State Abstraction Discovery from Progressive Disaggregation Methods

Orso Forghieri
CMAP, École polytechnique, Institut Polytechnique de Paris
orso.forghieri@polytechnique.edu

Hind Castel
Télécom SudParis, Institut Polytechnique de Paris
hind.castel@telecom-sudparis.eu

Emmanuel Hyon
LIP6, Sorbonne Université
emmanuel.hyon@parisnanterre.fr

Erwan Le Pennec
CMAP, École polytechnique, Institut Polytechnique de Paris
erwan.le-pennec@polytechnique.edu

Abstract

The high dimensionality of model-based Reinforcement Learning (RL) and Markov Decision Processes (MDPs) can be reduced using abstractions of the state and action spaces. Although hierarchical learning and state abstraction methods have been explored over the past decades, explicit methods to build useful abstractions of models are rarely provided. In this work, we study the relationship between Approximate Dynamic Programming (ADP) and State Abstraction. We provide an estimation of the approximation made through abstraction, which can be explicitly calculated. We also introduce a way to solve large MDPs through an abstraction refinement process that can be applied to both discounted and total reward criteria. This method allows finding explicit state abstractions while solving any MDP with controlled error. We then integrate this state space disaggregation process into classical Dynamic Programming algorithms, namely Approximate Value Iteration, Q-Value Iteration, and Policy Iteration. We show that this method can decrease the solving time of a wide range of models and can also describe the underlying dynamics of the MDP without making any assumptions about the structure of the problem. We also conduct an extensive numerical comparison and compare our approach to existing aggregation methods to support our claims.

1 Introduction

The Markov Decision Process (MDP) serves as a comprehensive framework for addressing stochastic dynamic control problems. Within this framework, the environment undergoes stochastic evolution, influenced by the actions of an agent. The primary objective is to optimize expected gains through strategic decision-making [Puterman, 2014]. The global objective is to identify the optimal sequence of actions, referred to as a policy, that maximizes the overall return. This pursuit extends to a diverse array of problem domains, as highlighted in a recent overview [Boucherie and van Dijk, 2017]. These encompass challenges in inventory control, energy management, network optimization involving

[1]This work mainly extends the analysis of [Forghieri et al., 2024]. It summarizes the previous theoretical results and provides a wide range of experiments and further interpretations.

queues, and navigating stochastic shortest paths in robot exploration. Achieving near-optimal control is crucial, necessitating precise solutions to address these problems.

Model-based methods for MDPs often struggle with high-dimensional state and action spaces, requiring decomposition techniques. State Abstraction aims to cluster the original state space while minimizing information loss, but the partitioning problem is a complex combinatorial challenge. We introduce a class of iterative aggregation algorithms for infinite horizon problems with expected discounted and total criteria. Our approach integrates state abstraction with Approximate Value Iteration (AVI), Q-Value, and Policy Iteration (PI) algorithms, focusing on spatial abstraction to solve the exact process using aggregation and building abstractions with bounded error. A key innovation is progressive disaggregation during iterations, grouping states with similar evolution under the Bellman operator to enhance efficiency. We provide a convergence proof without structural assumptions, demonstrating reduced computational complexity and valuable abstractions. Numerical comparisons across various models highlight our algorithm's superiority, especially against other abstraction methods.

The structure of the article unfolds as follows: we first link Approximate Dynamic Programming and State Abstraction. We then explicit a way to extract an explicit State Abstraction while diminishing an error to optimal bound. We finally benchmark our disaggregation method over a wide range of models and establish a connection between approximate Bellman operators with State Aggregation and the optimal Bellman operator of the abstract MDP. Subsequently, we articulate a bound on error to optimal value function contingent on the quality of the aggregation employed and then introduce our algorithms. Lastly, we assess the efficacy of our method through benchmarking on classical models, showcasing its efficiency in comparison to alternative approaches.

Related Works Dealing with large spaces is a well-documented challenge in the MDP framework. To overcome this, various strategies decompose complex MDPs into more manageable counterparts. Notably, Factored MDPs [Guestrin et al., 2003] represent states as dynamic feature vectors and use Dynamic Bayesian Networks for compact representation and efficient computation. Another recent approach, Reduced-Rank MDPs [Siddiqi et al., 2010], expresses transition probabilities as scalar products of continuous functions, offering an effective dimensionality reduction technique. A general and promising method for MDP approximation is the hierarchical approach [Hengst, 2012], which considers either temporal abstractions for actions persisting over time [Sutton et al., 1999], or state abstractions by aggregating states into meaningful regions [Li et al., 2006], enhancing efficiency in handling complex MDPs. Considering Partially Observable MDPs, Point-based Value Iteration [Pineau et al., 2003] updates the value function of relevant states to approximate the optimal value function. Monte-Carlo Tree Search [Coulom, 2006] relies on a model-based local policy optimization, but will suffer from a state abstraction that loses the tree structure of the actions. Moreover, most policy-based approaches are geared towards Deep Learning methods using policy gradient. They are efficient in practice, but rarely ensure guarantees on quadratic or sup-norm error.

