Budgeted Reinforcement Learning in Continuous State Space

Nicolas Carrara [3] Edouard Leurent[3,4] Tanguy Urvoy [1]
Romain Laroche [2] Odalric-Ambrym Maillard [3] Olivier Pietquin [3,4]

[1]Orange Labs

[2]Microsoft Montréal.

[3]Univ. Lille, CNRS, Centrale Lille, INRIA UMR 9189 - CRIStAL

[4]Google Research, Brain Team, Paris

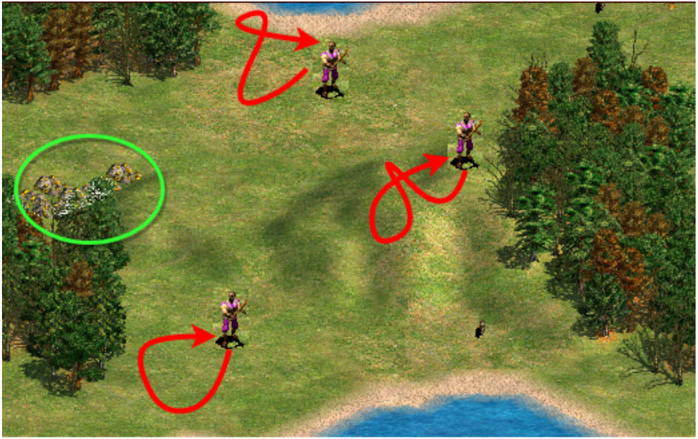
[4]Renault

10 august 2018.

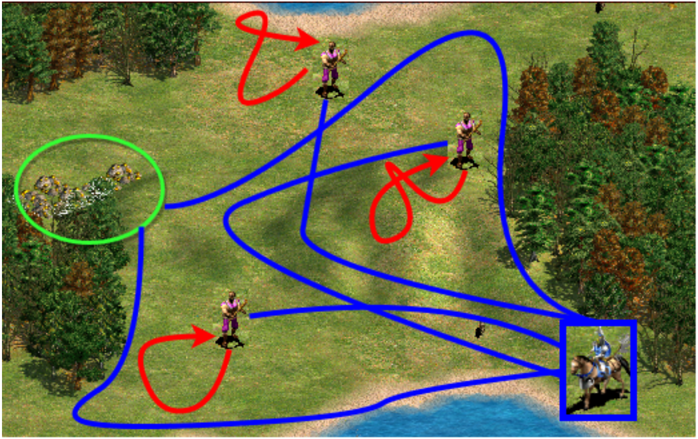
Introduction — Use-case

Introduction — Use-case

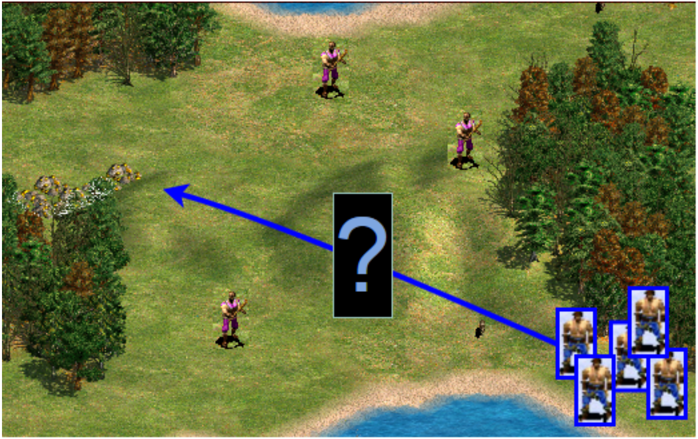
Introduction — Use-case



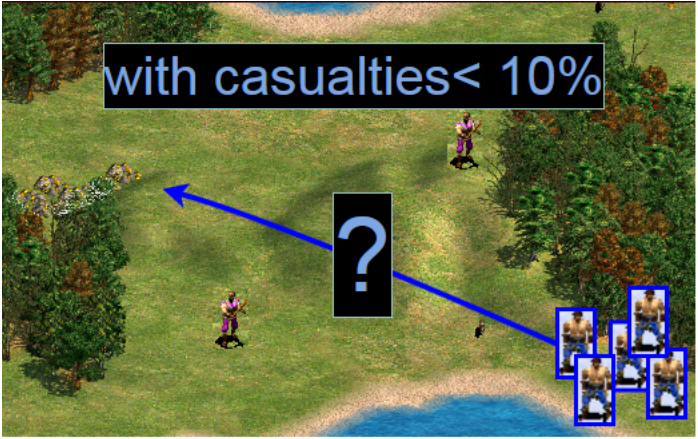
Introduction — Use-case



Introduction — Use-case



Introduction — Use-case



Introduction — Use-case

Problem

Given past trajectories, find a way to:

Introduction — Use-case

Problem

Given past trajectories, find a way to:

- gather gold as much as possible;

Introduction — Use-case

Problem

Given past trajectories, find a way to:

- gather gold as much as possible;
- limit the villager casualties under some budget;

Introduction — Use-case

Problem

Given past trajectories, find a way to:

- gather gold as much as possible;
- limit the villager casualties under some budget;
- being able to change the budget in real time.

Introduction — Use-case

Problem

Given past trajectories, find a way to:

- gather gold as much as possible;
- limit the villager casualties under some budget;
- being able to change the budget in real time.

Solution

- This problem can be cast as a Budgeted Markov Decision Process.

Setting

Markov Decision Process

We define a MDP as a tuple $(\mathcal{S}, \mathcal{A}, P, R_r, \gamma)$ where:

Setting

Markov Decision Process

We define a MDP as a tuple $(\mathcal{S}, \mathcal{A}, P, R_r, \gamma)$ where:

- $\mathcal{S}$ is the state space, $\mathcal{A}$ the action space,

Setting

Markov Decision Process

We define a MDP as a tuple $(\mathcal{S}, \mathcal{A}, P, R_r, \gamma)$ where:

- $\mathcal{S}$ is the state space, $\mathcal{A}$ the action space,
- $R_r \in \mathbb{R}^{\mathcal{S}\times\mathcal{A}}$ the rewards,

Setting

Markov Decision Process

We define a MDP as a tuple $(\mathcal{S}, \mathcal{A}, P, R_r, \gamma)$ where:

- $\mathcal{S}$ is the state space, $\mathcal{A}$ the action space,
- $R_r \in \mathbb{R}^{\mathcal{S}\times\mathcal{A}}$ the rewards,
- $P \in \mathcal{M}(\mathcal{S})^{\mathcal{S}\times\mathcal{A}}$ the dynamics,

Setting

Markov Decision Process

We define a MDP as a tuple $(\mathcal{S}, \mathcal{A}, P, R_r, \gamma)$ where:

- $\mathcal{S}$ is the state space, $\mathcal{A}$ the action space,
- $R_r \in \mathbb{R}^{\mathcal{S}\times\mathcal{A}}$ the rewards,
- $P \in \mathcal{M}(\mathcal{S})^{\mathcal{S}\times\mathcal{A}}$ the dynamics,
- and γ the discounted factor.

