State Abstraction Discovery in Model-Based Reinforcement Learning

Argo Seminar

Paris, France, September 16. 2024

Orso Forghieri (École polytechnique)

orso.forghieri@polytechnique.edu

Under supervision of

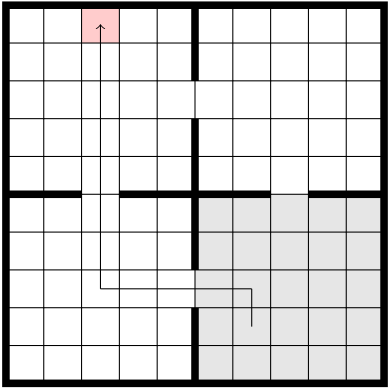
Hind Castel (Télécom SudParis)
Emmanuel Hyon (LIP6, Université Paris-Nanterre)
Erwan Le Pennec (École polytechnique)

Markov Decision Processes

- Observable State s_t, Action a_t, Reward r_t, Next state s_{t+1}
- Optimization problem : $\max_{\pi \in \mathcal{A}^{\mathcal{S}}} \sum_{t \geq 0} \gamma^t r_t$, $\gamma = 0.99$

slides/2024_04_24_semdoc/images/rl_principle.png

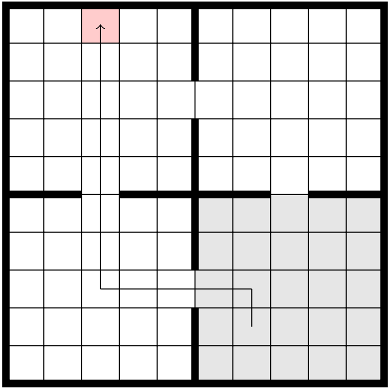
Four Rooms instance



Four rooms:

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

Four Rooms instance



Naturally hierarchical!

- Spatial: each room as a single state
- Temporal: learn to exit a room

Hierarchical Reinforcement Learning

Why use HRL in MDPs?

- Solving large MDPs
- Enhancing explainability and interpretability of MDPs
- Ensuring solution quality

How to implement it?

- Subgoal discovery and meta-action learning (temporal abstraction)
- Building approximate MDPs using spatial abstraction, with limited explicit methods

Related works

Handling Large MDP Spaces:

- With factorization hypothesis[1]
- Hierarchical approach[2]
- State information abstraction[3]

State aggregation and abstraction discovery:

- Abstraction discovery[4]
- Abstraction and MDP approximation[5]
- MDP solving acceleration[6]

[1] [Guestrin et al., 2003, Siddiqi et al., 2010]

[2] [Sutton et al., 1999, Li et al., 2006, Hengst, 2012]

[3] [Pineau et al., 2003, Coulom, 2006]

[4] [Singh et al., 1994, Dean and Givan, 1997, Abel et al., 2016, Ferrer-Mestres et al., 2020]

[5] [Tsitsiklis and Van Roy, 1996, Abel, 2019, Gopalan et al., 2017]

[6] [Bean et al., 1987, Bertsekas et al., 1988, Ciosek and Silver, 2015, Abel et al., 2020, Jothimurugan et al., 2021]

Context and work

In this talk, we present:

- MDP solving context
- The HRL approach
- Our contribution[7]

In state abstraction, we:

- Link MDP abstraction and Approximate Dynamic Programming
- Estimate error induced by abstraction
- Present a practical way to abstract MDP and solve them exactly
- Conduct a numerical comparison using real-world models

[7] [Forghieri et al., 2024]

Table of Contents

MDP solving approach

$$
\begin{aligned}
\text{Solving the MDP} &\iff \text{Maximizing upcoming rewards relatively to } \pi \\
&\iff \max_{\text{policy}} \mathop{\mathbb{E}}_{a_t = \pi(s_t)} \left[\sum_{t \geq 0} \gamma^t r_t \,\middle|\, s_0 = s \right] \\
&\iff \max_{\pi \in \mathcal{A}^{\mathcal{S}}} V^\pi(s) := V^*(s)
\end{aligned}
$$

Value functions definitions

- The expected reward applying π, V^π:

$$V^\pi(s) = \underset{a_t = \pi(s_t)}{\mathbb{E}} \left[\sum_{t \geq 0} \gamma^t r_t \middle| s_0 = s \right]$$

- The optimal value function:

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

Moreover, V^* is solution of the optimal Bellman equation (fixed point equation):

$$V^* = \max_{a \in \mathcal{A}} (R_a + \gamma \cdot T_a \cdot V^*) := \mathcal{T}^* V$$

How to find V^* or π^* ?

