Adaptive Aggregation for Approximate Dynamic Programming Methods

Workshop on restless bandits, index policies and applications in reinforcement learning

Université Grenoble Alpes, November 20–21 2023

Orso Forghieri

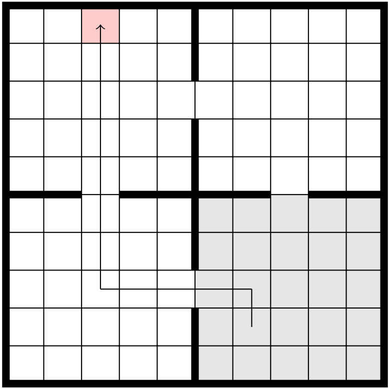
Erwan Le Pennec (École Polytechnique)

Hind Castel (Télécom SudParis)

Emmanuel Hyon (Sorbonne Université)

November 20, 2023

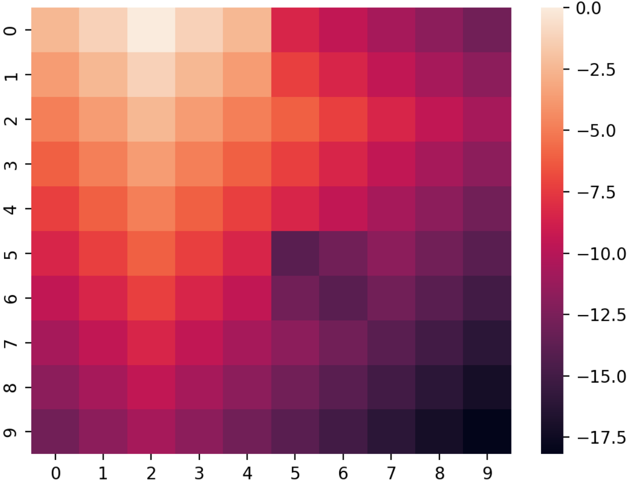
Markov Decision Processes and Four Rooms instance



Four rooms:

- $\mathcal{S} = [\![0 \ ; \ 100]\!]$, $\mathcal{A} = \{N, S, E, W\}$
- Reward: -1 until exit is reached, 0 otherwise.

Four Rooms optimal Value Function



Increasing complexity

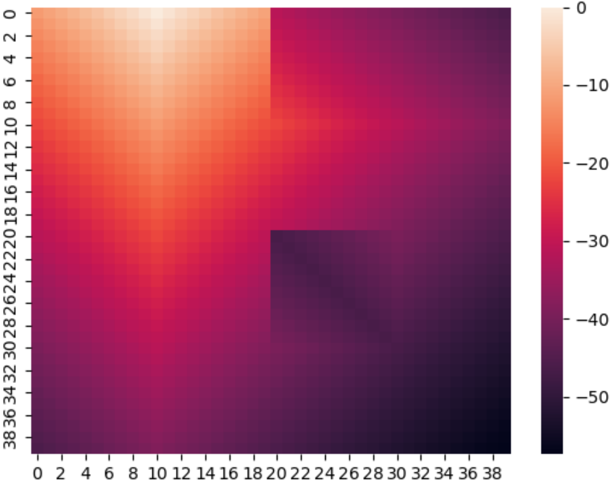


Table of Contents

Table of Contents

Litterature context

Our objectives:

- Find a good approximation of a large MDP
- Find the optimal policy on this approximation

To this end, we need:

- State Abstraction context to have intuition on how to build a final "good" abstraction
- Approximate Dynamic Programming to solve the new simplified problem iterating a contracting operator

Hierarchical Reinforcement Learning[1]

Two types of learnable hierarchy to divide a MDP:

- Temporal Abstraction: apply a series of actions that skip timesteps (like a given skill)
- Spatial Abstraction: Divide state space into regions and jump from one to another

[1] [Abel, 2022] gives a good insight of it.

