:

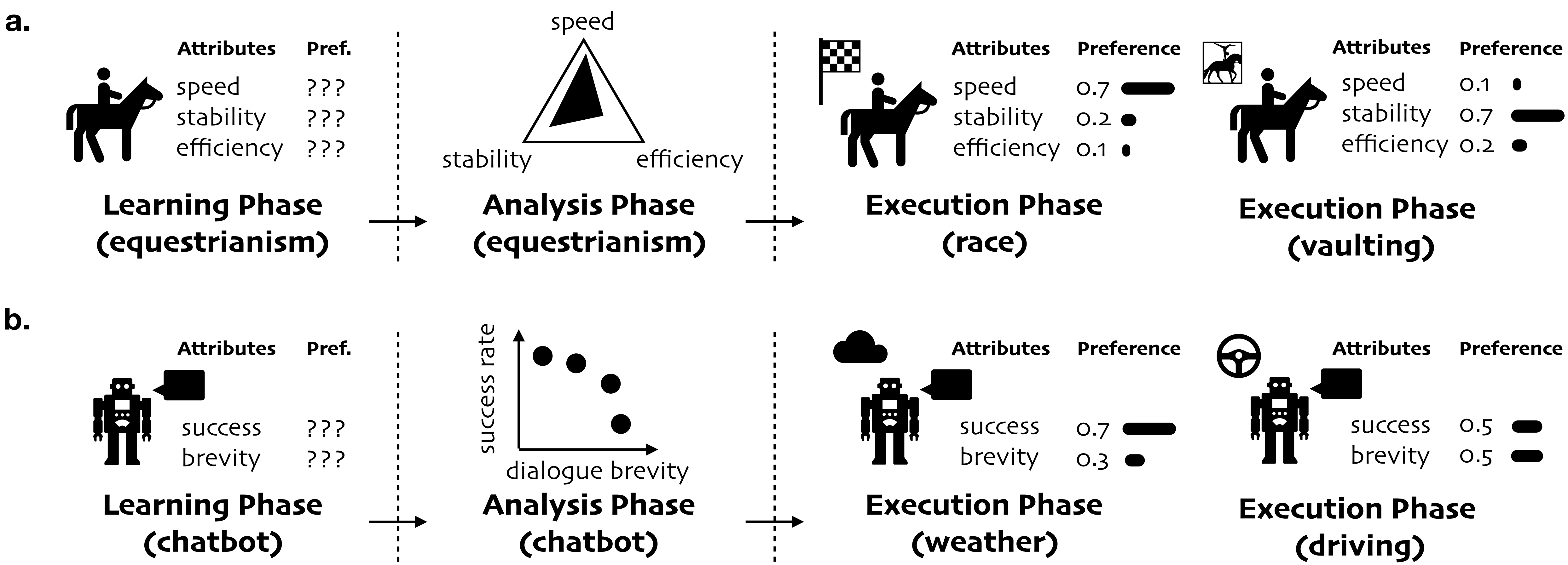
User can analyze the trade-off between multiple objectives.

Execution Phase:

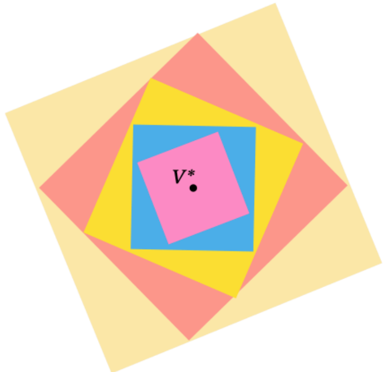
A specific linear preference function $\boldsymbol{\omega}$ will be given
Required to respond with an optimal policy π_{ω} from $\Pi_{\mathcal{L}}$ to the given preference, using limited computational resources.

Background - Problem Formulation

Delayed Linear Preference Scenarios



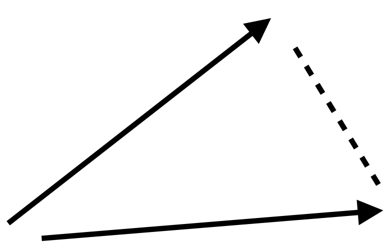
Theory - Framework for Value-Based RL



Please refer to my blog: https://runzhe-yang.science

Definition 1. (Metric Space) A *metric space* is an ordered pair (X, d) consists of an *underlying set* X and a real-valued function $d(x, y)$, called *metric*, defined for $x, y \in X$ such that for any $x, y, z \in X$ the following conditions are satisfied:

1. $d(x, y) \geq 0$ [*non-negativity*]
2. $d(x, y) = 0 \Leftrightarrow x = y$ [*identity of indiscernibles*]
3. $d(x, y) = d(y, x)$ [*symmetry*]
4. $d(x, y) \leq d(x, z) + d(z, y)$ [*triangle inequality*]



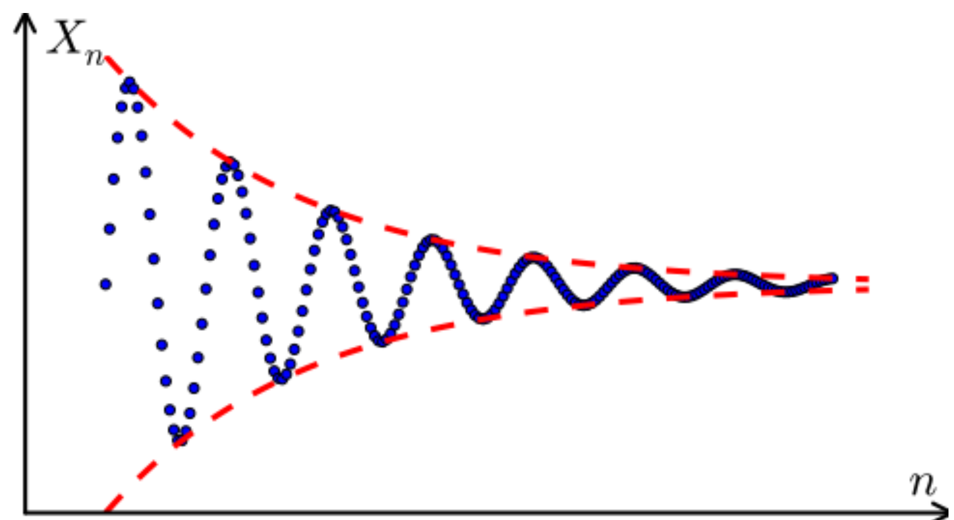
All these four conditions are in harmony with our intuition of distance. Indeed, the Euclidean distance $d(x, y) = \sqrt{(x_1 - y_1)^2 + (x_2 - y_2)^2 + (x_3 - y_3)^2}$ is a valid metric.

Theory - Framework for Value-Based RL

Definition 2. (Contraction) Let (X, d) be a metric space and $f : X \to X$. We say that f is a *contraction*, or a *contraction mapping*, if there is a real number $k \in [0, 1)$, such that

$$d(f(x), f(y)) \leq k d(x, y)$$

for all x and y in X, where the term k is called a *Lipschitz coefficent* for f.

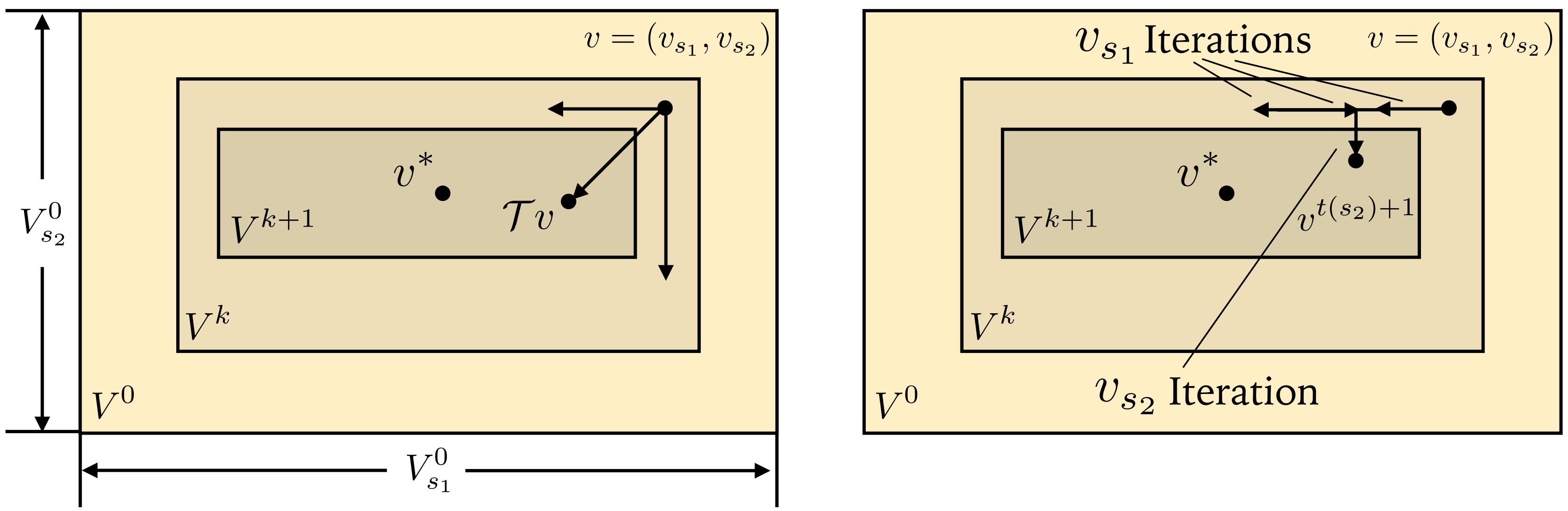


Theorem 1. (Contraction Mapping Theorem) Let (X, d) be a complete metric space and let $f : X \to X$ be a contraction. Then there is one and only one fixed point x^* such that

$$f(x^*) = x^*.$$

Moreover, if x is any point in X and $f^n(x)$ is inductively defined by $f^2(x) = f(f(x))$, $f^3(x) = f(f^2(x)), \ldots, f^n(x) = f(f^{n-1}(x))$, then $f^n(x) \to x^*$ as $n \to \infty$.

Theory - Framework for Value-Based RL



Topological interpretation of the asynchronous value iteration.

Single-objective Reinforcement Learning algorithms:

1) Value Space: all the bounded functions in $\mathcal{Q} = \mathbb{R}^{\mathcal{S} \times A}$

2) Value Metric: $d(Q, Q') = \sup_{s,a} |Q(s,a) - Q'(s,a)|$

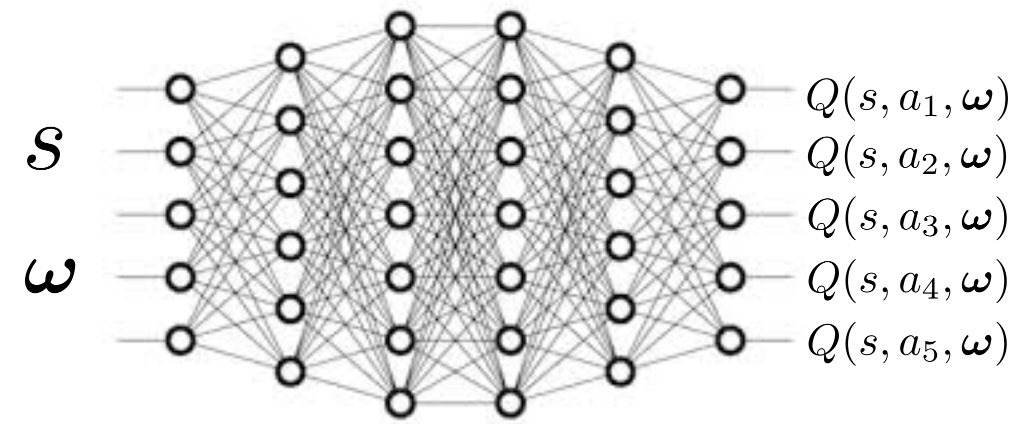
3) Optimality Operator: $(\mathcal{T}Q)(s,a) := r(s,a) + \gamma \mathbb{E}_{s' \sim \mathcal{P}(\cdot|s,a)} \sup_{a' \in \mathcal{A}} Q(s', a')$

is a contraction
with the fixed-point Q^*

Optimality Filter

4) Updating Scheme: asynchronous value iteration

Theory - Deep MORL Algorithms



Utility-Based Multi-Objective Q-Network

(Naive Version)

Multi-Objective Reinforcement Learning (MORL) algorithm:

1) **Value Space:** all the bounded functions in $\mathcal{Q} = (\Omega \to \mathbb{R})^{\mathcal{S}\times\mathcal{A}}$

2) **Value Metric:** $d(Q, Q') = \sup_{s,a} \sup_{\omega} |Q(s,a,\omega) - Q'(s,a,\omega)|$

$\langle \mathcal{Q}, d \rangle$ is still complete.

