Budgeted Reinforcement Learning in Continuous State Space

Nicolas Carrara[1], Edouard Leurent[1,2], Tanguy Urvoy[3], Romain Laroche[4], Odalric Maillard[1], Olivier Pietquin[1,5]

[1]Inria SequeL, [2]Renault Group, [3]Orange Labs, [4]Microsoft Montréal, [5]Google Research, Brain Team

Contents

01

Motivation and Setting

Optimal Decision-Making

Which action a_t should we choose in state s_t to maximise a cumulative reward R?

$$\max_{\pi} \mathop{\mathbb{E}}_{a_t \sim \pi(a_t|s_t)} \left[\sum_{t=0}^{\infty} \gamma^t R(s_t, a_t) \right]$$

Optimal Decision-Making

Which action a_t should we choose in state s_t to maximise a cumulative reward R?

$$\max_{\pi} \mathop{\mathbb{E}}_{a_t \sim \pi(a_t|s_t)} \left[\sum_{t=0}^{\infty} \gamma^t R(s_t, a_t) \right]$$

- ✓ A very general formulation
- ✓ Widely used in the industry

Optimal Decision-Making

Which action a_t should we choose in state s_t to maximise a cumulative reward R?

$$\max_{\pi} \mathop{\mathbb{E}}_{a_t \sim \pi(a_t|s_t)} \left[\sum_{t=0}^{\infty} \gamma^t R(s_t, a_t) \right]$$

✓ A very general formulation

✗ **Not** widely used in the industry

Optimal Decision-Making

Which action a_t should we choose in state s_t to maximise a cumulative reward R?

$$\max_{\pi} \mathop{\mathbb{E}}_{a_t \sim \pi(a_t|s_t)} \left[\sum_{t=0}^{\infty} \gamma^t R(s_t, a_t) \right]$$

- ✓ A very general formulation
- ✗ **Not** widely used in the industry
  - > Sample efficiency
  - > Trial and error
  - > Unpredictable behaviour

Reinforcement learning relies on a single reward function R

Reinforcement learning relies on a single reward function R

- ✓ A convenient formulation, but;

Reinforcement learning relies on a single reward function R

- ✓ A convenient formulation, but;
- ✗ R is not always easy to design.

Reinforcement learning relies on a single reward function R

- ✓ A convenient formulation, but;
- ✗ R is not always easy to design.

Conflicting Objectives

Complex tasks require multiple contradictory aspects. Typically:

Task completion vs Safety

Reinforcement learning relies on a single reward function R

- ✓ A convenient formulation, but;
- ✗ R is not always easy to design.

Conflicting Objectives

Complex tasks require multiple contradictory aspects. Typically:

Task completion vs Safety

For example...

Inria

Dialogue systems

A slot-filling problem: the agent fills a form by asking the user each slot. It can either:

- ask to answer using voice (safe/slow);
- ask to answer with a numeric pad (unsafe/fast).

Dialogue systems

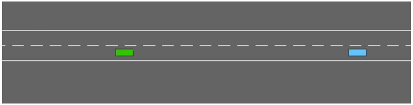
A slot-filling problem: the agent fills a form by asking the user each slot. It can either:

- ask to answer using voice (safe/slow);
- ask to answer with a numeric pad (unsafe/fast).

Autonomous Driving

The agent is driving on a two-way road with a car in front of it,

- it can stay behind (safe/slow);
- it can overtake (unsafe/fast).



Reinforcement learning relies on a single reward function R

- ✓ A convenient formulation, but;
- ✗ R is not always easy to design.

