State Abstraction discovery for Model-Based Reinforcement Learning

Séminaire des doctorants du CMAP

Palaiseau City, April 24. 2024

Orso Forghieri

Reinforcement Learning

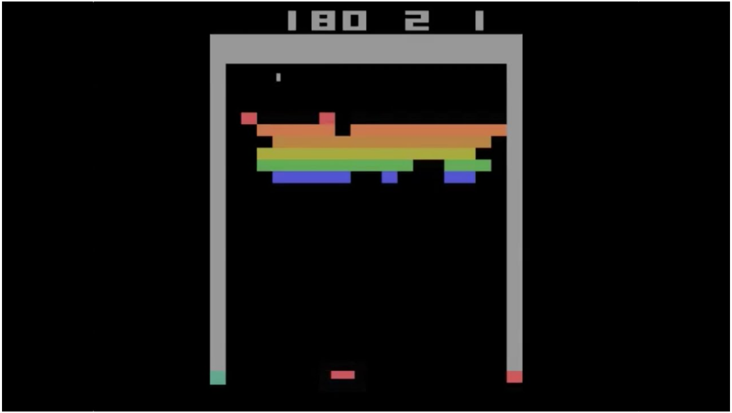


Figure 1: Atari breakout game, 1976.

- Current state: image
- Two actions: left, right
- Objective: maximizing upcoming rewards

Principle

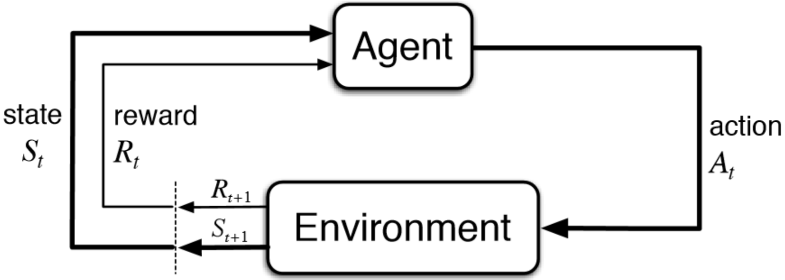


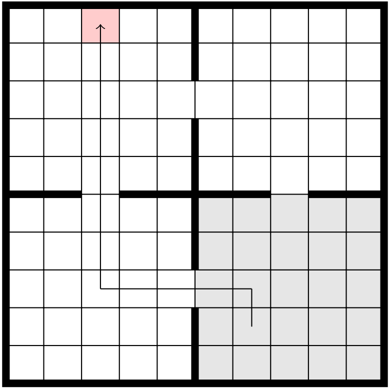
Figure 2: Principle of Reinforcement Learning [Sutton and Barto, 2018]. Agent is modeled by a learned function $\pi : \mathcal{S} \mapsto \mathcal{A}$.

Maximize the expected reward:

$$\max_{\pi \in \mathcal{A}^{\mathcal{S}}} \sum_{t \geq 0} \gamma^t r_t$$

with typically $\gamma = 0.99$

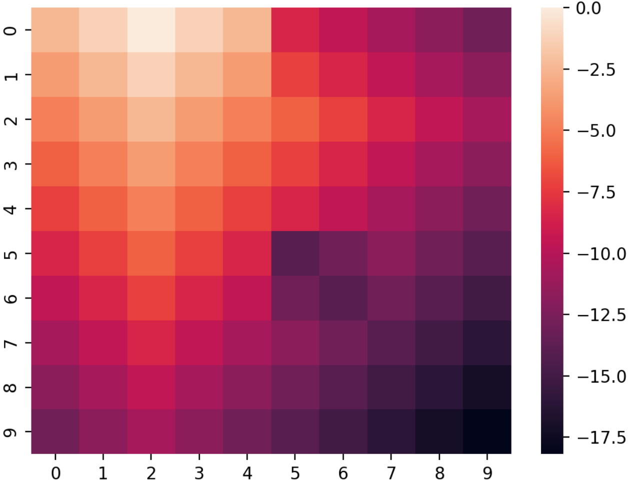
Markov Decision Processes: Four Rooms instance



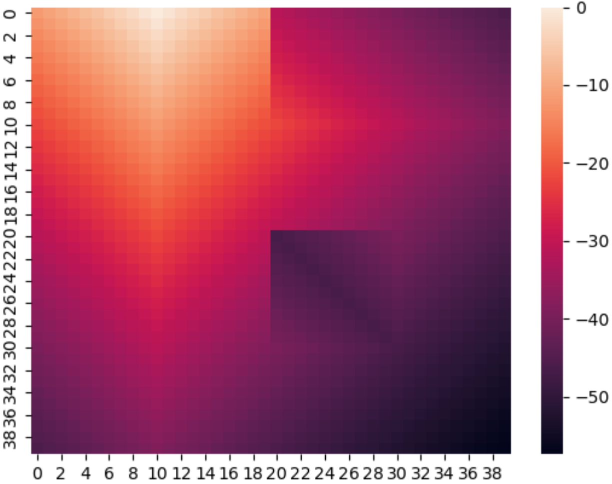
Four rooms:

- $\mathcal{S} = [\![1 \; ; \; 100]\!]$, $\mathcal{A} = \{N, S, E, W\}$
- Reward: -1 until exit is reached, 0 otherwise.
- Step forward with probability .8 (if step is doable)

