Progressive State Space Disaggregation for Infinite Horizon Dynamic Programming

Orso Forghieri[1]**, Hind Castel**[2]**, Emmanuel Hyon**[3, 4]**, Erwan Le Pennec**[1]**,**

[1] CMAP, École Polytechnique, Institut Polytechnique de Paris,
[2] Télécom SudParis, Institut Polytechnique de Paris,
[3] LIP6, Sorbonne Université,
[4] Université Paris Nanterre
{orso.forghieri, erwan.le-pennec}@polytechnique.edu, hind.castel@telecom-sudparis.eu, emmanuel.hyon@parisnanterre.fr

Abstract

High dimensionality of model-based Reinforcement Learning and Markov Decision Processes 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 provide a new state abstraction method for solving infinite horizon problems in the discounted and total settings. Our approach is to progressively disaggregate abstract regions by iteratively slicing aggregations of states relatively to a value function. The distinguishing feature of our method, in contrast to previous approximations of the Bellman operator, is the disaggregation of regions during value function iterations (or policy evaluation steps). The objective is to find a more efficient aggregation that reduces the error on each piece of the partition. We provide a proof of convergence for this algorithm without making any assumptions about the structure of the problem. We also show that this process decreases the computational complexity of the Bellman operator iteration and provides useful abstractions. We then plug this state space disaggregation process in classical Dynamic Programming algorithms, namely Approximate Value Iteration, Q-Value Iteration and Policy Iteration. Finally, we conduct a numerical comparison, which shows that our algorithm is faster than both traditional dynamic programming approach and recent aggregative methods that use a fixed number of partitions.

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

Regarding the MDP state and action spaces, most model-based methods suffers from high dimensionality and requires decomposition techniques. To this end, the State Abstraction approach aims to cluster the original state space while minimizing the information loss. The partition problem remains an open complex combinatorics challenge. In this context, we introduce a class of iterative aggregation algorithms for solving infinite horizon problems with both expected discounted and total criteria. Our approach integrates state abstraction with Approximate Value Iteration (AVI), Q-Value, and Policy Iteration (PI) algorithms. Our work focuses on spatial abstraction, not only solving the exact process using aggregation but also introducing a novel method for building abstractions with bounded error.A key innovation is the progressive disaggregation along iteration steps, grouping states with similar evolution under the Bellman operator application and enhancing algorithm efficiency. We provide a convergence proof without structural assumptions, showcasing reduced computational complexity and valuable abstractions. Numerical comparisons across various models highlight our algorithm's superiority in the MDP literature, especially against other abstraction methods.

The structure of the article unfolds as follows: we first 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, Precup, and Singh 1999), or state abstractions by aggregating states into meaningful regions (Li, Walsh, and Littman 2006), enhancing efficiency in handling complex MDPs. Considering Partially Observable MDPs, Point-based Value Iteration (Pineau et al. 2003) uses one state to represent a given region. Monte-Carlo Tree Search (Coulom 2006) rely on a model-based local policy optimization, but will suffer from a state abstraction that loose 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, as comprehensively reviewed by 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. 1995), 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, the approach pioneered by Bertsekas, Castanon et al. (1988) introduced aggregation based on the Bellman Residual, contributing significantly to the acceleration of optimization processes. Recent works have emphasized the use of options in approximating MDPs for efficient planning. For instance, Ciosek and Silver (2015) and Abel et al. (2020) incorporate options into state abstractions to expedite planning processes. Jothimurugan et al. (2021) propose identifying relevant subgoals to accelerate planning by temporal abstraction. However, these techniques often increase computational complexity. In contrast, Chen et al. (2022) apply aggregation to Value Iteration (VI) to speed up computations by grouping states with similar values. 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. Tagorti et al. (2013) further demonstrate the significance of aggregation, but highlight the challenge of refining aggregation without worsening the distance to the optimal value function.

Problem Setup

Our approach is grounded in MDPs, a well-documented field. We clarify notations, outline Dynamic Programming methods for model resolution, and integrate recent advancements in State Abstraction and Approximate Dynamic Programming.

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, it 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 a MDP, can be done either by maximizing the expected return V^π for any state or by minimizing the Bellman residual namely $\|V - \mathcal{T}^* V\|_\infty$. The Dynamic Programming methods generally aim to decrease the Bellman residual. In the VI algorithm (respectively Q-VI), one iterates the contracting optimal Bellman operator to approximate the fixed point solution of the optimal Bellman equation $V^* = \mathcal{T}^* V^*$ (respectively $Q^* = \mathcal{T}_Q^* Q^*$) (Puterman 2014). PI algorithm alternates between finding the solution to $V = \mathcal{T}^\pi V$ and updating the current policy π according to

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

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, Walsh, and Littman 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). *Let $\mathcal{M}_G$ be an MDP and $\mathcal{S} = \bigsqcup_{k=1}^K S_k$ a partition of the state space. Let us assume we give a weight $\omega_k(s)$ to each state of a region S_k relatively to the other states of this region S_k. The weights ω_k are positive and sum to 1 on each region. Moreover, if $s \notin S_k$, $\omega_k(s) = 0$. We finally store the weights into a matrix $\omega \in [0,1]^{K\times|\mathcal{S}|}$ and the state-region correspondence in a matrix ϕ such that $\phi[s,k] = \mathbb{1}_{s\in S_k}$. It is now possible to define a state aggregation by the tuple $((S_k)_{1\leq k\leq K}, \phi, \omega)$.*

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, Castanon 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 from this aggregation.