Conflicting Objectives

Complex tasks require multiple contradictory aspects. Typically:

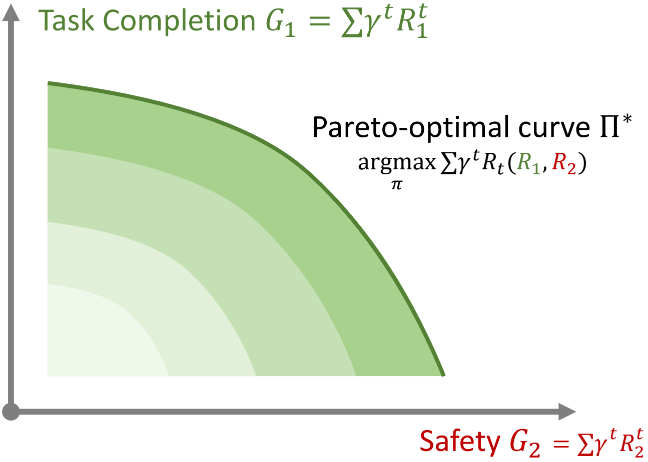
Task completion vs Safety

For example...

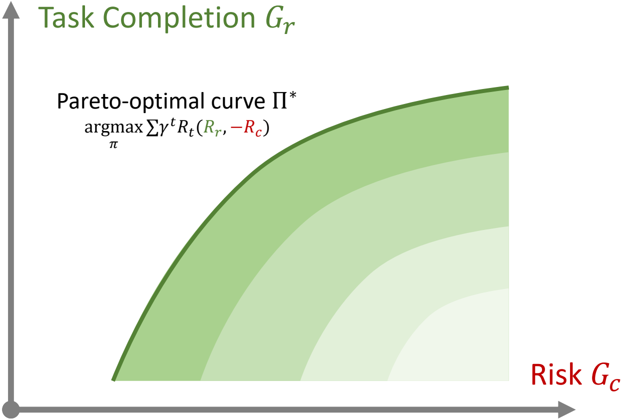
For a fixed reward function R,

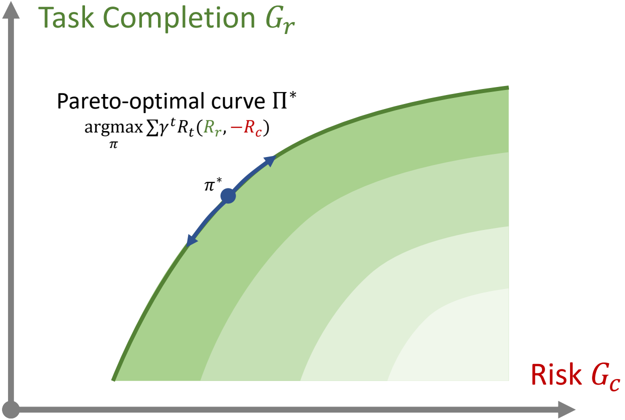
↳ no control over the $\frac{\text{Task Completion}}{\text{Safety}}$ trade-off

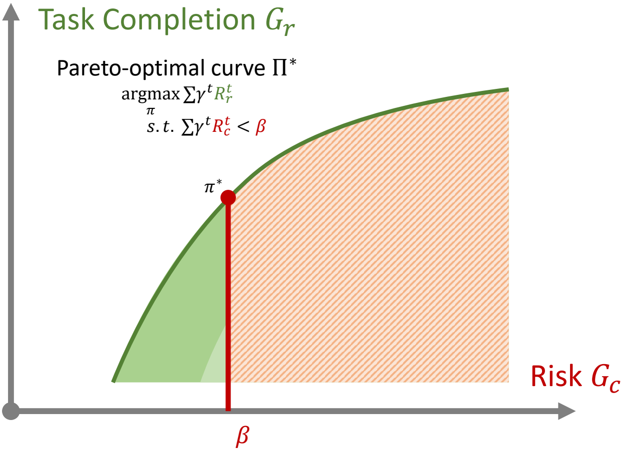
↳ π^* is only guaranteed to lie on a Pareto-optimal curve Π^*



Task Completion $G_1 = \sum \gamma^t R_1^t$
Pareto-optimal curve Π^*
$\underset{\pi}{\operatorname{argmax}} \sum \gamma^t R_t(R_1, R_2)$
Safety $G_2 = \sum \gamma^t R_2^t$


Task Completion G_r
Pareto-optimal curve Π^*
$\underset{\pi}{\operatorname{argmax}} \sum \gamma^t R_t(R_r, -R_c)$
Risk G_c