The challenge of state aggregation in Reinforcement Learning (RL) involves effectively grouping states while ensuring the quality of the abstraction. In Model-Based RL, spatial aggregation methods range from metric-relative approaches to deep learned representations [Starre et al., 2022]. The selection of merging criteria is crucial, with various approaches proposed in the literature. These include bisimulation for state grouping [Dean and Givan, 1997], soft aggregation techniques where states have probabilities of belonging to an aggregated region [Singh et al., 1994], metric-based grouping [Abel et al., 2016] or limiting the number of regions [Ferrer-Mestres et al., 2020]. Evaluating the quality of aggregation typically involves leveraging results from approximated dynamic programming and stochastic optimization [Tsitsiklis and Van Roy, 1996, Abel, 2019]. Additionally, literature explores planning on a fixed aggregation [Gopalan et al., 2017], although such approaches often require information that is not available before addressing the original MDP.

Several techniques have been proposed to construct abstractions without relying on information about the optimal solution of MDPs [Bean et al., 1987, Rogers et al., 1991]. Notably, one introduced aggregation based on the Bellman Residual, contributing significantly to the acceleration of optimization processes [Bertsekas et al., 1988]. Recent works have emphasized the use of options in approximating MDPs for efficient planning. For instance, ones incorporate options into state abstractions to expedite planning processes [Ciosek and Silver, 2015, Abel et al., 2020]. Others propose identifying relevant subgoals to accelerate planning by temporal abstraction [Jothimurugan et al., 2021]. However, these techniques often increase computational complexity. In contrast, one can apply aggregation to Value Iteration (VI) to speed up computations by grouping states with similar values [Chen et al., 2022]. While this approach improves computational speed, it does not fully leverage spatial MDP structure. Our method addresses this gap by maintaining previous aggregations, resulting in a more effective grouping of states with similar optimal values and trajectories through optimal Bellman operator iteration. One further demonstrates the significance of aggregation, but highlight the challenge of refining aggregation without worsening the distance to the optimal value function [Tagorti et al., 2013].

2 Problem Setup

As our approach is grounded in the MDP framework, we first provide notations for Dynamic Programming and then integrate State Abstraction definitions as well as Dynamic Programming ones.

Markov Decision Processes MDPs provide a framework for decision-making optimization [Puterman, 2014]. Formally, a finite MDP is specified as a tuple $\langle \mathcal{S}, \mathcal{A}, T, R, \gamma \rangle$, where $\mathcal{S}$ is the set of possible states, $\mathcal{A}$ is the set of actions that the agent can select, $T(s, a, s') \in [0, 1]$ is the environment transition probability from s to s' under action a and $R(s, a) \in \mathbb{R}$ describes the reward received by the agent in s triggering action a. Finally, we consider bounded rewards and a discount factor $\gamma \in (0, 1]$ to weight the incoming reward priority.

The objective is to maximize the expected sum of discounted immediate rewards in the upcoming trajectory of states for an infinite horizon. The researched solution is a deterministic policy $\pi : \mathcal{S} \mapsto \mathcal{A}$ that can decide which action to select when in state $s \in \mathcal{S}$. For a given policy π, it is thus possible to define the value function that gives a value to each state. It is defined as the expected return applying the policy π, and we have $\forall s \in \mathcal{S}$:

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

The planning problem is centered on maximizing the expected return. In our setting, there exists a non necessarily unique policy π^* such that $V^{\pi^*}(s) = \max_{\pi} V^{\pi}(s)$ simultaneously for all states s. It is worth noting that the optimal value function V^{π^*} (denoted as V^*) is the unique solution to the optimal Bellman Equation

$$V(s) = \max_{a \in \mathcal{A}} \left(R(s, a) + \gamma \sum_{s' \in \mathcal{S}} T(s, a, s') \cdot V(s') \right) , \tag{1}$$

for all $s \in \mathcal{S}$ [Puterman, 2014]. Along this article, we denote by $(\mathcal{T}^* V)(s)$ the right term of Equation (1).

Dynamic Programming Any value function can be computed recursively. Hence, for a given policy $\pi \in \mathcal{A}^{\mathcal{S}}$, we consider here the Bellman operator

$$\mathcal{T}^{\pi} : V \in \mathbb{R}^{\mathcal{S}} \mapsto R^{\pi} + \gamma T^{\pi} \cdot V \in \mathbb{R}^{\mathcal{S}}.$$

with $R^{\pi}(s) = R(s, \pi(s))$ and $T^{\pi}(s, s') = T(s, \pi(s), s')$. This Bellman operator updates any value function V relatively to the reward and transition functions. It is a contraction for the sup-norm and its iteration can lead to a value function solution of the Bellman equation $V = \mathcal{T}^{\pi} V$. One also considers the optimal Bellman operator $\mathcal{T}^*$ defined by Equation (1).