3) **Optimality Operator:** $(\mathcal{T}Q)(s, a, \boldsymbol{\omega}) := \boldsymbol{\omega}^{\intercal}\boldsymbol{r}(s, a) + \gamma \mathbb{E}_{s' \sim \mathcal{P}(\cdot|s,a)} (\mathcal{H}Q)(s', \boldsymbol{\omega}).$

is a contraction with the fixed-point Q^*

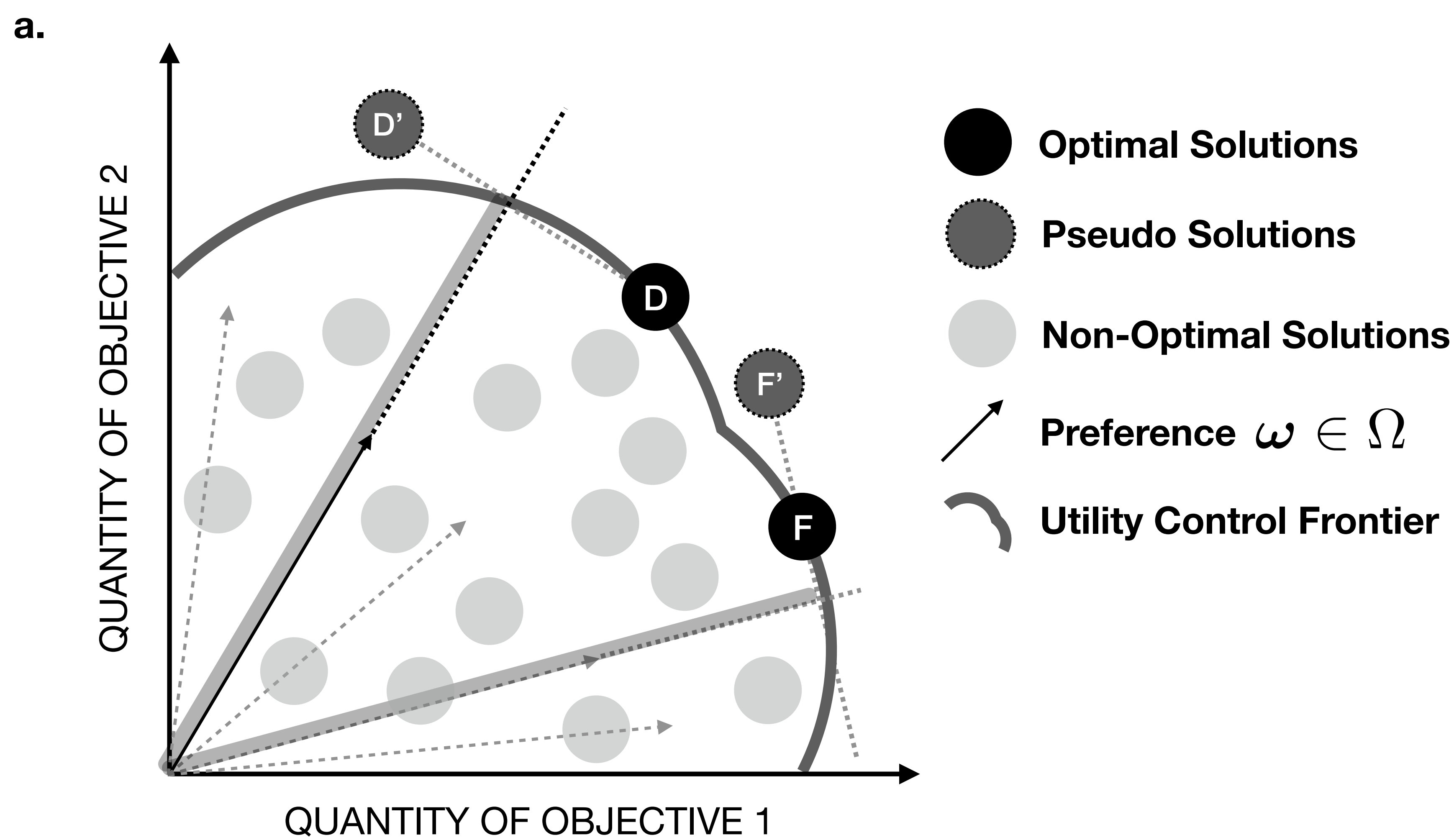
Optimality Filter

$(\mathcal{H}Q)(s, \omega) := \sup_{a'} Q(s, a', \omega)$

4) **Updating Scheme:** Hindsight Experience Reply (HER) [OpenAI, NIPS2017]

Theory - Deep MORL Algorithms

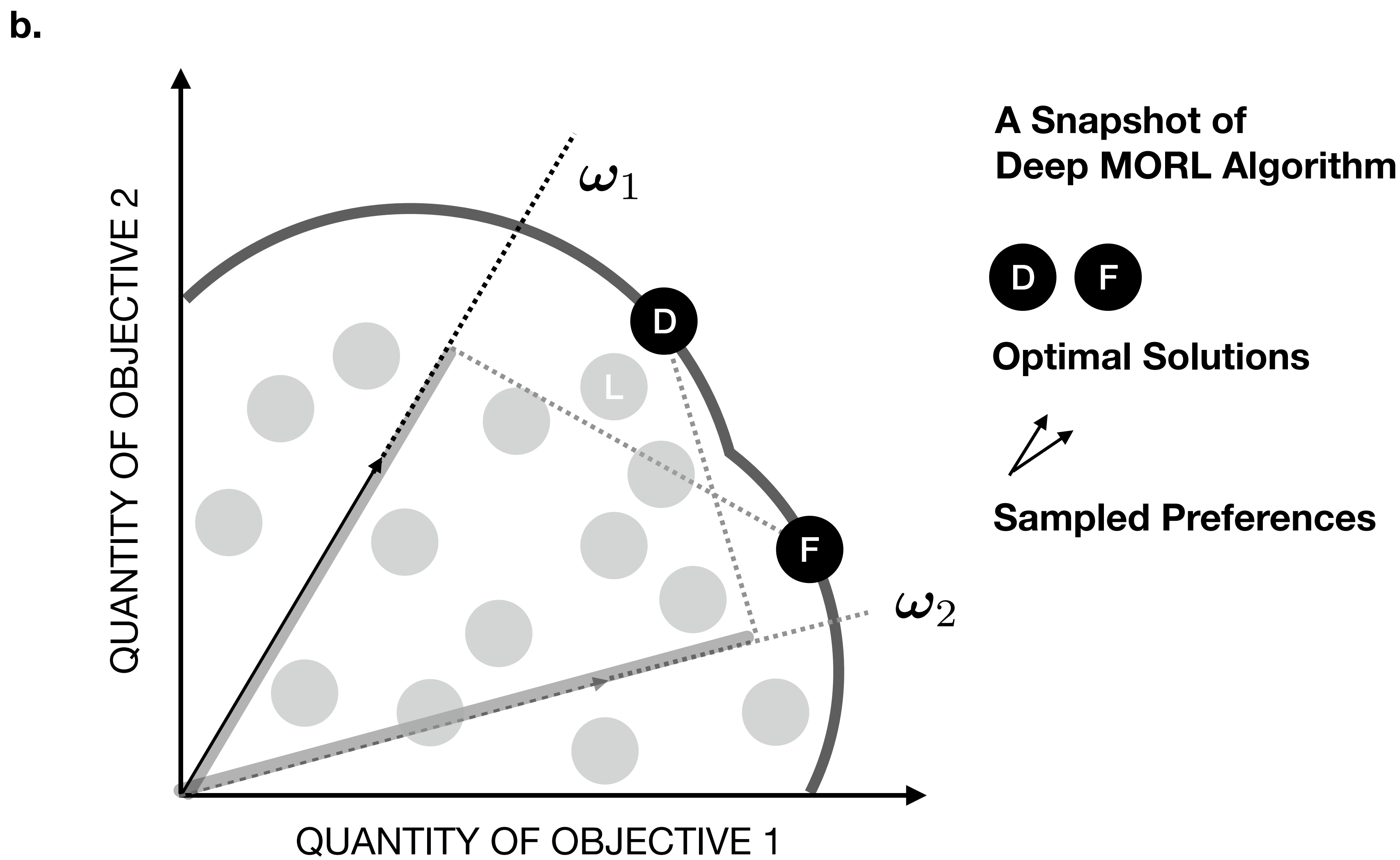
Problem 1: predictions are not informative;



Several predicted utilities are not enough to recover the Pareto optimal solutions, unless we have known the whole utility frontier.

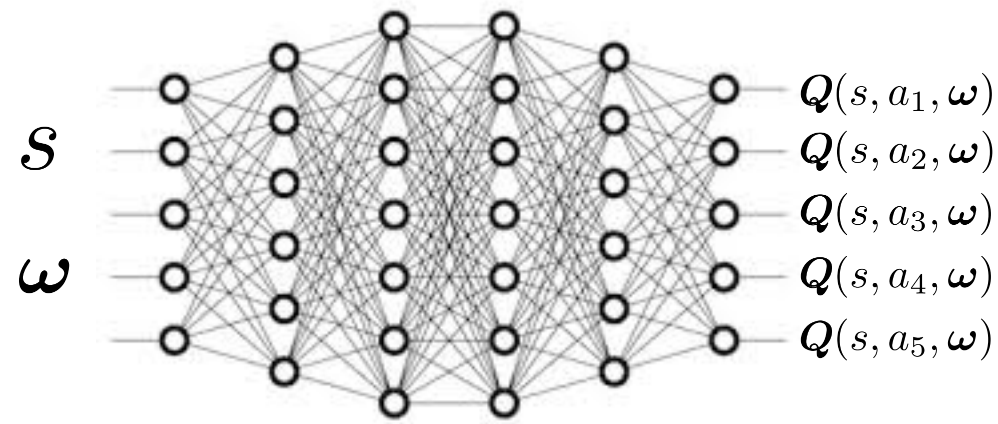
Theory - Deep MORL Algorithms

Problem 2: sample inefficiency.



At some stage, the naive algorithm finds the optimal solutions while they are not aligned with preferences. It still requires many iterations for the value-preference alignment.