Task Completion G_r
Pareto-optimal curve Π^*
$\underset{\pi}{\operatorname{argmax}} \sum \gamma^t R_t(R_r, -R_c)$
π^*
Risk G_c



Task Completion G_r
Pareto-optimal curve Π^*
$\underset{\pi}{\text{argmax}} \sum \gamma^t R_r^t$
$s.t. \sum \gamma^t R_c^t < \beta$
π^*
Risk G_c
β

Markov Decision Process

An MDP is a tuple $(\mathcal{S}, \mathcal{A}, P, R_r, \gamma)$ with:

- Rewards $R_r \in \mathbb{R}^{\mathcal{S}\times\mathcal{A}}$

Objective

Maximise rewards

$$\max_{\pi\in\mathcal{M}(\mathcal{A})^{\mathcal{S}}} \quad \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t R_r(s_t, a_t) \mid s_0 = s\right]$$

Constrained Markov Decision Process

A CMDP is a tuple $(\mathcal{S}, \mathcal{A}, P, R_r, R_c, \gamma, \beta)$ with:

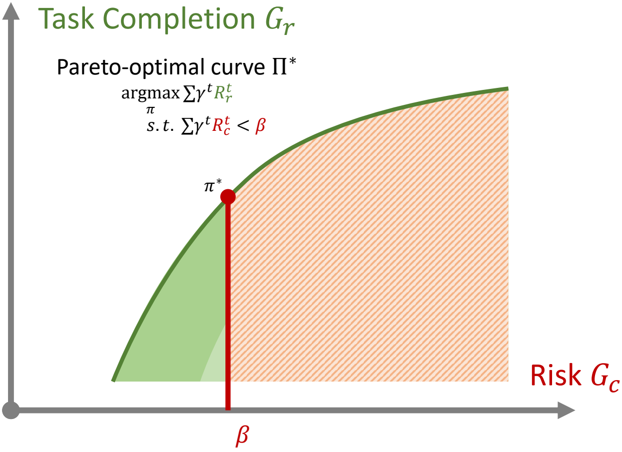
- Rewards $R_r \in \mathbb{R}^{\mathcal{S}\times\mathcal{A}}$
- Costs $R_c \in \mathbb{R}^{\mathcal{S}\times\mathcal{A}}$
- Budget β

Objective

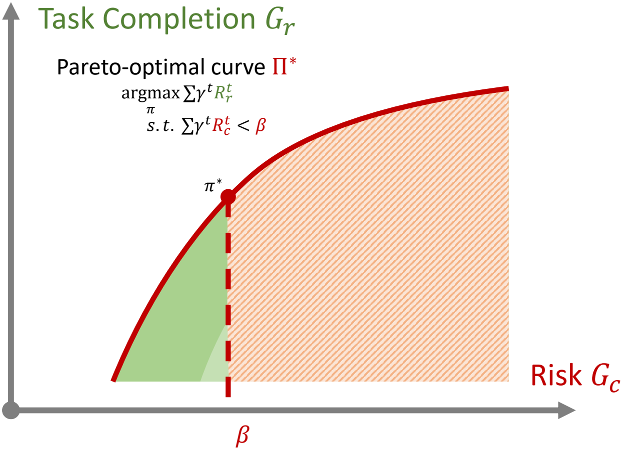
Maximise rewards while keeping costs under a fixed budget

$$\begin{aligned} \max_{\pi\in\mathcal{M}(\mathcal{A})^{\mathcal{S}}} \quad & \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t R_r(s_t, a_t) \mid s_0 = s\right] \\ \text{s.t.} \quad & \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t R_c(s_t, a_t) \mid s_0 = s\right] \leq \beta \end{aligned}$$

We want to learn Π^* rather than π_β^*



We want to learn Π^* rather than π^*_β



Budgeted Markov Decision Process

A BMDP is a tuple $(\mathcal{S}, \mathcal{A}, P, R_r, R_c, \gamma, \mathcal{B})$ with:

- Rewards $R_r \in \mathbb{R}^{\mathcal{S}\times\mathcal{A}}$
- Costs $R_c \in \mathbb{R}^{\mathcal{S}\times\mathcal{A}}$
- Budget space $\mathcal{B}$