Setting

Markov Decision Process

We define a MDP as a tuple $(\mathcal{S}, \mathcal{A}, P, R_r, \gamma)$ where:

- $\mathcal{S}$ is the state space, $\mathcal{A}$ the action space,
- $R_r \in \mathbb{R}^{\mathcal{S}\times\mathcal{A}}$ the rewards,
- $P \in \mathcal{M}(\mathcal{S})^{\mathcal{S}\times\mathcal{A}}$ the dynamics,
- and γ the discounted factor.

Objective

Setting

Markov Decision Process

We define a MDP as a tuple $(\mathcal{S}, \mathcal{A}, P, R_r, \gamma)$ where:

- $\mathcal{S}$ is the state space, $\mathcal{A}$ the action space,
- $R_r \in \mathbb{R}^{\mathcal{S}\times\mathcal{A}}$ the rewards,
- $P \in \mathcal{M}(\mathcal{S})^{\mathcal{S}\times\mathcal{A}}$ the dynamics,
- and γ the discounted factor.

Objective

- $G_r^\pi = \sum_{t=0}^{\infty} \gamma^t R_r(s_t, a_t)$ the γ-discounted return of rewards.

Setting

Markov Decision Process

We define a MDP as a tuple $(\mathcal{S}, \mathcal{A}, P, R_r, \gamma)$ where:

- $\mathcal{S}$ is the state space, $\mathcal{A}$ the action space,
- $R_r \in \mathbb{R}^{\mathcal{S}\times\mathcal{A}}$ the rewards,
- $P \in \mathcal{M}(\mathcal{S})^{\mathcal{S}\times\mathcal{A}}$ the dynamics,
- and γ the discounted factor.

Objective

- $G_r^\pi = \sum_{t=0}^{\infty} \gamma^t R_r(s_t, a_t)$ the γ-discounted return of rewards.
- Find π^* s.t $\forall s \in \mathcal{S}$:

$$\pi^* \in \underset{\pi \in \mathcal{M}(\mathcal{A})^{\mathcal{S}}}{\arg\max}\ \mathbb{E}[G_r^\pi | s_0 = s] \tag{1}$$

Setting

Constrained Markov Decision Process

We define a CMDP as a tuple $(\mathcal{S}, \mathcal{A}, P, R_r, R_c, \gamma, \beta)$ where:

Setting

Constrained Markov Decision Process

We define a CMDP as a tuple $(\mathcal{S}, \mathcal{A}, P, R_r, R_c, \gamma, \beta)$ where:

- $\mathcal{S}$ is the state space, $\mathcal{A}$ the action space,

Setting

Constrained Markov Decision Process

We define a CMDP as a tuple $(\mathcal{S}, \mathcal{A}, P, R_r, R_c, \gamma, \beta)$ where:

- $\mathcal{S}$ is the state space, $\mathcal{A}$ the action space,
- $R_r \in \mathbb{R}^{\mathcal{S}\times\mathcal{A}}$ the rewards, and $R_c \in \mathbb{R}^{\mathcal{S}\times\mathcal{A}}$ the costs

Setting

Constrained Markov Decision Process

We define a CMDP as a tuple $(\mathcal{S}, \mathcal{A}, P, R_r, R_c, \gamma, \beta)$ where:

- $\mathcal{S}$ is the state space, $\mathcal{A}$ the action space,
- $R_r \in \mathbb{R}^{\mathcal{S}\times\mathcal{A}}$ the rewards, and $R_c \in \mathbb{R}^{\mathcal{S}\times\mathcal{A}}$ the costs
- $P \in \mathcal{M}(\mathcal{S})^{\mathcal{S}\times\mathcal{A}}$ the dynamics,

Setting

Constrained Markov Decision Process

We define a CMDP as a tuple $(\mathcal{S}, \mathcal{A}, P, R_r, R_c, \gamma, \beta)$ where:

- $\mathcal{S}$ is the state space, $\mathcal{A}$ the action space,
- $R_r \in \mathbb{R}^{\mathcal{S}\times\mathcal{A}}$ the rewards, and $R_c \in \mathbb{R}^{\mathcal{S}\times\mathcal{A}}$ the costs
- $P \in \mathcal{M}(\mathcal{S})^{\mathcal{S}\times\mathcal{A}}$ the dynamics,
- γ the discounted factor, and β the budget.

Objective

Setting

Constrained Markov Decision Process

We define a CMDP as a tuple $(\mathcal{S}, \mathcal{A}, P, R_r, R_c, \gamma, \beta)$ where:

- $\mathcal{S}$ is the state space, $\mathcal{A}$ the action space,
- $R_r \in \mathbb{R}^{\mathcal{S}\times\mathcal{A}}$ the rewards, and $R_c \in \mathbb{R}^{\mathcal{S}\times\mathcal{A}}$ the costs
- $P \in \mathcal{M}(\mathcal{S})^{\mathcal{S}\times\mathcal{A}}$ the dynamics,
- γ the discounted factor, and β the budget.

Objective

- $G_r^\pi = \sum_{t=0}^{\infty} \gamma^t R_r(s_t, a_t)$ the γ-discounted return of rewards.

Setting

Constrained Markov Decision Process

We define a CMDP as a tuple $(\mathcal{S}, \mathcal{A}, P, R_r, R_c, \gamma, \beta)$ where:

- $\mathcal{S}$ is the state space, $\mathcal{A}$ the action space,
- $R_r \in \mathbb{R}^{\mathcal{S}\times\mathcal{A}}$ the rewards, and $R_c \in \mathbb{R}^{\mathcal{S}\times\mathcal{A}}$ the costs
- $P \in \mathcal{M}(\mathcal{S})^{\mathcal{S}\times\mathcal{A}}$ the dynamics,
- γ the discounted factor, and β the budget.

Objective

- $G_r^\pi = \sum_{t=0}^\infty \gamma^t R_r(s_t, a_t)$ the γ-discounted return of rewards.
- $G_c^\pi = \sum_{t=0}^\infty \gamma^t R_c(s_t, a_t)$ the γ-discounted return of costs.

Setting

Constrained Markov Decision Process

We define a CMDP as a tuple $(\mathcal{S}, \mathcal{A}, P, R_r, R_c, \gamma, \beta)$ where:

- $\mathcal{S}$ is the state space, $\mathcal{A}$ the action space,
- $R_r \in \mathbb{R}^{\mathcal{S}\times\mathcal{A}}$ the rewards, and $R_c \in \mathbb{R}^{\mathcal{S}\times\mathcal{A}}$ the costs
- $P \in \mathcal{M}(\mathcal{S})^{\mathcal{S}\times\mathcal{A}}$ the dynamics,
- γ the discounted factor, and β the budget.