Theory - Deep MORL Algorithms



Multi-Objective Q-Network

(Envelope Version)

Multi-Objective Reinforcement Learning (MORL) algorithm:

1) Value Space: all the bounded functions in $\mathcal{Q} = (\Omega \to \mathbb{R}^m)^{\mathcal{S}\times\mathcal{A}}$

2) Value Metric: $d(\boldsymbol{Q}, \boldsymbol{Q}') := \sup_{\substack{s\in\mathcal{S}, a\in\mathcal{A} \\ \boldsymbol{\omega}\in\Omega}} |\boldsymbol{\omega}^\intercal(\boldsymbol{Q}(s, a, \boldsymbol{\omega}) - \boldsymbol{Q}'(s, a, \boldsymbol{\omega}))|$ ← Pseudo-metric

3) Optimality Operator: $(\mathcal{T}\boldsymbol{Q})(s, a, \boldsymbol{\omega}) := \boldsymbol{r}(s, a) + \gamma \mathbb{E}_{s'\sim\mathcal{P}(\cdot|s,a)} (\mathcal{H}\boldsymbol{Q})(s', \boldsymbol{\omega})$

is a generalized contraction with the fixed-point class $[Q^*]$

Optimality Filter

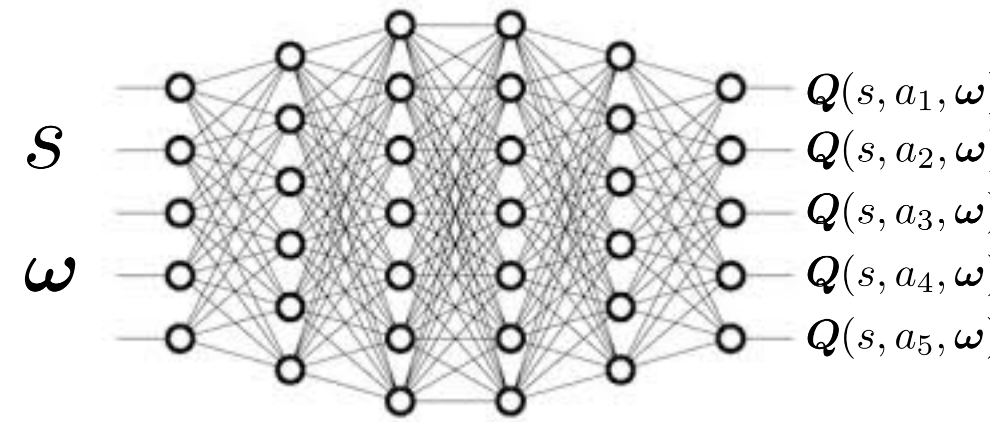
$(\mathcal{H}\boldsymbol{Q})(s, \boldsymbol{\omega}) := \arg_{\boldsymbol{Q}} \sup_{a', \boldsymbol{\omega}'} \boldsymbol{\omega}^\intercal \boldsymbol{Q}(s, a', \boldsymbol{\omega}')$

4) Updating Scheme: hindsight experience reply + **homotopy method**

Theory - Deep MORL Algorithms

(Envelope Version)

Multi-Objective Reinforcement Learning algorithms:



4) Updating Scheme: hindsight experience reply + **homotopy method**

$$L_k^{\mathtt{A}}(\theta) = \mathbb{E}_{s,a,\boldsymbol{\omega}}\left[\|\boldsymbol{y}_k - \boldsymbol{Q}(s,a,\boldsymbol{\omega};\theta)\|_2^2\right] \qquad L_k^{\mathtt{B}}(\theta) = \mathbb{E}_{s,a,\boldsymbol{\omega}}[|\boldsymbol{\omega}^{\intercal}\boldsymbol{y}_k - \boldsymbol{\omega}^{\intercal}\boldsymbol{Q}(s,a,\boldsymbol{\omega};\theta)|]$$

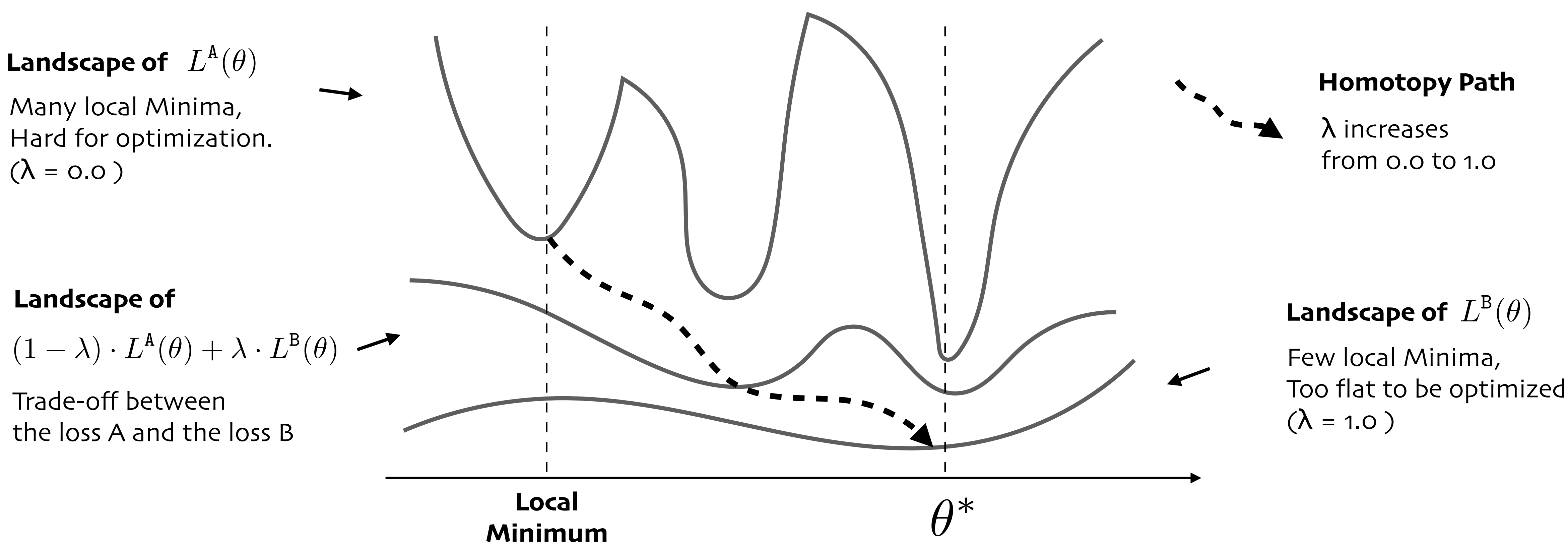
$$\boldsymbol{y}_k = \mathbb{E}_{s'}\left[\boldsymbol{r}(s,a) + \gamma(\mathcal{H}\boldsymbol{Q})(s', a', \boldsymbol{\omega}; \theta_k)\right]$$

Homotopy loss functions: $L_k(\theta) = (1-\lambda_k)\cdot L_k^{\mathtt{A}}(\theta) + \lambda_k \cdot L_k^{\mathtt{B}}(\theta)$

Theory - Deep MORL Algorithms

(Envelope Version)

Multi-Objective Reinforcement Learning algorithms:

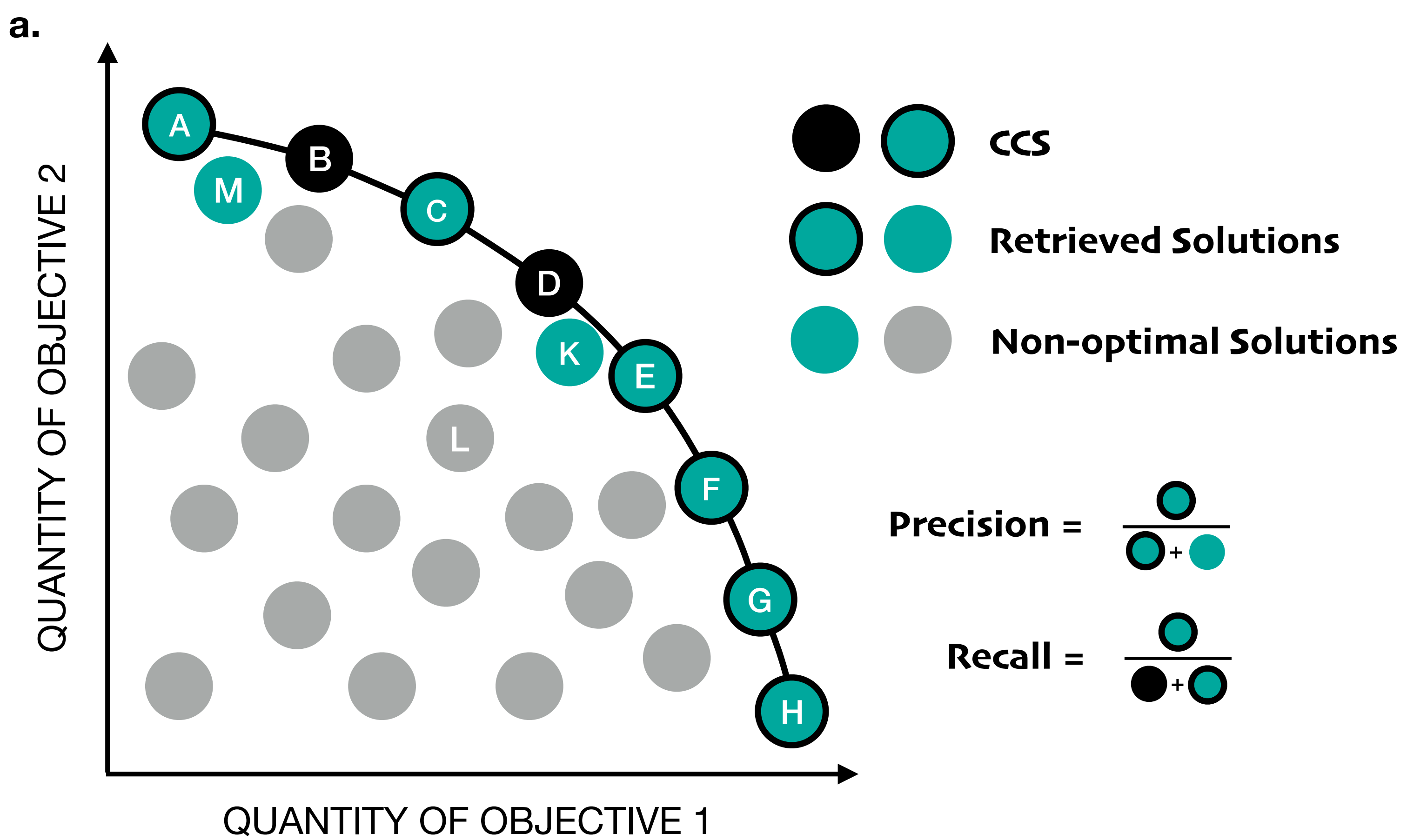


The homotopy path connecting two loss functions provides better opportunities to find the global optimal parameters.

Evaluation - Metrics

(1) An agent's ability to find all the potential optimal solutions in the convex coverage set of Pareto frontier.

(2) An agent's ability to adapt its policy to real-time specified preferences in the execution phase.