Objective

Maximise rewards while keeping costs under an adjustable budget.
$\forall \beta \in \mathcal{B}$,

$$\begin{aligned} \max_{\pi\in\mathcal{M}(\mathcal{A}\times\mathcal{B})^{\mathcal{S}\times\mathcal{B}}} \quad & \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t R_r(s_t, a_t) \mid s_0 = s, \beta_0 = \beta\right] \\ \text{s.t.} \quad & \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t R_c(s_t, a_t) \mid s_0 = s, \beta_0 = \beta\right] \leq \beta \end{aligned}$$

Budgeted policies π

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

↳ Augment the spaces with the budget β

Definition (Augmented spaces)

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

$$\text{state } (s, \beta), \text{ action } (a, \beta_a) \rightarrow \text{ next state } \begin{cases} s' \sim P(s'|s,a) \\ \beta' = \beta_a \end{cases}$$

Definition (Augmented signals)

1. Rewards $R = (R_r, R_c)$
2. Returns $G^\pi = (G_r^\pi, G_c^\pi) \stackrel{\text{def}}{=} \sum_{t=0}^{\infty} \gamma^t R(\overline{s}_t, \overline{a}_t)$
3. Value $V^\pi(\overline{s}) = (V_r^\pi, V_c^\pi) \stackrel{\text{def}}{=} \mathbb{E}\left[G^\pi \mid \overline{s_0} = \overline{s}\right]$
4. Q-Value $Q^\pi(\overline{s}, \overline{a}) = (Q_r^\pi, Q_c^\pi) \stackrel{\text{def}}{=} \mathbb{E}\left[G^\pi \mid \overline{s_0} = \overline{s}, \overline{a_0} = \overline{a}\right]$

02

Budgeted Dynamic Programming

Proposition (Budgeted Bellman Expectation)

The Bellman Expectation equations are preserved

$$V^\pi(\overline{s}) = \sum_{\overline{a}\in\overline{\mathcal{A}}} \pi(\overline{a}|\overline{s}) Q^\pi(\overline{s}, \overline{a})$$

$$Q^\pi(\overline{s}, \overline{a}) = R(\overline{s}, \overline{a}) + \gamma \sum_{\overline{s}'\in\overline{\mathcal{S}}} \overline{P}\left(\overline{s}' \mid \overline{s}, \overline{a}\right) V^\pi(\overline{s}')$$

Proposition (Budgeted Bellman Expectation)

The Bellman Expectation equations are preserved

$$V^\pi(\overline{s}) = \sum_{\overline{a}\in\overline{\mathcal{A}}} \pi(\overline{a}|\overline{s}) Q^\pi(\overline{s},\overline{a})$$

$$Q^\pi(\overline{s},\overline{a}) = R(\overline{s},\overline{a}) + \gamma \sum_{\overline{s}'\in\overline{\mathcal{S}}} \overline{P}\left(\overline{s}' \mid \overline{s},\overline{a}\right) V^\pi(\overline{s}')$$

Proposition (Contraction)

The Bellman Expectation Operator $\mathcal{T}^\pi$ is a γ-contraction.

$$\mathcal{T}^\pi Q(\overline{s},\overline{a}) \overset{def}{=} R(\overline{s},\overline{a}) + \gamma \sum_{\overline{s}'\in\overline{\mathcal{S}}} \sum_{\overline{a}'\in\overline{\mathcal{A}}} \overline{P}(\overline{s}'|\overline{s},\overline{a})\pi(\overline{a}'|\overline{s}')Q(\overline{s}',\overline{a}')$$

✓ We can evaluate a budgeted policy π

Definition (Budgeted Optimality)

In that order, we want to:

(i) Respect the budget β:

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

(ii) Maximise the rewards:

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

(iii) Minimise the costs:

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

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

Theorem (Budgeted Bellman Optimality Equation)

Q^* verifies the following equation:

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

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

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

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

Proposition (Optimality of the policy)