So far, we have considered the state value function V, but a similar analysis can be conducted for the state-action value function Q defined by

$$Q^{\pi}(s, a) = \mathop{\mathbb{E}}_{(s_t, a_t)_t} \left[\sum_{t=0}^{\infty} \gamma^t R(s_t, \pi(s_t)) \,|\, s_0 = s, a_0 = a \right] .$$

The optimal Bellman operator in the Q-value case exists and is defined as

$$\mathcal{T}_Q^* : Q \in \mathbb{R}^{\mathcal{S} \times \mathcal{A}} \to R + \gamma T \cdot \max_{a \in \mathcal{A}} (Q) \in \mathbb{R}^{\mathcal{S} \times \mathcal{A}} .$$

The practical solving of an MDP can be done either by maximizing the expected return V^π for any state or by minimizing the Bellman residual, namely $\|V - \mathcal{T}^* V\|_\infty$, usually iterating the Bellman operator $\mathcal{T}^*$ [Puterman, 2014]. The Policy Iteration algorithm alternates between finding the solution to $V = \mathcal{T}^\pi V$ and updating the current policy π according to

$$\pi_{t+1}(s) \leftarrow \underset{a\in\mathcal{A}}{\arg\max}\left(R\left(s,a\right) + \gamma \sum_{s'\in\mathcal{S}} T\left(s,a,s'\right) V^{\pi_t}(s')\right).$$

State Abstraction The concept of constructing a new MDP through state aggregation has been explored in the literature, particularly in the examination of abstract MDPs [Li et al., 2006]. This involves considering a ground MDP, denoted as $\mathcal{M}_G = \langle \mathcal{S}, \mathcal{A}, T, R, \gamma \rangle$, and creating a new abstract MDP, denoted as $\mathcal{M}_A = \langle \mathcal{S}_A, \mathcal{A}, T_A, R_A, \gamma \rangle$, from it. We first define State Aggregation.

Definition 1 (State aggregation [Tsitsiklis and Van Roy, 1996])**.** *Let $\mathcal{M}_G$ be an MDP and $\mathcal{S} = \bigsqcup_{k=1}^K S_k$ a partition of the state space. We attribute a weight $\omega_k(s) \geq 0$ to each state of a region S_k such that $\sum_{s\in S_k} \omega_k(s) = 1$ and $s \notin S_k \implies \omega_k(s) = 0$. We finally store the weights into a matrix $\omega \in [0,1]^{K\times|\mathcal{S}|}$ and the state-region matching in a matrix ϕ such that $\phi[s,k] = \mathbb{1}_{s\in S_k}$. We then name the tuple $\left((S_k)_{1\leq k\leq K}, \phi, \omega\right)$ a state aggregation.*

When all states of a region are equally weighted, ω can be computed as follows: $\omega_k(s) = \frac{1}{|S_k|}$, which corresponds to $\omega = \left(\phi^T \cdot \phi\right)^{-1} \cdot \phi^T$ [Bertsekas et al., 1988]. Let us note that the following analysis can also be done in the general case of unequally weighted states. From now, the State Abstraction simply consists in building a new MDP using this aggregation.

Definition 2 (Abstract MDP [Li et al., 2006])**.** *Let $\mathcal{M}_G$ be an MDP and $\left((S_k)_{1\leq k\leq K}, \phi, \omega\right)$ a state aggregation. We represent each region S_k by an abstract state s_k. The abstract MDP $\mathcal{M}_A$ can be therefore defined by $\mathcal{S}_A = \{s_k,\ 1 \leq k \leq K\}$, $\mathcal{A}_A = \mathcal{A}$, $T_A = \omega \cdot T \cdot \phi$ and $R_A = \omega \cdot R$.*

The interest of State Abstraction is to reduce the size of the original MDP, gathering states with similar properties like a close optimal value, a close optimal policy or a close optimal Q-value [Abel et al., 2016]. It can be used to approximate the ground optimal policy, but also to highlight a structure in the ground MDP.

Approximate Dynamic Programming While Dynamic Programming involves applying an operator to enhance the current solution, Approximate Dynamic Programming focuses on updating an approximated version of the value function [Powell, 2007]. In our context, we adopt the linear parameterization

$$V_\theta(s) = \sum_{k=1}^K \theta_k \mathbb{1}_{s\in S_k}\ ,$$

with $(S_k)_{1\leq k\leq K}$ a state aggregation. Those value functions are constant over each region S_k. The approximate Bellman operator relative to this family of functions (denoted $\Pi\mathcal{T}^*$) is made of the optimal Bellman operator and a projection matrix Π that averages the value on each region to obtain a piecewise constant value function.