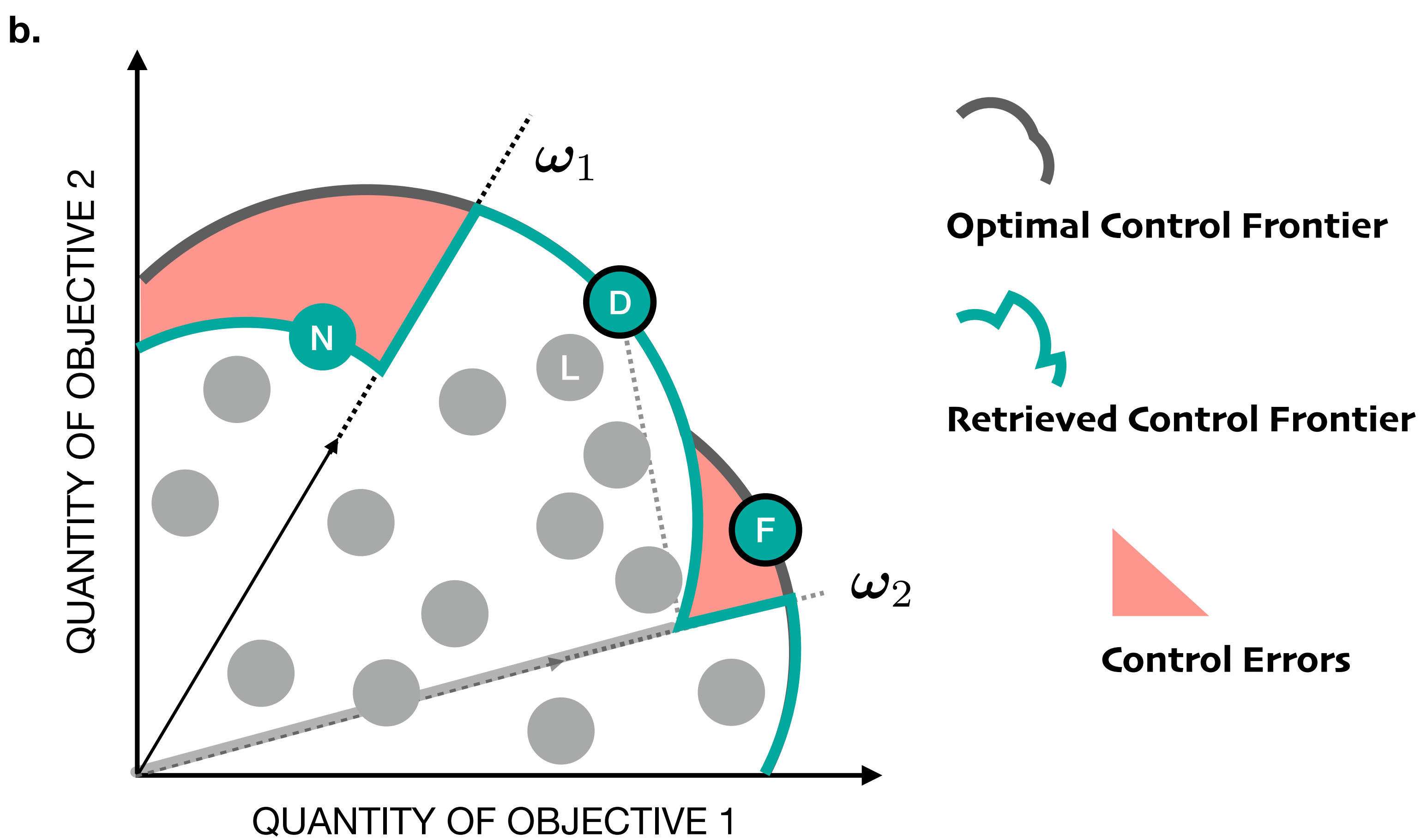


Coverage Ratio: $$\mathrm{CR}_{\mathrm{F1}}(\mathcal{F}) = 2 \cdot \frac{\mathtt{precision} \cdot \mathtt{recall}}{\mathtt{precision} + \mathtt{recall}}$$

Evaluation - Metrics

(1) An agent's ability to find all the potential optimal solutions in the convex coverage set of Pareto frontier.

(2) An agent's ability to adapt its policy to real-time specified preferences in the execution phase.



Adaptation Quality: $\mathtt{AQ}(\mathcal{C}) = \dfrac{1}{1 + \alpha \cdot \mathtt{err}_{\mathcal{D}_\omega}}$

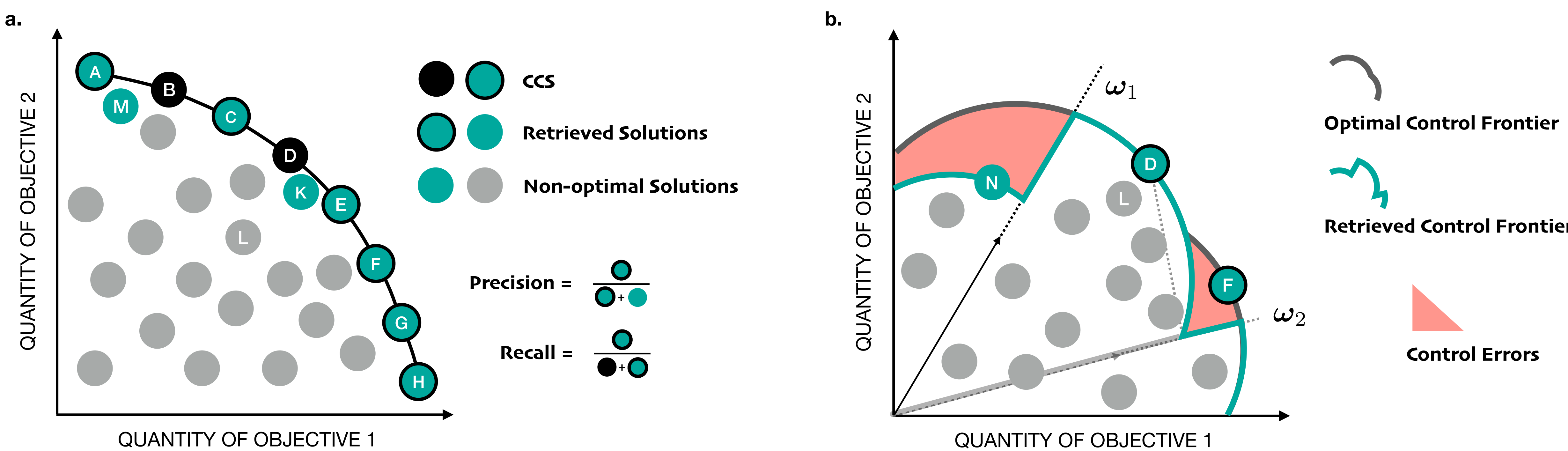
Evaluation - Synthetic Environments

Why Synthetic Environments?

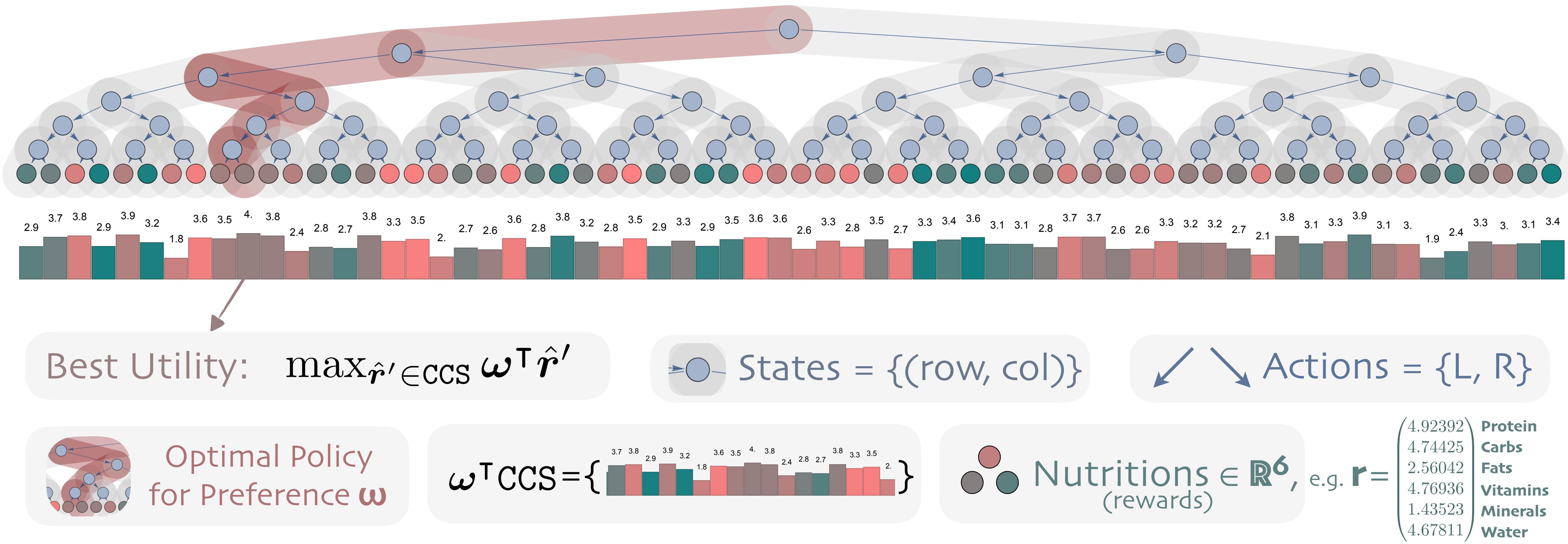
Evaluation - Synthetic Environments

Why Synthetic Environments?

The Access to Ground Truth for Evaluation!



Evaluation - Synthetic Environments



Fruit Tree Navigation (FTN): An agent travels from the root node to one of the leaf node to pick a fruit according to a post-assigned preference ω on the components of nutrition, treated as different objectives. The observation of an agent is its current coordinates (row, col), and its valid actions are moving to the left or the right child node.

Evaluation - Synthetic Environments

# Samples	Coverage Ratio Recall (execution)		Coverage Ratio F1 (execution)		Adaptation Quality (execution)	
	Naive	Envelope	Naive	Envelope	Naive	Envelope
1	0.4562±0.058	0.8626±0.084	0.625±0.057	0.924±0.051	0.7037±0.012	0.759±0.066
4	0.6254±0.097	0.972±0.007	0.7654±0.077	0.9856±0.004	0.7701±0.026	0.9101±0.006
8	0.753±0.101	0.9624±0.014	0.856±0.067	0.9808±0.007	0.8205±0.023	0.9261±0.015
16	0.8188±0.096	0.9904±0.009	0.8976±0.062	0.9952±0.004	0.8255±0.044	0.9306±0.007
32	0.85±0.061	0.975±0.041	0.914±0.044	0.987±0.021	0.8597±0.035	0.9402±0.011
64	0.8968±0.036	0.9812±0.013	0.9452±0.02	0.9904±0.007	0.877±0.031	0.9506±0.001
128	0.8626±0.042	0.9906±0.021	0.9258±0.024	0.9952±0.011	0.8705±0.03	0.9536±0.002

Sample Efficiency - Coverage Ratio (CR) & Adaptation Quality (AQ) comparison of two deep MORL algorithms tested on fruit tree navigation task, where the tree depth d=6. Trained on 5000 episode.

Evaluation - Synthetic Environments

# Samples	Coverage Ratio Recall (execution)		Coverage Ratio F1 (execution)		Adaptation Quality (execution)	
	Naive	Envelope	Naive	Envelope	Naive	Envelope
1	0.4562±0.058	0.8626±0.084	0.625±0.057	0.924±0.051	0.7037±0.012	0.759±0.066
4	0.6254±0.097	0.972±0.007	0.7654±0.077	0.9856±0.004	0.7701±0.026	0.9101±0.006
8	0.753±0.101	0.9624±0.014	0.856±0.067	0.9808±0.007	0.8205±0.023	0.9261±0.015
16	0.8188±0.096	0.9904±0.009	0.8976±0.062	0.9952±0.004	0.8255±0.044	0.9306±0.007
32	0.85±0.061	0.975±0.041	0.914±0.044	0.987±0.021	0.8597±0.035	0.9402±0.011
64	0.8968±0.036	0.9812±0.013	0.9452±0.02	0.9904±0.007	0.877±0.031	0.9506±0.001
128	0.8626±0.042	0.9906±0.021	0.9258±0.024	0.9952±0.011	0.8705±0.03	0.9536±0.002

Sample Efficiency - Coverage Ratio (CR) & Adaptation Quality (AQ) comparison of two deep MORL algorithms tested on fruit tree navigation task, where the tree depth d=6. Trained on 5000 episode.