State Abstraction for Markov Decision Processes

State Abstraction characteristics:

- Gather similar states[2] into regions to form a new simpler MDP[3]
- Reasonable loss of information[4]
- Fastly built abstraction

[2] same value, same Q-value or same policy

[3] [Abel et al., 2016]

[4] [Abel et al., 2019]

State Abstraction buildings[Tsitsiklis and Van Roy, 1996]

Let $\mathcal{S} = \{s_1, s_2, s_3\} = S_1 \sqcup S_2 = \{s_1, s_2\} \sqcup \{s_3\}$. We define:

$$\phi := \left(\mathbb{1}_{s \in S_k}\right)_{s \in \mathcal{S}, 1 \leq k \leq K} = \begin{pmatrix} 1 & 0 \\ 1 & 0 \\ 0 & 1 \end{pmatrix}$$

and

$$\omega := \left(\phi^T \cdot \phi\right)^{-1} \cdot \phi^T = \begin{pmatrix} \frac{1}{2} & \frac{1}{2} & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

Then

$$V \in \mathbb{R}^{\mathcal{S}} \quad \xrightarrow{\Pi := \phi \cdot \omega} \quad \tilde{V} \in \mathbb{R}^{\mathcal{S}} \quad \underset{\phi}{\overset{\omega}{\rightleftarrows}} \quad \underline{\mathrm{V}} \in \mathbb{R}^{K}$$

$$V = \begin{pmatrix} 3 \\ 4 \\ 5 \end{pmatrix} \xrightarrow{\Pi := \phi \cdot \omega} \begin{pmatrix} 3.5 \\ 3.5 \\ 5 \end{pmatrix} \quad \underset{\phi}{\overset{\omega}{\rightleftarrows}} \quad \begin{pmatrix} 3.5 \\ 5 \end{pmatrix}$$

Abstract MDP definition[5]

For an original MDP

$$\mathcal{M} = (\mathcal{S}, \mathcal{A}, T, R, \gamma)$$

Abstract transition and reward depending on the original:

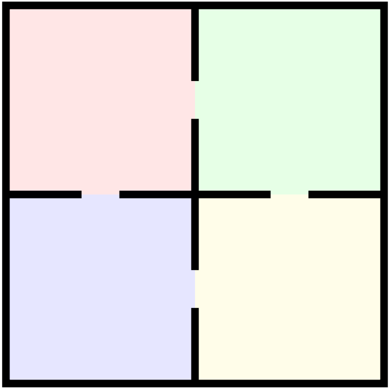
$$R_{abs} = \omega \cdot R, \quad T_{abs} = \omega \cdot T \cdot \phi$$

and its optimal value function $\underline{\mathrm{V}}^*$ is the solution of

$$\underline{\mathrm{V}}^* = \mathcal{T}^*_{abs} \underline{\mathrm{V}}^*$$

[5] [Abel et al., 2016]

State Abstraction application



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

Approximate Value Iteration

VI:

$$\begin{aligned} V_{t+1} &\leftarrow \mathcal{T}^* V_t \\ &= \max_{a \in \mathcal{A}} \left(R_a + \gamma T_a \cdot V \right) \end{aligned}$$

AVI[6]:

$$V_{t+1} \leftarrow \Pi \mathcal{T}^* V_t$$

where Π is a projector on a subspace of $\mathbb{R}^{\mathcal{S}}$.

For State Aggregation:

$$\Pi = \phi \cdot \omega := \phi \cdot \left(\phi^T \cdot \phi \right)^{-1} \cdot \phi^T$$

so we iterate

$$V_{t+1} \leftarrow \phi \cdot \omega \cdot \mathcal{T}^* V_t$$

[6] [Powell, 2007]

General context and work

In this work, we achieve to

- Aggregate states *having close value*
- *Approximate optimal value* function
- *Adapt the method* through Q-Value Iteration and Policy Iteration algorithms

Table of Contents

State Aggregation and Approximate Dynamic Programming

Lemma (Optimal Error Bound with arbitrary partition, O.F.)

For any piecewise constant value function $\tilde{V}$

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

where $\mathrm{Span}_{S_k} V := \max_{s\in S_k} V(s) - \min_{s\in S_k} V(s)$, *and* V^* *is the optimal value function.*

Sketch of proof

Proof.

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

□

$\rightarrow$ Also true for $\mathcal{T}_Q^*$ and $\mathcal{T}^\pi$ for any π

Speed of Approximate Dynamic Programming

Remark

Compared to $\mathcal{T}^*$, we lose in complexity:

- $\frac{|\mathcal{S}|}{K}$ *computing* $\Pi\mathcal{T}^*$
- $\left(\frac{|\mathcal{S}|}{K}\right)^3$ *computing* $\Pi\mathcal{T}_Q^*$
- $\left(\frac{|\mathcal{S}|}{K}\right)^2$ *computing* $\Pi\mathcal{T}^\pi$

where K is the number of regions.