Dynamic Programming approaches[8]:

- Value Iteration

$$V_{t+1} \leftarrow \mathcal{T}^* V_t \text{ until } \|V^* - V_t\|_\infty \leq \varepsilon$$

- Policy Iteration Modified, where we iterate

$$\begin{cases} V_{t+1} \leftarrow (\mathcal{T}^\pi)^m V_t \\ \pi_{t+1} \leftarrow \arg\max_{a \in \mathcal{A}} (R_a + \gamma \cdot T_a \cdot V^{\pi_t}) \end{cases}$$

[8] [Sutton and Barto, 2018]

MDP solving

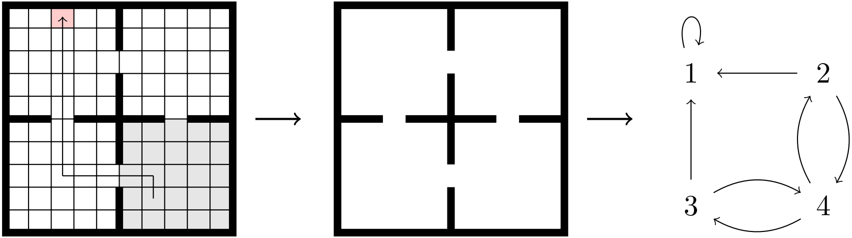
In general,

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

Table of Contents

Hierarchical RL — State Abstraction

MDP approximation from state space partition:



Less states, same action space:

$$\mathcal{S} = \bigsqcup_k S_k \to \mathcal{K} = \{s_1, \ldots, s_k\}$$

Averaged transition and reward:

$$T \to \omega \cdot T \cdot \phi$$

$$R \to \omega \cdot R$$

State Abstraction Definition

Definition (Abstract MDP [Li et al., 2006])

Given $\mathcal{M} = (\mathcal{S}, \mathcal{A}, T, R)$ s.t. $\mathcal{S} = \bigsqcup_k S_k$, the abstract MDP

$$(\mathcal{K}, \mathcal{A}, \tilde{T}, \tilde{R})$$

is defined using

- Averaged transition $\tilde{T} = \omega \cdot T \cdot \phi$
- Averaged reward $\tilde{R} = \omega \cdot R$
- Abstract state space $\mathcal{K} = \{s_k\}$

where

- $\omega \in [0, 1]^{K \times \mathcal{S}}$ weights with sum 1 on each region
- $\phi = (\mathbb{1}_{s \in S_k})_{s,k}$

Approximate Bellman Operator

An $L2$ approximation of the optimal Bellman operator

$$\mathcal{T}^* = V \to \max_{a \in \mathcal{A}} \left(R_a + \gamma \cdot T_a \cdot V \right)$$

is its averaged version[9]

$$\Pi \mathcal{T}^*(V) = \phi \cdot \omega \cdot \mathcal{T}^*(V) = \phi \cdot \arg\min_{V_A \in \mathbb{R}^K} \|\phi \cdot V_A - \mathcal{T}^* V\|_2$$

which is less complex to compute!

Moreover, $\tilde{V}^*$ solution of $V = \Pi \mathcal{T}^* V$ checks

$$\|\tilde{V}^* - V^*\|_\infty \leq \max_{1 \leq k \leq K} \frac{\max_{S_k} V - \min_{S_k} V}{1 - \gamma} .$$

[9] [Tsitsiklis and Van Roy, 1996]

Notes on State Abstraction

We note that:

- Grouping similar states limits approximation made[10]:

$$\forall s, s' \in S_k,\ |Q^*(s,a) - Q^*(s',a)| \leq \varepsilon \implies \|V^* - V^{\tilde{\pi}}\|_\infty \leq K \cdot \varepsilon$$

- Efficient partition criterion often depends on V^*, Q^*, π^*...

We propose to:

- Estimate the quality of any valuation of state abstraction
- Refine abstraction along VI steps

[10] [Abel et al., 2016]

Table of Contents

Approximate Bellman Operator and Abstraction

Approximate Bellman $\Pi\mathcal{T}_Q^* = \phi \cdot \omega \cdot \mathcal{T}_Q^*$ is the exact Bellman of an abstract MDP:[11]

Lemma (O.F.)