Evaluation - Synthetic Environments

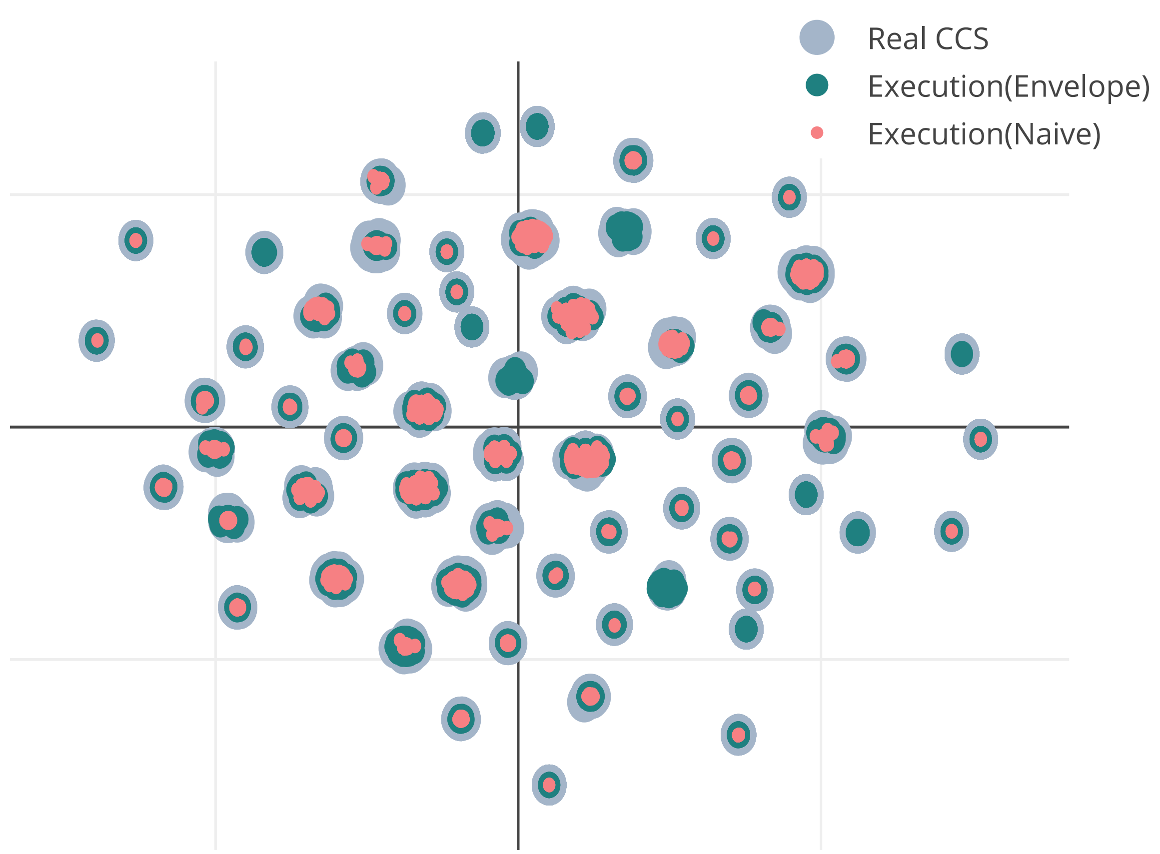
# Samples	Coverage Ratio Recall (execution)		Coverage Ratio F1 (execution)		Adaptation Quality (execution)	
	Naive	Envelope	Naive	Envelope	Naive	Envelope
1	0.4562±0.058	0.8626±0.084	0.625±0.057	0.924±0.051	0.7037±0.012	0.759±0.066
4	0.6254±0.097	0.972±0.007	0.7654±0.077	0.9856±0.004	0.7701±0.026	0.9101±0.006
8	0.753±0.101	0.9624±0.014	0.856±0.067	0.9808±0.007	0.8205±0.023	0.9261±0.015
16	0.8188±0.096	0.9904±0.009	0.8976±0.062	0.9952±0.004	0.8255±0.044	0.9306±0.007
32	0.85±0.061	0.975±0.041	0.914±0.044	0.987±0.021	0.8597±0.035	0.9402±0.011
64	0.8968±0.036	0.9812±0.013	0.9452±0.02	0.9904±0.007	0.877±0.031	0.9506±0.001
128	0.8626±0.042	0.9906±0.021	0.9258±0.024	0.9952±0.011	0.8705±0.03	0.9536±0.002

Previous Problem 2

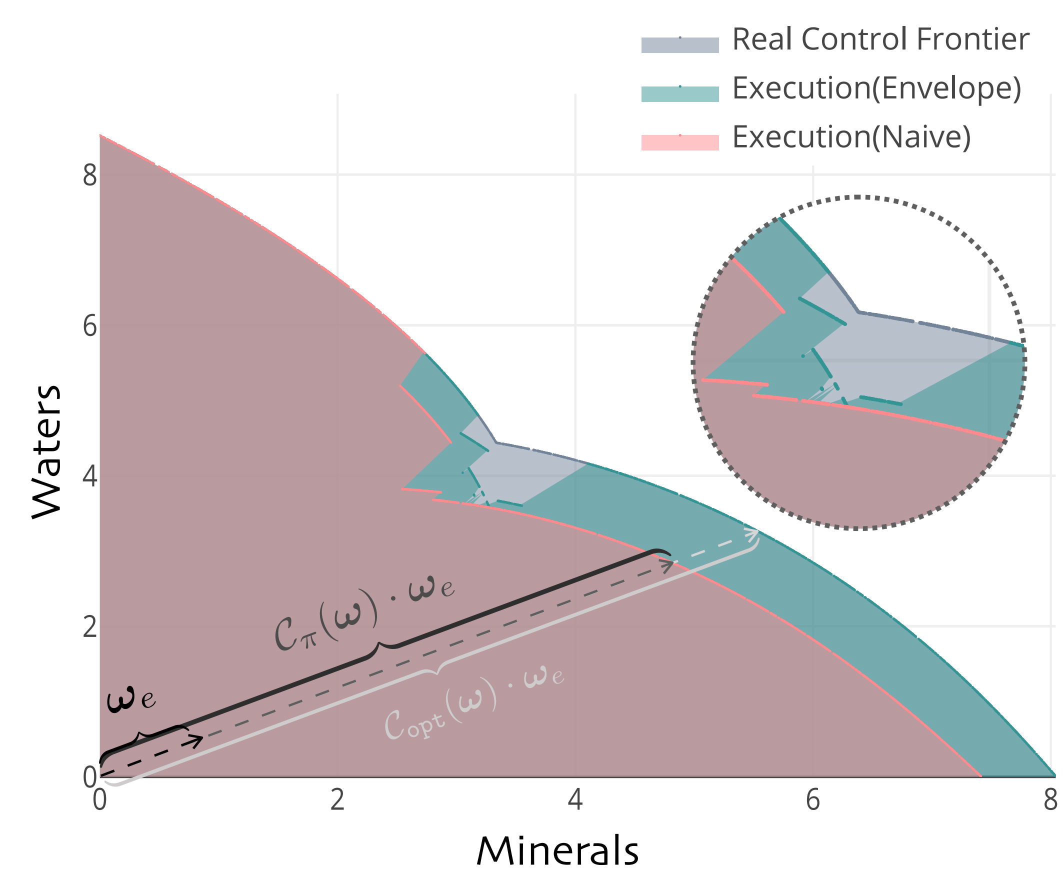
Sample Efficiency - Coverage Ratio (CR) & Adaptation Quality (AQ) comparison of two deep MORL algorithms tested on fruit tree navigation task, where the tree depth d=6. Trained on 5000 episode.

Evaluation - Synthetic Environments

a.



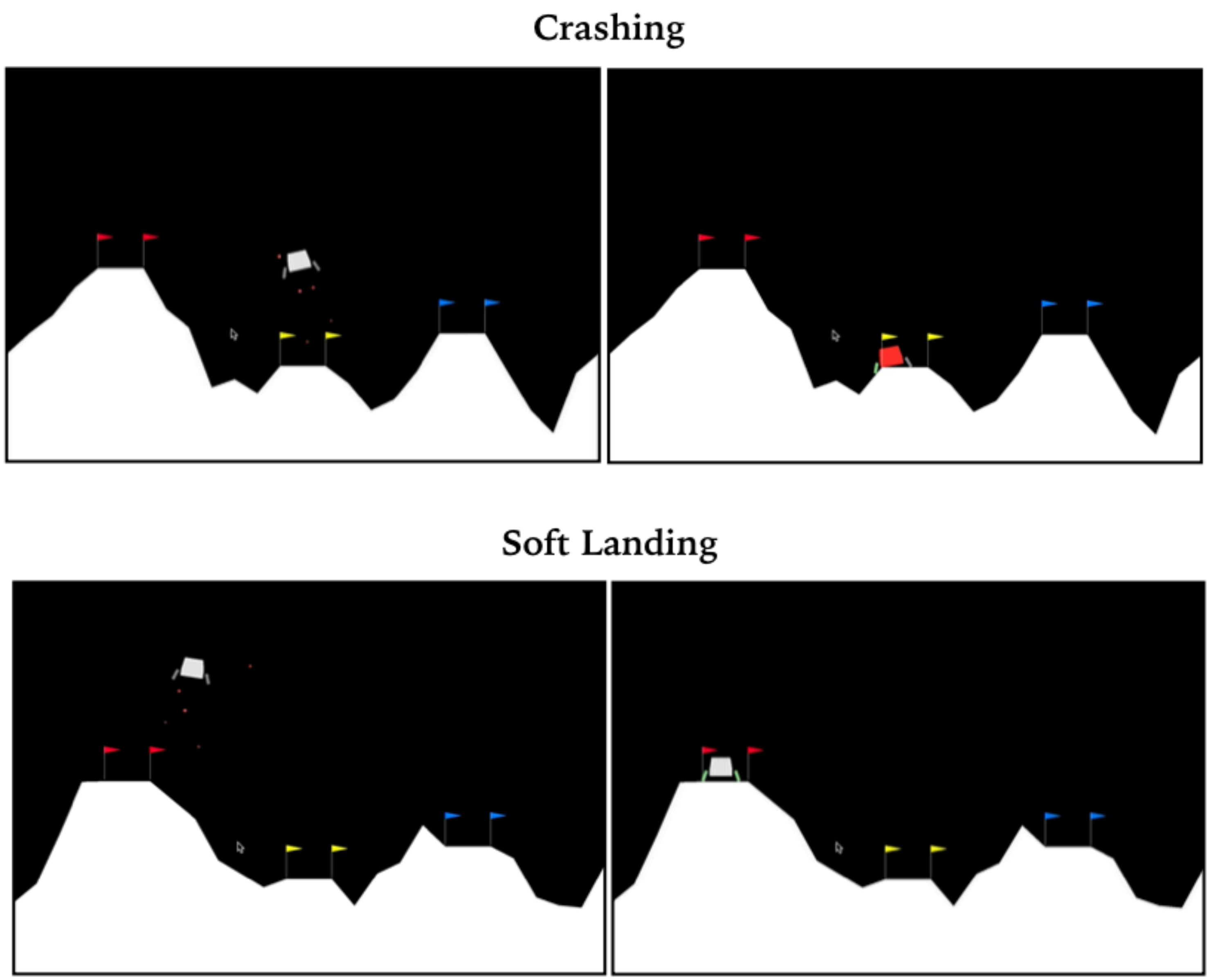
b.