Objective

- $G_r^\pi = \sum_{t=0}^\infty \gamma^t R_r(s_t, a_t)$ the γ-discounted return of rewards.
- $G_c^\pi = \sum_{t=0}^\infty \gamma^t R_c(s_t, a_t)$ the γ-discounted return of costs.
- Find π^* s.t $\forall s \in \mathcal{S}$:

$$\begin{aligned} \pi^* \in \; & \underset{\pi \in \mathcal{M}(\mathcal{A})^{\mathcal{S}}}{\arg\max} \; \mathbb{E}[G_r^\pi | s_0 = s] \\ \text{s.t. } & \mathbb{E}[G_c^\pi | s_0 = s] \leq \beta \end{aligned} \tag{2}$$

Setting

Budgeted Markov Decision Process

We define a BMDP as a tuple $(\mathcal{S}, \mathcal{A}, P, R_r, R_c, \gamma, \mathcal{B})$ where:

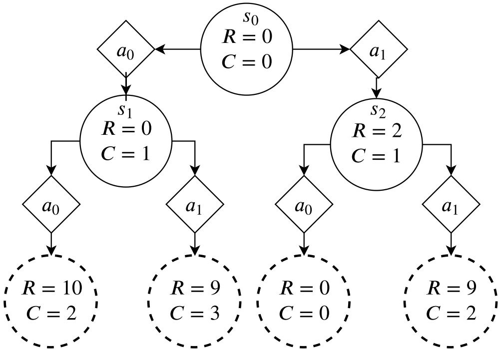
- $\mathcal{S}$ is the state space, $\mathcal{A}$ the action space,
- $R_r \in \mathbb{R}^{\mathcal{S}\times\mathcal{A}}$ the rewards, and $R_c \in \mathbb{R}^{\mathcal{S}\times\mathcal{A}}$ the costs
- $P \in \mathcal{M}(\mathcal{S})^{\mathcal{S}\times\mathcal{A}}$ the dynamics,
- γ the discounted factor, and $\mathcal{B}$ the budget space.

Objective

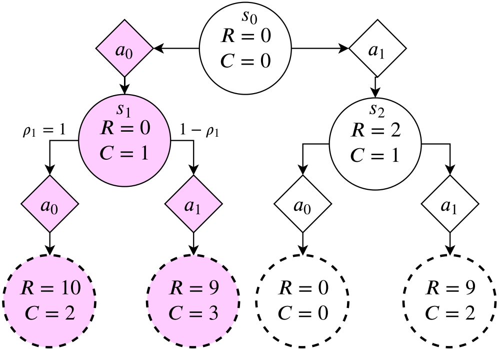
- $G_r^\pi = \sum_{t=0}^\infty \gamma^t R_r(s_t, a_t)$ the γ-discounted return of rewards.
- $G_c^\pi = \sum_{t=0}^\infty \gamma^t R_c(s_t, a_t)$ the γ-discounted return of costs.
- Find π^* s.t $\forall (s, \beta) \in \mathcal{S} \times \mathcal{B}$:

$$\begin{aligned} \pi^* \in \; & \mathop{\arg\max}_{\pi \in \mathcal{M}(\mathcal{A}\times\mathcal{B})^{\mathcal{S}\times\mathcal{B}}} \mathbb{E}[G_r^\pi | s_0 = s, \beta_0 = \beta] \\ \text{s.t.} \; & \mathbb{E}[G_c^\pi | s_0 = s, \beta_0 = \beta] \le \beta \end{aligned} \tag{3}$$

The Dynamic Programming solution: intuition

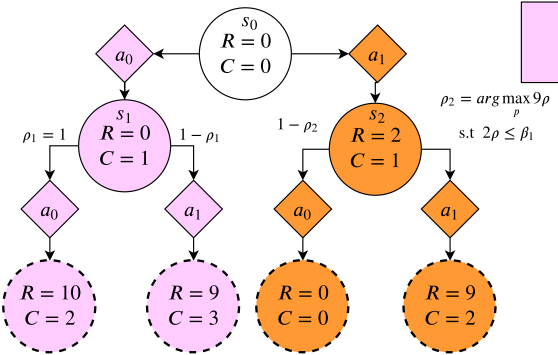


The Dynamic Programming solution: intuition



$r = 10$
$c = 2$

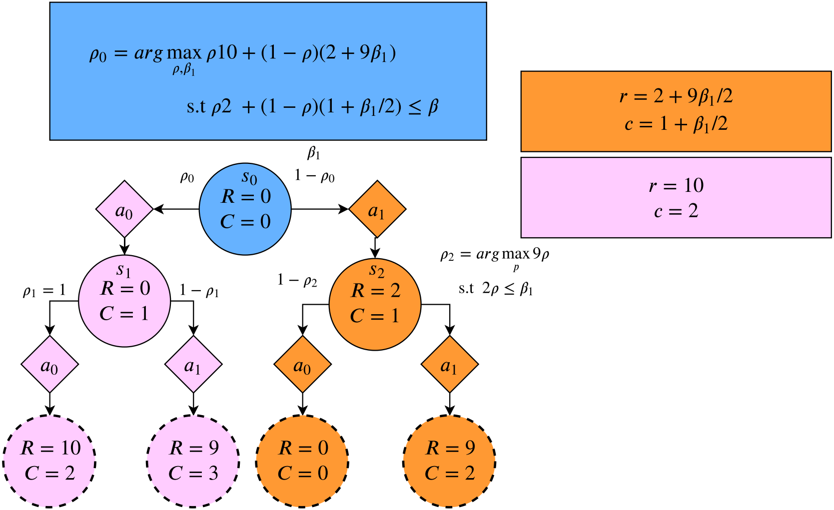
The Dynamic Programming solution: intuition



$r = 2 + 9\beta_1/2$
$c = 1 + \beta_1/2$

$r = 10$
$c = 2$

The Dynamic Programming solution: intuition



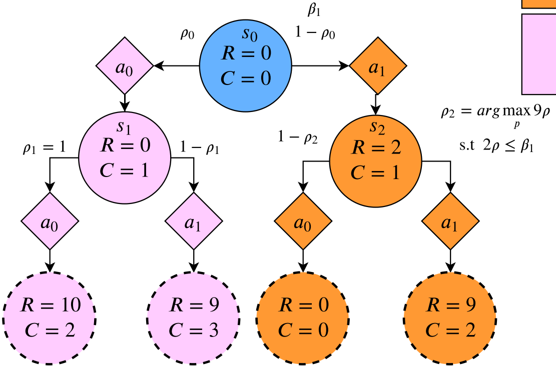
The Dynamic Programming solution: intuition

$$\rho_0 = arg\max_{\rho,\beta_1} \rho 10 + (1-\rho)(2+9\beta_1)$$

$$\text{s.t } \rho 2 + (1-\rho)(1+\beta_1/2) \leq \beta$$

$r = 2 + 9\beta_1/2$
$c = 1 + \beta_1/2$

$r = 10$
$c = 2$



Augmented Settings

Budgeted policies π

Augmented Settings

Budgeted policies π

- Take a budget β as an additional input

Augmented Settings

Budgeted policies π

- Take a budget β as an additional input
- Output a next budget β'