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

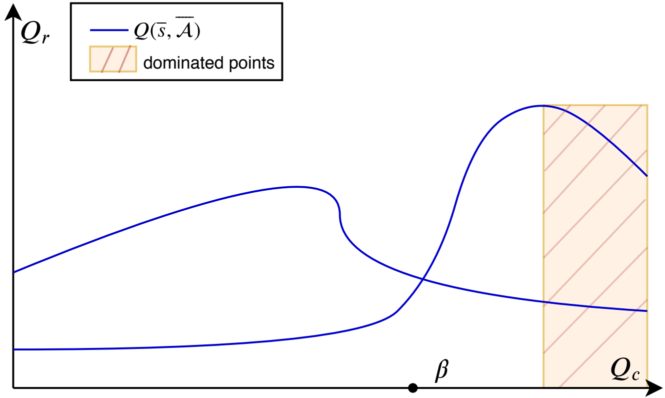
$$\pi_{greedy}(\cdot\ ; Q^*) \in \Pi^*(\bar{s})$$

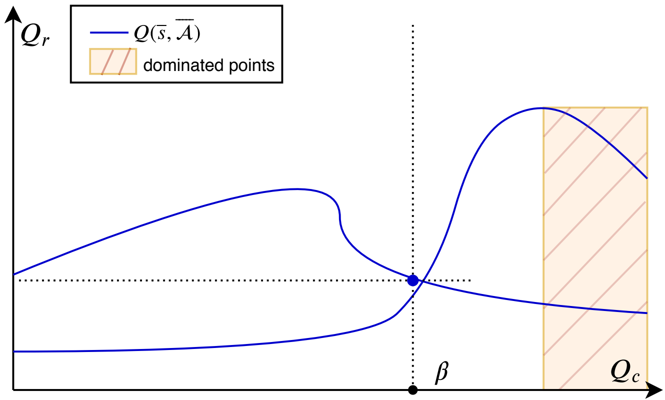
In particular, $V^{\pi_{greedy}(\cdot;Q^*)} = V^*$ *and* $Q^{\pi_{greedy}(\cdot;Q^*)} = Q^*$*.*

Proposition (Solving the non-linear program)

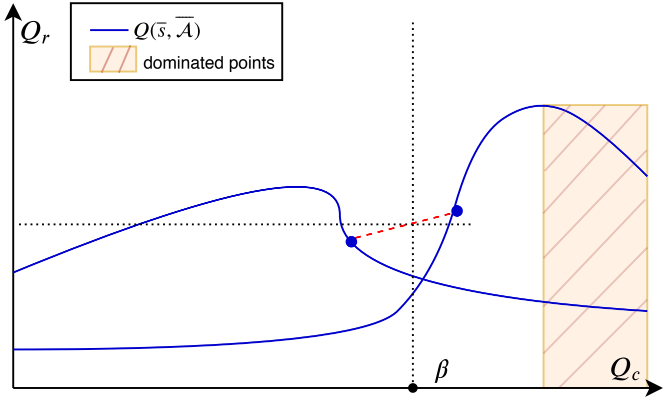
π_{greedy} *can be computed efficiently, as a mixture* π_{hull} *of two points that lie on the convex hull of* Q*.*

