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

November 20, 2024

Journées du GT Vérif 2024

UMONS Université de Mons

fnrs FREEDOM TO RESEARCH

UNIVERSITAS MASARYKIANA BRUNENSIS

Motivations

- Checking multi-reachability objectives in MDPs is **computationally hard**.
- **Approach:** construct an abstraction of our model which has a smaller size.

Motivations

- Checking multi-reachability objectives in MDPs is **computationally hard**.
- **Approach:** construct an abstraction of our model which has a smaller size.

We give an abstraction-refinement algorithm that:

1. **builds** an abstraction of an MDP and;
2. **refines** it until its behavior is similar "enough" to the one of the MDP with respect to a multi-reachability objective.

Motivations

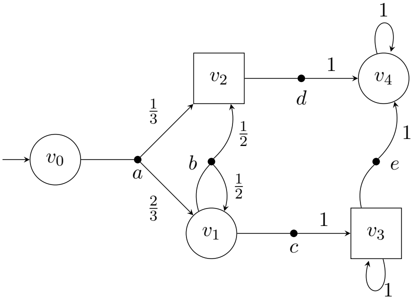
- Checking multi-reachability objectives in MDPs is **computationally hard**.
- **Approach:** construct an abstraction of our model which has a smaller size.

We give an abstraction-refinement algorithm that:

1. **builds** an abstraction of an MDP and;
2. **refines** it until its behavior is similar "enough" to the one of the MDP with respect to a multi-reachability objective.

- We abstract our model as a two-player stochastic game in order to compute a lower and an upper **approximation** of the Pareto frontier.

Context

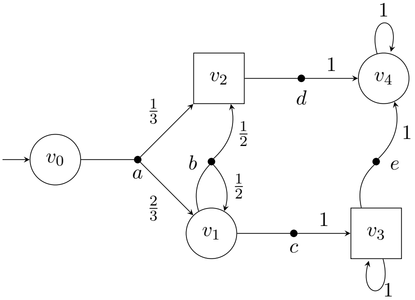


Two-player stochastic game:

- A finite set of states $V = V_1 \cup V_2$
- An initial state v_{init}
- A set of actions A
- A probabilistic transition function $\tau : V \times A \to \mathcal{D}(V)$

A **Markov decision process** is a stochastic game where $V_2 = \emptyset$.

Context



Two-player stochastic game:

- A finite set of states $V = V_1 \cup V_2$
- An initial state v_{init}
- A set of actions A
- A probabilistic transition function $\tau : V \times A \to \mathcal{D}(V)$

A **Markov decision process** is a stochastic game where $V_2 = \emptyset$.

- Plays are **infinite sequences** $\pi = v_0 a_0 v_1 a_1 \dots$ where $\tau(v_i, a_i, v_{i+1}) > 0$ for all $i \in \mathbb{N}$.
- Histories are **finite prefixes** $h = v_0 a_0 \dots a_{n-1} v_n$ of a play ending in a state, the last state of h is $\text{last}(h)$.

Strategies

A strategy for $\mathcal{P}_i$ is a function $\sigma_i : \mathsf{Hists_i}(\mathcal{G}) \to \mathcal{D}(A)$ that **respects the structure** of $\mathcal{G}$.

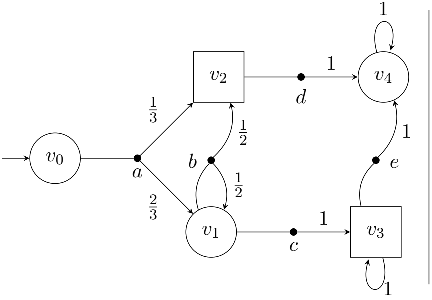
- **Memoryless** strategies are of the form $\sigma_i : V_i \to \mathcal{D}(A)$.
- **Pure** strategies are of the form $\sigma_i : \mathsf{Hists_i}(\mathcal{G}) \to A$.

Strategies

A strategy for $\mathcal{P}_i$ is a function $\sigma_i : \mathsf{Hists_i}(\mathcal{G}) \to \mathcal{D}(A)$ that **respects the structure** of $\mathcal{G}$.

- **Memoryless** strategies are of the form $\sigma_i : V_i \to \mathcal{D}(A)$.
- **Pure** strategies are of the form $\sigma_i : \mathsf{Hists_i}(\mathcal{G}) \to A$.

We denote by $\mathcal{G}^{\sigma_1,\sigma_2}$, the Markov chain **induced** by strategies σ_1 and σ_2.
Example:



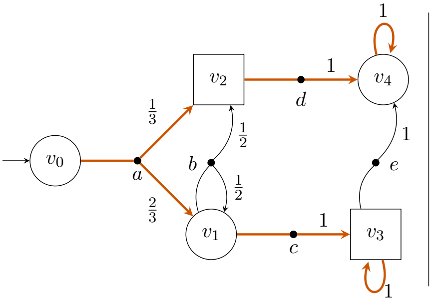
Strategies

A strategy for $\mathcal{P}_i$ is a function $\sigma_i : \mathsf{Hists_i}(\mathcal{G}) \to \mathcal{D}(A)$ that **respects the structure** of $\mathcal{G}$.