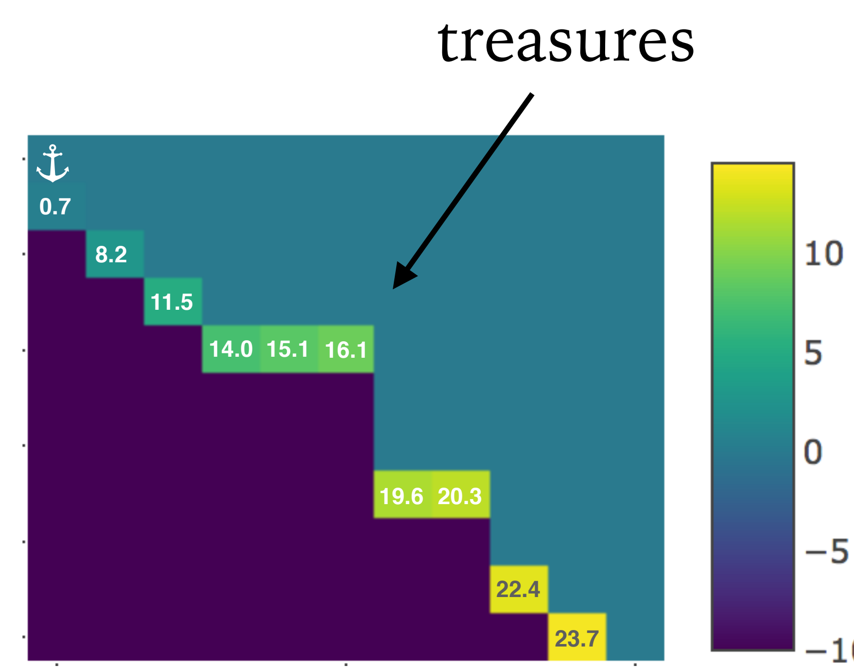
Comparison of CCS and control frontiers of deep MORL algorithms. Both figures are measured on a fruit tree navigation task of the depth 6 containing total 64 solutions. The figure (a.) visualizes the real CCS and retrieved CCS of naive and envelope MORL algorithms using t-SNE. The figure (b.) presents the slices of optimal control frontier and the control frontier of two MORL algorithms along the Mineral-Waters plane.

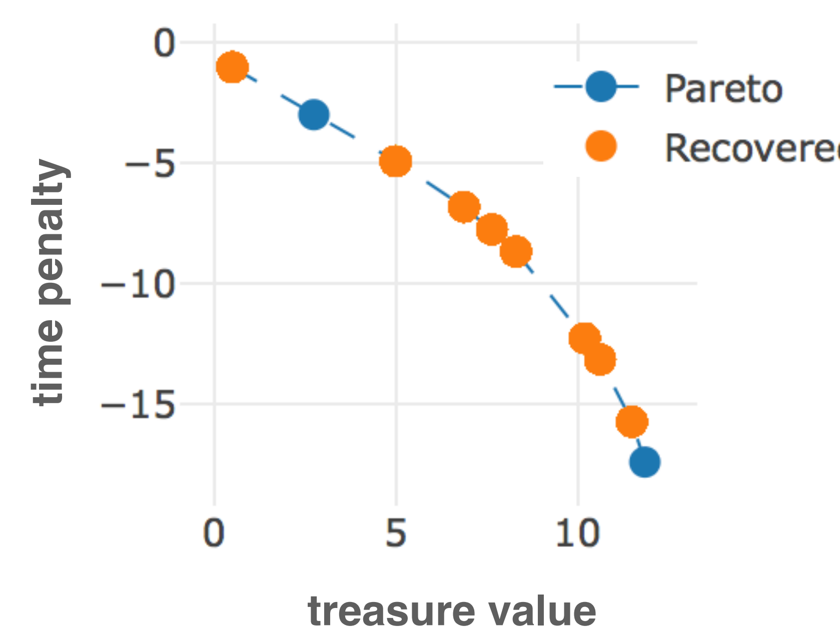
Evaluation - Synthetic Environments

More Environments…



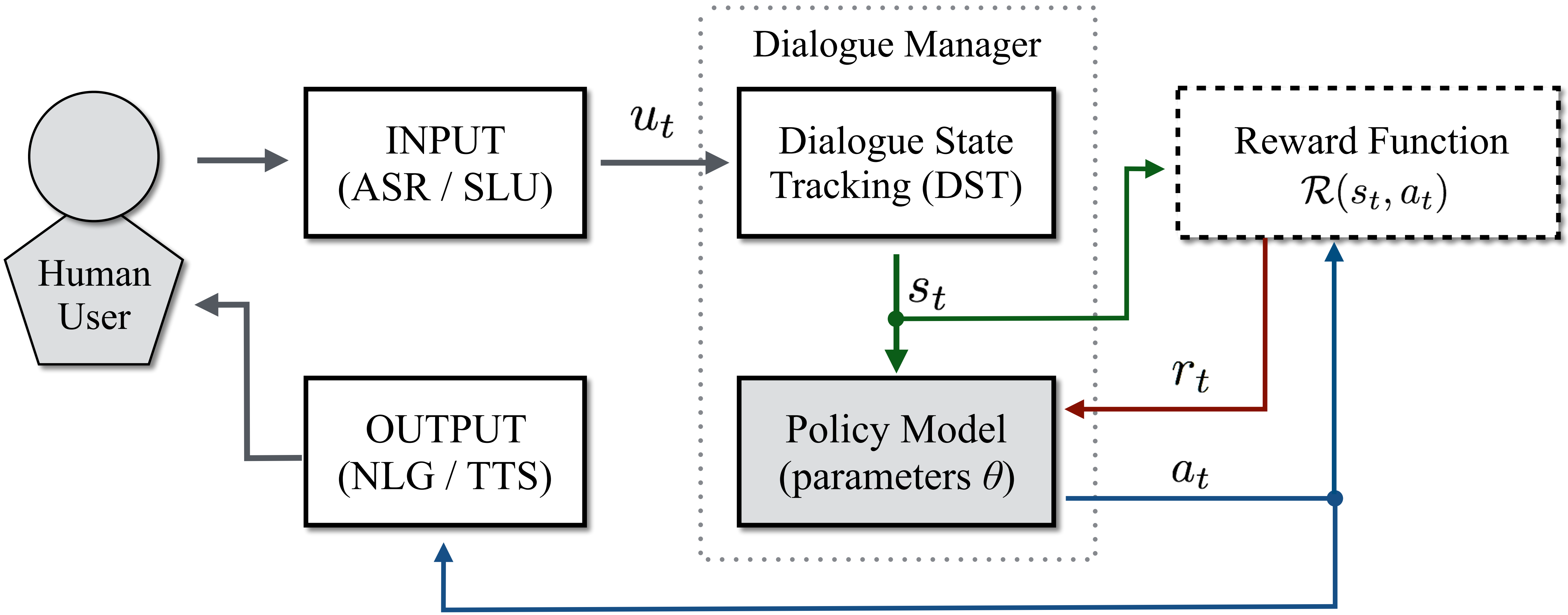
Multi-Objective Lunar Lander
(fuel/speed/height)





Deep Sea Treasure (DST)
(time/treasure)

Application - Task-Oriented Dialogue Systems



The RL-Based framework of task-oriented dialogue systems.

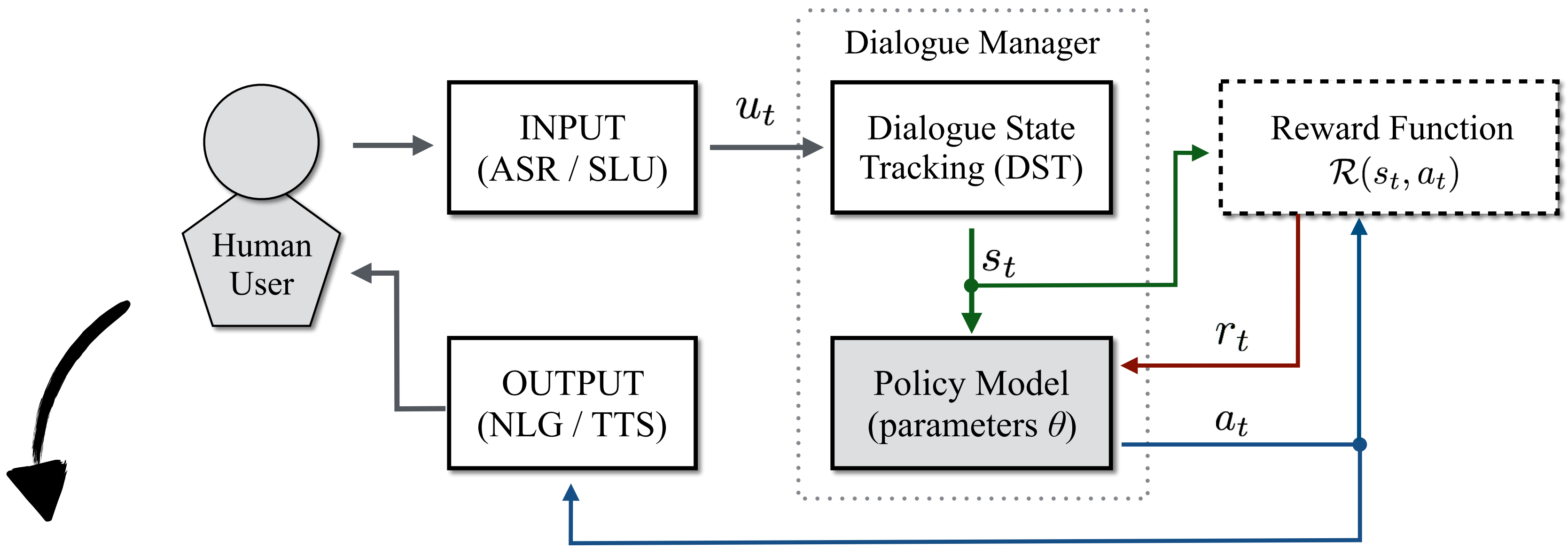
Reward Function:

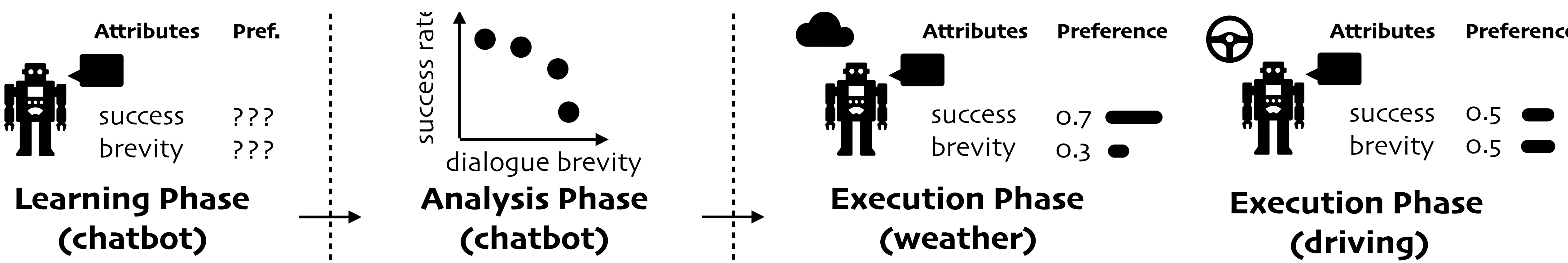
$$r_t = 0.5 \cdot r_t^{\texttt{turn}} + 0.5 \cdot r_t^{\texttt{succ}}$$

Objective 1 - **Dialogue brevity**: users prefer shorter dialogue.