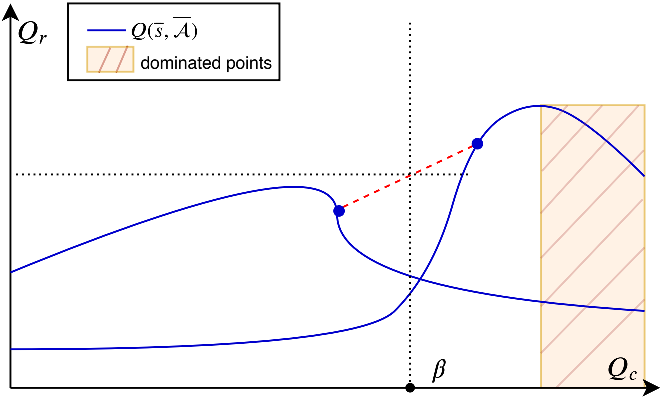
$$\pi_{greedy} = \pi_{hull}$$


Q_r
$Q(\overline{s}, \overline{\mathcal{A}})$
dominated points
β
Q_c

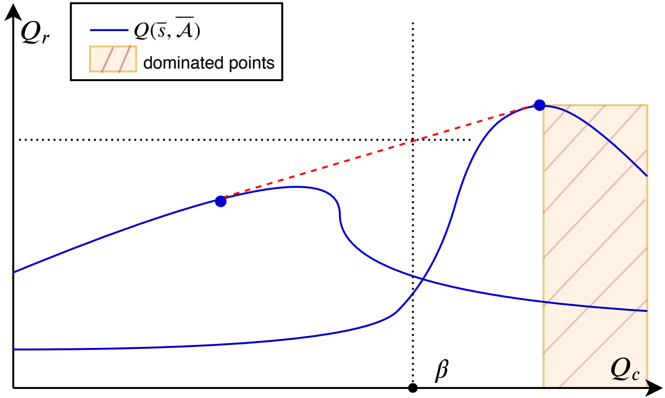

Q_r
$Q(\overline{s}, \overline{\mathcal{A}})$
dominated points
β
Q_c

Solving the non-linear program: intuition




Q_r
$Q(\overline{s}, \overline{\mathcal{A}})$
dominated points
β
Q_c

Solving the non-linear program: intuition



Recall what we've shown so far:

$$\mathcal{T} \xrightarrow[fixed-point]{} Q^* \xrightarrow[tractable]{} \pi_{\text{hull}}(Q^*) \xrightarrow[equal]{} \pi_{\text{greedy}}(Q^*) \xrightarrow[optimal]{}$$

Recall what we've shown so far:

$$\mathcal{T} \xrightarrow[fixed-point]{} Q^* \xrightarrow[tractable]{} \pi_{\text{hull}}(Q^*) \xrightarrow[equal]{} \pi_{\text{greedy}}(Q^*) \xrightarrow[optimal]{}$$

We're almost there!
All that is left is to perform Fixed-Point Iteration to compute Q^*.

Recall what we've shown so far:

$$\mathcal{T} \xrightarrow[fixed-point]{} Q^* \xrightarrow[tractable]{} \pi_{\text{hull}}(Q^*) \xrightarrow[equal]{} \pi_{\text{greedy}}(Q^*) \xrightarrow[optimal]{}$$

We're almost there!
All that is left is to perform Fixed-Point Iteration to compute Q^*.

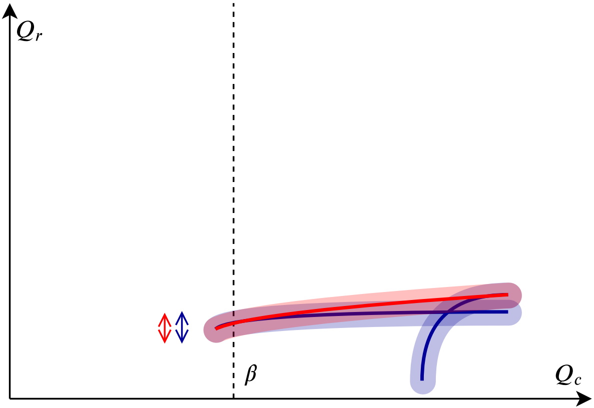
Theorem (Non-Contractivity)

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

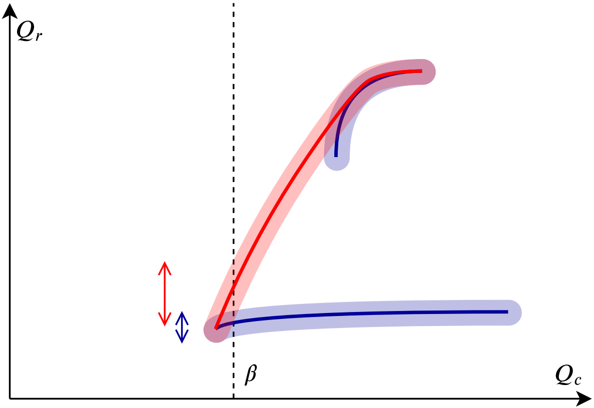
$$\forall \varepsilon > 0, \exists Q^1, Q^2 \in (\mathbb{R}^2)^{\overline{\mathcal{S}\mathcal{A}}} : \|\mathcal{T}Q^1 - \mathcal{T}Q^2\|_\infty \geq \frac{1}{\varepsilon}\|Q^1 - Q^2\|_\infty$$