Four Rooms optimal Value Function V^*



Increasing complexity



Our strategy

Assuming exact knowledge of transition and reward functions, we

- assimilate all states in a **single region**
- create **new regions** for outlying states
- **update the value** function on each region

Which results in

- A partition of the problem that describes its **structure**
- An approximation of the **optimal solution** with arbitrary precision

Table of Contents

Markov Decision Process

Definition (Markov Decision Process)

A Markov Decision Process is defined as:

- A discrete state space $\mathcal{S}$
- An discrete action space $\mathcal{A}$
- Stochastic transition $s_{t+1} \sim T(s_t, a_t, .)$ (one step memory)
- Immediate reward $R(s_t, a_t)$

Markov Decision Process

$$
\begin{aligned}
\text{Solve the MDP} &\iff \text{Maximizing upcoming rewards relatively to } \pi \\
&\iff \max_{\pi \in \mathcal{A}^{\mathcal{S}}} \mathop{\mathbb{E}}_{s_{t+1} \sim T(s_t, a_t, \cdot)} \left[\sum_{t=0}^{\infty} \gamma^t R\left(s_t, \pi(s_t)\right) | s_0 = s \right] \\
&\iff \max_{\pi \in \mathcal{A}^{\mathcal{S}}} V^{\pi}(s)
\end{aligned}
$$

Value Function

Definition (Value function, optimal value function)

Value function of a policy:

$$V^{\pi}(s) = \mathop{\mathbb{E}}_{s_{t+1} \sim T(s_t, a_t, \cdot)} \left[\sum_{t=0}^{\infty} \gamma^t R\left(s_t, \pi(s_t)\right) | s_0 = s \right]$$

Optimal Value Function:

$$V^*(s) = \max_{\pi} V^{\pi}(s)$$

Bellman equations

Theorem (Optimal Bellman equation)

V^* is the unique solution of the optimal Bellman equation:

$$V^*(s) = \max_{a \in \mathcal{A}} \left(R(s,a) + \gamma \sum_{s' \in \mathcal{S}} T(s,a,s') \cdot V^*(s') \right) := \mathcal{T}^* V^*$$

Moreover, the Bellman operator $\mathcal{T}^* : \mathbb{R}^{\mathcal{S}} \mapsto \mathbb{R}^{\mathcal{S}}$ contracts space with factor $\gamma < 1$.

→ Fixed point theorem : iterating $\mathcal{T}^*$ make any V converge to the solution of $V^* = \mathcal{T}^* V^*$

→ But : necessity to update each state n times for large spaces

$$\|V^* - (\mathcal{T}^*)^n V\|_\infty \leq \gamma^n \|V^* - V\|_\infty$$

Bellman operators

Definition (Bellman operators)

For a given policy π, we define the Bellman operator

$$\mathcal{T}^{\pi} : V \to R(s, \pi(s)) + \gamma \sum_{s' \in \mathcal{S}} T(s, \pi(s), s') V(s')$$

with optimal Bellman operator

$$\mathcal{T}^{*} = \max_{\pi} \mathcal{T}^{\pi}$$

Bellman operators

Definition (Q-value)

Let us define the Q-value

$$Q(s, a) = \mathbb{E}\left[\sum_{t \geq 0} \gamma^t r_t | s_0 = s, a_0 = a\right]$$

$\rightarrow$ It is the value function but we also set the first action

we define its Bellman operator

$$\mathcal{T}^\pi : Q(s, a) \rightarrow R(s, a) + \gamma \sum_{s' \in \mathcal{S}} T(s, a, s').Q(s', \pi(s'))$$

MDP solving

Finally,

$$
\begin{aligned}
& \text{Solve an MDP} \\
\iff & \text{Solve } \max_{\pi} V^{\pi} \\
& \text{(Policy Gradient, Actor-Critic, Deep Reinforcement Learning...)} \\
\iff & \text{Solve } \min_{V \in \mathbb{R}^{\mathcal{S}}} \|V - \mathcal{T}^* V\|_{\infty} \\
& (\textbf{Dynamic Programming}, \text{TD-Learning...})
\end{aligned}
$$

Dynamic Programming

Two main approches:

- Value Iteration:

$$\begin{cases} V_0 = 0 \\ V_{t+1} \leftarrow \mathcal{T}^* V_t \end{cases} \quad \text{until } \|V_{t+1} - V_t\|_\infty \leq (1-\gamma)\varepsilon$$

- Policy Iteration:

$$\begin{cases} V_0 = 0 \\ \pi_0 = 0 \\ V_{t+1} = (\mathcal{T}^\pi)^n V_t \text{ (Policy Evaluation)} \\ \pi_{t+1} = \arg\max_{a \in \mathcal{A}} \left(R_a + \gamma T_a \cdot V_{t+1}\right) \end{cases} \quad \text{until } \pi_{t+1} = \pi_t$$