Definition 3 (Projected optimal Bellman operator [Tsitsiklis and Van Roy, 1996])**.** *Let us note $\mathcal{P}$ the set of value functions that are piecewise constant relatively to $(S_k)_k$. We assume that the states are equally weighted in each partition: $\omega_k(s) = \frac{1}{|S_k|}$. Then, the operator $\Pi\mathcal{T}^*$ that checks*

$$\forall V \in \mathbb{R}^{\mathcal{S}},\ \Pi\mathcal{T}^* V \in \underset{V'\in\mathcal{P}}{\arg\min} \|V' - \mathcal{T}^* V\|_2$$

is the projected optimal Bellman operator $\Pi\mathcal{T}^* = \phi \cdot \omega \cdot \mathcal{T}^*$ where ϕ and ω are described in Definition 1.

We define in the same way the projected Bellman operators $\Pi\mathcal{T}^*_Q$ and $\Pi\mathcal{T}^\pi$ for any policy π with $\Pi = \phi \cdot \omega$.

3 Projected Bellman Operators and State Abstraction

We describe here the relationship between the Least Squares approximate Bellman operator and State Abstraction and provide evaluations for its complexity.

Projected Bellman operator and Abstract MDP We consider here the unique fixed point of the projected Bellman operator that satisfies $Q = \Pi\mathcal{T}_Q^* Q$. We claim that this fixed point $\tilde{Q}$, which is piecewise constant as ω averages values, is exactly the optimal Q-value function of the associated Abstract MDP when written in a contracted form *i.e.* without repetitions. We note that this property generalizes to the Bellman operator $\mathcal{T}^\pi$, but not to $\mathcal{T}^*$.

Proposition 1 (Projected Bellman fixed point and Abstract MDP [Forghieri et al., 2024]). *Let $((S_k)_k, \phi, \omega)$ be an aggregation of $\mathcal{S}$ and $\tilde{Q} = \phi \cdot Q$ be the unique fixed point of the operator $\Pi\mathcal{T}_Q^*$ that checks $Q = \Pi\mathcal{T}_Q^* Q$. Then Q is the optimal Q-value function of the abstract MDP $\mathcal{M}_A$ associated to $((S_k)_k, \phi, \omega)$ as described in Definition 2.*

We notice that this property is also checked for the operator $\mathcal{T}^{\tilde{\pi}}$ where $\tilde{\pi}$ is a piecewise-constant policy relatively to $(S_k)_k$ but not for $\mathcal{T}^*$. In Proposition 1, we claim that the solution to $Q = \Pi\mathcal{T}_Q^* Q$ is also the optimal Q-value function of the abstract MDP.

In Proposition 2, we state the convergence of iteration of the operator $\Pi\mathcal{T}^*$ when applied to any $V_0 \in \mathbb{R}^{\mathcal{S}}$. This property is also true for the operators $\Pi\mathcal{T}_Q^*$ and $\Pi\mathcal{T}^\pi$ [Forghieri et al., 2024].

Proposition 2 (Convergence of Projected Bellman operator iteration [Bertsekas et al., 1988]). *Let $((S_k)_k, \phi, \omega)$ be any aggregation of $\mathcal{S}$. Then the sequence*

$$\begin{cases} V_0 \in \mathbb{R}^{\mathcal{S}} \\ V_{t+1} := \phi \cdot \omega \cdot \mathcal{T}^* V_t \end{cases}$$

converges to the fixed point equation solution $V = \phi \cdot \omega \cdot \mathcal{T}^* V$.

In Table 1, we summarize the complexity of each projected Bellman operator.

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

Table 1: Complexity of the different projected Bellman operators.

The complexities of projected operators are smaller than the $|\mathcal{S}|^3 |\mathcal{A}|$ complexity for $\mathcal{T}^*$. Having established that computing projected operators is more straightforward than traditional ones, we introduce an algorithm that combines region value updates and disaggregation process.

4 Disaggregation and State Abstraction production

In this section, we first estimate the approximation made using State Abstraction through an explicit bound and then describe a disaggregation process that tends to diminish that bound.

Theorem 1 (Optimal Error Bound with Arbitrary Partition [Forghieri et al., 2024]). *Let $\tilde{V} \in \mathbb{R}^{\mathcal{S}}$ be any piecewise constant value function. Its distance to the optimal value function V^* can be bounded as follows:*

$$\|\tilde{V} - V^*\|_\infty \leq \frac{1}{1-\gamma} \max_{1\leq k\leq K} \operatorname{Span}_{S_k}\left(\mathcal{T}^*\tilde{V}\right) + \frac{1}{1-\gamma}\|\tilde{V} - \Pi\mathcal{T}^*\tilde{V}\|_\infty\,, \tag{2}$$

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

We furthermore note that $\max_{1\leq k\leq K} \operatorname{Span}_{S_k}\left(\mathcal{T}^*\tilde{V}\right)$ measures how much the aggregation groups states having the same value and that $\|\tilde{V} - \Pi\mathcal{T}^*\tilde{V}\|_\infty$ estimates the optimality of the current piecewise value function relatively to the projected Bellman operator. Let us note that the quantity $\|\tilde{V} - \Pi\mathcal{T}^*\tilde{V}\|_\infty$ can be arbitrarily reduced iterating $\Pi\mathcal{T}^*$ as the operator $\Pi\mathcal{T}^*$ contracts space with a factor γ. Inequation 2 can also be formulated using $\Pi\mathcal{T}_Q^*$ and $\Pi\mathcal{T}^{\pi_G}$ for any piecewise constant policy π_G.