For any state abstraction $(\mathcal{K}, \omega, \phi)$ *and its value function* $Q_A = \omega \cdot \tilde{Q} \in \mathbb{R}^K$,

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

[11] [Forghieri et al., 2024]

Approximate Bellman Operator and Abstraction

Proof.

For any $Q_A \in \mathbb{R}^K$,

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

$\square$

Quality of a piecewise constant value function

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

Given $\mathcal{M}$*, its abstraction* $(\mathcal{K}, \omega, \phi)$*, and the piecewise constant value function* $\tilde{V}$*,*

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

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

$\rightarrow$ Dependence on the $\tilde{V}$ ($\equiv$ value function on the abstract MDP) and on the aggregation structure !

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

Quality of a piecewise constant value function

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

Two terms:

- $\max_{1\leq k\leq K} \operatorname{Span}_{S_k} \mathcal{T}^*\tilde{V}$: 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 theorem

Proof.

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

□

Quality of a piecewise constant value function

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

Fortunately,

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

Approximate VI is cheaper to compute

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

Table 1: Number of operations necessary to update a value function. $K \ll S$ generally.

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

Progressive State Space Disaggregation Process

We propose starting with a trivial abstraction:

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

Then iterate as follows:

1. Apply $\Pi\mathcal{T}^*$ until $\|\tilde{V} - \Pi\mathcal{T}^*\tilde{V}\|_\infty \leq \epsilon$
2. Refine each region by splitting until

$$\max_{S_k} \mathcal{T}^* V_t - \min_{S_k} \mathcal{T}^* V_t \leq \varepsilon$$

   holds for each region k.

$\rightarrow$ This process converges to V^* with arbitrary accuracy.

Disaggregation process

slides/2024_04_24_semdoc/images/rooms_avi_0.png

First disaggregation step

slides/2024_04_24_semdoc/images/rooms_avi_1.png

Second disaggregation step

slides/2024_04_24_semdoc/images/rooms_avi_2.png

Progressive Disaggregation Convergence

Theorem (Final Abstraction Quality, O.F.)

Considering the final value and abstraction $(V_A, (S_k)_k)$*,*

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

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

Moreover, for any region k*,*

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

→ Abstraction quality is ensured!

Progressive Disaggregation Convergence

Proof.

Two main arguments :

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

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

ensures the claimed final precision.

□

Remarks

Advantages :

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

Risks :

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

Performance Evaluation

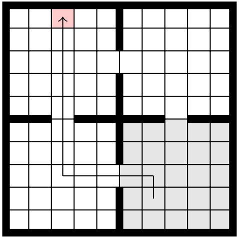
Solving an MDP depends on

- Its structure ($|\mathcal{S}|$, $|\mathcal{A}|$, density of the transition matrix...)
- Wanted final precision to approximate V^* ($\varepsilon = 10^{-3} \nRightarrow \pi = \pi^*$...)
- 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

Toys models

- Four Rooms [Hengst, 2012]
- Racetrack [Sutton and Barto, 2018]



slides/2024_09_16_argo/images/ra

Figure 5: Four Rooms and Sutton's racectrack models

Real-world models

- Servers in tandem and stochastic arrivals [Tournaire et al., 2022]
- Hydro-valley electricity production management [Carpentier et al., 2018]
- Inventory Control [Winston, 2004]

slides/2024_09_16_argo/images/damm.png

Solving methods

Traditional Dynamic Programming

- Value Iteration
- Modified Policy Iteration

Our (Progressive Disaggregation) adapted to

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

Alternative Aggregation approach

- Modified Policy Iteration with adaptive aggregation boosting [Bertsekas et al., 1988]
- Aggregation-disaggregation for Temporal-Difference learning [Chen et al., 2022]
- Work in Progress: Abstraction building and solving [Ciosek and Silver, 2015], no precision guarantee

Runtime comparison

$\lvert\mathcal{S}\rvert$	Rooms 193.6k	Barto 1M	Tandem 19.6k	Hydro-valley 118k	Inventory 250k
PDVI	11712.5	73501.4	979.4	>24h	2877.4
PDQVI	**160.3**	**46.8**	657.4	>24h	**1458.5**
VI	2520.2	46524.7	553.2	>24h	4032.7
Chen	12844.1	33238.3	>24h	>24h	>24h
PDPIM	>24h	18044.8	**254.2**	**2051.0**	>24h
PIM	>24h	20164.2	348.8	2273.9	>24h
Bertsekas	>24h	33238.3	>24h	>24h	>24h

Table 2: Runtimes for solving different models. Discount $\gamma = 0.9999$, final precision $\varepsilon = 10^{-3}$.