Value Iteration on Four Rooms

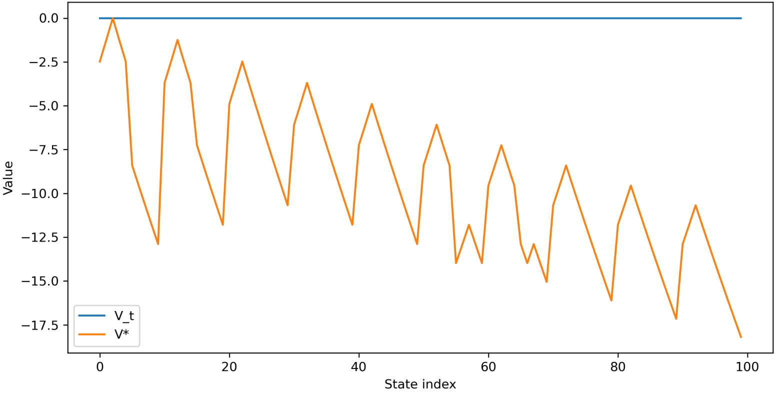


Figure 3: Application of Value Iteration to Four Rooms instance. $\gamma = 0.99$, $|\mathcal{S}| = 100$

Value Iteration on Four Rooms

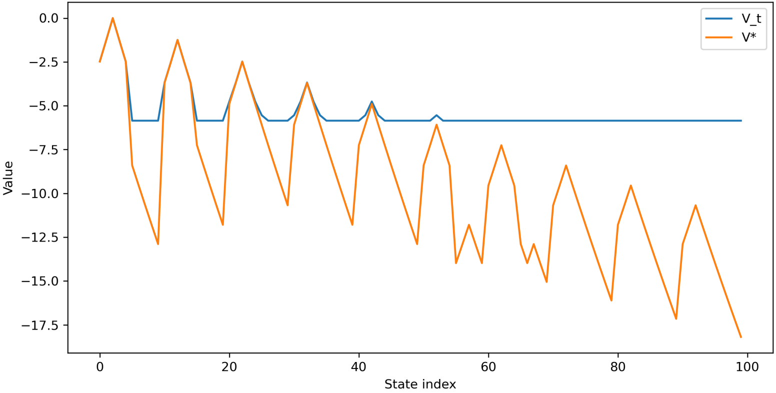


Figure 4: Application of Value Iteration to Four Rooms instance. $\gamma = 0.99$, $|\mathcal{S}| = 100$

Value Iteration on Four Rooms

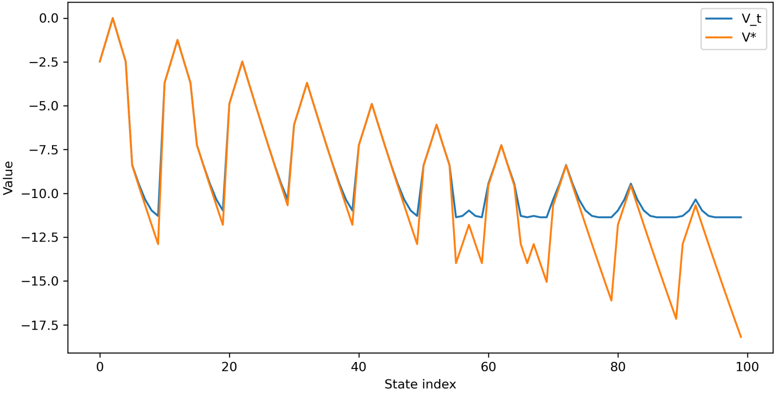


Figure 5: Application of Value Iteration to Four Rooms instance. $\gamma = 0.99$, $|\mathcal{S}| = 100$

Value Iteration on Four Rooms

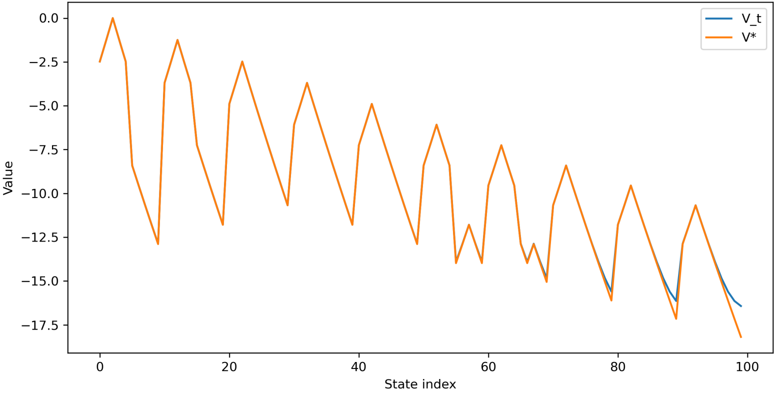


Figure 6: Application of Value Iteration to Four Rooms instance. $\gamma = 0.99$, $|\mathcal{S}| = 100$

Approximate Value Iteration [Powell, 2007]

Value Iteration

$$\begin{cases} V_0 = 0 \\ V_{t+1} \leftarrow \mathcal{T}^* V_t \end{cases}$$

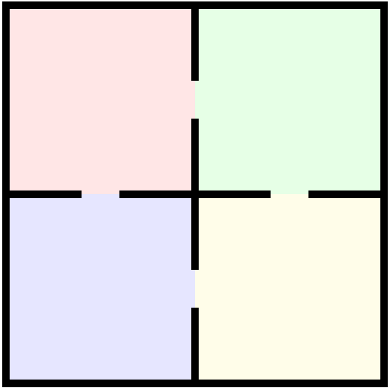
is replaced by

$$\begin{cases} V_0 = 0 \\ V_{t+1} \leftarrow \arg\min_{V \in \mathcal{V}} \|V - \mathcal{T}^* V_t\| \end{cases}$$

where $\mathcal{V} \subset \mathcal{S}$. In our work
$\mathcal{V} = \{$ piecewise constant value functions with fixed partition$\}$.

