Taming Large MDPs Through Stochastic Games

Chloé Capon [1] Nicolas Lecomte [1]
Petr Novotný [3] Mickaël Randour [1,2]

[1]UMONS – Université de Mons, Belgium

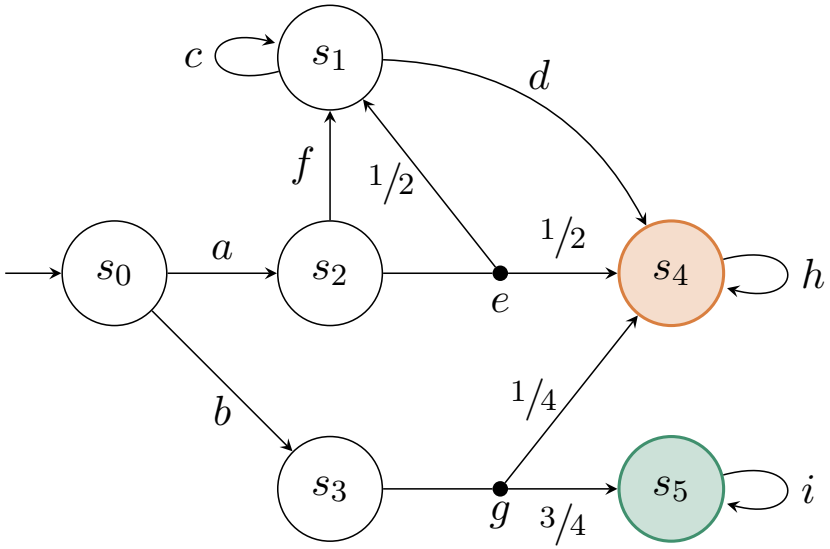
[2]F.R.S.-FNRS, Belgium

[3]Masaryk University, Brno, Czech Republic

September 19, 2024

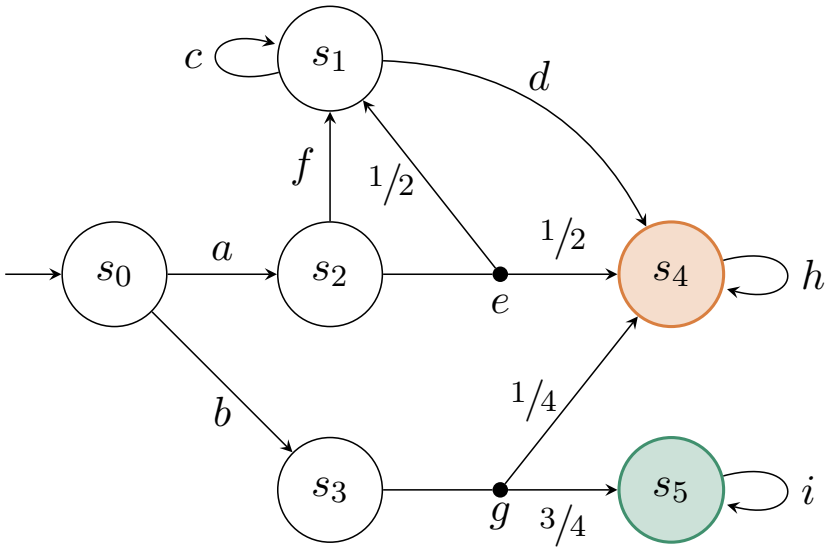
Highlights of Logic, Games and Automata 2024

Context



- A finite set of states S;
- An initial state s_{init};
- A finite set of actions A;
- A probabilistic transition function $\delta : S \times A \to \text{Dist}(S)$.

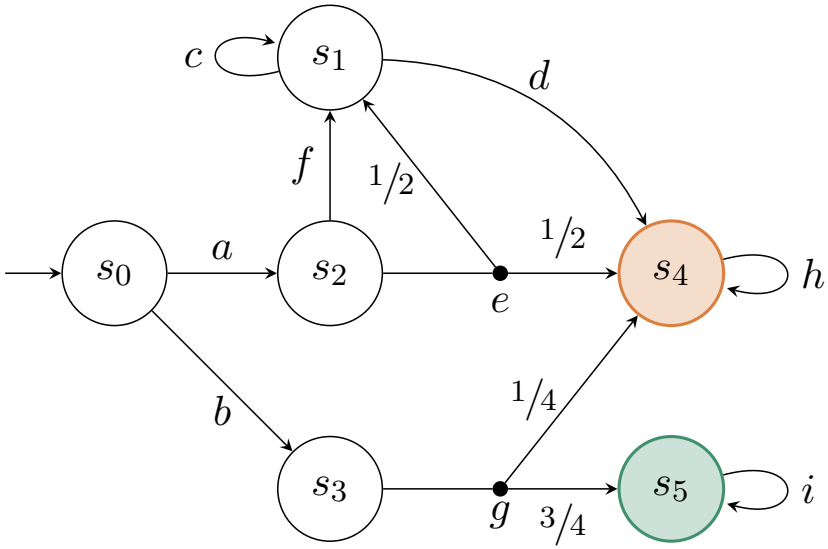
Context

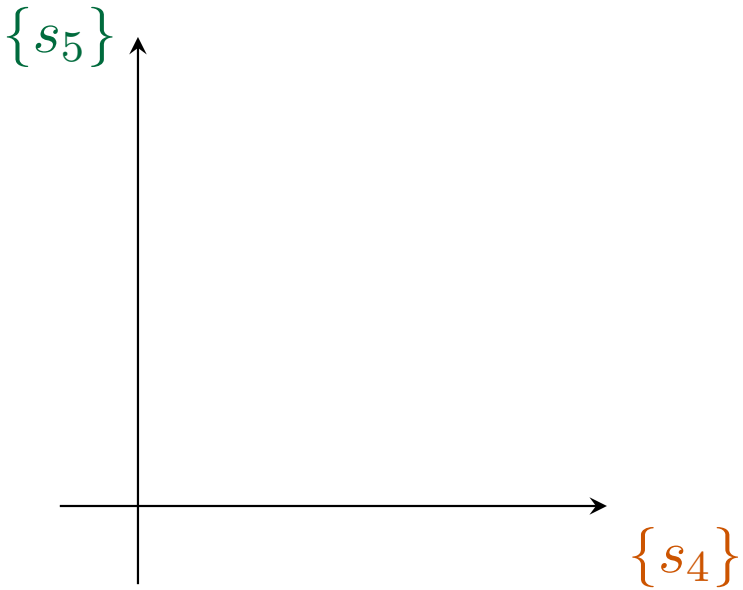


- A finite set of states S;
- An initial state s_{init};
- A finite set of actions A;
- A probabilistic transition function $\delta : S \times A \to \text{Dist}(S)$.

Multi-reachability objectives are vectors of sets of states that we want to **reach**.