- **Memoryless** strategies are of the form $\sigma_i : V_i \to \mathcal{D}(A)$.
- **Pure** strategies are of the form $\sigma_i : \mathsf{Hists_i}(\mathcal{G}) \to A$.

We denote by $\mathcal{G}^{\sigma_1,\sigma_2}$, the Markov chain **induced** by strategies σ_1 and σ_2.

Example:

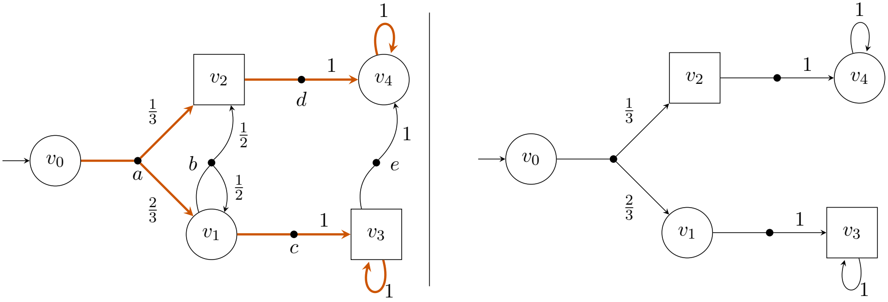


Strategies

A strategy for $\mathcal{P}_i$ is a function $\sigma_i : \mathsf{Hists_i}(\mathcal{G}) \to \mathcal{D}(A)$ that **respects the structure** of $\mathcal{G}$.

- **Memoryless** strategies are of the form $\sigma_i : \; V_i \; \to \mathcal{D}(A)$.
- **Pure** strategies are of the form $\sigma_i : \mathsf{Hists_i}(\mathcal{G}) \to \; A \;$.

We denote by $\mathcal{G}^{\sigma_1,\sigma_2}$, the Markov chain **induced** by strategies σ_1 and σ_2.
Example:



Reachability objectives

For $T \subseteq V$ a set of states, a reachability objective is defined by

$$\Diamond T = \{\pi \in \mathsf{Plays}(\mathcal{G}) \mid \exists\, i \in \mathbb{N},\, \pi[i] \in T\}.$$

- We denote by $\mathbb{P}_s^{\sigma_1,\sigma_2}(\Diamond T)$ the **probability to reach** T from s in the Markov chain induced by σ_1 and σ_2.

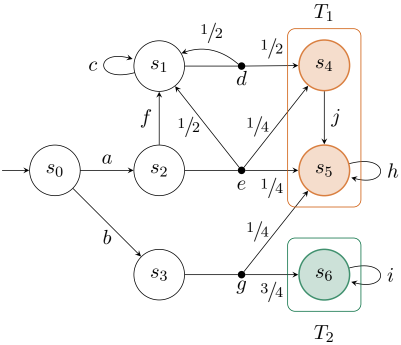
Multi-reachability objectives

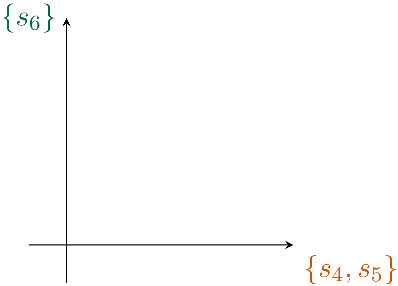
Multi-reachability objectives are vectors of sets of states $(T_1, \ldots, T_n)$ that we want to **reach**.

Multi-reachability objectives

Multi-reachability objectives are vectors of sets of states $(T_1, \ldots, T_n)$ that we want to **reach**.

> A threshold vector $\boldsymbol{\alpha} \in [0,1]^n$ is **achievable** if there exists a strategy σ_1 of $\mathcal{P}_1$ such that for all strategies σ_2 of $\mathcal{P}_2$, we have that $\mathbb{P}_{\mathcal{G}}^{\sigma_1,\sigma_2}(\Diamond T_i) \geq \alpha_i$ for all $1 \leq i \leq n$.

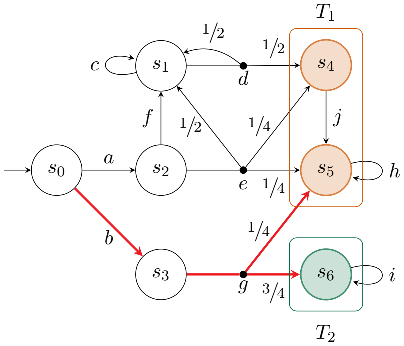


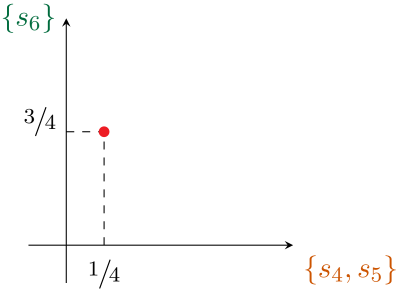


Multi-reachability objectives

Multi-reachability objectives are vectors of sets of states $(T_1, \ldots, T_n)$ that we want to **reach**.