$\rightarrow$ Cheaper iterations but slower...

Natural State Abstraction for Four Rooms



Context: AVI for piecewise constant functions

Property (Projection of the Bellman operator [Bertsekas and Tsitsiklis, 1996])

Let be

- $\mathcal{S} = \bigsqcup_k S_k$ *a partition of the state space*
- $\tilde{V} \in \mathbb{R}^{\mathcal{S}}$ *a piecewise constant value function relatevely to* $(S_k)_k$
- $\mathcal{V} = \{\tilde{V}\}$

Then:

$$\arg\min_{V \in \mathcal{V}} \|V - \mathcal{T}^* V_t\|_\infty = \phi \cdot (\phi^T \cdot \phi)^{-1} \cdot \phi^T \cdot \mathcal{T}^* V$$

where $\phi := (\mathbb{1}_{s \in S_k})_{k,s} \in \{0,1\}^{K \times \mathcal{S}}$

We note $\omega = (\phi^T \cdot \phi)^{-1} \cdot \phi^T$ and $\Pi := \phi \cdot \omega \in \mathbb{R}^{\mathcal{S} \times \mathcal{S}}$.

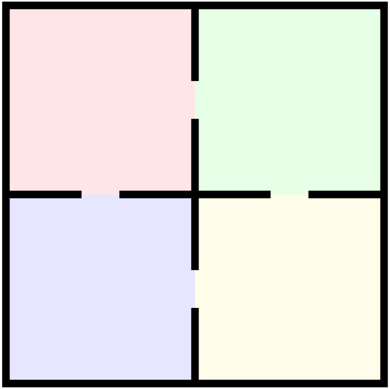
$\rightarrow \Pi \mathcal{T}^*$ contracts space with factor γ

Hierarchical Reinforcement Learning

HRL consists in

- State Abstraction : build abstract MDP from state space partition
- Action abstraction : train and apply sequence of actions to develop skills

A State Abstraction for Four Rooms



$$\underline{V}^* = \begin{pmatrix} -96 \\ -96.96 \\ -96.96 \\ -97.37 \end{pmatrix} \in \mathbb{R}^4$$

Action abstraction discovery for Four Rooms

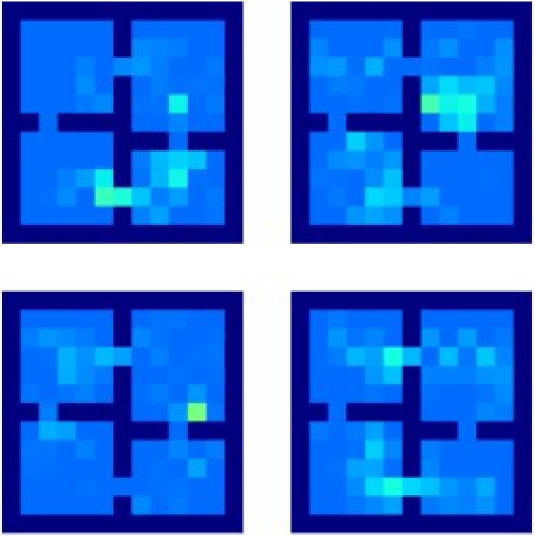


Figure 7: Option termination close to a door [Bacon et al., 2017] → Room exit skill !

State Abstraction

Definition (Abstract MDP)

Let be

- An MDP $\mathcal{M} = (\mathcal{S}, \mathcal{A}, T, R)$
- A partition $\mathcal{S} = \bigsqcup_k S_k$
- $\omega \in [0,1]^{K \times \mathcal{S}}$ a matrix of weights summing to 1 : $\sum_{s \in S_k} \omega_{k,s} = 1$

We define the associated Abstract MDP $\mathcal{M}_A = (\mathcal{K}, \mathcal{A}, T, R)$ with

- Abstract transition function : for any $k, k' \in \mathcal{K}$, for any $\forall a \in \mathcal{A}$,

$$\underline{\mathrm{T}}(k, a, k') = \sum_{s \in S_k} \sum_{s' \in S_{k'}} \omega_{k,s}.T(s, a, s') = \omega \cdot T \cdot \phi$$

- Abstract reward function

$$\underline{\mathrm{R}}(k, a) = \sum_{s \in S_k} \omega_{k,s}.R(s, a) = \omega \cdot R$$

Abstraction and information loss

Theorem (Similar state aggregation [Abel et al., 2016])

Let be

- *A real value* $\epsilon > 0$
- *A partition* $\mathcal{S} = \bigsqcup_k S_k$ *where, for any* $k \in \mathcal{K}$*, for any* $\forall s, s' \in S_k$ *and* $a \in \mathcal{A}$*,*
$$|Q^*(s,a) - Q^*(s',a)| \leq \epsilon$$
- $\mathcal{M}_A$ *the MDP associated to this partition*