Context

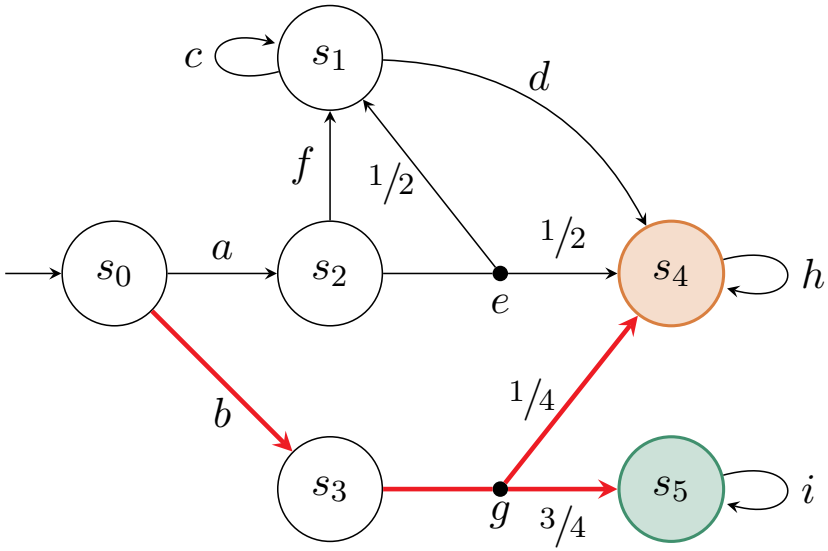


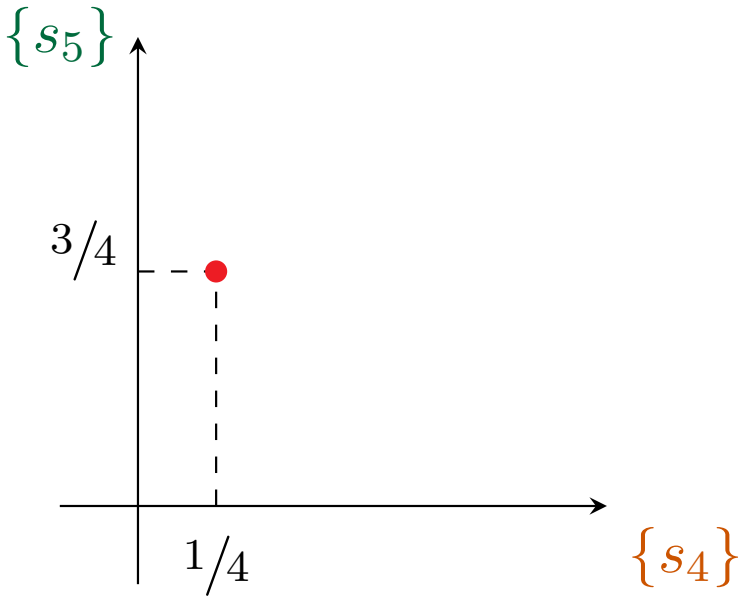


Multi-reachability objectives are vectors of sets of states that we want to **reach**.

A threshold vector $\boldsymbol{\alpha} \in [0,1]^n$ is **achievable** if there exists a strategy σ where $\mathbb{P}_s^\sigma(\Diamond T_i) \geq \alpha_i$ for all $1 \leq i \leq n$.

Context

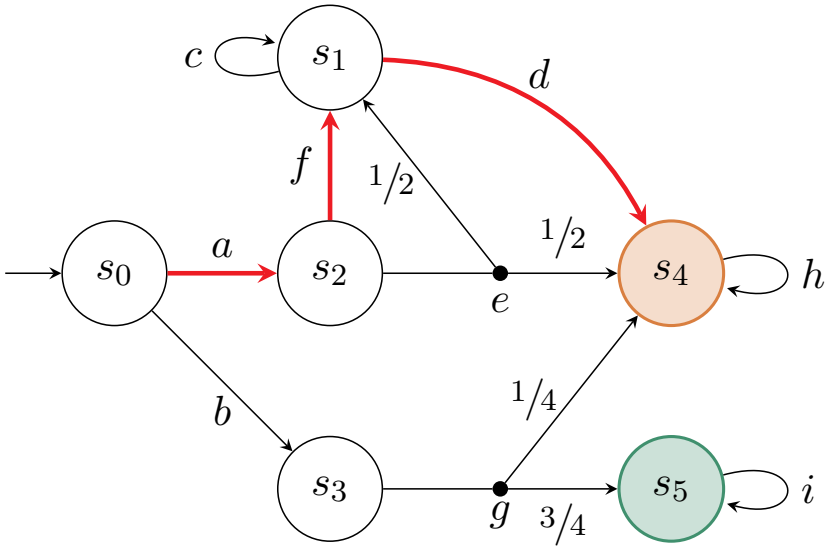


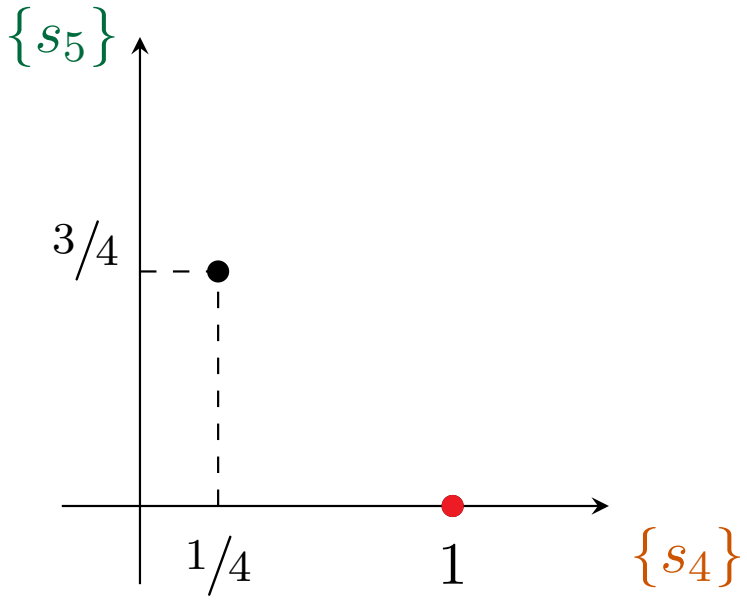


Multi-reachability objectives are vectors of sets of states that we want to **reach**.

A threshold vector $\boldsymbol{\alpha} \in [0,1]^n$ is **achievable** if there exists a strategy σ where $\mathbb{P}_s^\sigma(\Diamond T_i) \geq \alpha_i$ for all $1 \leq i \leq n$.

Context

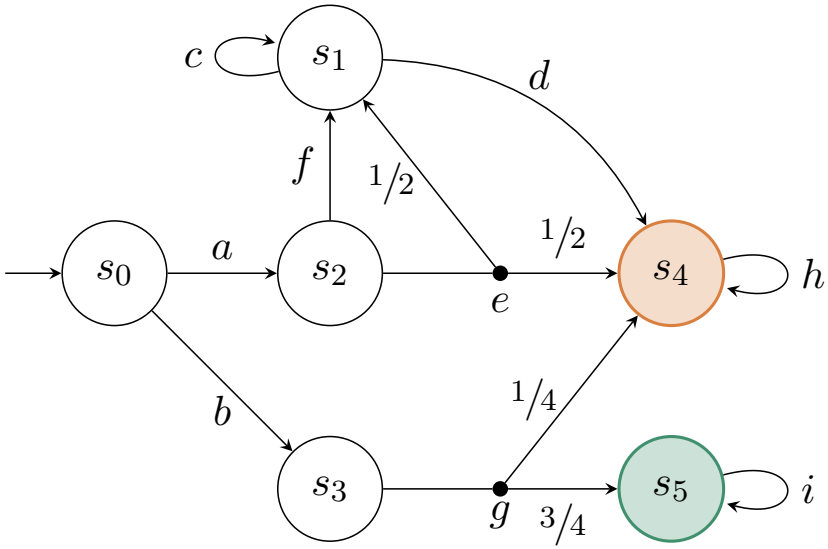


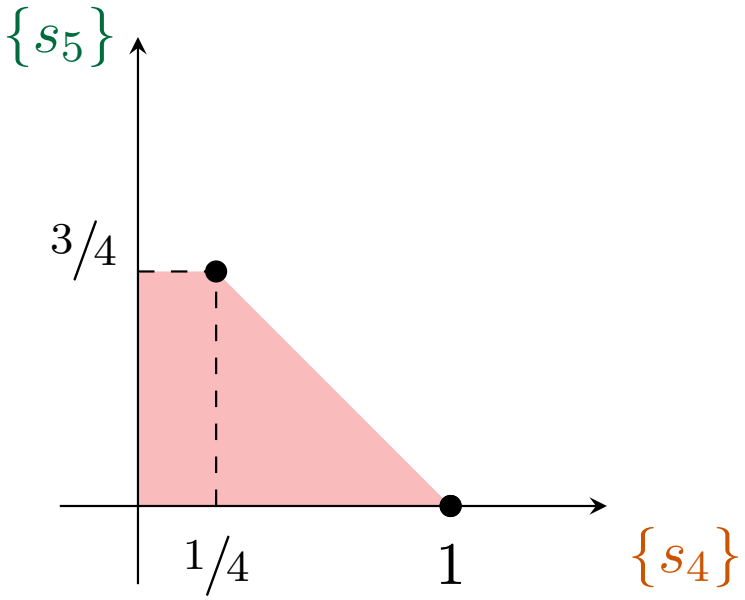


Multi-reachability objectives are vectors of sets of states that we want to **reach**.