A threshold vector $\boldsymbol{\alpha} \in [0,1]^n$ is **achievable** if there exists a strategy σ_1 of $\mathcal{P}_1$ such that for all strategies σ_2 of $\mathcal{P}_2$, we have that $\mathbb{P}_{\mathcal{G}}^{\sigma_1,\sigma_2}(\Diamond T_i) \geq \alpha_i$ for all $1 \leq i \leq n$.

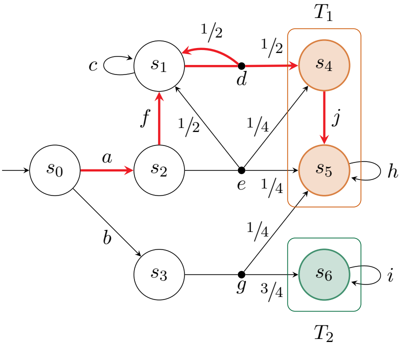


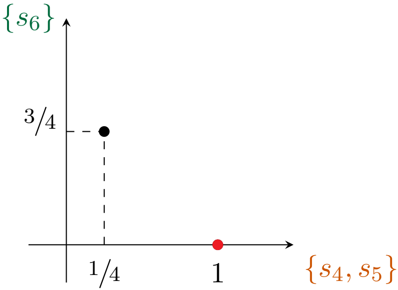


Multi-reachability objectives

Multi-reachability objectives are vectors of sets of states $(T_1, \ldots, T_n)$ that we want to **reach**.

A threshold vector $\boldsymbol{\alpha} \in [0,1]^n$ is **achievable** if there exists a strategy σ_1 of $\mathcal{P}_1$ such that for all strategies σ_2 of $\mathcal{P}_2$, we have that $\mathbb{P}_{\mathcal{G}}^{\sigma_1,\sigma_2}(\Diamond T_i) \geq \alpha_i$ for all $1 \leq i \leq n$.

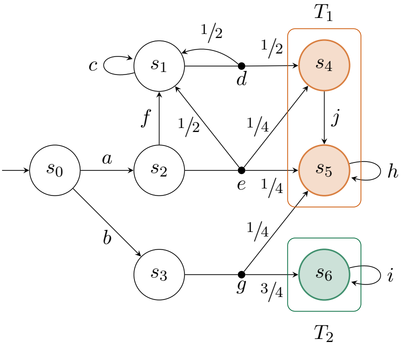


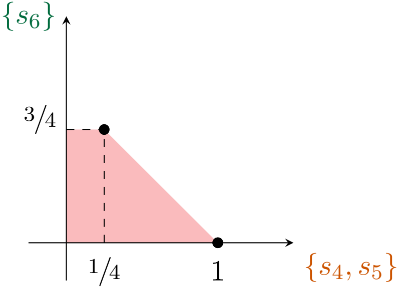


Multi-reachability objectives

Multi-reachability objectives are vectors of sets of states $(T_1, \ldots, T_n)$ that we want to **reach**.

> A threshold vector $\boldsymbol{\alpha} \in [0,1]^n$ is **achievable** if there exists a strategy σ_1 of $\mathcal{P}_1$ such that for all strategies σ_2 of $\mathcal{P}_2$, we have that $\mathbb{P}_{\mathcal{G}}^{\sigma_1,\sigma_2}(\Diamond T_i) \geq \alpha_i$ for all $1 \leq i \leq n$.



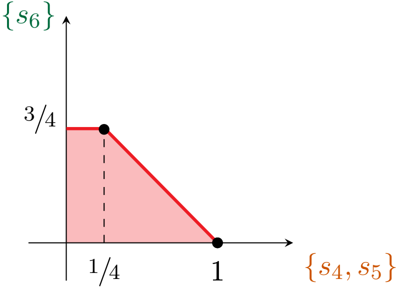


Multi-reachability objectives

Multi-reachability objectives are vectors of sets of states $(T_1, \ldots, T_n)$ that we want to **reach**.

A threshold vector $\boldsymbol{\alpha} \in [0,1]^n$ is **achievable** if there exists a strategy σ_1 of $\mathcal{P}_1$ such that for all strategies σ_2 of $\mathcal{P}_2$, we have that $\mathbb{P}_{\mathcal{G}}^{\sigma_1,\sigma_2}(\Diamond T_i) \geq \alpha_i$ for all $1 \leq i \leq n$.