Then,

$$\|V^* - V^\pi\|_\infty \leq \frac{\max_{s,a} R(s,a)}{(1-\gamma)^2}\,\varepsilon$$

where $\pi = \arg\max_{a \in \mathcal{A}} \left(\underline{R}_a + \gamma \underline{T}_a \cdot \mathcal{M}_A\right)$

$\rightarrow$ Similar states aggregation $\implies$ bounded loss of performance

Practical discovery of State Abstraction

We noticed

- A few practical build of abstraction (without use of V^* or Q^*)
- A link betweek Approximate VI and abstract MDPs

It follows

- A disaggregation process (succession of Abstract MDPs)
- Optimal value function approximation of each Abstract MDP

Table of Contents

Progressive State Space Disaggregation process [1]

In the following, we

- link the projected Bellman operator and abstract MDPs
- **estimate the quality** of a given piecewise value function
- suggest a way to **produce useful abstraction**
- **efficiently solve MDPs** taking advantage of redundant states

[0] *Progressive State Space Disaggregation for Infinite Horizon Dynamic Programming*, Forghieri, Castel, Hyon and Le Pennec, ICAPS2024

Approximate Value Iteration and State Abstraction

Theorem (Project Bellman operator and Approximate Value Iteration, O.F.)

Let us consider

- $\mathcal{S} = \bigsqcup_k S_k$ *a partition of an MDP* $\mathcal{M}$
- $\mathcal{M}_A$ *the associate abstract MDP*
- $\Pi \cdot \mathcal{T}^*_Q$ *the projected Bellman operator on the set of piecewise constant* Q *value functions*

Then, for any $\underline{Q} \in \mathbb{R}^K$,

$$\phi \cdot \mathcal{T}^*_{Q,A}\underline{Q} = \Pi\mathcal{T}^*_Q(\phi \cdot \underline{Q})$$

$\rightarrow$ projected Bellman operator $\approx$ abstract MDP Bellman operator

$\rightarrow$ projected Bellman operator is cheap to compute !

Approximate Value Iteration and State Abstraction

Proof.

For any $\underline{\mathrm{Q}} \in \mathbb{R}^K$,

$$\begin{aligned}
\phi \cdot \mathcal{T}^*_{Q,A}\underline{\mathrm{Q}} &= \phi \cdot \left(\underline{\mathrm{R}} + \gamma.\underline{\mathrm{T}} \cdot \max_{a\in\mathcal{A}} \underline{\mathrm{Q}}\right) \\
&= \phi \cdot \left(\omega \cdot R + \gamma.\omega \cdot T \cdot \phi \cdot \max_{a\in\mathcal{A}} \underline{\mathrm{Q}}\right) \\
&= \phi \cdot \omega \cdot \left(R + \gamma.T \cdot \max_{a\in\mathcal{A}} (\phi \cdot \underline{\mathrm{Q}})\right) \\
&= \Pi \cdot \left(R + \gamma.T \cdot \max_{a\in\mathcal{A}} \tilde{Q}\right) \\
&= \Pi \mathcal{T}^*_Q \tilde{Q}
\end{aligned}$$

□

Quality of a piecewise constant value function

Theorem (Quality of a piecewise constant value function, O.F.)

Let us consider

- *A partition* $\mathcal{S} = \bigsqcup_k S_k$ *of* $\mathcal{M}$
- *A piecewise constant value* $\tilde{V}$ *relatively to* $(S_k)_k$
- *The projected optimal Bellman operator* $\Pi\mathcal{T}^*$

Then,

$$\|\tilde{V} - V^*\|_\infty \leq \frac{1}{1-\gamma}\left(\max_{1\leq k\leq K} \mathrm{Span}_{S_k}\left(\mathcal{T}^*\tilde{V}\right) + \|\tilde{V} - \Pi\mathcal{T}^*\tilde{V}\|_\infty\right)$$

where $\mathrm{Span}_{S_k}(V) := \max_{s\in S_k} V(s) - \min_{s\in S_k} V(s)$.

→ Dependence on the piecewise constant $\tilde{V}$ and on the aggregation !

→ True for $\mathcal{T}^*_Q$, $\mathcal{T}^\pi$

Quality of a piecewise constant value function

$$\|\tilde{V} - V^*\|_\infty \leq \frac{1}{1-\gamma}\left(\max_{1\leq k\leq K} \mathrm{Span}_{S_k}\left(\mathcal{T}^*\tilde{V}\right) + \|\tilde{V} - \Pi\mathcal{T}^*\tilde{V}\|_\infty\right)$$

Two terms:

- $\max_{1\leq k\leq K} \mathrm{Span}_{S_k}\left(\mathcal{T}^*\tilde{V}\right)$: do we lose information aggregating ?
- $\|\tilde{V} - \Pi\mathcal{T}^*\tilde{V}\|_\infty$: is $\tilde{V}$ close to optimal value of abstract MDP ?

Proof of the bound

Lemma

$$\forall V \in \mathbb{R}^{\mathcal{S}},\ \|V^* - V\|_\infty \leq \frac{1}{1-\gamma}\|V - \mathcal{T}^* V\|_\infty$$

Proof.