✗ We cannot guarantee the convergence of $\mathcal{T}^n(Q_0)$ to Q^*

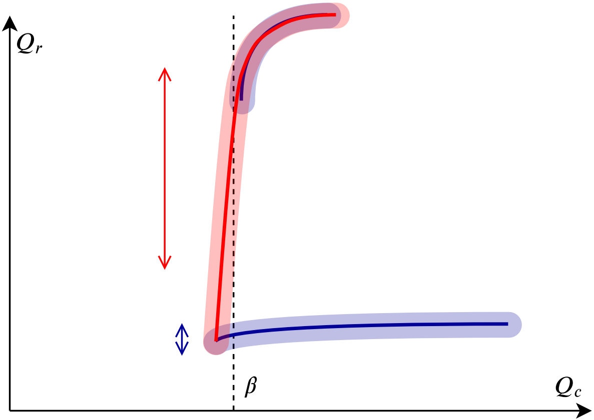
Not a contraction: intuition



Not a contraction: intuition



Not a contraction: intuition



Thankfully,

Theorem (Contractivity on smooth Q-functions)

$\mathcal{T}$ 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 $L < \frac{1}{\gamma} - 1$.

$$\mathcal{L}_\gamma = \left\{ \begin{array}{c} Q \in (\mathbb{R}^2)^{\overline{\mathcal{S}\mathcal{A}}} \text{ s.t. } \exists L < \frac{1}{\gamma} - 1 : \forall \overline{s} \in \overline{\mathcal{S}}, \overline{a}_1, \overline{a}_2 \in \overline{\mathcal{A}}, \\ |Q_r(\overline{s}, \overline{a}_1) - Q_r(\overline{s}, \overline{a}_2)| \leq L |Q_c(\overline{s}, \overline{a}_1) - Q_c(\overline{s}, \overline{a}_2)| \end{array} \right\}$$

- ✓ We guarantee convergence under some (strong) assumptions
- ✓ We observe empirical convergence

Inria

Algorithm 1: Budgeted Value-Iteration

Data: P, R_r, R_c

Result: Q^*

1 $Q_0 \leftarrow 0$

2 **repeat**

3 $\quad Q_{k+1} \leftarrow \mathcal{T} Q_k$

4 **until** *convergence*

03

Budgeted Reinforcement Learning

We address several limitations of Budgeted Value-Iteration

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

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 $\mathcal{D} = \{(\overline{s}_i, \overline{a}_i, r_i, \overline{s}'_i\}_{i\in[0,N]}$

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 $\hat{\mathcal{T}}$:

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

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:

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 $\hat{\mathcal{T}}$:

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

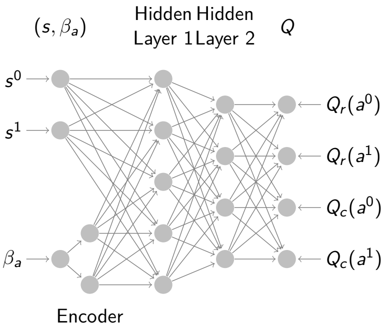
2. If $\mathcal{S}$ is continuous:
   - Employ function approximation Q_θ, and minimise a regression loss

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

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

- CPU parallel computing of the targets $\sum_{\overline{a_i'} \in \mathcal{A}_i} \pi_{\text{greedy}}(\overline{a_i'} | \overline{s_i'}; Q) Q(\overline{s_i'}, \overline{a_i'})$
- Same for interactions with the environment.