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**.



Game-based abstraction

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

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

⇝ Approximate the Pareto frontier through a smaller model.

Introducing a new form of nondeterminism

Pessimistic

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 MDP by **merging** states together.

⇝ 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 MDP 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

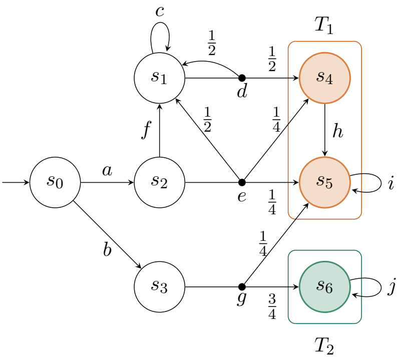
- ▷ Players are cooperating (MDP)
- ▷ Upper approximation of the Pareto frontier

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

Example of GBA

Let us consider the partition $P = \{\{s_0, s_1, s_2, s_3\}, \{s_4, s_5\}, \{s_6\}\}$.

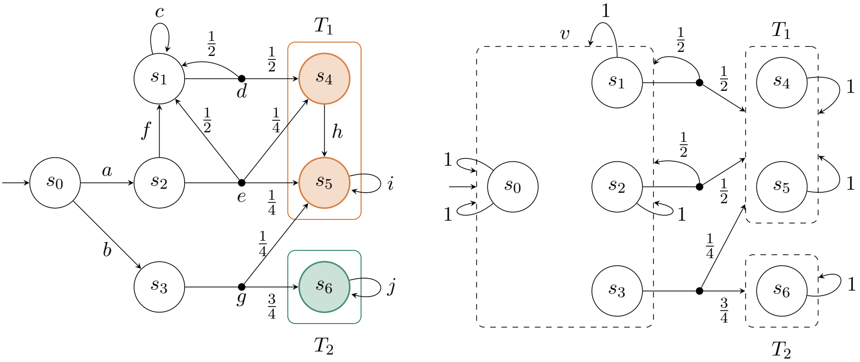
1. Lift the transition function to the partition.



Example of GBA

Let us consider the partition $P = \{\{s_0, s_1, s_2, s_3\}, \{s_4, s_5\}, \{s_6\}\}$.

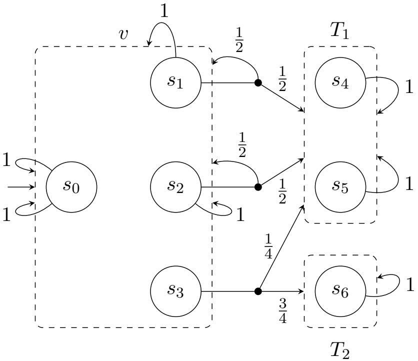
1. Lift the transition function to the partition.



Example of GBA

Let us consider the partition $P = \{\{s_0, s_1, s_2, s_3\}, \{s_4, s_5\}, \{s_6\}\}$.

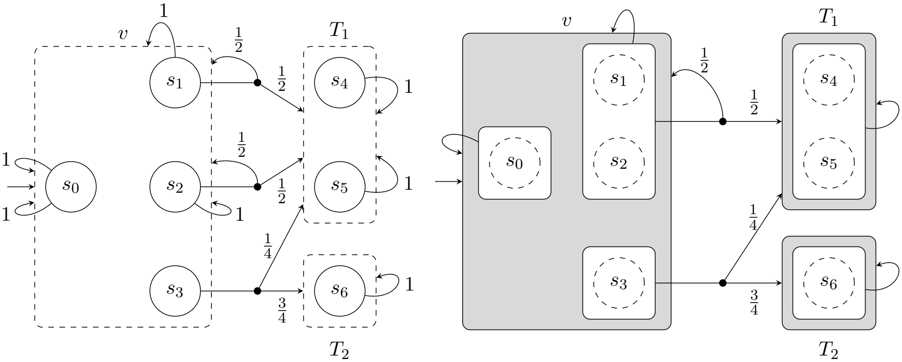
1. Lift the transition function to the partition.
2. Group states that have similar behavior with respect to the partition.



Example of GBA

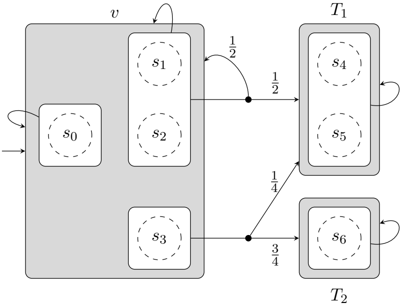
Let us consider the partition $P = \{\{s_0, s_1, s_2, s_3\}, \{s_4, s_5\}, \{s_6\}\}$.

1. Lift the transition function to the partition.
2. Group states that have similar behavior with respect to the partition.



Resolving the nondeterminism

- ▷ White states are **concrete** states from the MDP,
- ▷ Grey states are **abstract** states, i.e., group of concrete states.