Adaptive Aggregation algorithm

From

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

we propose the following process:

1. Approximate the original state space by a unique trivial region
2. Then alternate between
    1. Progressively refine the partitioning along $\mathcal{T}^* \tilde{V}$ to reduce $\max_{1 \leq k \leq K} \mathrm{Span}_{S_k} \mathcal{T}^* \tilde{V}$
    2. Iterate the contracting operator $\Pi \mathcal{T}^*$ to reduce Projected Bellman Residual $\|\tilde{V} - \Pi \mathcal{T}^* \tilde{V}\|_\infty$
3. Finish if the two terms are each bounded by ϵ

Theoretical guarantee

Corollary (Final precision and aggregation criterion)

The algorithm result $(\tilde{V}, \{S_k\})$ *checks:*

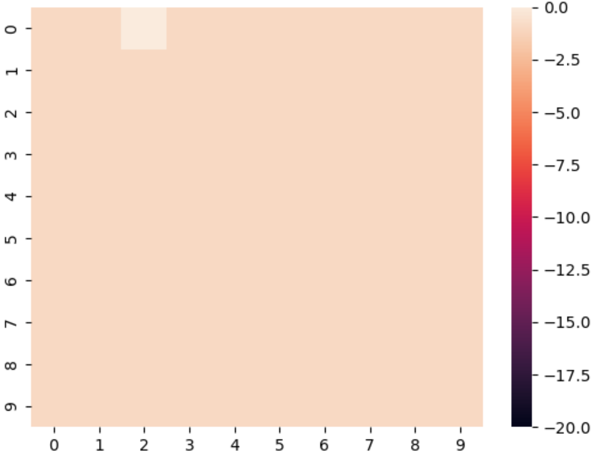
$$\|V - V^*\|_\infty \leq \frac{2\epsilon}{1-\gamma}$$

and

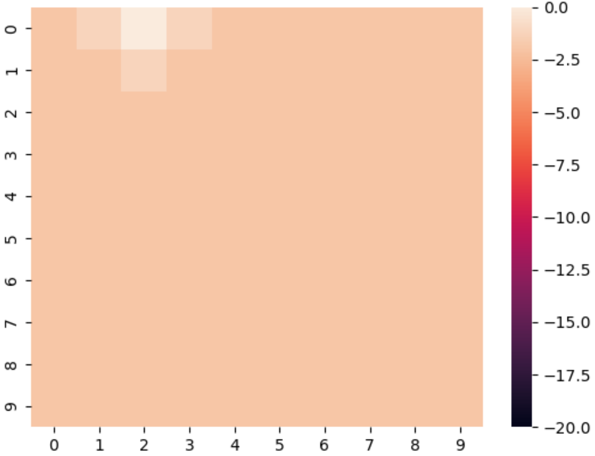
$$\forall k \in [\![1\ ;\ K]\!],\ \forall s, s' \in S_k,\ |V^*(s) - V^*(s')| \leq \frac{4\epsilon}{1-\gamma}$$