Disaggregation process We therefore propose an algorithm with initialization $\tilde{V}_0 = (0)_{s\in\mathcal{S}}$ and a unique region $S_1 = \mathcal{S}$. We then iterate the two following steps successively:

- 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 $k \in [\![1\ ;\ K]\!]$.

When applying this process, we separate states having different trajectories through VI. In fact, it considers an abstract MDP, computes an approximation of its value function, and then refines the current abstraction through the operator $\mathrm{UpdateRegion}$, which breaks heterogeneous regions. Moreover, although the $\Pi\mathcal{T}^*$ operator changes at each region division step, the goal is also to take advantage of the time savings from the projected Bellman operator application compared to the optimal ground one. We provide the pseudocode of Progressive Disaggregation Value Iteration (PDVI) in Algorithm 1. We also name PDQVI and PDPI the action-value and policy versions of this method.

Algorithm 1 Progressive Disaggregation Value Iteration

```
Input: $\mathcal{M} = \langle \mathcal{S}, \mathcal{A}, T, R, \gamma \rangle$, $\varepsilon > 0$
Output: A value $V$, an aggregation $(S_k)_k$ of the state space
 1: $K := 1$, $S_1 := \mathcal{S}$, $\underline{V}_0 := (0)_{1\le k\le K}$
 2: while
        $\|\underline{V}_t - \Pi\mathcal{T}^*\underline{V}_t\|_\infty + \max_{1\le k\le K} \mathrm{Span}_{S_k}(\mathcal{T}^*\underline{V}_t) > 2\varepsilon$
    do
 3:     $V_{t+1} := \mathcal{T}^*(\phi \cdot \underline{V}_t)$
 4:     if $\max_{1\le k\le K} \mathrm{Span}_{S_k}(V_{t+1}) > \varepsilon$ then
 5:         $(S_k)_k = \mathrm{UpdateRegion}(k, V_k, (S_k)_k, \varepsilon)$
 6:     end if
 7:     while $\|\underline{V}_t - \Pi\mathcal{T}^*\underline{V}_t\|_\infty > \varepsilon$ do
 8:         $\underline{V}_{t+1} \leftarrow \Pi\mathcal{T}^*\underline{V}_t$
 9:     end while
10: end while
11: return $(\underline{V}, (S_k)_k)$
```

The operator $\mathrm{UpdateRegion}(k, V_k, \{S_k\}_k, \varepsilon)$ simply divides any region that has a highly variable value function $V(S_k)$: if region k satisfies $\max_{s\in S_k} V(s) - \min_{s\in S_k} V(s) > \varepsilon$, it is divided into smaller blocks where the value function V varies by less than ε. Algorithm 1 is based on the criterion specified in Equation 1. We also note that this method can be transposed to Q-VI (that we name Progressive Disaggregation Q-Value Iteration, PDQVI) and to the Policy Evaluation step of Modified Policy Iteration (that we name Progressive Disaggregation Policy Iteration Modified, PDPIM). Moreover, we emphasize that the operator $\mathrm{UpdateRegion}$ presented here can be replaced with any operator that refines regions, as long as the new operator divides at least one region when applied.

Building adapted abstraction In Algorithm 1, starting from a single region, we separated states having a different optimal Bellman update $\mathcal{T}^*\tilde{V}_t$. Consequently, we grouped states with similar optimal values in the same region. This property is formalized in the Proposition 4.

Proposition 3 (Final abstraction characterization [Forghieri et al., 2024]). *Let $\mathcal{S} = \bigsqcup_k S_k$ the final state space partition. Then, the following inequation is checked:*

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

Assuming the final abstraction is not trivial, we built an abstraction that contains information about the MDP structure: each region contains homogeneous states with similar behavior, as indicated by their similar optimal Bellman updates. Since the Bellman update reflects the link between a state and the related ones, we grouped states with similar relationships to the rest of the state space and similar rewards. In this work, we succeeded in building abstractions that associate states with similar optimal values. For instance, the Four Rooms model with 100 states was reduced to a smaller model with only 18 states. In the following section, we conduct an experiment that includes a runtime benchmark and descriptions of the practical abstraction constructed.