In a play, we alternate between:

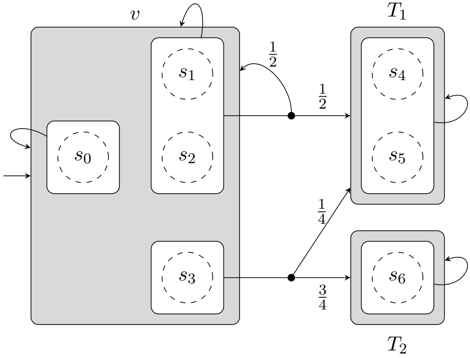
1. In an abstract state $\rightsquigarrow$ choosing a concrete state;
2. In a concrete state $\rightsquigarrow$ choosing an action of the MDP.

Resolving the nondeterminism

Depending on whether we want a lower or an upper approximation of the Pareto frontier, we have:

Optimistic: both types of states are controlled by only one player (MDP).

Pessimistic: abstract states are controlled by $\mathcal{P}_2$ and the concrete ones by $\mathcal{P}_1$ (SG).



Abstraction-refinement algorithm

1. Build a **game-based abstraction** $\mathcal{G}$ of the MDP following a partition;

Abstraction-refinement algorithm

1. Build a **game-based abstraction** $\mathcal{G}$ of the MDP following a partition;
2. For each state of $\mathcal{G}$: compute its **lower and upper frontier**;

Abstraction-refinement algorithm

1. Build a **game-based abstraction** $\mathcal{G}$ of the MDP following a partition;
2. For each state of $\mathcal{G}$: compute its **lower and upper frontier**;
3. For each abstract state:

Abstraction-refinement algorithm

1. Build a **game-based abstraction** $\mathcal{G}$ of the MDP following a partition;
2. For each state of $\mathcal{G}$: compute its **lower and upper frontier**;
3. For each abstract state:
   - Compute the **maximal distance** between its lower and upper frontier;

Abstraction-refinement algorithm

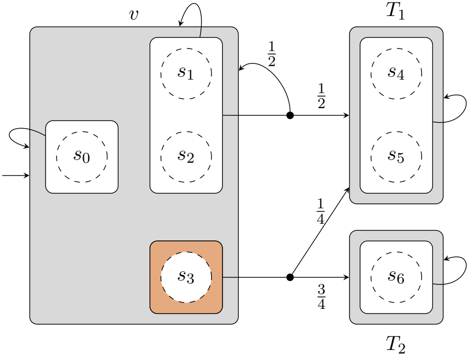
1. Build a **game-based abstraction** $\mathcal{G}$ of the MDP following a partition;
2. For each state of $\mathcal{G}$: compute its **lower and upper frontier**;
3. For each abstract state:
   - ▷ Compute the **maximal distance** between its lower and upper frontier;
   - ▷ If this distance is **too big**: **refine the partition** by splitting the abstract state;

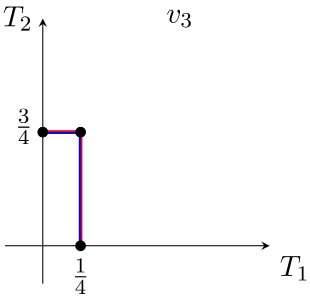
Abstraction-refinement algorithm

1. Build a **game-based abstraction** $\mathcal{G}$ of the MDP following a partition;
2. For each state of $\mathcal{G}$: compute its **lower and upper frontier**;
3. For each abstract state:
   - Compute the **maximal distance** between its lower and upper frontier;
   - If this distance is **too big**: **refine the partition** by splitting the abstract state;
4. Repeat until the abstract states no longer need to be refined.

Computing lower and upper frontiers

- Approximating the upper frontier via [FKP12][2].
- Approximating the lower frontier using the value-iteration approach of [ACK+20][3].

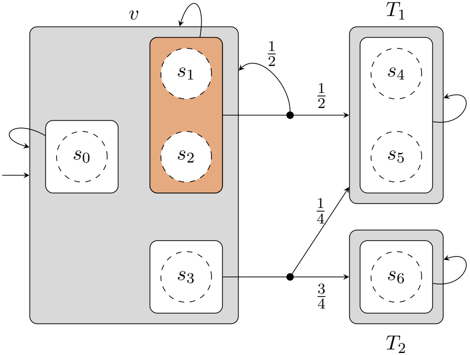


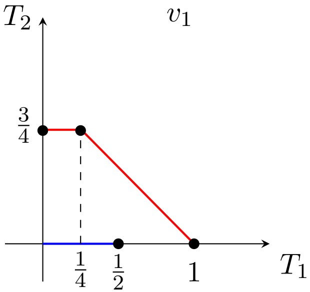


[2] Forejt, Kwiatkowska, and Parker, "Pareto Curves for Probabilistic Model Checking".
[3] Ashok et al., "Approximating Values of Generalized-Reachability Stochastic Games".