Objective 2 - **Dialogue success**: users get correct responses.

Application - Task-Oriented Dialogue Systems





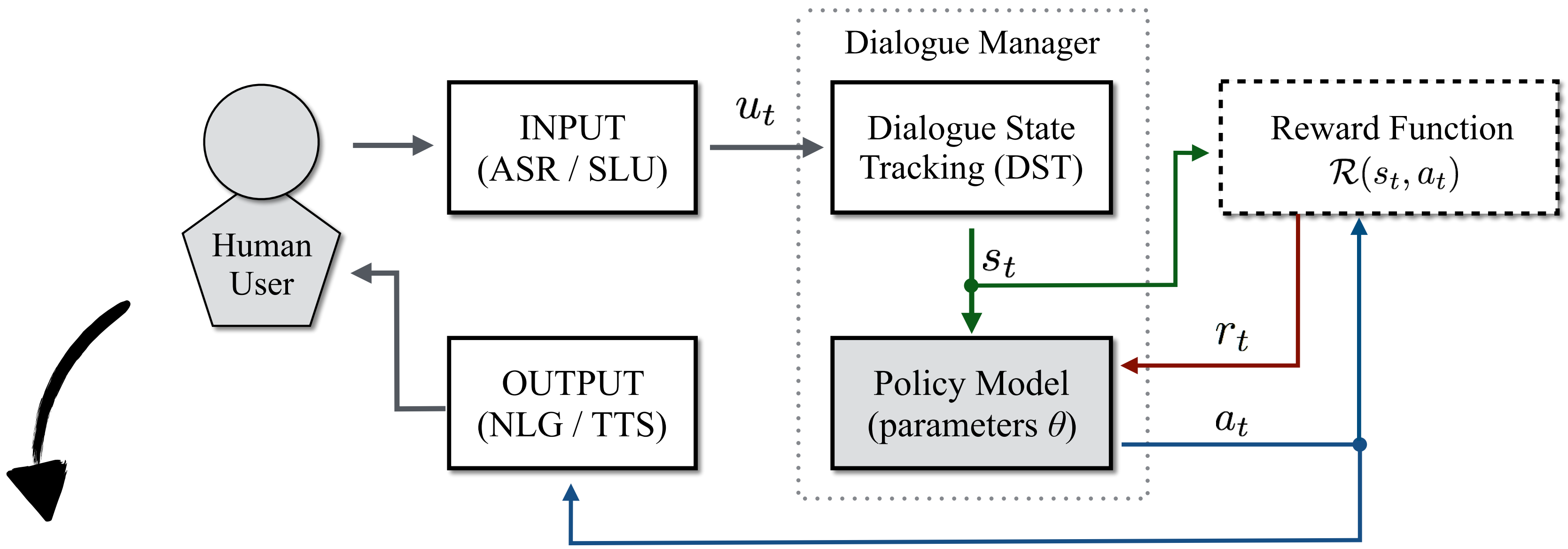
Reward Function:

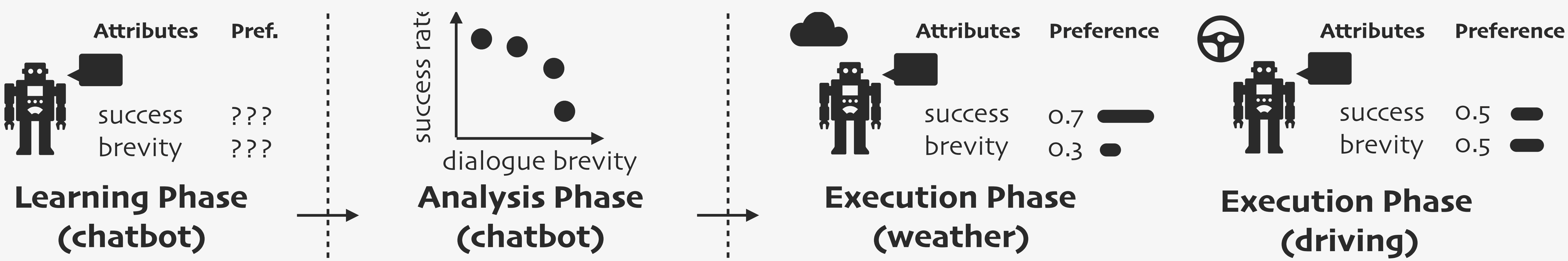
$$r_t = \begin{bmatrix} r_t^{\text{turn}} & r_t^{\text{succ}} \end{bmatrix}^\mathsf{T}$$

Objective 1 - **Dialogue brevity**: users prefer shorter dialogue.

Objective 2 - **Dialogue success**: users get correct responses.

Application - Task-Oriented Dialogue Systems





Reward Function:

$$r_t = \begin{bmatrix} r_t^{\texttt{turn}} & r_t^{\texttt{succ}} \end{bmatrix}^\top$$

Delayed Linear Preference Scenarios

Objective 1 - **Dialogue brevity**: users prefer shorter dialogue.

Objective 2 - **Dialogue success**: users get correct responses.

Application - Task-Oriented Dialogue Systems

Experimental Settings:

PyDial Agenda-base user simulator with an **error model**

error rate = 15%

Application - Task-Oriented Dialogue Systems

Experimental Settings:

PyDial Agenda-base user simulator with an **error model**

error rate = 15%

The **turn reward = -1** for each turn, and the **success reward = 20**.
The maximal length of dialogue is 25.

Application - Task-Oriented Dialogue Systems

Experimental Settings:

PyDial Agenda-base user simulator with an **error model**

error rate = 15%

The **turn reward = -1** for each turn, and the **success reward = 20**.
The maximal length of dialogue is 25.

All the single-objective and multi-objective reinforcement learning are **trained for 3,000 sessions.**

Application - Task-Oriented Dialogue Systems

Experimental Settings:

PyDial Agenda-base user simulator with an **error model**

error rate = 15%

The **turn reward = -1** for each turn, and the **success reward = 20**.
The maximal length of dialogue is 25.

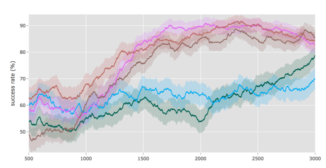
All the single-objective and multi-objective reinforcement learning are **trained for 3,000 sessions.**

We evaluate learned policies on **5,000 sessions** with **near-uniformly randomly assigned user preferences**.

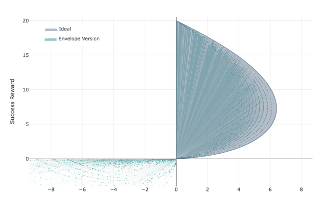
Application - Task-Oriented Dialogue Systems

Experimental Goals:

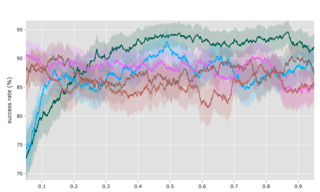
To investigate the applicability of our proposed deep MORL algorithms in task-oriented dialogue policy learning.



- first, how will the multi-objective reinforcement learning affect **the efficiency of training process**?

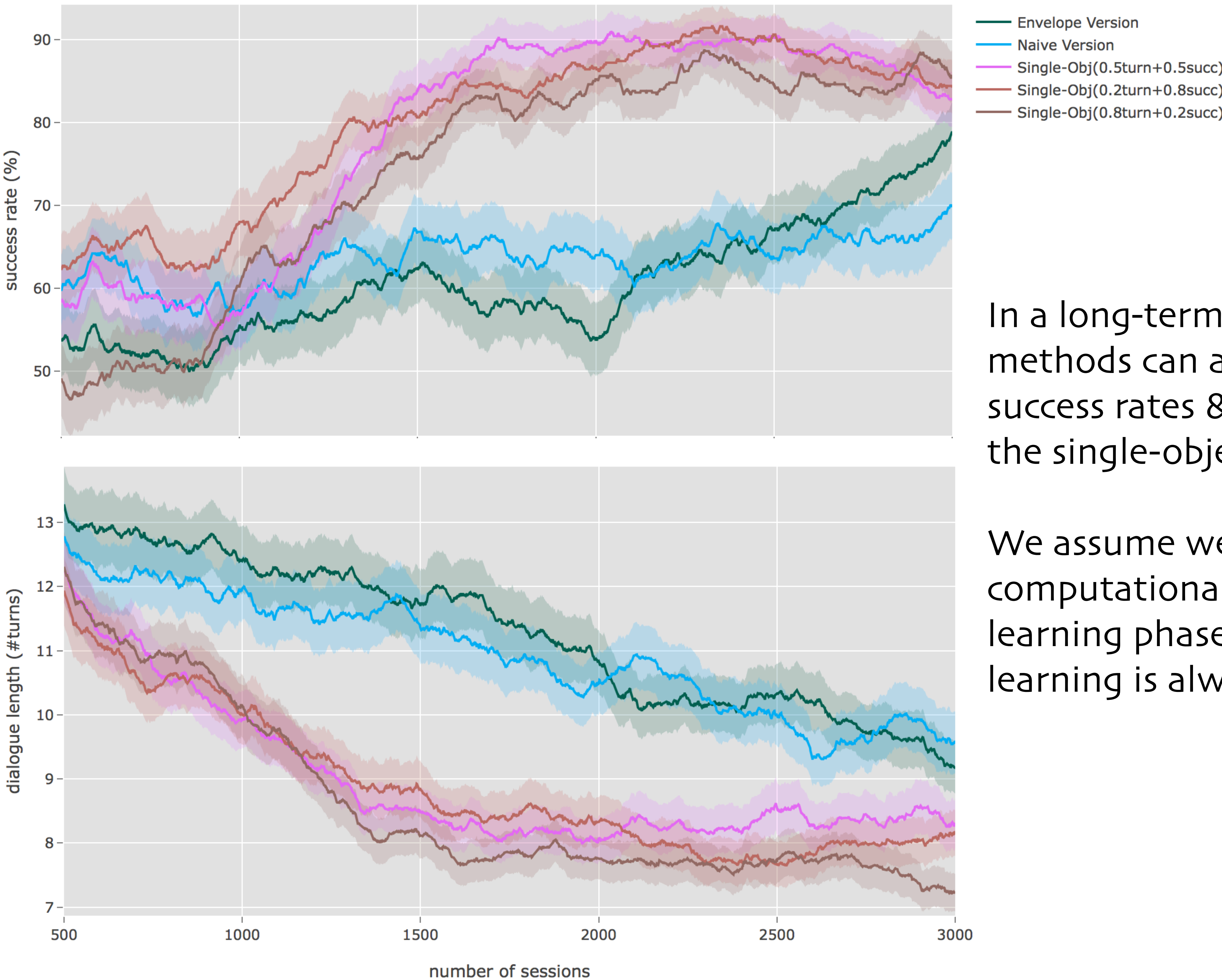


- Second, what is the **optimality frontier for the brevity and success objectives** in a dialogue application?



- Third, how do our proposed deep MORL algorithms **better fit users' preferences**?