A threshold vector $\boldsymbol{\alpha} \in [0,1]^n$ is **achievable** if there exists a strategy σ where $\mathbb{P}_s^\sigma(\Diamond T_i) \geq \alpha_i$ for all $1 \leq i \leq n$.

Context

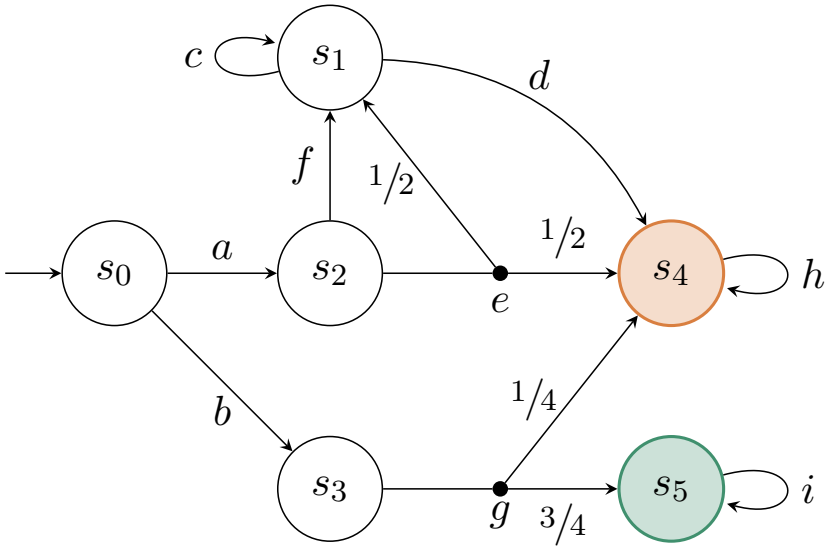


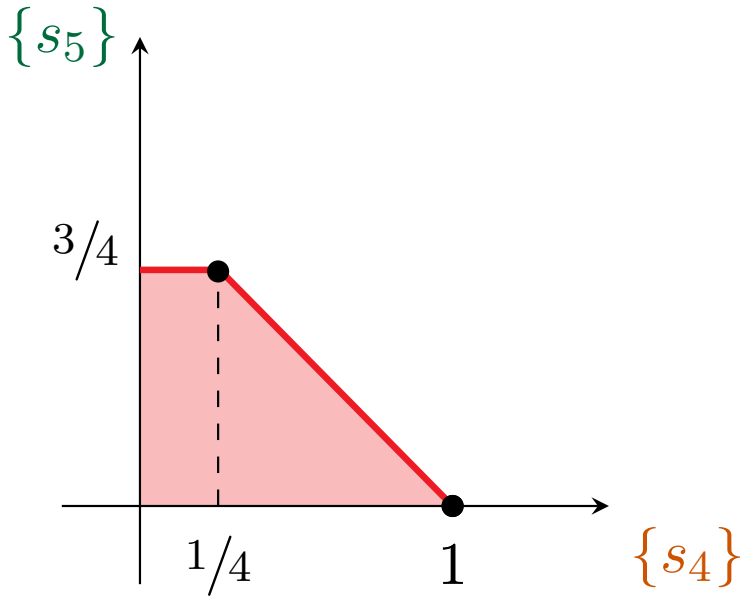


Multi-reachability objectives are vectors of sets of states that we want to **reach**.

A threshold vector $\boldsymbol{\alpha} \in [0,1]^n$ is **achievable** if there exists a strategy σ where $\mathbb{P}_s^\sigma(\Diamond T_i) \geq \alpha_i$ for all $1 \leq i \leq n$.

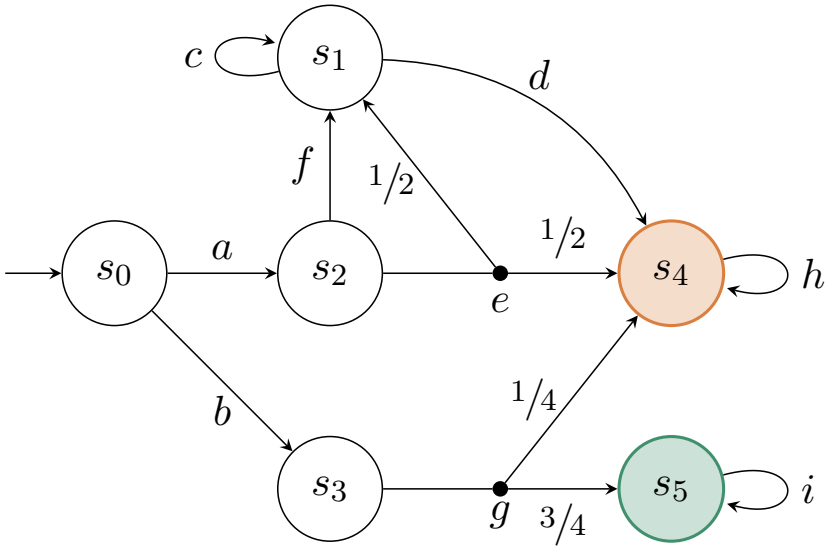
Context

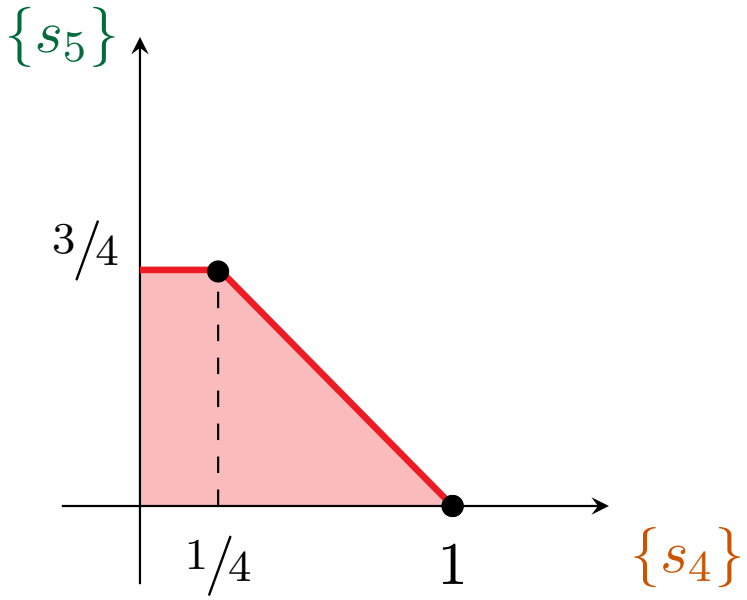




The Pareto frontier of Ach(s) is the set of points in the downward-closure of the convex hull of Ach(s) that are **not dominated**.