Computing lower and upper frontiers

- Approximating the upper frontier via [FKP12][2].
- Approximating the lower frontier using the value-iteration approach of [ACK+20][3].

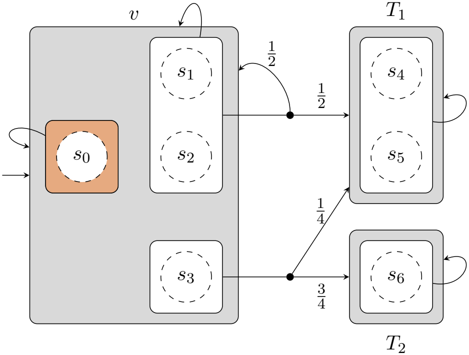


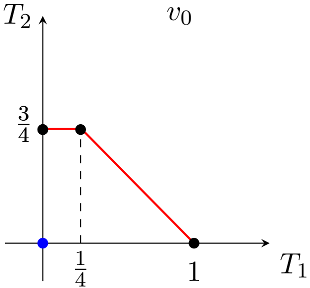


[2]Forejt, Kwiatkowska, and Parker, "Pareto Curves for Probabilistic Model Checking".
[3]Ashok et al., "Approximating Values of Generalized-Reachability Stochastic Games".

Computing lower and upper frontiers

- Approximating the upper frontier via [FKP12][2].
- Approximating the lower frontier using the value-iteration approach of [ACK+20][3].

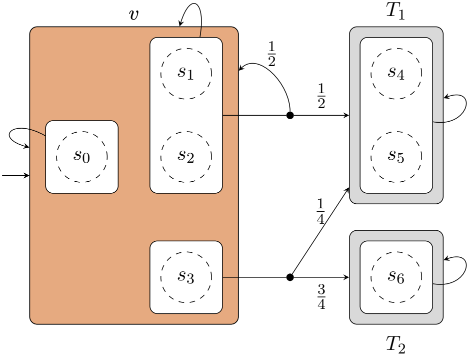


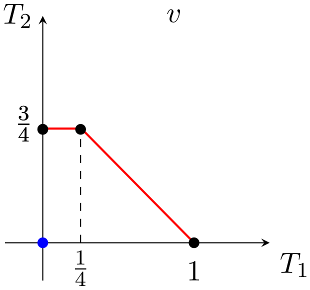


[2] Forejt, Kwiatkowska, and Parker, "Pareto Curves for Probabilistic Model Checking".
[3] Ashok et al., "Approximating Values of Generalized-Reachability Stochastic Games".

Computing lower and upper frontiers

- Approximating the upper frontier via [FKP12][2].
- Approximating the lower frontier using the value-iteration approach of [ACK+20][3].



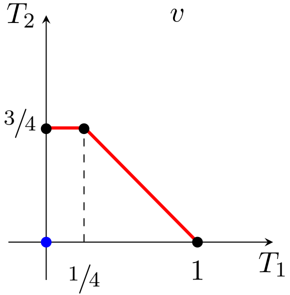


[2] Forejt, Kwiatkowska, and Parker, "Pareto Curves for Probabilistic Model Checking".
[3] Ashok et al., "Approximating Values of Generalized-Reachability Stochastic Games".

Refining the abstraction

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

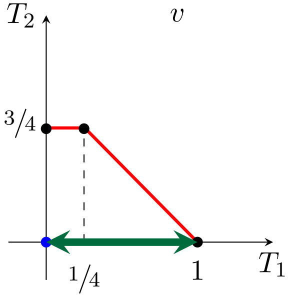
Example.



Refining the abstraction

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

Example.



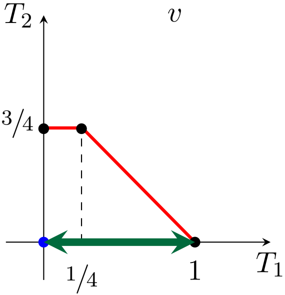
- Consider an abstract state v
- Check the distance between the approximations
- If the distance is too big $\rightsquigarrow$ split the state v

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.



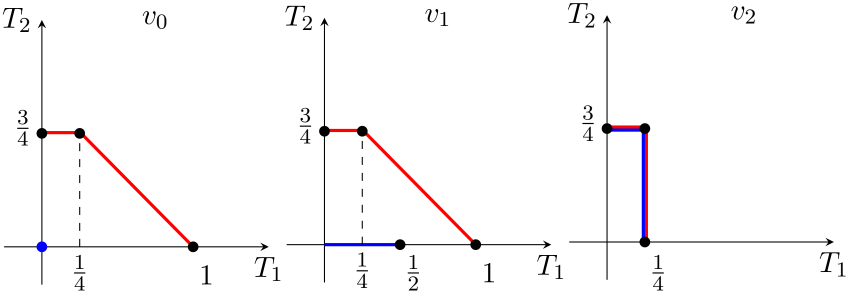
- Consider an abstract state v
- Check the distance between the approximations
- If the distance is too big $\rightsquigarrow$ split the state v

But **how** do we split ?