5 Application

We conducted a benchmark[1] of our approach on multiple scalable models. We compared PDVI, PDQVI and PDPI to usual VI, Modified PI, as well as Bertsekas' approach [Bertsekas et al., 1988] and Chen's Adaptive Aggregation [Chen et al., 2022]. These last two methods leverage aggregation-disaggregation processes to accelerate dynamic programming updates. However, unlike our approach, Bertsekas and Chen do not gather information on the MDP throughout the process. In particular, Bertsekas' method is based on Policy Iteration. Its policy evaluation benefits from accelerated convergence by grouping states and updating blocks of the current value function using a linear approximation of the exact Bellman iteration. However, this method involves small matrix inversions, which slows down the process. Chen's approach consists of applying Temporal-Difference block updates to the current value function alongside exact Value Iteration steps. This approach, however, suffers from slow transition sampling and stochastic approximation updates.

In our method, we first start with a trivial one-state abstraction. As explicited in Proposition 1, we used the span criterion to refine those abstraction. One main point of the method is that we disaggregate and thus distinguish states having different trajectories through the exact Bellman update. At the end of the process, it is then possible to extract a non-trivial State Abstraction which transcripts the underlying dynamics of the model. Our comparison with a diverse set of solving methods shows that our disaggregation algorithms outperform other methods on most of the models while explaining their structures. We selected multiple types of models for our study, including a Stochastic Shortest Path problem (Four Rooms [Hengst, 2012]), two toy control models (Mountain Car [Moore, 1990] and Sutton's racetrack [Sutton and Barto, 2018]), two real-world models (tandem queues [Tournaire et al., 2022] and hydro-valley management [Carpentier et al., 2018]) and randomly generated MDPs [Archibald et al., 1995]. As VI has been shown to be efficient for small discount factors and Policy Iteration for larger ones [Kaelbling et al., 1996], we considered a high discount factor (up to 0.9999) and diverse problems where both value-based and policy-based approaches perform well. The state space size was also increased to up to a million states. We ran our experiments on one thread of a CPU Intel Xeon @ 3.00GHz, using Python with `numpy` and `scipy` sparse matrices with at most 64GB of RAM. We present the results in the following.

5.1 Random models

Random models have been largely used in the literature to benchmark MDP solvers [Archibald et al., 1995, Grand-Clément and Kroer, 2021, Bhatnagar et al., 2009]. The transition functions $T(s, a, .)$ are randomly drawn on $\mathcal{S}$ and the reward follows a centered normal distribution with choosen variance. We set $|\mathcal{S}| = 500$ and $|\mathcal{A}| = 50$ and a variable proportion of nonzero entries (named density) in the transition matrix. As the density of the transition matrix impacted the most the optimal value function shape, we set a maximum of diversity in this parameter going from 1% (almost empty matrix) to 65% (two over three pairs of state are connected by a nonzero transition) of nonzero entries. As shown in Table 2, our disaggregation methods demonstrate their advantages for both value and policy-based approaches. Small transition matrices densities induce independent states while higher densities of transition matrices smooth the optimal value function.

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

Density	PDVI	PDQVI	PDPI	VI	PIM	Bertsekas	Chen
1%	6.6	8.0	**1.1**	113.3	3.0	2.8	>1h
10%	7.5	15.2	**1.6**	300	1.7	2.5	>1h
25%	6.2	24.1	**0.7**	752	1.1	1.5	>1h
45%	7.6	36.3	**0.6**	1398	1.8	2.0	>1h
65%	6.7	50.3	**1.6**	1915	2.7	3.3	>1h

[1]The code is available at `https://github.com/OrsoF/state_space_disaggregation.git`

5.2 Toy models

This section introduces the results we obtained applying our method to the Four Rooms, Mountain Car and racetrack models. The rooms grid model [Hengst, 2012] is made of four rooms with doors. The goal is to reach a given square, the exit. Each action (N, S, E or W) leads to the adjacent state with a probability 0.8, otherwise we stay in place.

The Mountain Car problem [Moore, 1990] is an MDP where the car, starting in a valley, must use limited acceleration actions to reach the goal by optimizing its position and velocity through two actions, left and right accelerations. The state space is made of couple of real values representing the car position and velocity. The reward penalizes each move with a -1 value until the upper hill is reached. We present the solving time in Table 3, Table 4 and Table 5.

The Sutton's racetrack model [Sutton and Barto, 2018] involves driving a racecar while selecting directions and speed. It is a stochastic model where there is a probability of slipping and going off the road. The goal is to reach the exit of the track as quickly as possible.

Table 3: Four Rooms model solving time (s). Discount $\gamma = 0.9999$, final precision $\varepsilon = 10^{-3}$.