→ Reasonable gain of time for real-world models.
→ Necessity of a very high discount!

Remarks

Experimental Results:

- For toy models, a suitable decomposition generally exists $\rightarrow$ runtime reduced by a factor of 15
- For real-world models: runtime reduced by a factor of 1.3

Limitations:

- Modest improvements for real-world models with non-smooth optimal value functions: leads to trivial abstraction almost immediately
- For maze-like models, state space exploration is slow, with no improvement in Value Iteration or Policy Iteration Methods

Table of Contents

Conclusion

In those slides:

- Hierarchical RL context
- Link between **Approximate DP and State Abstraction**
- Practical **State Abstraction discovery** criterion
- MDP **solving benchmark**

Upcoming work :

- Total reward convergence proof
- Larger benchmark
- How to transpose it to model free ?

Thank you for listening!

Abel, D. (2019).
A theory of state abstraction for reinforcement learning.
In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 9876–9877.

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.

Abel, D., Umbanhowar, N., Khetarpal, K., Arumugam, D., Precup, D., and Littman, M. (2020).
Value preserving state-action abstractions.
In *International Conference on Artificial Intelligence and Statistics*, pages 1639–1650. PMLR.

Bean, J. C., Birge, J. R., and Smith, R. L. (1987).
Aggregation in dynamic programming.
Operations Research, 35(2):215–220.

Bertsekas, D. P., Castanon, D. A., et al. (1988).

Adaptive aggregation methods for infinite horizon dynamic programming.
IEEE Transactions on Automatic Control.

Carpentier, P., Chancelier, J.-P., Leclère, V., and Pacaud, F. (2018).
Stochastic decomposition applied to large-scale hydro valleys management.
European Journal of Operational Research, 270(3):1086–1098.

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.*

Ciosek, K. and Silver, D. (2015).
Value iteration with options and state aggregation.
In *Proceedings of the ICAPS Workshop on Planning and Learning.*

Coulom, R. (2006).

Efficient selectivity and backup operators in monte-carlo tree search.
In *International conference on computers and games*, pages 72–83. Springer.

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

Ferrer-Mestres, J., Dietterich, T. G., Buffet, O., and Chades, I. (2020).
Solving k-mdps.
In *Proceedings of the International Conference on Automated Planning and Scheduling*, volume 30, pages 110–118.

Forghieri, O., Le Pennec, E., Castel, H., and Hyon, E. (2024).
Progressive state space disaggregation for infinite horizon dynamic programming.
In *34th International Conference on Automated Planning and Scheduling*.

Gopalan, N., Littman, M., MacGlashan, J., Squire, S., Tellex, S., Winder, J., Wong, L., et al. (2017).
Planning with abstract markov decision processes.
In *Proceedings of the International Conference on Automated Planning and Scheduling*, volume 27, pages 480–488.

Guestrin, C., Koller, D., Parr, R., and Venkataraman, S. (2003).
Efficient solution algorithms for factored mdps.
Journal of Artificial Intelligence Research, 19:399–468.

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

Jothimurugan, K., Bastani, O., and Alur, R. (2021).
Abstract value iteration for hierarchical reinforcement learning.
In *International Conference on Artificial Intelligence and Statistics*, pages 1162–1170. PMLR.

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

In *AI&M*.

Pineau, J., Gordon, G., Thrun, S., et al. (2003).
Point-based value iteration: An anytime algorithm for pomdps.
In *Ijcai*, volume 3, pages 1025–1032.

Siddiqi, S., Boots, B., and Gordon, G. (2010).
Reduced-rank hidden markov models.
In *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, pages 741–748. JMLR Workshop and Conference Proceedings.

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

Sutton, R. S. and Barto, A. G. (2018).
Reinforcement learning: An introduction.
MIT press.

Sutton, R. S., Precup, D., and Singh, S. (1999).

Between mdps and semi-mdps: A framework for temporal abstraction in reinforcement learning.
Artificial intelligence, 112(1-2):181–211.

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.

Tsitsiklis, J. N. and Van Roy, B. (1996).
Feature-based methods for large scale dynamic programming.
Machine Learning, 22(1-3):59–94.

Winston, W. L. (2004).
Operations research: applications and algorithm.
Thomson Learning, Inc.