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

Splitting an abstract state

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

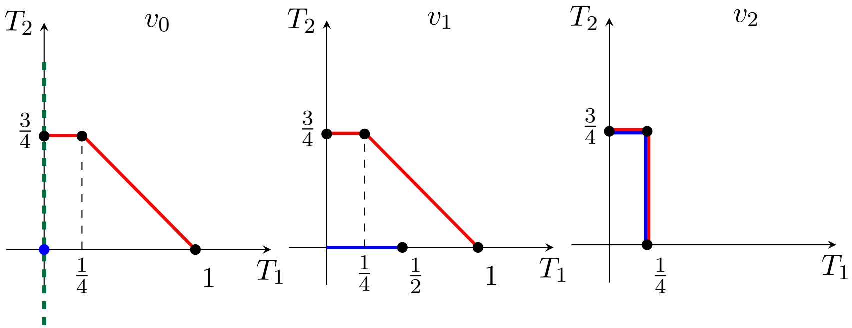


$\mathcal{S}^{\boldsymbol{d}}_{\min}(v) =$ and $\mathcal{S}^{\boldsymbol{d}}_{\max}(v) =$

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

Splitting an abstract state

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

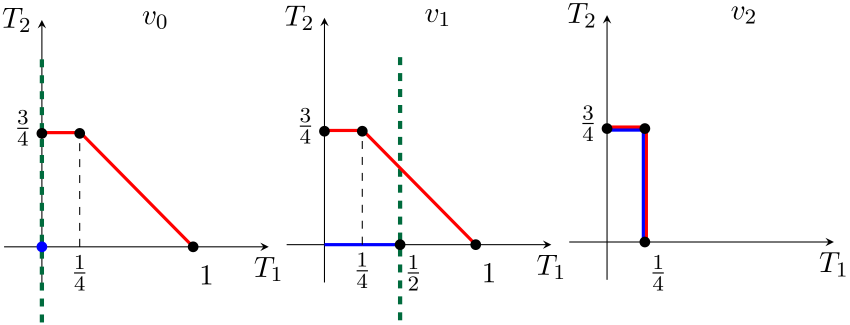


$\mathcal{S}^{\boldsymbol{d}}_{\min}(v) =$ and $\mathcal{S}^{\boldsymbol{d}}_{\max}(v) =$

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

Splitting an abstract state

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

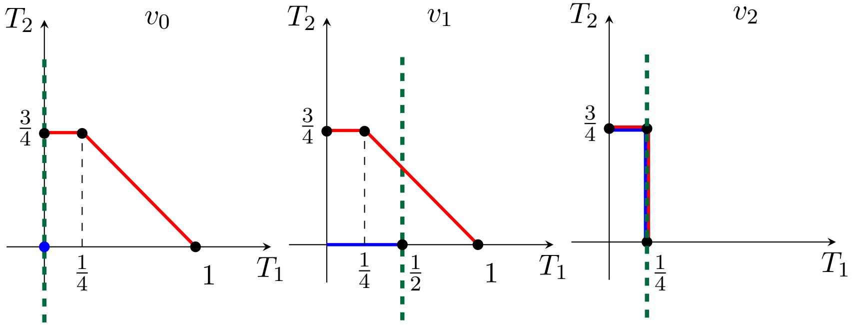


$\mathcal{S}^{\boldsymbol{d}}_{\min}(v) =$ and $\mathcal{S}^{\boldsymbol{d}}_{\max}(v) =$

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

Splitting an abstract state

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

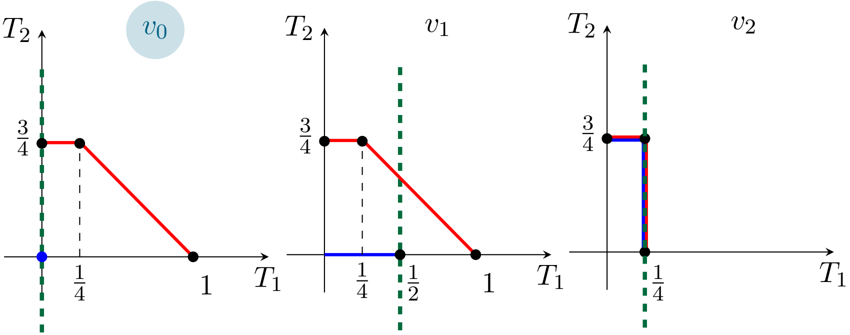


$\mathcal{S}^{\boldsymbol{d}}_{\min}(v) =$ and $\mathcal{S}^{\boldsymbol{d}}_{\max}(v) =$