Augmented Settings

Budgeted policies π

- Take a budget β as an additional input
- Output a next budget β'
- $\pi : \underbrace{(s, \beta)}_{\bar{s}} \rightarrow \underbrace{(a, \beta')}_{\bar{a}}$

Domain

Augmented Settings

Budgeted policies π

- Take a budget β as an additional input
- Output a next budget β'
- $\pi : \underbrace{(s, \beta)}_{\bar{s}} \rightarrow \underbrace{(a, \beta')}_{\bar{a}}$

Domain

- States $\overline{\mathcal{S}} = \mathcal{S} \times \mathcal{B}$.

Augmented Settings

Budgeted policies π

- Take a budget β as an additional input
- Output a next budget β'
- $\pi : \underbrace{(s, \beta)}_{\overline{s}} \to \underbrace{(a, \beta')}_{\overline{a}}$

Domain

- States $\overline{\mathcal{S}} = \mathcal{S} \times \mathcal{B}$.
- Actions $\overline{\mathcal{A}} = \mathcal{A} \times \mathcal{B}$.

Augmented Settings

Budgeted policies π

- Take a budget β as an additional input
- Output a next budget β'
- $\pi : \underbrace{(s, \beta)}_{\bar{s}} \to \underbrace{(a, \beta')}_{\bar{a}}$

Domain

- States $\overline{\mathcal{S}} = \mathcal{S} \times \mathcal{B}$.
- Actions $\overline{\mathcal{A}} = \mathcal{A} \times \mathcal{B}$.
- Dynamics $\overline{P}\left((s', \beta') \mid (s, \beta), (a, \beta_a)\right) \stackrel{\text{def}}{=} P(s'|s, a)\delta(\beta' - \beta_a)$.

2D signals

Augmented Settings

Budgeted policies π

- Take a budget β as an additional input
- Output a next budget β'
- $\pi : \underbrace{(s, \beta)}_{\overline{s}} \rightarrow \underbrace{(a, \beta')}_{\overline{a}}$

Domain

- States $\overline{\mathcal{S}} = \mathcal{S} \times \mathcal{B}$.
- Actions $\overline{\mathcal{A}} = \mathcal{A} \times \mathcal{B}$.
- Dynamics $\overline{P}\left((s', \beta') \mid (s, \beta), (a, \beta_a)\right) \stackrel{\text{def}}{=} P(s'|s, a)\delta(\beta' - \beta_a)$.

2D signals

1. Rewards $R = (R_r, R_c)$

Augmented Settings

Budgeted policies π

- Take a budget β as an additional input
- Output a next budget β'
- $\pi : \underbrace{(s, \beta)}_{\overline{s}} \to \underbrace{(a, \beta')}_{\overline{a}}$

Domain

- States $\overline{\mathcal{S}} = \mathcal{S} \times \mathcal{B}$.
- Actions $\overline{\mathcal{A}} = \mathcal{A} \times \mathcal{B}$.
- Dynamics $\overline{P}\left((s', \beta') \mid (s, \beta), (a, \beta_a)\right) \overset{\text{def}}{=} P(s'|s, a)\delta(\beta' - \beta_a)$.

2D signals

1. Rewards $R = (R_r, R_c)$
2. Returns $G^\pi = (G_r^\pi, G_c^\pi)$

Augmented Settings

Budgeted policies π

- Take a budget β as an additional input
- Output a next budget β'
- $\pi : \underbrace{(s, \beta)}_{\overline{s}} \rightarrow \underbrace{(a, \beta')}_{\overline{a}}$

Domain

- States $\overline{\mathcal{S}} = \mathcal{S} \times \mathcal{B}$.
- Actions $\overline{\mathcal{A}} = \mathcal{A} \times \mathcal{B}$.
- Dynamics $\overline{P}\left((s', \beta') \mid (s, \beta), (a, \beta_a)\right) \stackrel{\text{def}}{=} P(s'|s, a)\delta(\beta' - \beta_a)$.

2D signals

1. Rewards $R = (R_r, R_c)$
2. Returns $G^\pi = (G_r^\pi, G_c^\pi)$
3. $V^\pi(\overline{s}) = (V_r^\pi, V_c^\pi) \stackrel{\text{def}}{=} \mathbb{E}\left[G^\pi \mid \overline{s_0} = \overline{s}\right]$

Augmented Settings

Budgeted policies π

- Take a budget β as an additional input
- Output a next budget β'
- $\pi : \underbrace{(s, \beta)}_{\overline{s}} \to \underbrace{(a, \beta')}_{\overline{a}}$

Domain

- States $\overline{\mathcal{S}} = \mathcal{S} \times \mathcal{B}$.
- Actions $\overline{\mathcal{A}} = \mathcal{A} \times \mathcal{B}$.
- Dynamics $\overline{P}\left((s', \beta') \mid (s, \beta), (a, \beta_a)\right) \stackrel{\text{def}}{=} P(s'|s, a)\delta(\beta' - \beta_a)$.

2D signals

1. Rewards $R = (R_r, R_c)$
2. Returns $G^\pi = (G_r^\pi, G_c^\pi)$
3. $V^\pi(\overline{s}) = (V_r^\pi, V_c^\pi) \stackrel{\text{def}}{=} \mathbb{E}\left[G^\pi \mid \overline{s_0} = \overline{s}\right]$
4. $Q^\pi(\overline{s}, \overline{a}) = (Q_r^\pi, Q_c^\pi) \stackrel{\text{def}}{=} \mathbb{E}\left[G^\pi \mid \overline{s_0} = \overline{s}, \overline{a_0} = \overline{a}\right]$

Policy Evaluation

The Bellman Expectation equationsare preserved, and the Bellman Expectation Operator $\mathcal{T}^\pi$ is a γ-contraction.