Context

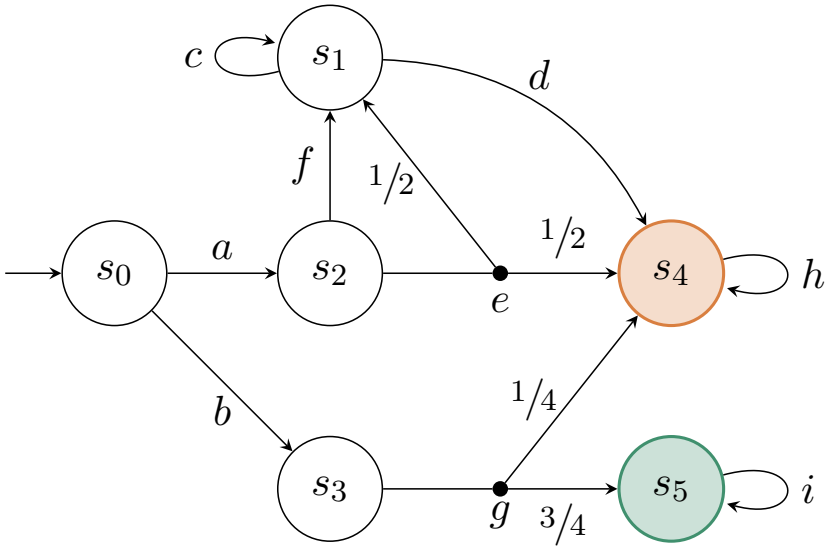


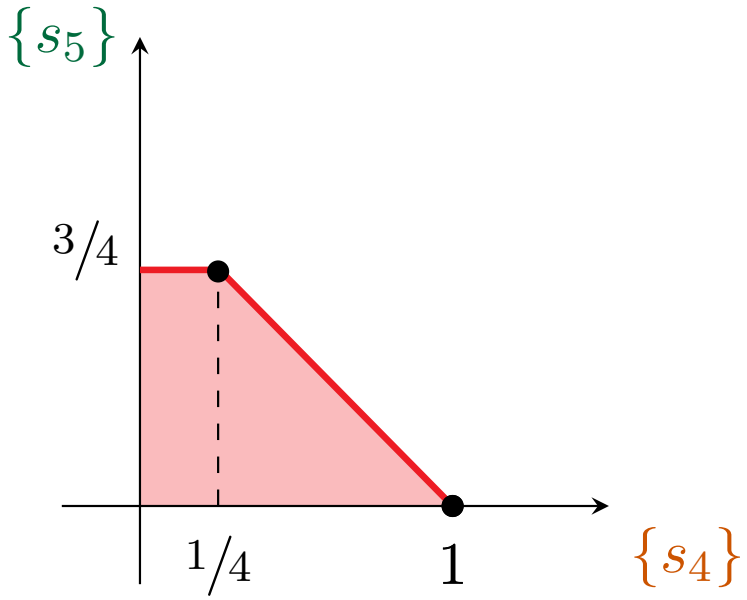


The Pareto frontier of $\mathrm{Ach}(s)$ is the set of points in the downward-closure of the convex hull of $\mathrm{Ach}(s)$ that are **not dominated**.

We want to compute Pareto frontiers

Context





The Pareto frontier of $\mathrm{Ach}(s)$ is the set of points in the downward-closure of the convex hull of $\mathrm{Ach}(s)$ that are **not dominated**.

We want to compute Pareto frontiers $\rightsquigarrow$ Develop **efficient** methods in practice.

Game-based abstraction

We extend the work of [KKNP10][1] from one to multiple dimensions.

Goal. Abstract our model by **merging** states together.

$\rightsquigarrow$ Approximate the Pareto frontier through a smaller model.

[1] Kattenbelt et al., "A game-based abstraction-refinement framework for Markov decision processes".

Game-based abstraction

We extend the work of [KKNP10][1] from one to multiple dimensions.

Goal. Abstract our model by **merging** states together.

$\rightsquigarrow$ Approximate the Pareto frontier through a smaller model.

Introducing a new form of nondeterminism

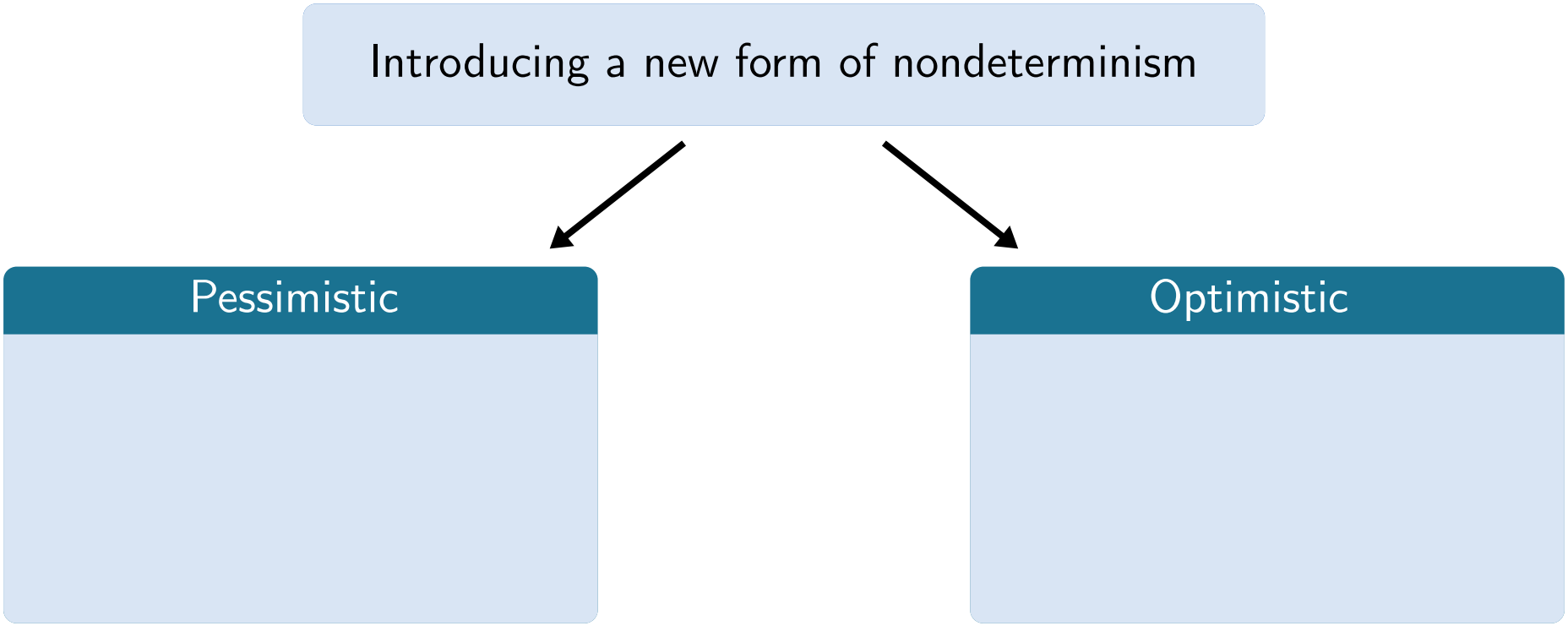
[1] Kattenbelt et al., "A game-based abstraction-refinement framework for Markov decision processes".

Game-based abstraction

We extend the work of [KKNP10][1] from one to multiple dimensions.

Goal. Abstract our model by **merging** states together.

$\rightsquigarrow$ Approximate the Pareto frontier through a smaller model.



[1] Kattenbelt et al., "A game-based abstraction-refinement framework for Markov decision processes".

Game-based abstraction

We extend the work of [KKNP10][1] from one to multiple dimensions.

Goal. Abstract our model by **merging** states together.

$\rightsquigarrow$ Approximate the Pareto frontier through a smaller model.

Introducing a new form of nondeterminism

Pessimistic

- Players are antagonistic (SG)
- Lower approximation of the Pareto frontier

Optimistic

[1] Kattenbelt et al., "A game-based abstraction-refinement framework for Markov decision processes".

Game-based abstraction

We extend the work of [KKNP10][1] from one to multiple dimensions.

Goal. Abstract our model by **merging** states together.

$\rightsquigarrow$ Approximate the Pareto frontier through a smaller model.

Introducing a new form of nondeterminism

Pessimistic

- Players are antagonistic (SG)
- Lower approximation of the Pareto frontier

Optimistic

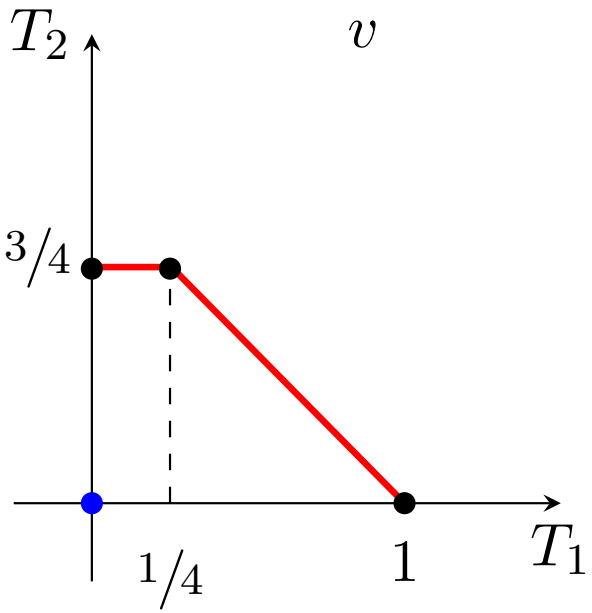
- Players are cooperating (MDP)
- Upper approximation of the Pareto frontier

[1] Kattenbelt et al., "A game-based abstraction-refinement framework for Markov decision processes".