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

Splitting an abstract state

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

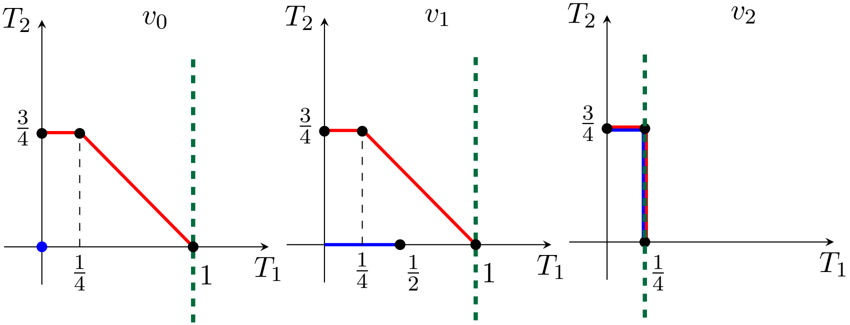


$\mathcal{S}^{\boldsymbol{d}}_{\min}(v) = \{v_0\}$ and $\mathcal{S}^{\boldsymbol{d}}_{\max}(v) =$

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

Splitting an abstract state

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

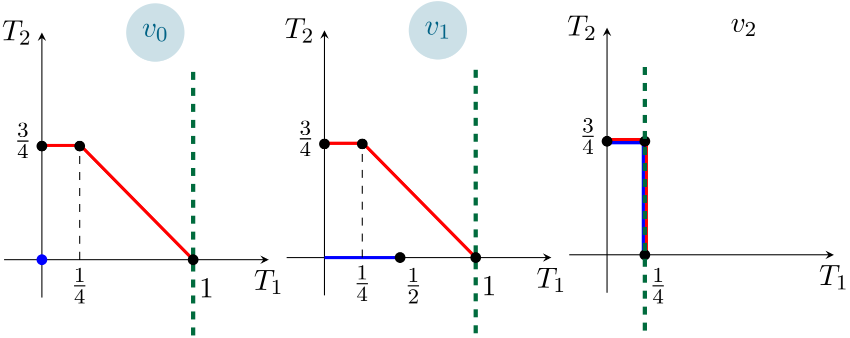


$\mathcal{S}^{\boldsymbol{d}}_{\min}(v) = \{v_0\}$ and $\mathcal{S}^{\boldsymbol{d}}_{\max}(v) =$

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

Splitting an abstract state

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

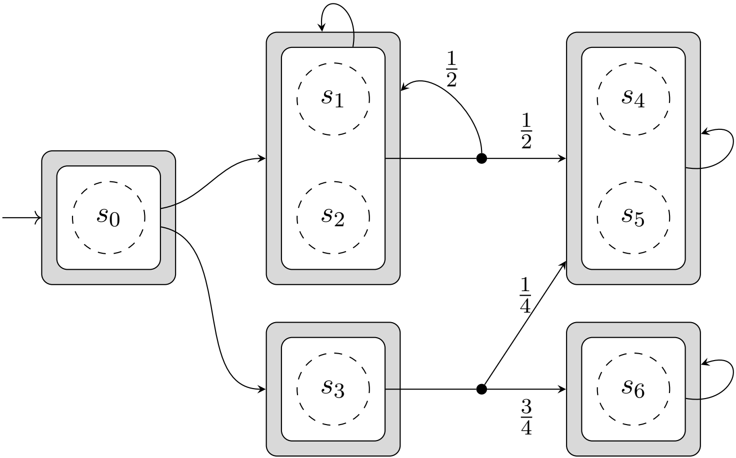


$\mathcal{S}^{\boldsymbol{d}}_{\min}(v) = \{v_0\}$ and $\mathcal{S}^{\boldsymbol{d}}_{\max}(v) = \{v_0, v_1\}$

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

New abstraction

$P_1 = \{\{s_0\}, \{s_1, s_2\}, \{s_3\}, \{s_4, s_5\}, \{s_6\}\}$



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

Ashok, Pranav et al. "Approximating Values of Generalized-Reachability Stochastic Games". In: *LICS '20: 35th Annual ACM/IEEE Symposium on Logic in Computer Science, Saarbrücken, Germany, July 8-11, 2020*. Ed. by Holger Hermanns et al. ACM, 2020, pp. 102–115.

Forejt, Vojtech, Marta Z. Kwiatkowska, and David Parker. "Pareto Curves for Probabilistic Model Checking". In: *Automated Technology for Verification and Analysis - 10th International Symposium, ATVA 2012, Thiruvananthapuram, India, October 3-6, 2012. Proceedings*. Ed. by Supratik Chakraborty and Madhavan Mukund. Vol. 7561. Lecture Notes in Computer Science. Springer, 2012, pp. 317–332. DOI: 10.1007/978-3-642-33386-6_25.

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.