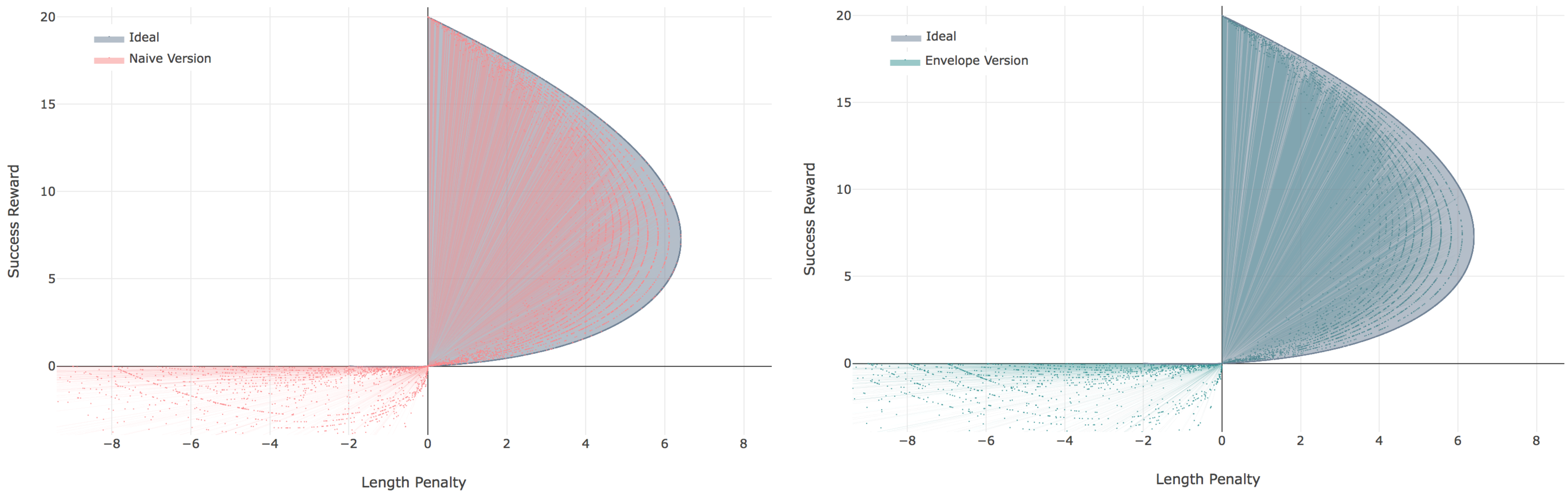
Application - Task-Oriented Dialogue Systems



In a long-term, the multi-objective methods can achieve competitive success rates & dialogue length to the single-objective methods.

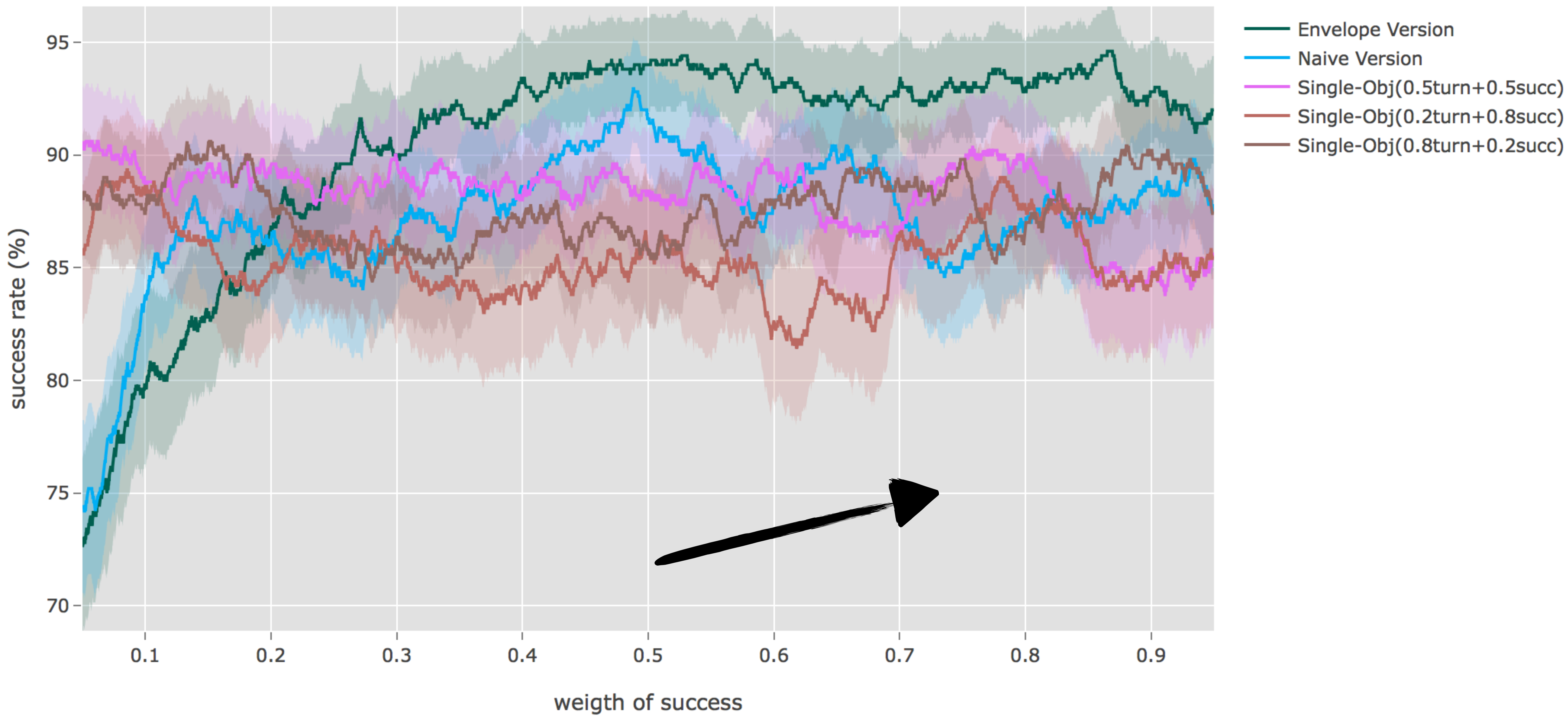
We assume we have abundant computational resources in the learning phase, and off-policy learning is always available.

Application - Task-Oriented Dialogue Systems



	Single-(0.5,0.5)	Single-(0.2,0.8)	Single-(0.8,0.2)	Naive	Envelope
Success Rate	88.18 ± 0.90	85.30 ± 0.98	87.62 ± 0.91	86.38 ± 0.95	**89.52 ± 0.85**
# Turns	8.93 ± 0.13	9.40 ± 0.16	**7.42 ± 0.10**	8.08 ± 0.12	8.08 ± 0.12
User Utility	2.13 ± 0.23	1.84 ± 0.23	2.53 ± 0.22	2.38 ± 0.22	**2.65 ± 0.22**
AQ (α = 0.1)	0.660	0.279	0.728	0.614	**0.814**

Application - Task-Oriented Dialogue Systems

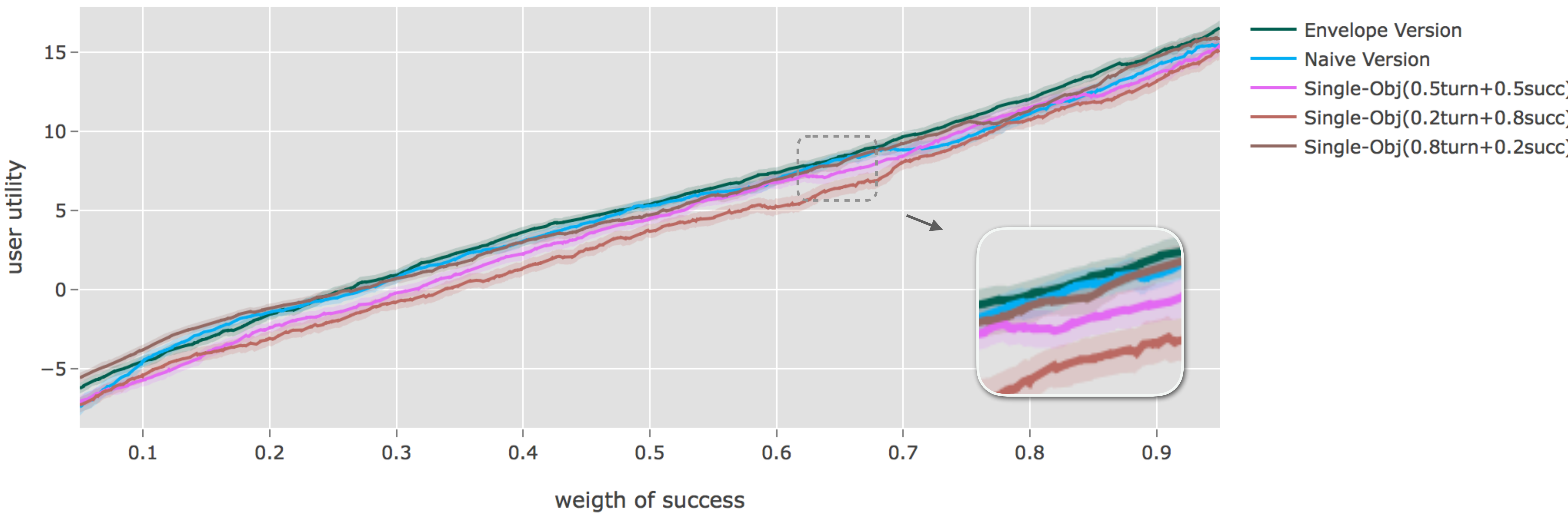


Our MORL methods can adapt to the user's preference, while the single-objective methods cannot.

Application - Task-Oriented Dialogue Systems

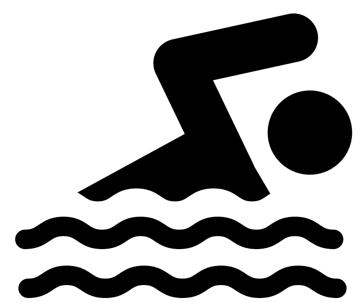
When the length of the dialogue is not important, our MORL algorithms can sacrifice a bit brevity to ensure the success rate is above 90%

Application - Task-Oriented Dialogue Systems

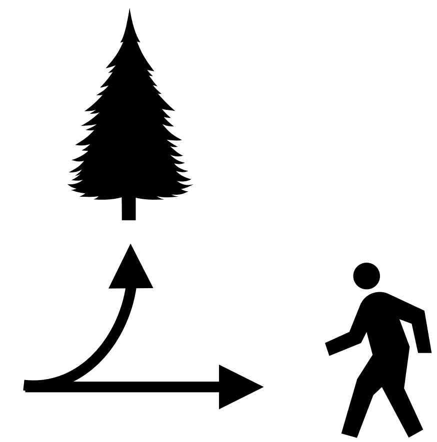


The envelope deep MORL algorithm is almost always better than other methods in terms of utility, and the naive version keeps a good level of utility under almost all user preferences. Single-objective methods are good only when the user's weight of success is close to their fixed preferences while training.

Conclusion



Multiple Competing Objectives



Human Preferences

Can we design an efficient learning algorithm, which learns **all potentially optimal policies**, and adapts optimally to any real-time specified **preference**?

Conclusion

0. Background

- Reinforcement Learning
- Problem Formulation
  - MO-MDPs
  - Optimality Concepts
  - Delayed Linear Preference Scenarios

1. Theory Contributions

- Theoretical Framework for Value-Based RL
- Two Value-Based Deep MORL Algorithms
  - Naive Version: A simple extension
  - Envelope Version

Yes We Can!

2. Evaluation Contributions

- Quantitive Evaluation Metrics
  - Coverage Ratio
  - Adaptation Quality
- Synthetic Environments

3. Application Contributions

- Task-Oriented Dialogue Systems
- RL-Based Dialogue Policy Learning
  - Objectives: Brevity v.s. Success
  - User Adaptive Policies