$\lvert\mathcal{S}\rvert$	PDVI	PDQVI	PDPI	VI	PIM	Bertsekas	Chen
900	25.3	**9.2**	12.4	15.0	66.0	170.6	640.9
19600	375.6	**14.6**	1131.9	223.2	4199.7	4726.9	3240.3
193600	11712.5	**160.3**	>24h	2520.2	>24h	>24h	12844.1
1000000	>24h	**1622.4**	>24h	16658.8	>24h	>24h	>24h

Table 4: Mountain Car model solving time (s). Discount $\gamma = 0.9999$, final precision $\varepsilon = 10^{-3}$.

$\lvert\mathcal{S}\rvert$	PDVI	PDQVI	PDPI	VI	PIM	Bertsekas	Chen
400	24.8	12.0	4.0	**0.1**	0.2	3.8	**0.1**
19600	1463.8	**92.2**	761.6	146.4	2041.1	1878.8	849.1
193600	16271.8	**239.4**	>24h	1561.8	>24h	>24h	2697.3
1000000	>24h	**437.1**	>24h	>24h	>24h	>24h	>24h

Table 5: Sutton's racetrack model solving time (s). Discount $\gamma = 0.9999$, final precision $\varepsilon = 10^{-3}$.

$\lvert\mathcal{S}\rvert$	PDVI	PDQVI	PDPI	VI	PIM	Bertsekas	Chen
210	35.8	22.5	**4.1**	18.4	5.7	10.6	75.5
735	55.7	23.8	**4.5**	28.5	8.6	45.4	388.8
1260	75.7	25.9	**4.8**	48.3	12.4	77.3	407.7
2310	125.6	26.4	**5.7**	60.2	16.1	31.2	426.0
3360	163.9	26.5	**6.8**	81.1	20.2	84.1	457.3

Considering the Four Rooms model, we succeeded in extracting an exactly equivalent abstraction through our process. For instance, the 10×10 model boils down to an MDP with 17 states. Considering the model structure, we first recover the exit and then, step by step, extract states that are at the same distance from the exit. At the same time, the current exit value information is transmitted to the related regions, including the furthest states, which is still accelerating the learning process. These properties were also observed in the Mountain Car model, where we reduced the state space from 100 states to only 20 with an exact contraction of the MDP. We also found that our disaggregation method is quite robust to small perturbations: it still discovers the shortest path structure of the Four Rooms model even when adding a small random value to the reward (e.g., -1.1 instead of -1) or making slight changes to the transition probabilities. The method uncovers the structure of the problem by distinguishing between dangerous states (near the track boundaries) and safer states (towards the center of the track).

5.3 Real-world models

In this section, we first develop two models, Tandem Queue and Hydro Valley. We then consider the runtimes that we obtain solving them. Finally, we describe the structure of the problem that our method highlighted.

We considered tandem queues management inspired from a real-world server operation [Ohno and Ichiki, 1987]. Here, the agent scales two servers relatively to the load of two tandem queues with parallel servers. There are 3 actions (add, keep or remove a server) for each queue, which gives 9 actions in total. We could scale here the size of the queue and the size of the server to adjust the state space dimension.

The Hydro Valley model [Carpentier et al., 2018] consists of a chain of dams located on the same river. Each dam can turbinate a given amount of water, which produces electricity. The turbinated water then flows to the next dam downstream. The reward is determined by electricity prices. The state space comprises the volumes of water in the different dams, and the action space consists of the turbination capacities. Both the state space and the action space can be scaled according to the dam capacities. Using these two models, we obtained the runtimes in Table 6 and Table 7.

Table 6: Tandem Queue model solving time (s). Discount $\gamma = 0.9999$, final precision $\varepsilon = 10^{-3}$.

$\lvert\mathcal{S}\rvert$	PDVI	PDQVI	PDPI	VI	PIM	Bertsekas	Chen
441	36.8	27.2	**8.4**	20.5	9.8	23.7	9775.0
5929	280.1	151.3	**44.0**	155.8	69.6	427.0	>24h
19600	979.4	657.4	**254.2**	553.2	348.8	>24h	>24h

Table 7: Hydro Valley model solving time (s). Discount $\gamma = 0.9999$, final precision $\varepsilon = 10^{-3}$.

$\lvert\mathcal{S}\rvert$	PDVI	PDQVI	PDPI	VI	PIM	Bertsekas	Chen
729	43.8	31.7	**4.0**	23.7	6.7	>24h	>24h
15625	6008.4	4288.8	**89.2**	3740.4	130.3	>24h	>24h
117649	>24h	>24h	**2051.0**	>24h	2273.9	>24h	>24h

We therefore note an improvement of 10% to 50% in the computing times, and that the best approach is policy-based. This last observation can be explained by the high irregularity of the optimal value function. Despite this irregularity, the approximate step still improves the basic Dynamic Programming method.