Table of Contents

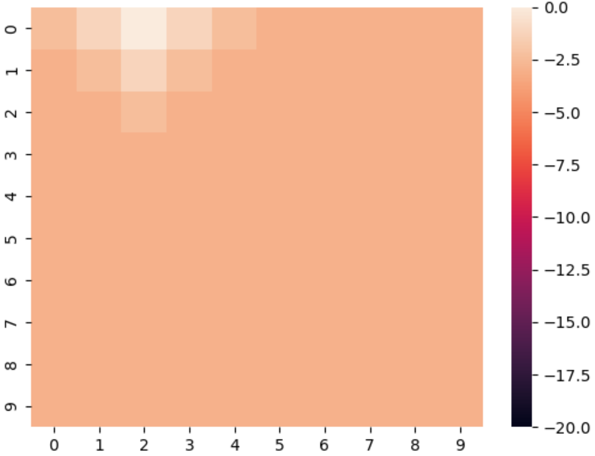
Adaptive Aggregation Value Iteration



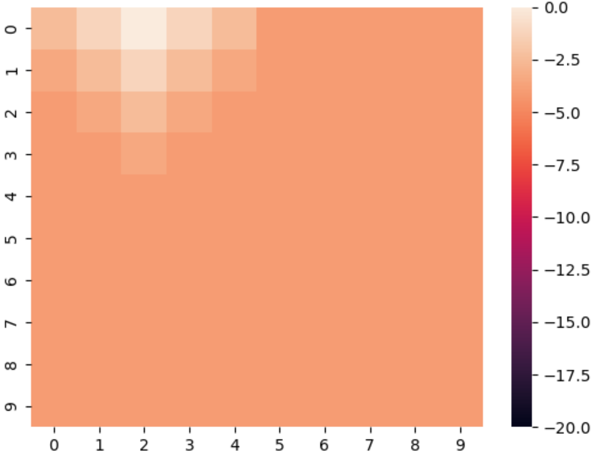
Adaptive Aggregation Value Iteration



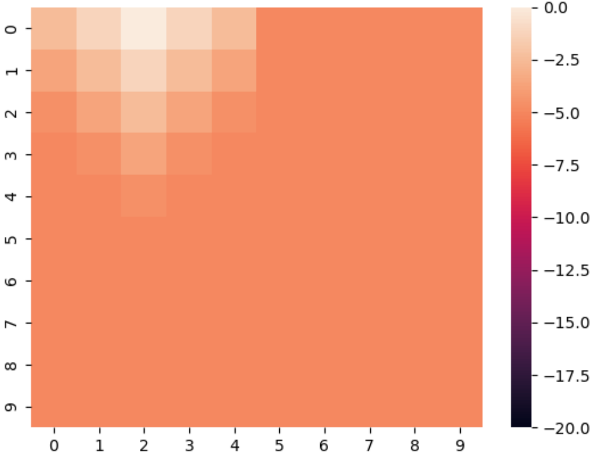
Adaptive Aggregation Value Iteration



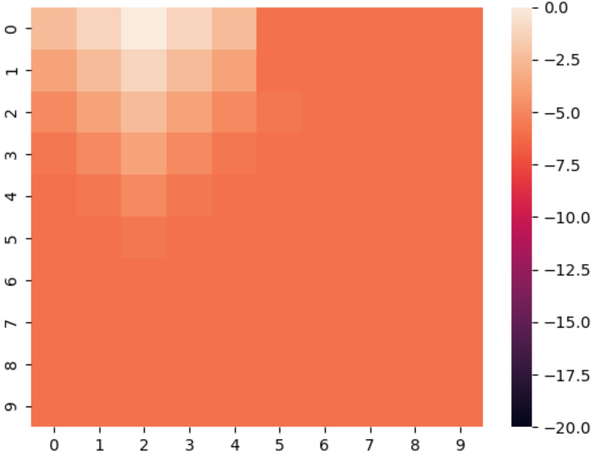
Adaptive Aggregation Value Iteration



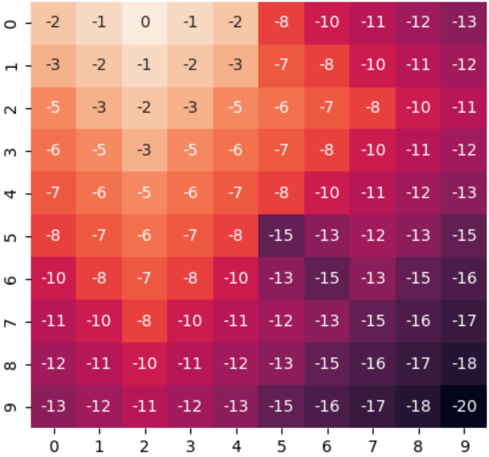
Adaptive Aggregation Value Iteration



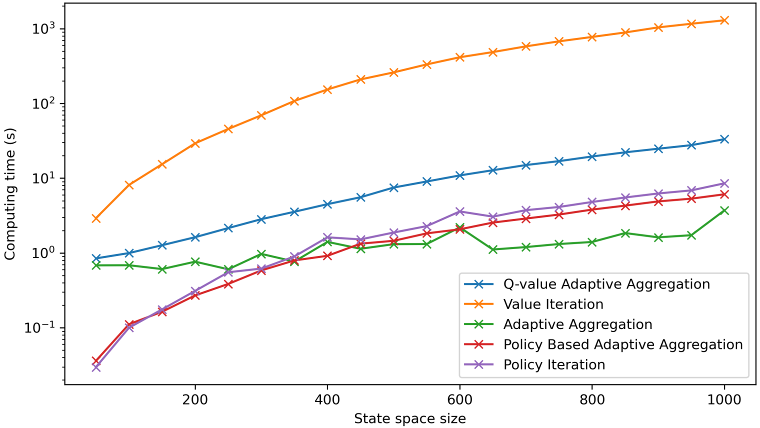
Adaptive Aggregation Value Iteration



Final Partition



Runtime comparison — Random MDPs



Average runtime for variable $|\mathcal{S}| \in [\![50\ ;\ 1000]\!]$
($\mathcal{A} = 10$, 10 exp/point, random MDP with 95% non-zero, precision 10^{-2}, $\gamma = 0.99$)

Runtime comparison — Random MDPs

Sparsity	Agg PI	Agg Value	Agg Q-Value	VI	PI
10%	1.52	3.78	8.31	216.28	2.35
40%	2.72	4.06	21.94	783.85	4.36
70%	3.82	4.59	34.91	1311.30	6.28
100%	3.18	4.59	31.08	1048.93	8.85

Average runtime (s) for variable sparsity.
$\big(\mathcal{A} = \{10, 50, 100\}$, $\mathcal{S} \in [\![50\ ;\ 1000]\!]$, 5 exp/point, random MDP, precision 10^{-2}, $\gamma = 0.99\big)$

Results

Discussion:

- Low sparsity $\implies$ slow classical Value Iteration
- Higher accuracy and greater discount $\implies$ our algorithm struggle
- Value Iteration $\approx$ Adaptive Aggregation Value Iteration for sparse MDPs?

Table of Contents

Conclusion on State Abstraction

We provided:

- An efficient Approximate algorithm to compute optimal Policy/Value Function
- Useful State Abstractions

Perspective:

- More simulations needed (more models, impact of $\mathcal{A}$, bigger state space, lower ϵ)
- Improve speed of the Policy Iteration-like algorithm
- Generalization over model-free problems

Thank you for your attention!

Abel, D. (2022).
A theory of abstraction in reinforcement learning.