- CPU parallel computing of the targets $\sum_{\overline{a_i'} \in \mathcal{A}_i} \pi_{\text{greedy}}(\overline{a_i'}|\overline{s_i'}; Q) Q(\overline{s_i'}, \overline{a_i'})$
- Same for interactions with the environment.
- Neural Network for function approximation:



04

Experiments

Lagrangian Relaxation

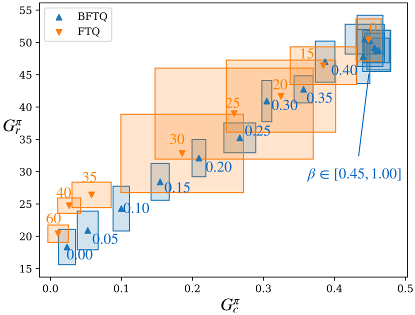
Consider the dual problem so as to replace the hard constraint by a soft constraint penalised by a Lagrangian multiplier λ:

$$\max_{\pi} \mathbb{E} \sum_{t} \gamma^t R_r(s,a) - \lambda \gamma^t R_c(s,a)$$

- Train many policies π_k with penalties λ_k and recover the cost budgets β_k
- Very data/memory-heavy

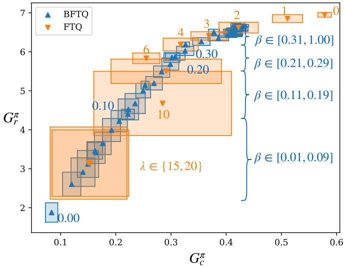
A slot-filling problem: the agent (the dialogue system) fills a form by asking the user each slot. It can either:

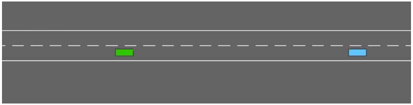
- ask to answer using voice (safe/slow);
- ask to answer with a numeric pad (unsafe/fast).



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).





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

We propose an ε-greedy exploration procedure:

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

We propose an ε-greedy exploration procedure:

- Sample an initial budget $\beta_0 \sim \mathcal{U}(\mathcal{B})$

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

We propose an ε-greedy exploration procedure:

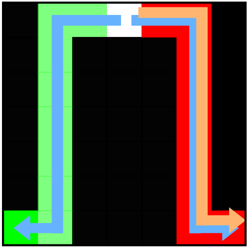
- Sample an initial budget $\beta_0 \sim \mathcal{U}(\mathcal{B})$
- At each step, where $\overline{s} = (s, \beta)$ only explore feasible budgets:

$$\overline{a} = (a, \beta_a) \sim \mathcal{U}(\Delta_{\mathcal{AB}})$$

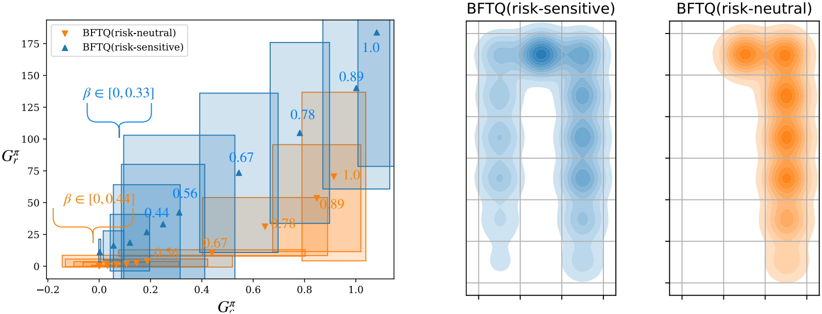
where Δ is such that $\mathbb{P}\,(a, \beta_a | s, \beta)$ verifies $\mathbb{E}[\beta_a] \leq \beta$

Two corridors:

1. one with high costs / high rewards
2. the other with no costs / low rewards



→ Validate the risk-sensitive exploration procedure


BFTQ(risk-neutral)
BFTQ(risk-sensitive)
$\beta \in [0, 0.33]$
$\beta \in [0, 0.44]$
1.0
0.89
0.78
0.67
0.56
0.44
1.0
0.89
0.78
0.67
0.56
175
150
125
100
75
50
25
0
G_r^π
−0.2
0.0
0.2
0.4
0.6
0.8
1.0
G_c^π
BFTQ(risk-sensitive)
BFTQ(risk-neutral)

Thank You!