$\forall V \in \mathbb{R}^{\mathcal{S}}$,

$$\begin{aligned}\|V^* - V\|_\infty &\leq \|V^* - \mathcal{T}^* V\|_\infty + \|\mathcal{T}^* V - V\|_\infty \\ &= \|\mathcal{T}^* V^* - \mathcal{T}^* V\|_\infty + \|\mathcal{T}^* V - V\|_\infty \\ &\leq \gamma \|V^* - V\|_\infty + \|\mathcal{T}^* V - V\|_\infty\end{aligned}$$

Therefore,

$$\|V^* - V\|_\infty - \gamma\|V^* - V\|_\infty \leq \|\mathcal{T}^* V - V\|_\infty$$

which concludes. □

Proof of the theorem

Proof.

$$
\begin{aligned}
(1-\gamma)\|V^* - \tilde{V}\|_\infty &\leq \|\tilde{V} - \mathcal{T}^*\tilde{V}\|_\infty \\
&\leq \|\tilde{V} - \Pi\mathcal{T}^*\tilde{V}\|_\infty + \|\Pi\mathcal{T}^*\tilde{V} - \mathcal{T}^*\tilde{V}\|_\infty \\
&\leq \|\tilde{V} - \Pi\mathcal{T}^*\tilde{V}\|_\infty + \max_k \mathrm{Span}_{S_k}\left(\mathcal{T}^*\tilde{V}\right)
\end{aligned}
$$

□

Quality of a piecewise constant value function

$$\|\tilde{V} - V^*\|_\infty \leq \frac{1}{1-\gamma}\left(\max_{1\leq k\leq K} \mathrm{Span}_{S_k}\left(\mathcal{T}^*\tilde{V}\right) + \|\tilde{V} - \Pi\mathcal{T}^*\tilde{V}\|_\infty\right)$$

Fortunately,

- $\max_{1\leq k\leq K} \mathrm{Span}_{S_k}\left(\mathcal{T}^*\tilde{V}\right)$ can decrease refining aggregation $(S_k)_k$
- $\|\tilde{V} - \Pi\mathcal{T}^*\tilde{V}\|_\infty$ can decrease iterating contracting $\Pi\mathcal{T}^*$ over $\tilde{V}$
- $\Pi\mathcal{T}^*$ is cheaper to compute

Approximate VI is cheaper to compute

Operator	Complexity	Approximation	Complexity
$\mathcal{T}^*$	$\mathcal{S}^3\mathcal{A}$	$\Pi\mathcal{T}^*$	$\mathcal{S}^2K\mathcal{A}$
$\mathcal{T}^\pi$	$\mathcal{S}^3$	$\Pi\mathcal{T}^\pi$	K^3
$\mathcal{T}_Q^*$	$\mathcal{S}^3\mathcal{A}$	$\Pi\mathcal{T}_Q^*$	$K^3\mathcal{A}$

Table 1: Number of operations necessary to update a value function.

$\rightarrow$ Cheaper to compute, contract space with factor γ, but converge to $\tilde{V} \neq V^*$... Need to refine aggregation !

Progressive State Space Disaggregation process[2]

Let be ε the final precision to approximate V^*. Starting with

- $K = 1$, $S_1 = \mathcal{S}$
- $\tilde{V}_0 = (0)_{s \in \mathcal{S}}$

We iterate

- Apply $\Pi\mathcal{T}^*$ until $\|\tilde{V} - \Pi\mathcal{T}^*\tilde{V}\|_\infty$ is smaller than ϵ
- Compute $V_{t+1} := \mathcal{T}^* V_t$. Divide each region until $\max_{s \in S_k} V_{t+1} - \min_{s \in S_k} V_{t+1}$ is smaller than ϵ for each region $k \in [\![1 \ ; \ K]\!]$.

[2] *Progressive State Space Disaggregation for Infinite Horizon Dynamic Programming*, Forghieri, Castel, Hyon and Le Pennec, ICAPS2024

Disaggregation process

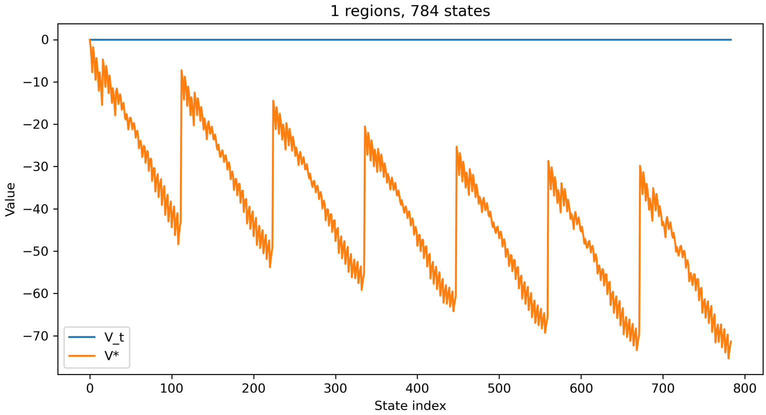


Figure 8: Disaggregation process applied to Tandem Queues model [Tournaire et al., 2022]