arXiv preprint arXiv:2203.00397.

Abel, D., Arumugam, D., Asadi, K., Jinnai, Y., Littman, M. L., and Wong, L. L. (2019).
State abstraction as compression in apprenticeship learning.
In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 3134–3142.

Abel, D., Arumugam, D., Lehnert, L., and Littman, M. (2018).
State abstractions for lifelong reinforcement learning.
In *International Conference on Machine Learning*, pages 10–19. PMLR.

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.

Andre, D. and Russell, S. J. (2002).

State abstraction for programmable reinforcement learning agents.
In *Aaai/iaai*, pages 119–125.

Bertsekas, D. P. (2018).
Feature-based aggregation and deep reinforcement learning: A survey and some new implementations.
IEEE/CAA Journal of Automatica Sinica, 6(1):1–31.

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

Dean, T. and Givan, R. (1997).
Model minimization in markov decision processes.
In *AAAI/IAAI*, pages 106–111.

Dearden, R. and Boutilier, C. (1997).
Abstraction and approximate decision-theoretic planning.
Artificial Intelligence, 89(1-2):219–283.

Hengst, B. (2012).

Hierarchical approaches.
In *Reinforcement Learning: State-of-the-Art*, pages 293–323. Springer.

Jong, N. K. and Stone, P. (2005).
State abstraction discovery from irrelevant state variables.
In *IJCAI*, volume 8, pages 752–757.

Li, L., Walsh, T. J., and Littman, M. L. (2006).
Towards a unified theory of state abstraction for mdps.
In *AI&M*.

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

Singh, S., Jaakkola, T., and Jordan, M. (1994).
Reinforcement learning with soft state aggregation.
Advances in neural information processing systems, 7.

Tsitsiklis, J. N. and Van Roy, B. (1996).

Feature-based methods for large scale dynamic programming.
Machine Learning, 22(1-3):59–94.