We also discovered such underlying structure in both models. First, the Tandem Queue model abstraction gathered states having a similar number of clients in the queue as well as a similar number of servers. This simplification made the problem much more manageable as we abstracted the essential description of the current model state. Concerning the dams, the model precisely describes the water quantity and the upcoming water for each dam. The aggregation found allowed us to also group states having similar quantities of water inside each dam as well as similar upcoming water quantities.

6 Conclusion

Approaching the exact MDP solution remains a question that deeply depends on the problem structure. In this context, we present an approximation method that combines state abstraction refinement and approximate value iteration to accelerate traditional dynamic programming algorithms while explaining the deep structure of the MDP.

We first focused on the link between approximate value iteration and state abstraction. We then provided an estimation of the approximation made through state abstraction and its value function. We concluded that this bound can be refined and contains a very interesting criterion to reduce the state space while approximating the original MDP. A benchmark of this method demonstrated its effectiveness when applied to specific problems in comparison to the traditional dynamic programming approach and two alternative aggregation methods.

However, algorithmic improvements can still be made. The method still struggles with some real-world models, particularly those associated with non-smooth optimal value functions. As an extension of this work, this challenge could be tackled through a weighted norm while conserving the convergence guarantee. Additionally, the disaggregation process would greatly benefit from being extended to more complex problems, such as partially observable MDPs, and in the model-free context when associated with deep learning methods.

References

[Abel, 2019] 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 et al., 2016] 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 et al., 2020] 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.

[Archibald et al., 1995] 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.

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

[Bertsekas et al., 1988] Bertsekas, D. P., Castanon, D. A., et al. (1988). Adaptive aggregation methods for infinite horizon dynamic programming. *IEEE Transactions on Automatic Control*.

[Bhatnagar et al., 2009] Bhatnagar, S., Sutton, R. S., Ghavamzadeh, M., and Lee, M. (2009). Natural actor–critic algorithms. *Automatica*, 45(11):2471–2482.

[Boucherie and van Dijk, 2017] Boucherie, R. J. and van Dijk, N. M. (2017). *Markov decision processes in practice*, volume 248. Springer.

[Carpentier et al., 2018] 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 et al., 2022] 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 and Silver, 2015] Ciosek, K. and Silver, D. (2015). Value iteration with options and state aggregation. *arXiv preprint arXiv:1501.03959*.

[Coulom, 2006] 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 and Givan, 1997] Dean, T. and Givan, R. (1997). Model minimization in markov decision processes. In *AAAI/IAAI*, pages 106–111.

[Ferrer-Mestres et al., 2020] 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 et al., 2024] 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 et al., 2017] 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.

[Grand-Clément and Kroer, 2021] Grand-Clément, J. and Kroer, C. (2021). Scalable first-order methods for robust mdps. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 12086–12094.

[Guestrin et al., 2003] 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, 2012] Hengst, B. (2012). Hierarchical approaches. In *Reinforcement learning*, pages 293–323. Springer.

[Jothimurugan et al., 2021] 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.

[Kaelbling et al., 1996] Kaelbling, L. P., Littman, M. L., and Moore, A. W. (1996). Reinforcement learning: A survey. *Journal of artificial intelligence research*, 4:237–285.

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

[Moore, 1990] Moore, A. W. (1990). Efficient memory-based learning for robot control. Technical report, University of Cambridge, Computer Laboratory.

[Ohno and Ichiki, 1987] Ohno, K. and Ichiki, K. (1987). Computing optimal policies for controlled tandem queueing systems. *Operations Research*, 35(1):121–126.

[Pineau et al., 2003] 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.

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

[Puterman, 2014] Puterman, M. L. (2014). *Markov decision processes: discrete stochastic dynamic programming*. John Wiley & Sons.

[Rogers et al., 1991] Rogers, D. F., Plante, R. D., Wong, R. T., and Evans, J. R. (1991). Aggregation and disaggregation techniques and methodology in optimization. *Operations research*, 39(4):553–582.

[Siddiqi et al., 2010] 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 et al., 1994] Singh, S., Jaakkola, T., and Jordan, M. (1994). Reinforcement learning with soft state aggregation. *Advances in neural information processing systems*, 7.

[Starre et al., 2022] Starre, R. A., Loog, M., and Oliehoek, F. A. (2022). Model-based reinforcement learning with state abstraction: A survey. In *Benelux Conference on Artificial Intelligence*, pages 133–148. Springer.

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

[Sutton et al., 1999] 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.

[Tagorti et al., 2013] Tagorti, M., Scherrer, B., Buffet, O., and Hoffmann, J. (2013). Abstraction pathologies in markov decision processes. In *8èmes Journées Francophones sur la Planification, la Décision et l'Apprentissage pour la conduite de systèmes*.

[Tournaire et al., 2022] 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 and Van Roy, 1996] Tsitsiklis, J. N. and Van Roy, B. (1996). Feature-based methods for large scale dynamic programming. *Machine Learning*, 22(1-3):59–94.