Augmented Optimality

Definition

In that order, we want to:

Augmented Optimality

Definition

In that order, we want to:

(i) Respect the budget β:

$$\Pi_a(\overline{s}) \stackrel{\text{def}}{=} \{\pi \in \Pi : V_c^{\pi}(s, \beta) \leq \beta\}$$

Augmented Optimality

Definition

In that order, we want to:

(i) Respect the budget β:

$$\Pi_a(\overline{s}) \stackrel{\text{def}}{=} \{\pi \in \Pi : V_c^\pi(s, \beta) \leq \beta\}$$

(ii) Maximise the rewards:

$$V_r^*(\overline{s}) \stackrel{\text{def}}{=} \max_{\pi \in \Pi_a(\overline{s})} V_r^\pi(\overline{s}) \qquad \Pi_r(\overline{s}) \stackrel{\text{def}}{=} \arg\max_{\pi \in \Pi_a(\overline{s})} V_r^\pi(\overline{s})$$

Augmented Optimality

Definition

In that order, we want to:

(i) Respect the budget β:

$$\Pi_a(\bar{s}) \stackrel{\text{def}}{=} \{\pi \in \Pi : V_c^\pi(s, \beta) \le \beta\}$$

(ii) Maximise the rewards:

$$V_r^*(\bar{s}) \stackrel{\text{def}}{=} \max_{\pi \in \Pi_a(\bar{s})} V_r^\pi(\bar{s}) \qquad \Pi_r(\bar{s}) \stackrel{\text{def}}{=} \arg\max_{\pi \in \Pi_a(\bar{s})} V_r^\pi(\bar{s})$$

(iii) Minimise the costs:

$$V_c^*(\bar{s}) \stackrel{\text{def}}{=} \min_{\pi \in \Pi_r(\bar{s})} V_c^\pi(\bar{s}), \qquad \Pi^*(\bar{s}) \stackrel{\text{def}}{=} \arg\min_{\pi \in \Pi_r(\bar{s})} V_c^\pi(\bar{s})$$

We define the budgeted action-value function Q^* similarly

Budgeted Bellman Optimality Equation

Theorem (Budgeted Bellman Optimality Equation)

Q^* verifies the following equation:

$$Q^*(\overline{s}, \overline{a}) = \mathcal{T} Q^*(\overline{s}, \overline{a})$$
$$\stackrel{\text{def}}{=} R(\overline{s}, \overline{a}) + \gamma \sum_{\overline{s'} \in \overline{\mathcal{S}}} \overline{P}(\overline{s'} | \overline{s}, \overline{a}) \sum_{\overline{a'} \in \overline{\mathcal{A}}} \pi_{greedy}(\overline{a'} | \overline{s'}; Q^*) Q^*(\overline{s'}, \overline{a'}),$$

where the greedy policy π_{greedy} is defined by:

$$\pi_{greedy}(\overline{a} | \overline{s}; Q) \in \boxed{\arg\min}_{\rho \in \Pi_r^Q} \mathop{\mathbb{E}}_{\overline{a} \sim \rho} Q_c(\overline{s}, \overline{a}),$$

$$\text{where} \quad \Pi_r^Q \stackrel{\text{def}}{=} \boxed{\arg\max}_{\rho \in \mathcal{M}(\overline{\mathcal{A}})} \mathop{\mathbb{E}}_{\overline{a} \sim \rho} Q_r(\overline{s}, \overline{a})$$
$$\text{s.t.} \quad \mathop{\mathbb{E}}_{\overline{a} \sim \rho} Q_c(\overline{s}, \overline{a}) \boxed{\leq \beta}$$

Optimality of the policy

Proposition (Optimality of the policy)

$\pi_{greedy}(\cdot\ ; Q^*)$ is simultaneously optimal in all states $\bar{s} \in \overline{\mathcal{S}}$:

$$\pi_{greedy}(\cdot\ ; Q^*) \in \Pi^*(\bar{s})$$

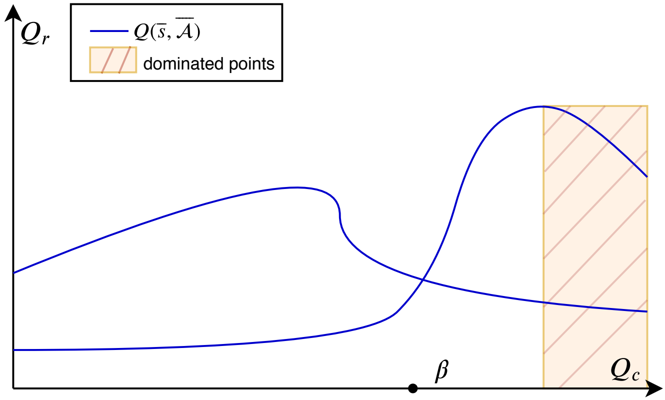
In particular, $V^{\pi_{greedy}(\cdot\,;Q^*)} = V^*$ and $Q^{\pi_{greedy}(\cdot\,;Q^*)} = Q^*$.

Solving the untractable program

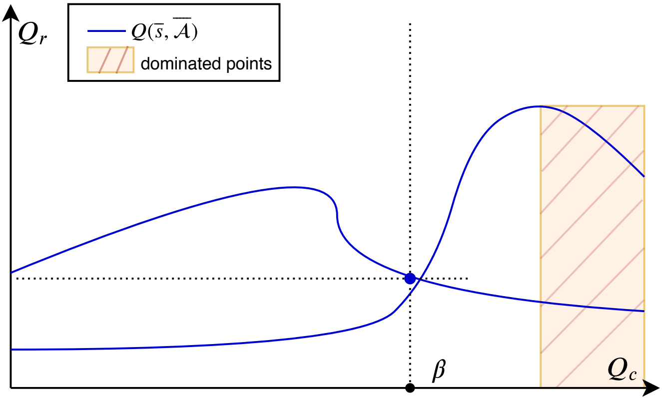
Proposition ($\pi_{\text{greedy}} = \pi_{\text{hull}}$)

π_{greedy} can be computed explicitly, as a mixture of two points that lie on the convex hull of Q.

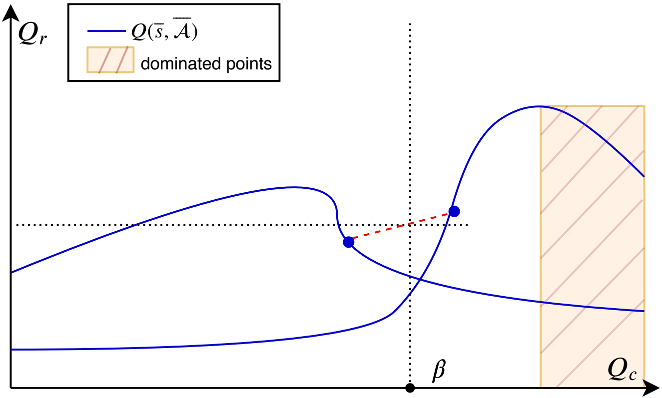
Solving the non-linear programming problem: intuition