Refining the abstraction

The approximations provide a **quantitative** evaluation of the abstraction's quality and indicate on how to **refine** it.

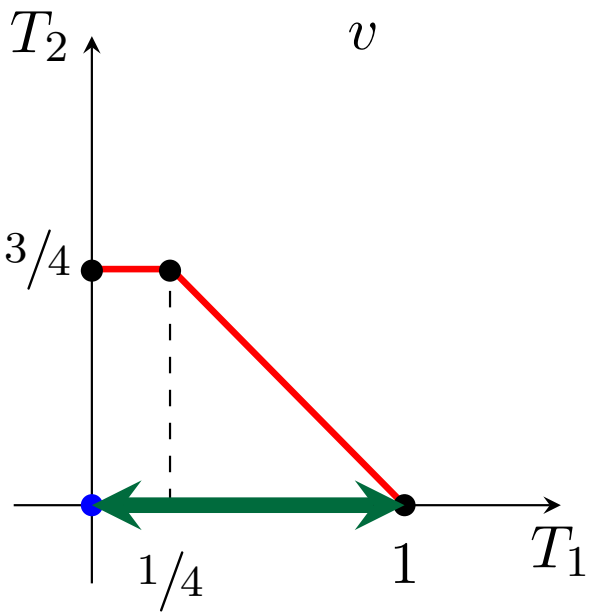
Example.



Refining the abstraction

The approximations provide a **quantitative** evaluation of the abstraction's quality and indicate on how to **refine** it.

Example.



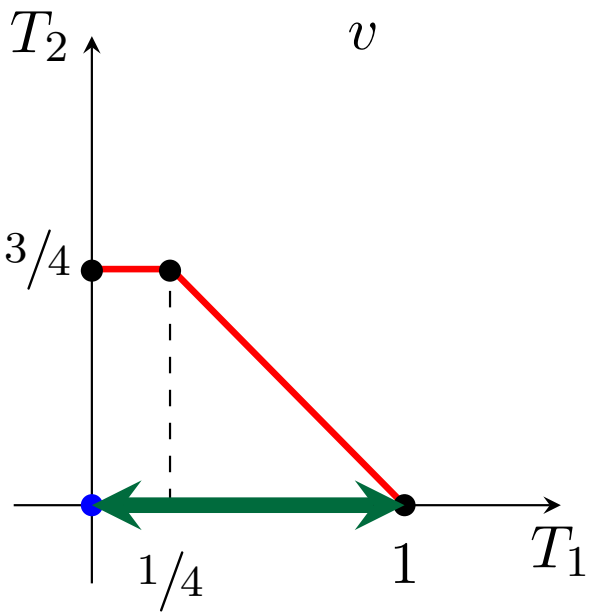
- Consider an abstract state v
- Check the distance between the approximations
- If the distance is too big $\rightsquigarrow$ split the state v

But **how** do we split ?

Refining the abstraction

The approximations provide a **quantitative** evaluation of the abstraction's quality and indicate on how to **refine** it.

Example.



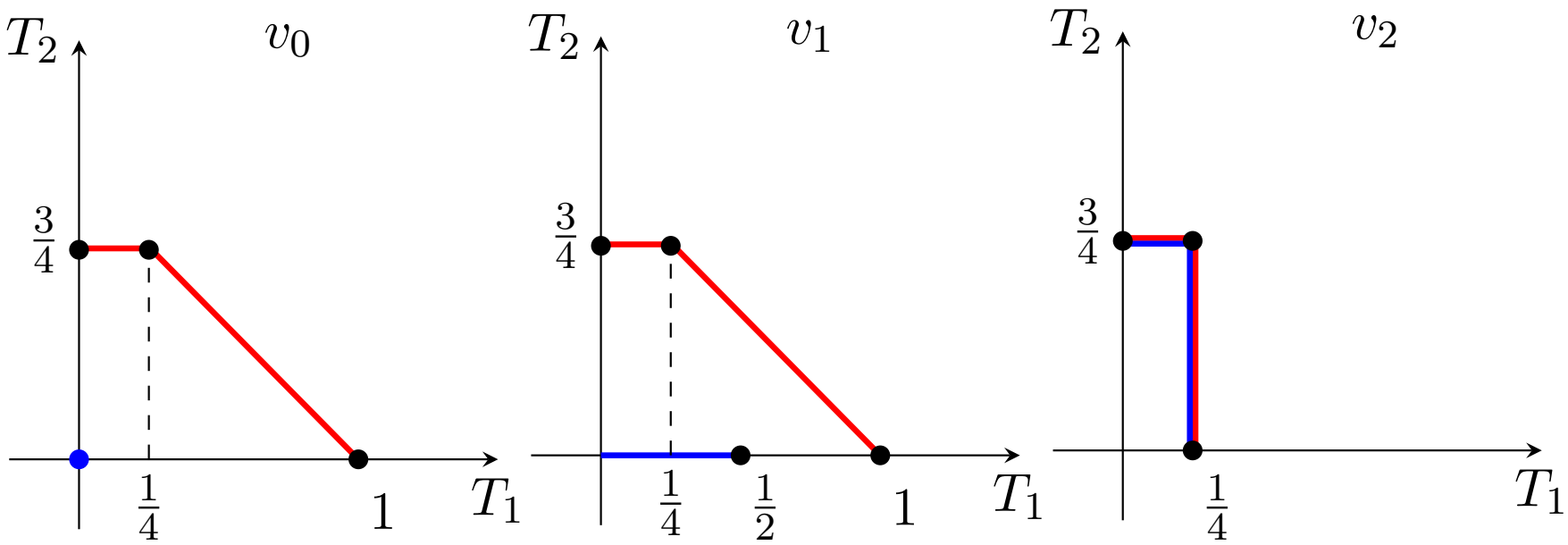
- Consider an abstract state v
- Check the distance between the approximations
- If the distance is too big $\rightsquigarrow$ split the state v

But **how** do we split ?

We look at the approximations of the concrete states **contained** in v.

Splitting an abstract state

Example. For $\boldsymbol{d} = (1, 0)$:

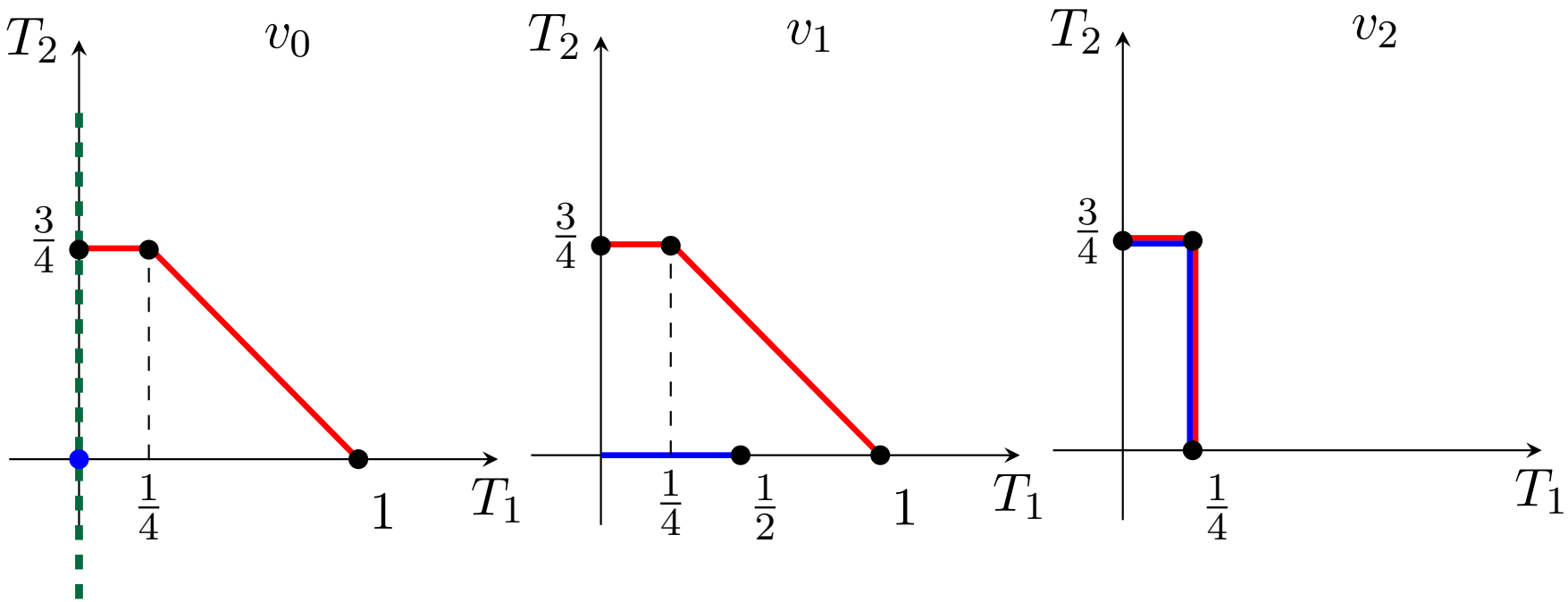


$\mathcal{S}^{\boldsymbol{d}}_{\min}(v) =$ and $\mathcal{S}^{\boldsymbol{d}}_{\max}(v) =$

$$\mathcal{S}^{\boldsymbol{d}}_{\min}(v) = \operatorname*{argmin}_{s \in v} \max_{\boldsymbol{x} \in s.\mathsf{L}} \langle \boldsymbol{x}, \boldsymbol{d} \rangle \quad \text{and} \quad \mathcal{S}^{\boldsymbol{d}}_{\max}(v) = \operatorname*{argmax}_{s \in v} \max_{\boldsymbol{x} \in s.\mathsf{U}} \langle \boldsymbol{x}, \boldsymbol{d} \rangle$$

Splitting an abstract state

Example. For $\boldsymbol{d} = (1, 0)$:

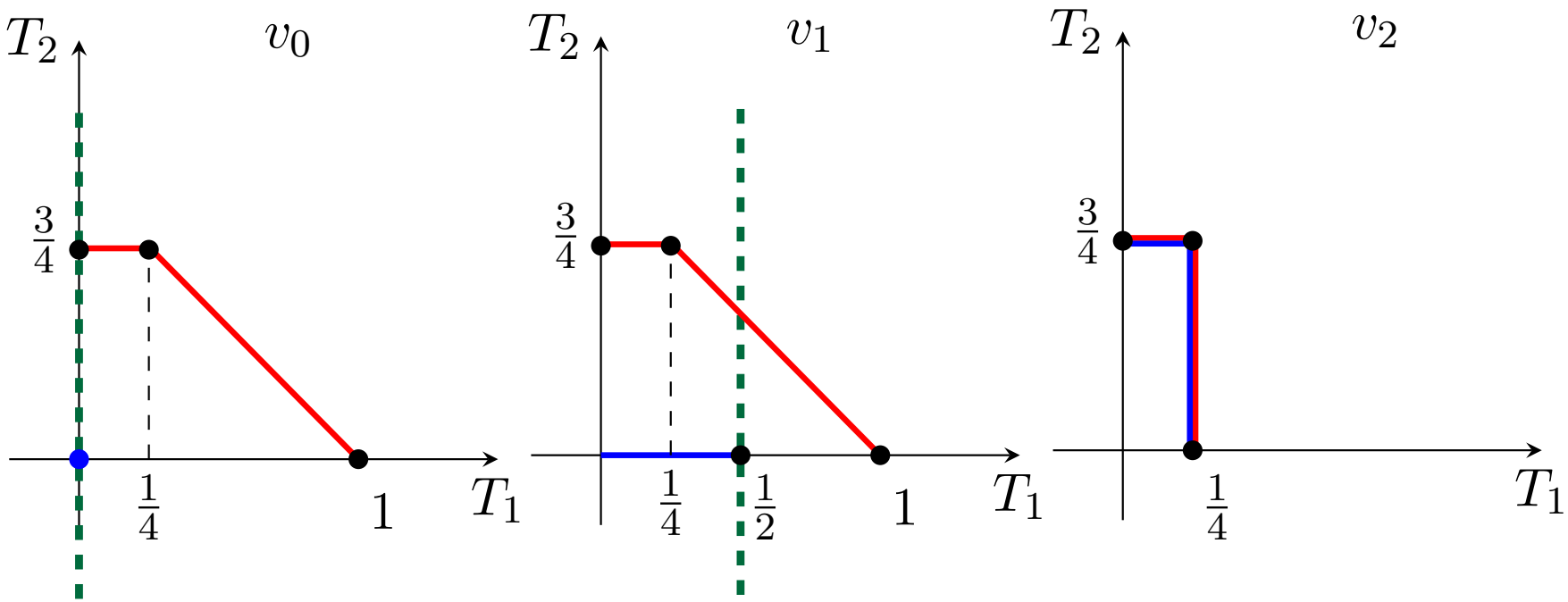


$\mathcal{S}^{\boldsymbol{d}}_{\min}(v) =$ and $\mathcal{S}^{\boldsymbol{d}}_{\max}(v) =$

$$\mathcal{S}^{\boldsymbol{d}}_{\min}(v) = \operatorname*{argmin}_{s \in v} \max_{\boldsymbol{x} \in s.\mathsf{L}} \langle \boldsymbol{x}, \boldsymbol{d} \rangle \quad \text{and} \quad \mathcal{S}^{\boldsymbol{d}}_{\max}(v) = \operatorname*{argmax}_{s \in v} \max_{\boldsymbol{x} \in s.\mathsf{U}} \langle \boldsymbol{x}, \boldsymbol{d} \rangle$$

Splitting an abstract state

Example. For $\boldsymbol{d} = (1, 0)$:

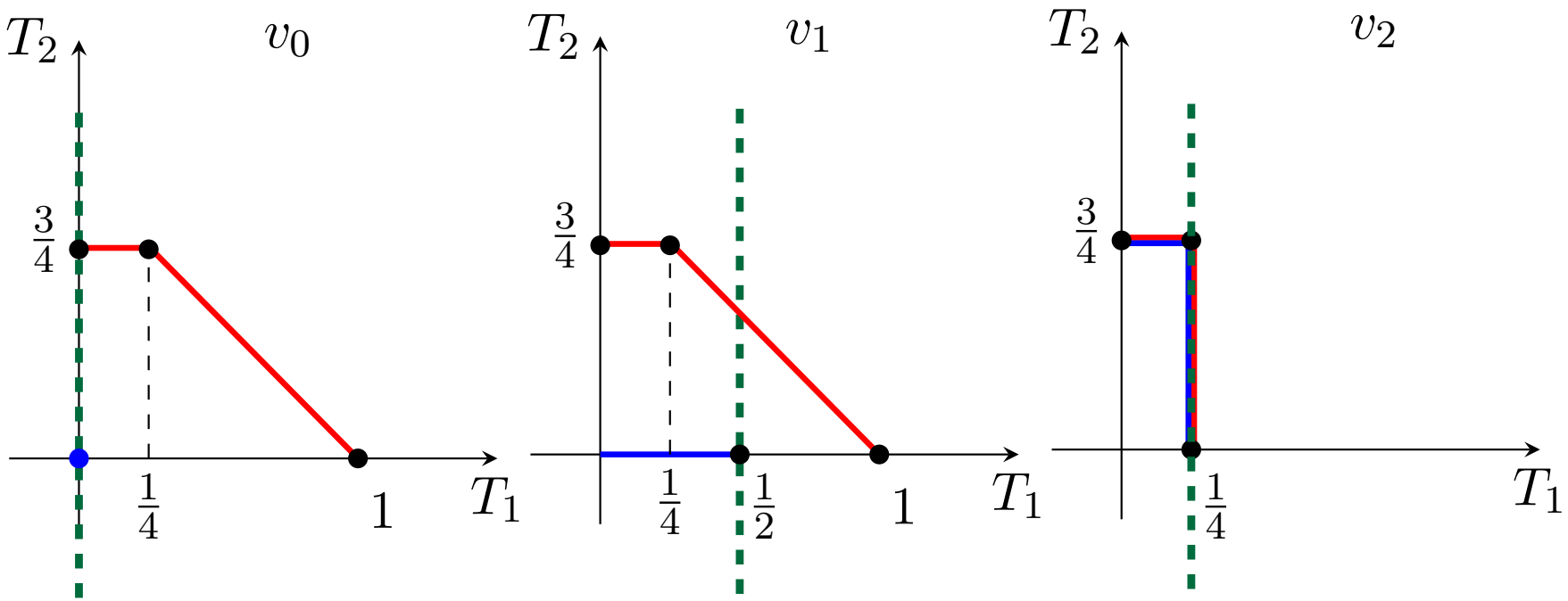


$\mathcal{S}^{\boldsymbol{d}}_{\min}(v) =$ and $\mathcal{S}^{\boldsymbol{d}}_{\max}(v) =$

$$\mathcal{S}^{\boldsymbol{d}}_{\min}(v) = \operatorname*{argmin}_{s \in v} \max_{\boldsymbol{x} \in s.\mathsf{L}} \langle \boldsymbol{x}, \boldsymbol{d} \rangle \quad \text{and} \quad \mathcal{S}^{\boldsymbol{d}}_{\max}(v) = \operatorname*{argmax}_{s \in v} \max_{\boldsymbol{x} \in s.\mathsf{U}} \langle \boldsymbol{x}, \boldsymbol{d} \rangle$$

Splitting an abstract state

Example. For $\boldsymbol{d} = (1, 0)$:

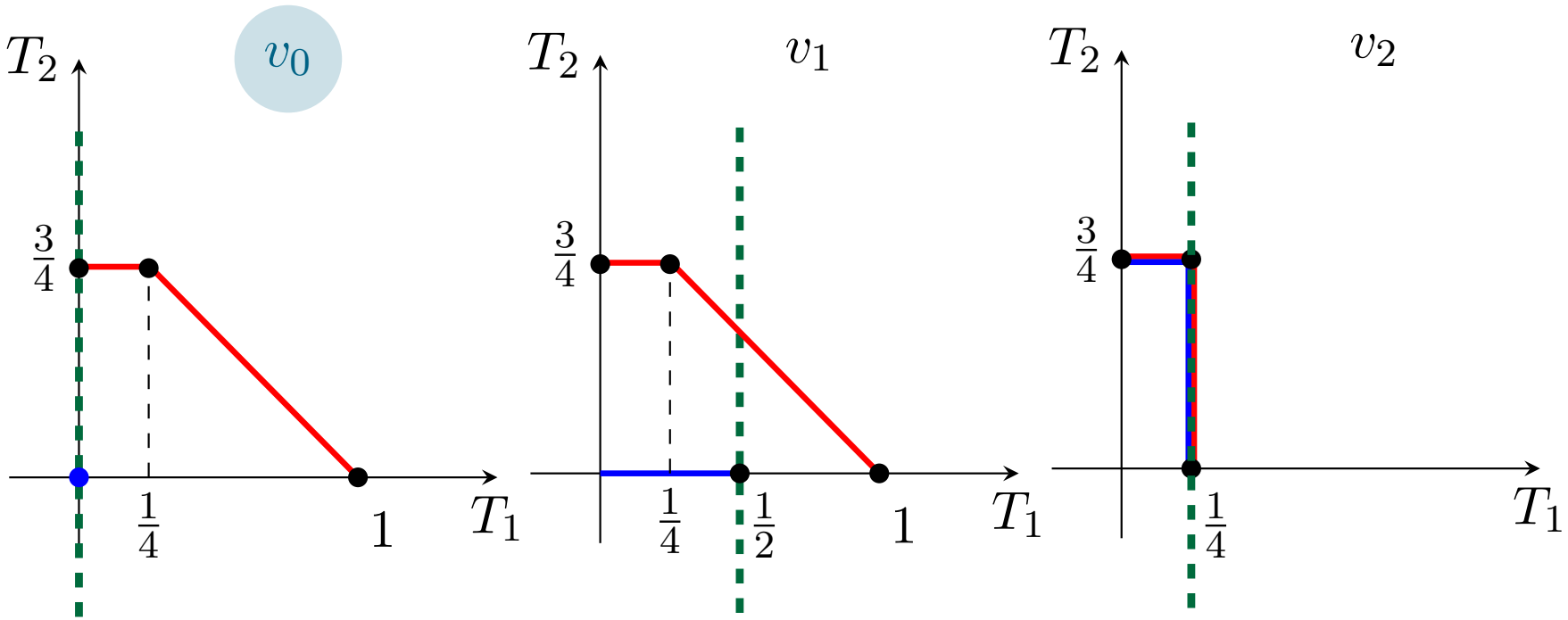


$\mathcal{S}^{\boldsymbol{d}}_{\min}(v) =$ and $\mathcal{S}^{\boldsymbol{d}}_{\max}(v) =$

$$\mathcal{S}^{\boldsymbol{d}}_{\min}(v) = \operatorname*{argmin}_{s \in v} \max_{\boldsymbol{x} \in s.\mathsf{L}} \langle \boldsymbol{x}, \boldsymbol{d} \rangle \quad \text{and} \quad \mathcal{S}^{\boldsymbol{d}}_{\max}(v) = \operatorname*{argmax}_{s \in v} \max_{\boldsymbol{x} \in s.\mathsf{U}} \langle \boldsymbol{x}, \boldsymbol{d} \rangle$$

Splitting an abstract state

Example. For $\boldsymbol{d} = (1, 0)$:

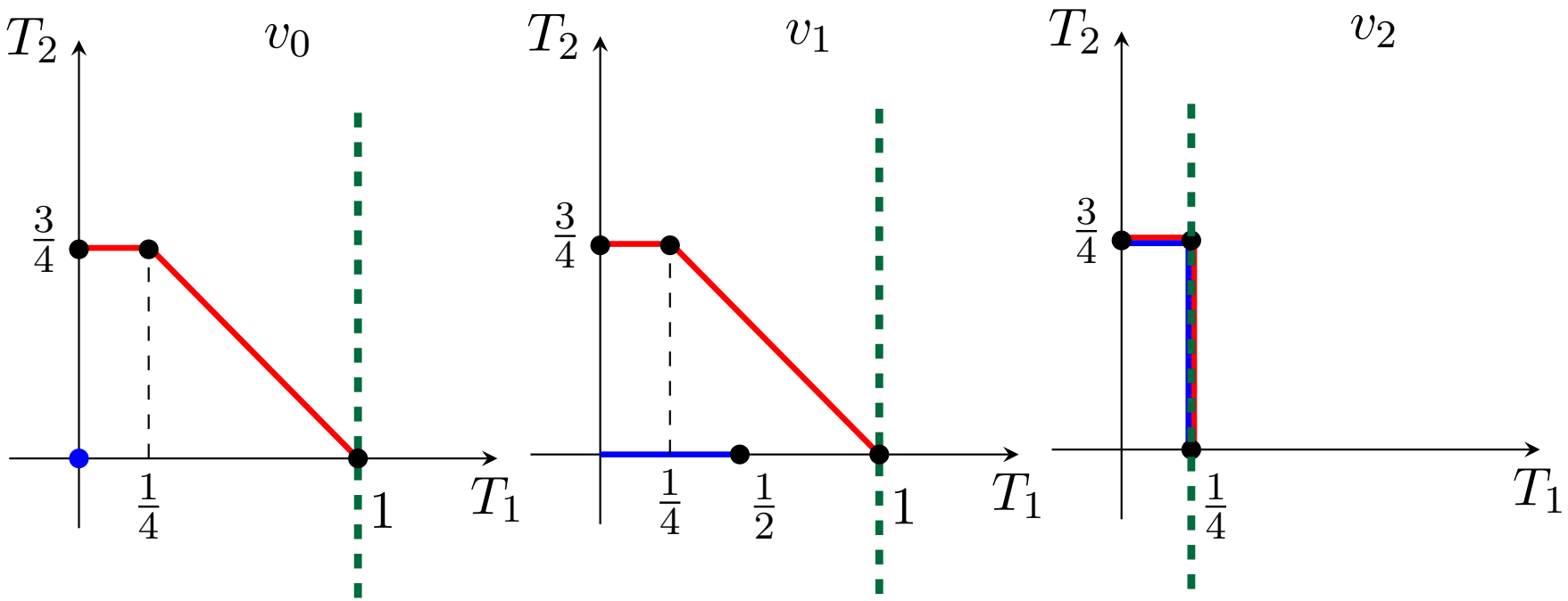


$\mathcal{S}^{\boldsymbol{d}}_{\min}(v) = \{v_0\}$ and $\mathcal{S}^{\boldsymbol{d}}_{\max}(v) =$

$$\mathcal{S}^{\boldsymbol{d}}_{\min}(v) = \operatorname*{argmin}_{s \in v} \max_{\boldsymbol{x} \in s.\mathsf{L}} \langle \boldsymbol{x}, \boldsymbol{d} \rangle \quad \text{and} \quad \mathcal{S}^{\boldsymbol{d}}_{\max}(v) = \operatorname*{argmax}_{s \in v} \max_{\boldsymbol{x} \in s.\mathsf{U}} \langle \boldsymbol{x}, \boldsymbol{d} \rangle$$

Splitting an abstract state

Example. For $\boldsymbol{d} = (1, 0)$:

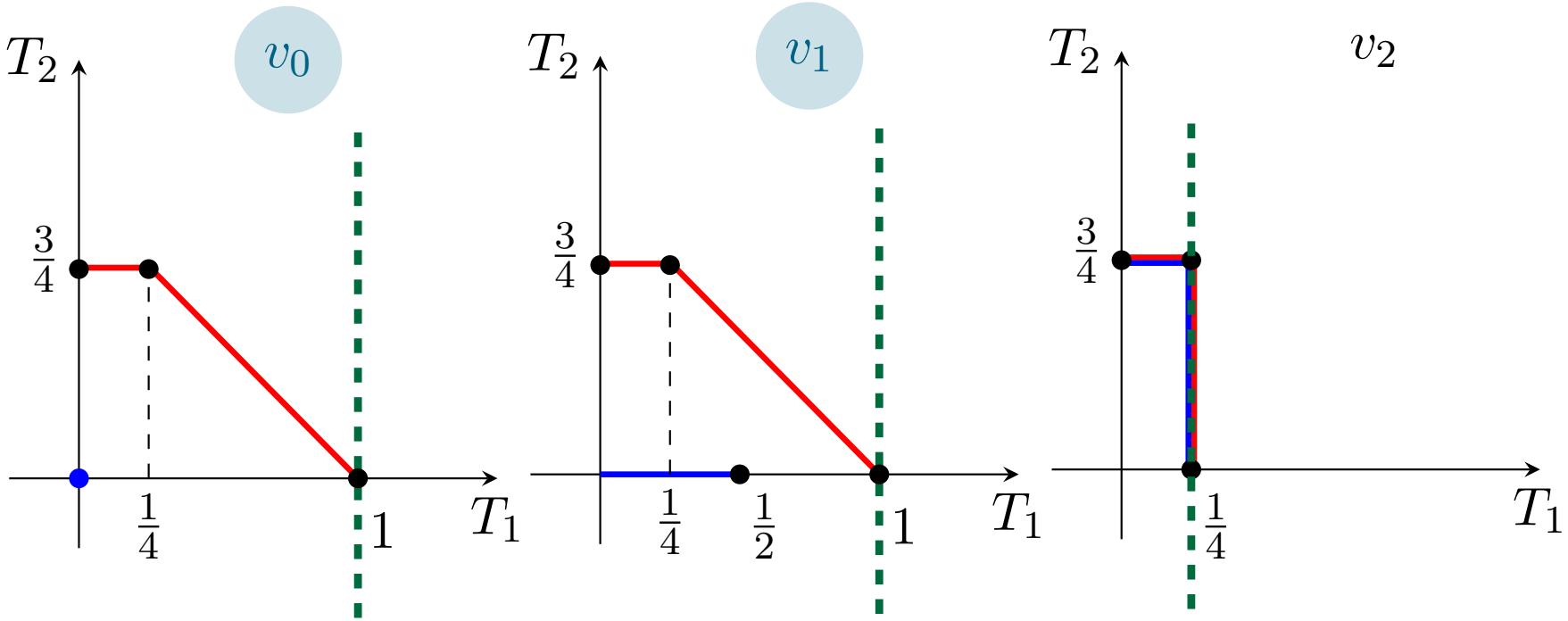


$\mathcal{S}^{\boldsymbol{d}}_{\min}(v) = \{v_0\}$ and $\mathcal{S}^{\boldsymbol{d}}_{\max}(v) =$

$$\mathcal{S}^{\boldsymbol{d}}_{\min}(v) = \operatorname*{argmin}_{s \in v} \max_{\boldsymbol{x} \in s.\mathsf{L}} \langle \boldsymbol{x}, \boldsymbol{d} \rangle \quad \text{and} \quad \mathcal{S}^{\boldsymbol{d}}_{\max}(v) = \operatorname*{argmax}_{s \in v} \max_{\boldsymbol{x} \in s.\mathsf{U}} \langle \boldsymbol{x}, \boldsymbol{d} \rangle$$

Splitting an abstract state

Example. For $\boldsymbol{d} = (1, 0)$:



$\mathcal{S}^{\boldsymbol{d}}_{\min}(v) = \{v_0\}$ and $\mathcal{S}^{\boldsymbol{d}}_{\max}(v) = \{v_0, v_1\}$

$$\mathcal{S}^{\boldsymbol{d}}_{\min}(v) = \operatorname*{argmin}_{s \in v} \max_{\boldsymbol{x} \in s.\mathsf{L}} \langle \boldsymbol{x}, \boldsymbol{d} \rangle \quad \text{and} \quad \mathcal{S}^{\boldsymbol{d}}_{\max}(v) = \operatorname*{argmax}_{s \in v} \max_{\boldsymbol{x} \in s.\mathsf{U}} \langle \boldsymbol{x}, \boldsymbol{d} \rangle$$

Results so far...

- If an abstraction needs to be **refined** then there always exists a direction such that an abstract state is splitted.

Results so far...

- If an abstraction needs to be **refined** then there always exists a direction such that an abstract state is splitted.
- We have an **iterative** algorithm that returns an ε-approximation of the Pareto frontier of an MDP.

Results so far...

- If an abstraction needs to be **refined** then there always exists a direction such that an abstract state is splitted.
- We have an **iterative** algorithm that returns an ε-approximation of the Pareto frontier of an MDP.

What's next ?

- **Implementing** the algorithm and;
- Assessing its performance **in practice**.

Results so far...

- If an abstraction needs to be **refined** then there always exists a direction such that an abstract state is splitted.
- We have an **iterative** algorithm that returns an ε-approximation of the Pareto frontier of an MDP.

What's next ?

- **Implementing** the algorithm and;
- Assessing its performance **in practice**.

Thank you for your attention!

Bibliography


Kattenbelt, Mark et al. “A game-based abstraction-refinement framework for Markov decision processes”. In: *Formal Methods Syst. Des.* 36.3 (2010), pp. 246–280. DOI: 10.1007/s10703-010-0097-6. URL: https://doi.org/10.1007/s10703-010-0097-6.