First disaggregation step

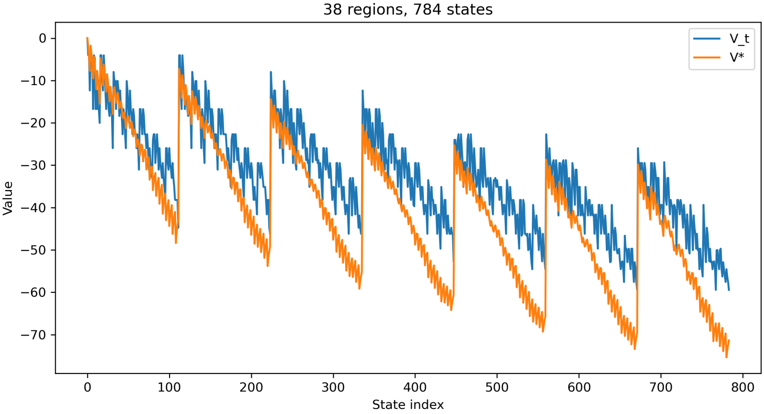


Figure 9: Disaggregation process applied to Tandem Queues model [Tournaire et al., 2022]

Second disaggregation step

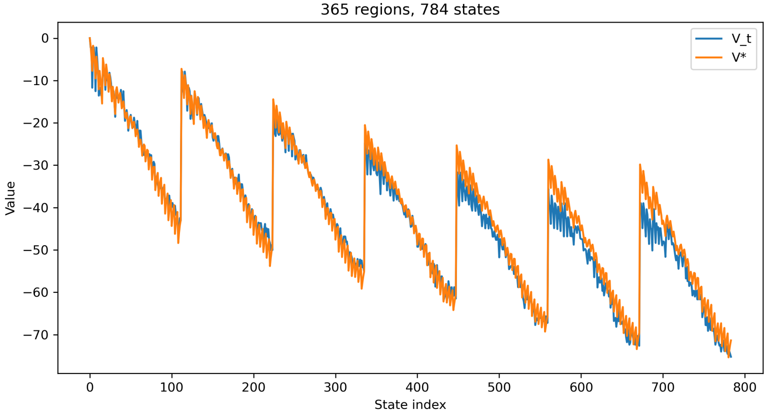


Figure 10: Disaggregation process applied to Tandem Queues model [Tournaire et al., 2022]

Progressive Disaggregation Convergence

Theorem

Let $(\underline{V}, (S_k)_k)$ denote the value and the abstraction computed by PDVI. Then, the following properties hold.

1. *The process finishes in a finite number of steps.*
2. *The distance to optimal value function checks:*

$$\|\phi \cdot \underline{V} - V^*\|_\infty \leq \frac{2\epsilon}{1-\gamma}$$

Moreover, for any region k,

$$\forall s, s' \in S_k,\ |V^*(s) - V^*(s')| \leq \frac{4\epsilon}{1-\gamma}\,.$$

Progressive Disaggregation Convergence

Proof.

Two main arguments :

1. The number of partition strictly increases at each step
2. The bound

$$\|\tilde{V} - V^*\|_\infty \leq \frac{1}{1-\gamma}\left(\max_{1\leq k\leq K} \mathrm{Span}_{S_k}\left(\mathcal{T}^*\tilde{V}\right) + \|\tilde{V} - \Pi\mathcal{T}^*\tilde{V}\|_\infty\right)$$

ensure the claimed final precision.

□

Remarks

Advantages :

- Saving time on projected Bellman operator iterations $\Pi\mathcal{T}^*$
- Final Abstraction much smaller than original mdp : $K \ll |\mathcal{S}|$
- Convergence guarantee !

Risks :

- Too many disaggregation steps (maximum $|\mathcal{S}|$)
- Final Abstract could be the original MDP itself !

Performance Evaluation

Solving an MDP depends on

- Its complexity ($|\mathcal{S}|$, $|\mathcal{A}|$, density of the transition matrix...)
- Wanted final precision to approximate V^* ($\varepsilon = 10^{-3} \not\Rightarrow \pi = \pi^*$...)
- Chosen discount γ and expected length of the trajectory ($\gamma \ll 1 \iff$ Value Iteration $\gg$ Policy Iteration)

$\rightarrow$ We compare algorithm on the runtime ensuring the same final precision

Models used

Three MDPs with large state spaces :

- Randomly drew stochastic transition matrix (Garnets, [Archibald et al., 1995, Clement and Kroer, 2021])
- Four Rooms environment [Hengst, 2012]
- Real world Tandem Server Queues [Tournaire et al., 2022] (Two servers in tandem, managing the number of VMs)

Solving methods

Traditional Dynamic Programming :

- Value Iteration
- Policy Iteration

Alternative Aggregation approach :

- Policy Iteration Modified with Adapative Aggregation Boosting [Bertsekas et al., 1988]
- Aggregation-Disaggregation for Temporal-Difference Learning [Chen et al., 2022]

Progressive Disaggregation applied to :

- Value Iteration, Q-Value Iteration
- Policy Iteration Modified

Random MDPs solving

Density	VI	PDVI	PDQVI
1%	113.3 ± 1.0	6.6 ± 0.5	8.0 ± 0.4
10%	300.3 ± 10.9	7.5 ± 0.1	15.2 ± 0.3
25%	751.7 ± 16.0	6.2 ± 0.6	24.1 ± 0.8
45%	1397.7 ± 23.7	7.6 ± 1.3	36.3 ± 1.7
65%	1915.4 ± 54.2	6.7 ± 0.4	50.3 ± 3.6