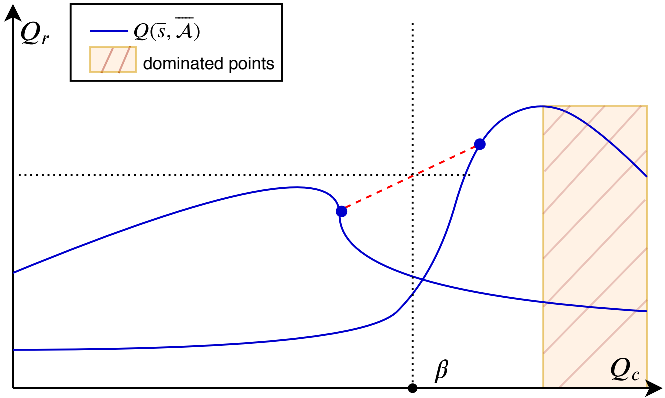
Solving the non-linear programming problem: intuition



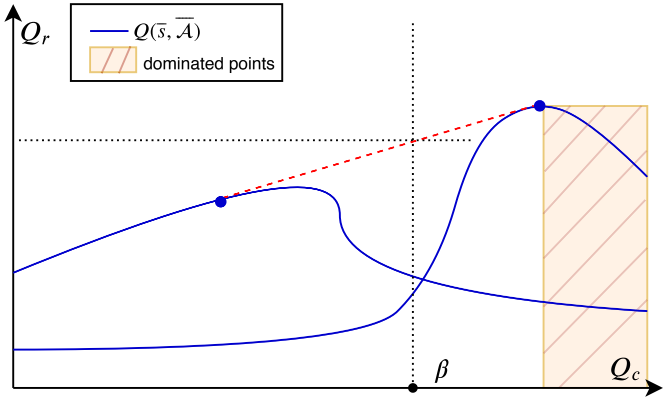
Solving the non-linear programming problem: intuition



Solving the non-linear programming problem: intuition



Solving the non-linear programming problem: intuition



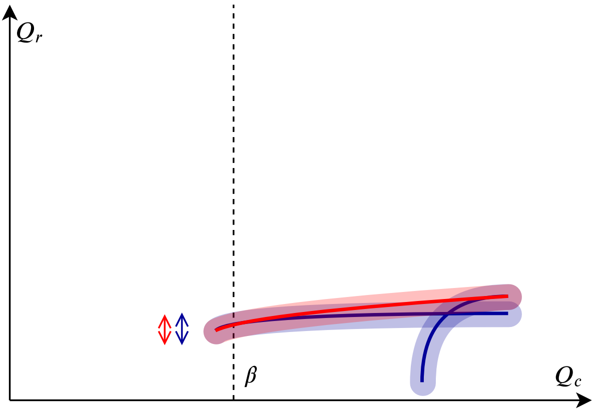
Not a contraction

Theorem (Non-Contractivity)

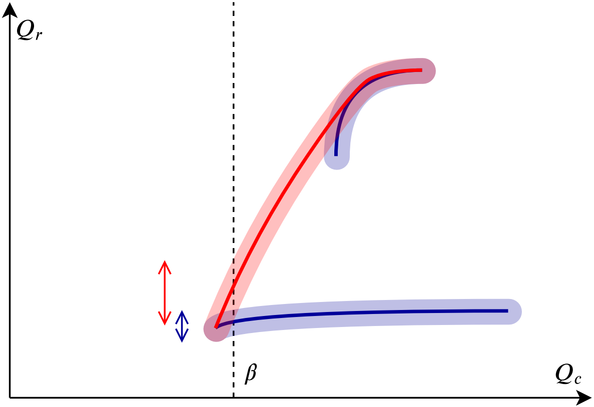
For any BMDP $(\mathcal{S}, \mathcal{A}, P, R_r, R_c, \gamma)$ with $|\mathcal{A}| \geq 2$, $\mathcal{T}$ is not a contraction.

$$\forall \varepsilon > 0, \exists Q^1, Q^2 \in (\mathbb{R}^2)^{\overline{\mathcal{S}\mathcal{A}}} : \|\mathcal{T} Q^1 - \mathcal{T} Q^2\|_\infty \geq \frac{1}{\varepsilon} \|Q^1 - Q^2\|_\infty$$

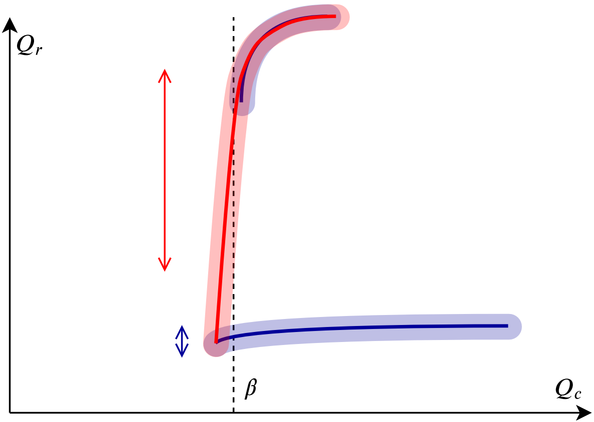
Not a contraction: intuition



Not a contraction: intuition



Not a contraction: intuition



Contractivity on smooth Q-functions

Conjecture (Contractivity $\mathcal{L}_\gamma$)

$\mathcal{T}$ is a contraction when restricted to the subset $\mathcal{L}_\gamma$ of Q-functions such that "Q_r is L-Lipschitz with respect to Q_c", with $L < \frac{1}{\gamma} - 1$.

Budgeted Dynamic Programming

Algorithm 1: Budgeted Value-Iteration

Data: P, R_r, R_c

Result: Q^*

$Q_0 \leftarrow 0$

repeat

  $Q_{k+1} \leftarrow \mathcal{T} Q_k$

until *convergence*

Budgeted Reinforcement Learning

We address several limitations of Budgeted Value-Iteration

Budgeted Reinforcement Learning

We address several limitations of Budgeted Value-Iteration

1. If the P, R_r and R_c are unknown:

Budgeted Reinforcement Learning

We address several limitations of Budgeted Value-Iteration

1. If the P, R_r and R_c are unknown:
   - Work with a batch of samples $\mathcal{D} = \{(\overline{s}_i, \overline{a}_i, r_i, \overline{s}'_i\}_{i\in[0,N]}$

Budgeted Reinforcement Learning

We address several limitations of Budgeted Value-Iteration

1. If the P, R_r and R_c are unknown:
   - Work with a batch of samples $\mathcal{D} = \{(\overline{s}_i, \overline{a}_i, r_i, \overline{s}'_i\}_{i \in [0,N]}$
   - Replace $\mathcal{T}$ with a sampling operator $\hat{\mathcal{T}}$:

$$\hat{\mathcal{T}} Q(\overline{s}_i, \overline{a}_i, r_i, \overline{s}'_i) \stackrel{\text{def}}{=} r_i + \gamma \sum_{\overline{a'_i} \in \mathcal{A}_i} \pi_{\text{greedy}}(\overline{a'_i} | \overline{s'_i}; Q) Q(\overline{s'_i}, \overline{a'_i}).$$

Budgeted Reinforcement Learning

We address several limitations of Budgeted Value-Iteration

1. If the P, R_r and R_c are unknown:
   - Work with a batch of samples $\mathcal{D} = \{(\overline{s}_i, \overline{a}_i, r_i, \overline{s}'_i\}_{i\in[0,N]}$
   - Replace $\mathcal{T}$ with a sampling operator $\hat{\mathcal{T}}$:

$$\hat{\mathcal{T}} Q(\overline{s}_i, \overline{a}_i, r_i, \overline{s}'_i) \stackrel{\text{def}}{=} r_i + \gamma \sum_{\overline{a'_i} \in \mathcal{A}_i} \pi_{\text{greedy}}(\overline{a'_i} | \overline{s'_i}; Q) Q(\overline{s'_i}, \overline{a'_i}).$$

2. If $\mathcal{S}$ is continuous:

Budgeted Reinforcement Learning

We address several limitations of Budgeted Value-Iteration

1. If the P, R_r and R_c are unknown:
   - Work with a batch of samples $\mathcal{D} = \{(\overline{s}_i, \overline{a}_i, r_i, \overline{s}'_i\}_{i\in[0,N]}$
   - Replace $\mathcal{T}$ with a sampling operator $\hat{\mathcal{T}}$:

$$\hat{\mathcal{T}} Q(\overline{s}_i, \overline{a}_i, r_i, \overline{s}'_i) \stackrel{\text{def}}{=} r_i + \gamma \sum_{\overline{a'_i} \in \mathcal{A}_i} \pi_{\text{greedy}}(\overline{a'_i} | \overline{s'_i}; Q) Q(\overline{s'_i}, \overline{a'_i}).$$

2. If $\mathcal{S}$ is continuous:
   - Employ function approximation Q_θ, and minimise a regression loss

$$\mathcal{L}(Q_\theta, Q_{\text{target}}; \mathcal{D}) = \sum_{\mathcal{D}} ||Q_\theta(\overline{s}, \overline{a}) - Q_{\text{target}}(\overline{s}, \overline{a}, r, \overline{s}')||_2^2$$

Budgeted Fitted-Q

Algorithm 2: Budgeted Fitted-Q Iteration

Data: $\mathcal{D}$

Result: Q^*

$Q_{\theta_0} \leftarrow 0$

repeat

  $\theta_{k+1} \leftarrow \arg\min_\theta \mathcal{L}(Q_\theta, \hat{\mathcal{T}} Q_{\theta_k}; \mathcal{D})$

until *convergence*

More scaling

More scaling

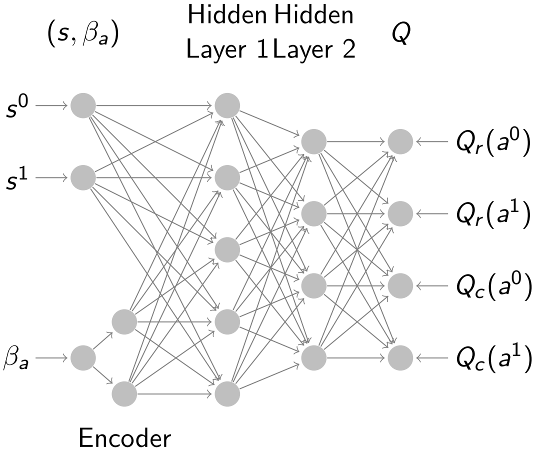
- CPU parallel computing of the targets $\sum_{\overline{a'_i} \in \mathcal{A}_i} \pi_{\text{greedy}}(\overline{a'_i} | \overline{s'_i}; Q) Q(\overline{s'_i}, \overline{a'_i}) \; \forall i$

More scaling

- CPU parallel computing of the targets $\sum_{\overline{a'_i} \in \mathcal{A}_i} \pi_{\text{greedy}}(\overline{a'_i} | \overline{s'_i}; Q) Q(\overline{s'_i}, \overline{a'_i}) \; \forall i$
- Same for samples generation.

More scaling

- CPU parallel computing of the targets $\sum_{\overline{a'_i} \in \mathcal{A}_i} \pi_{\text{greedy}}(\overline{a'_i} | \overline{s'_i}; Q) Q(\overline{s'_i}, \overline{a'_i}) \ \forall i$
- Same for samples generation.
- Neural Network as function approximator:



Experiments: performances of BFTQ

- Baseline: λ-FTQ, Lagrangian relaxation
  - $R_r(s, a) \leftarrow R_r(s, a) - \lambda R_c(s, a)$ where $\lambda \geq 0$

.

- Applications:
  - dialogue systems
  - autonomous driving

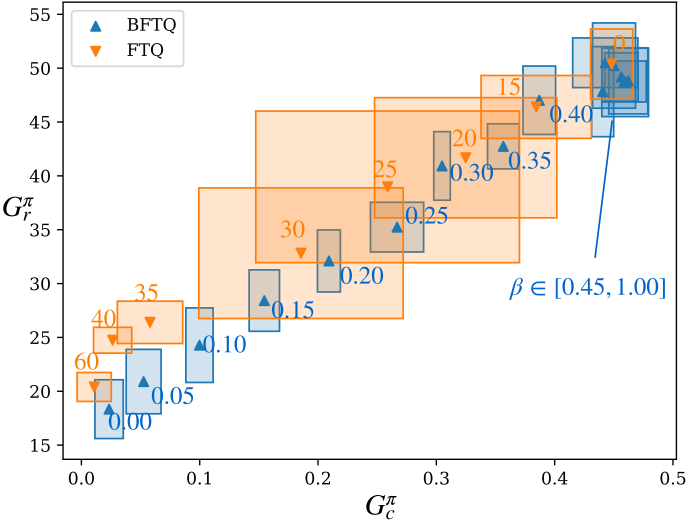
Experiments: dialogue systems

- A slot-filling problem: the agent (the dialogue system) fills a form by asking the user each slot.

Experiments: dialogue systems

- A slot-filling problem: the agent (the dialogue system) fills a form by asking the user each slot.
- Two ways to deal with recognition errors:
  - ask to repeat with voice (safe/slow),
  - Ask to repeat with numeric pad (unsafe/fast).

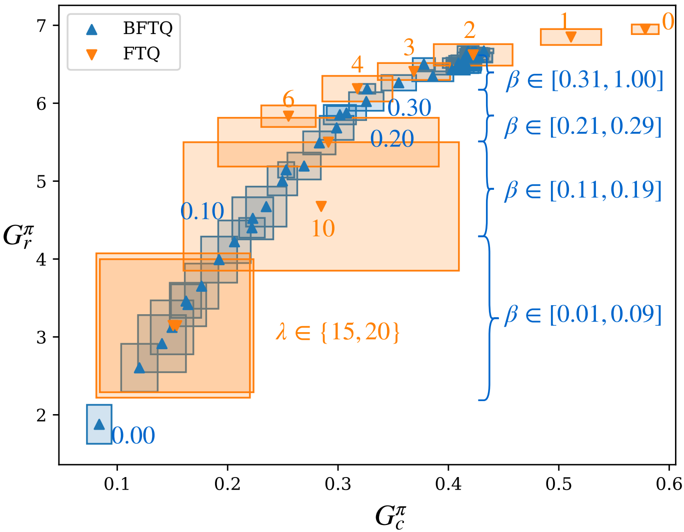
Experiments: dialogue systems



Experiments: autonomous driving

- the agent (the car) is on a two-way road with a car in front of it:
  - it can stay behind (safe/slow),
  - it can overtake (unsafe/fast).