Density	MPI	PDPIM	Bertsekas
1%	3.0 ± 1.25	$\mathbf{1.09 \pm 0.23}$	2.8 ± 0.6
10%	1.65 ± 0.46	$\mathbf{1.57 \pm 0.45}$	2.5 ± 0.3
25%	1.17 ± 0.08	$\mathbf{0.72 \pm 0.11}$	1.5 ± 0.4
45%	1.83 ± 0.32	$\mathbf{0.61 \pm 0.21}$	2.0 ± 0.2
65%	2.86 ± 1.03	$\mathbf{1.57 \pm 0.74}$	3.3 ± 0.7

Table 2: Random MDPs mean solving time (s). $|\mathcal{S}| = 500$, $|\mathcal{A}| = 50$, $\gamma = 0.99$, $\varepsilon = 10^{-2}$, 10 experiments.

Four Rooms solving

$\|\mathcal{S}\|$	VI	PDVI	PDQVI
36	2.72 ± 0.0	7.46 ± 0.4	103.28 ± 0.7
100	3.63 ± 0.1	6.77 ± 1.7	267.63 ± 2.6
196	3.57 ± 0.4	9.25 ± 2.7	276.04 ± 2.5
324	10.25 ± 0.8	14.16 ± 5.0	456.31 ± 7.9

$\|\mathcal{S}\|$	MPI	PDPIM	Bertsekas
36	2 ± 1	$\mathbf{1 \pm 0.1}$	1 ± 0.5
100	18 ± 3	$\mathbf{2 \pm 0.7}$	19 ± 0.9
196	29 ± 4	$\mathbf{3 \pm 0.4}$	29 ± 0.9
324	47 ± 7	$\mathbf{10 \pm 1.2}$	47 ± 0.6

Table 3: Four Rooms model mean policy-based solving time (s). Variable $|\mathcal{S}|$, $|\mathcal{A}| = 4$, $\gamma = 0.999$, $\varepsilon = 10^{-3}$, 10 experiments.

Tandem Queues solving

$\lvert\mathcal{S}\rvert$	VI	PDVI	PDQVI
8100	12.1 ± 0.5	$\mathbf{8.0 \pm 1.3}$	15.3 ± 0.7
12544	41.5 ± 0.8	$\mathbf{18.8 \pm 1.8}$	35.3 ± 1.6

$\lvert\mathcal{S}\rvert$	MPI	PDPI	Bertsekas
8100	1442.5 ± 39.2	267.5 ± 5.6	1626.1 ± 13.4
12544	4211.0 ± 63.1	994.7 ± 6.3	3577.2 ± 14.8

Table 4: Tandem Queues model mean solving time (s). Variable $\lvert\mathcal{S}\rvert$, $\lvert\mathcal{A}\rvert = 3$, $\gamma = 0.99$, $\varepsilon = 10^{-2}$, 10 experiments.

Table of Contents

Conclusion

Here, we

- linked Approximate Value Iteration and Abstract MDPs
- **Estimated** aggregation quality
- provided a **practical way** to build useful abstractions
- **evaluated** this method on various environments

Upcoming work :

- Total reward convergence proof
- A larger benchmark
- Opening piecewise constant approximation to model-free context

Abel, D., Hershkowitz, D., and Littman, M. (2016).
Near optimal behavior via approximate state abstraction.
In *International Conference on Machine Learning*, pages 2915–2923. PMLR.

Archibald, T., McKinnon, K., and Thomas, L. (1995).
On the generation of markov decision processes.
Journal of the Operational Research Society, 46(3):354–361.

Bacon, P.-L., Harb, J., and Precup, D. (2017).
The option-critic architecture.
In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 31.

Bertsekas, D. and Tsitsiklis, J. N. (1996).
Neuro-dynamic programming.
Athena Scientific.

Bertsekas, D. P., Castanon, D. A., et al. (1988).
Adaptive aggregation methods for infinite horizon dynamic programming.

IEEE Transactions on Automatic Control.

Chen, G., Gaebler, J. D., Peng, M., Sun, C., and Ye, Y. (2022).
An adaptive state aggregation algorithm for markov decision processes.
In *AAAI 2022 Workshop on Reinforcement Learning in Games.*

Clement, J. G. and Kroer, C. (2021).
First-order methods for wasserstein distributionally robust mdp.
In *International Conference on Machine Learning*, pages 2010–2019. PMLR.

Hengst, B. (2012).
Hierarchical approaches.
In *Reinforcement learning*, pages 293–323. Springer.

Powell, W. B. (2007).
Approximate Dynamic Programming: Solving the curses of dimensionality, volume 703.
John Wiley & Sons.

Sutton, R. S. and Barto, A. G. (2018).

Reinforcement learning: An introduction.
MIT press.

Tournaire, T., Jin, Y., Aghasaryan, A., Castel-Taleb, H., and Hyon, E. (2022).
Factored reinforcement learning for auto-scaling in tandem queues.
In *NOMS 2022-2022 IEEE/IFIP Network Operations and Management Symposium*, pages 1–7. IEEE.