Experiments: autonomous driving



Experiments: autonomous driving

▸ BFTQ on the highway environment

Risk-sensitive exploration

How to collect the batch $\mathcal{D}$?

- We propose an ε-greedy exploration procedure

Risk-sensitive exploration

How to collect the batch $\mathcal{D}$?

- We propose an ε-greedy exploration procedure
  - Sample an initial budget β_0

Risk-sensitive exploration

How to collect the batch $\mathcal{D}$?

- We propose an ε-greedy exploration procedure
  - Sample an initial budget β_0
  - At each step, where $\overline{s} = (s, \beta)$ only explore feasible budgets:

Risk-sensitive exploration

How to collect the batch $\mathcal{D}$?

- We propose an ε-greedy exploration procedure
  - Sample an initial budget β_0
  - At each step, where $\overline{s} = (s, \beta)$ only explore feasible budgets:

$$\overline{a} = (a, \beta_a) \sim \mathcal{U}(\Delta_{\mathcal{AB}})$$
$$\text{where } \Delta \text{is s.t. } \mathbb{P}\,(a, \beta_a | s, \beta) \,\text{verifies}\, \mathbb{E}[\beta_a] \leq \beta$$

Experiments: risk-sensitive exploration

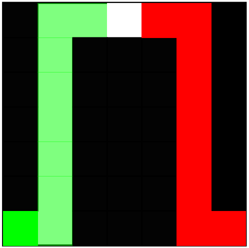
- Validate the risk-sensitive exploration procedure on the corridor environment

Experiments: risk-sensitive exploration

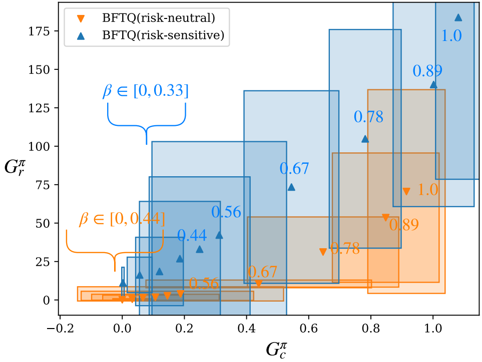
- Validate the risk-sensitive exploration procedure on the corridor environment
- Learn 2 BFTQ policies with respectively:
  - A batch generated by a risk-neutral ε-greedy procedure
  - A batch generated by a risk-sensitive ε-greedy procedure

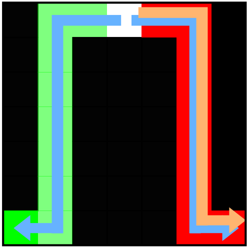
Experiments: corridors

- 2 corridors:
  - high costs/high rewards around the starting state
  - no costs/low rewards around the starting state
- The outermost cell is the one yielding the most reward



Experiments: corridors





Experiments: corridors

▸ Risk-sensitive vs Risk-Neutral on the corridors environment

Summary

Summary

+ Budgeted Bellman Optimality Operator.

Summary

+ Budgeted Bellman Optimality Operator.
    - Fixed point.

Summary

+ Budgeted Bellman Optimality Operator.
  - Fixed point.
  - Not a contraction but converging in practice.

Summary

+ Budgeted Bellman Optimality Operator.
  - Fixed point.
  - Not a contraction but converging in practice.
+ Scalable for RL in continuous state space.

Summary

+ Budgeted Bellman Optimality Operator.
  - Fixed point.
  - Not a contraction but converging in practice.
+ Scalable for RL in continuous state space.
  - Function approximation with Neural Network (dedicated architecture).

Summary

+ Budgeted Bellman Optimality Operator.
    - Fixed point.
    - Not a contraction but converging in practice.
+ Scalable for RL in continuous state space.
    - Function approximation with Neural Network (dedicated architecture).
    - Solving of the untractable program using convex hull.

Summary

+ Budgeted Bellman Optimality Operator.
  - Fixed point.
  - Not a contraction but converging in practice.
+ Scalable for RL in continuous state space.
  - Function approximation with Neural Network (dedicated architecture).
  - Solving of the untractable program using convex hull.
  - CPU parallel computing of the target.

Summary

+ Budgeted Bellman Optimality Operator.
    - Fixed point.
    - Not a contraction but converging in practice.
+ Scalable for RL in continuous state space.
    - Function approximation with Neural Network (dedicated architecture).
    - Solving of the untractable program using convex hull.
    - CPU parallel computing of the target.
    - Risk-sensitive exploration procedure.

Summary

- Budgeted Bellman Optimality Operator.
  - Fixed point.
  - Not a contraction but converging in practice.
- Scalable for RL in continuous state space.
  - Function approximation with Neural Network (dedicated architecture).
  - Solving of the untractable program using convex hull.
  - CPU parallel computing of the target.
  - Risk-sensitive exploration procedure.
- Experiments on two applications.

Summary

+ Budgeted Bellman Optimality Operator.
    - Fixed point.
    - Not a contraction but converging in practice.
+ Scalable for RL in continuous state space.
    - Function approximation with Neural Network (dedicated architecture).
    - Solving of the untractable program using convex hull.
    - CPU parallel computing of the target.
    - Risk-sensitive exploration procedure.
+ Experiments on two applications.
    - BFTQ reaches similar performances as Lagrangian relaxation,

Summary

+ Budgeted Bellman Optimality Operator.
  - Fixed point.
  - Not a contraction but converging in practice.
+ Scalable for RL in continuous state space.
  - Function approximation with Neural Network (dedicated architecture).
  - Solving of the untractable program using convex hull.
  - CPU parallel computing of the target.
  - Risk-sensitive exploration procedure.
+ Experiments on two applications.
  - BFTQ reaches similar performances as Lagrangian relaxation,
  - with no need for calibration,

Summary

- Budgeted Bellman Optimality Operator.
  - Fixed point.
  - Not a contraction but converging in practice.
- Scalable for RL in continuous state space.
  - Function approximation with Neural Network (dedicated architecture).
  - Solving of the untractable program using convex hull.
  - CPU parallel computing of the target.
  - Risk-sensitive exploration procedure.
- Experiments on two applications.
  - BFTQ reaches similar performances as Lagrangian relaxation,
  - with no need for calibration,
  - and less variance.