Definition 2 (Abstract MDP, (Li, Walsh, and Littman 2006)). *Let $\mathcal{M}_G$ be an MDP and $((S_k)_{1\leq k\leq K}, \phi, \omega)$ 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 state 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 function that are piecewise constant relatively to $(S_k)_k$. Then, the operator $\Pi\mathcal{T}^*$ that checks*

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

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

In the following sections, we will consider each of the projected Bellman operators $\Pi\mathcal{T}^*$, $\Pi\mathcal{T}_Q^*$ and $\Pi\mathcal{T}^\pi$ for any policy π with $\Pi = \phi \cdot \omega$.

Projected Bellman Operators and State Abstraction

In what follows, we describe the relationship between the projected Bellman operators and State Abstraction. We first prove that a projected Bellman operator is exactly the Bellman operator of a smaller abstract MDP. As we want to implement those operators, we evaluate theirs complexities and compare it to the optimal Bellman operator $\mathcal{T}^*$.

Projected Bellman Equations and Abstract MDP We are now interested in the unique solution of each of the equations $Q = \Pi\mathcal{T}_Q^* Q$ and $V = \Pi\mathcal{T}^\pi V$. We will namely prove that those projected equations are the Bellman equations for the associated abstract MDP. Let us note that it does not generalize to the equation $V = \Pi\mathcal{T}^* V$. Indeed, as Q and V^π contain action information through Q and π, any value function solution to $\tilde{V} = \Pi\mathcal{T}\tilde{V}$ is not necessarily associated with a piecewise constant policy. The solution of $V = \Pi\mathcal{T}^* V$ is therefore not necessarily the optimal value function of the abstract MDP.

It is now interesting to note that value functions that are solutions of these equations $Q = \Pi\mathcal{T}_Q^* Q$ and $V = \Pi\mathcal{T}^\pi V$ are piecewise constant. Indeed, the operator Π makes the function being constant over the regions S_k. We are therefore adopting the following notations. Let $\tilde{V} \in \mathbb{R}^{\mathcal{S}}$ be a piecewise constant value function relatively to a partition $(S_k)_{1\leq k\leq K}$. The entries of $\tilde{V}$ are in a way redundant: for any state $s \in S_k$, $\tilde{V}(s)$ has the same value. We therefore build a contracted representation $\underline{V} \in \mathbb{R}^K$ which contains a single value for each region: $\forall s \in S_k, \tilde{V}(s) = \underline{V}(k)$. This new value function $\underline{V}$ can be a value function to the associated abstract MDP. Moreover, it is possible to switch between $\tilde{V}$ and $\underline{V}$ using the relations $\tilde{V} = \phi \cdot \underline{V}$ and $\underline{V} = \omega \cdot \tilde{V}$. We use similar notations for the Q-value: $\tilde{Q}$ and $\underline{Q}$.

In the following proposition, we suggest that the solution to $Q = \Pi \mathcal{T}_Q^* Q$ is also the optimal Q-value function of the abstract MDP.

Proposition 1. *Let $((S_k)_k, \phi, \omega)$ be an aggregation of the state space. Let $\tilde{Q} = \phi \cdot \underline{Q}$ be the unique solution to the Q-projected optimal Bellman equation $Q = \Pi \mathcal{T}_Q^* Q$. Then $\underline{Q}$ is the optimal Q-value function of the abstract MDP $\mathcal{M}_A$ described in Definition 2.*

The proof simply consists in establishing that the equation $Q = \Pi \mathcal{T}_Q^* Q$ can be written as the optimal Bellman equation $Q = \mathcal{T}_Q^* Q$ for the abstract MDP.

Proof. Let $\tilde{Q} = \phi \cdot \underline{Q}$ be the unique solution to $Q = \Pi \mathcal{T}_Q^* Q$. Let Q_A^* be the optimal Q-value of the abstract MDP $\mathcal{M}_A$. Let us show that those Q-values are the solution to the same equation. The equation $\tilde{Q} = \Pi \mathcal{T}_Q^* \tilde{Q}$ can be written:

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

which is precisely the optimal Bellman equation for the abstract MDP $\mathcal{M}_A$. As the solution to each of the equation is unique, we can conclude that $\underline{Q} = Q_A^*$. □

As Abel et al. (2016), we focus here on an arbitrary policy π_A defined on the abstract state space $\mathcal{S}_A$. We define its generalization π_G to the ground state space $\mathcal{S}$, by the piecewise constant policy given by:

$$\pi(s) = \pi_A(s_k)\,,\ \forall s \in S_k\,, \quad \forall k \in [\![1\ ;\ K]\!].$$

Proposition 1 has an equivalent for the $\mathcal{T}^\pi$ operator. Hence, in Proposition 2, we state that the value of any abstract policy $\tilde{V}^{\pi_A}$ is the solution of a projected Bellman equation $\tilde{V}^{\pi_A} = \Pi \mathcal{T}^{\pi_A} \tilde{V}^{\pi_A}$ at the ground level.

Proposition 2. *Let $((S_k)_k, \phi, \omega)$ be an aggregation of the state space. Let $\pi_A : \mathcal{S}_A \mapsto \mathcal{A}$ be an arbitrary policy and $\pi_G : \mathcal{S} \mapsto \mathcal{A}$ its generalization to the ground state space $\mathcal{S}$. Then, the value of this policy on the abstract MDP $\underline{V}^{\pi_A}$ is the solution of the following projected Bellman equation*

$$\phi \cdot \underline{V}^{\pi_A} = \Pi \mathcal{T}^{\pi_G} \left(\phi \cdot \underline{V}^{\pi_A}\right).$$

The proof still relies on the unicity of the solution of a fixed-point equality.

Proof. In the following, we prove the equivalence of the equations

$$\tilde{V} = \Pi \mathcal{T}^{\pi_G} \tilde{V} \qquad \text{and} \qquad \underline{V} = \mathcal{T}^{\pi_A} \underline{V}$$

which suffices to conclude on the proposition.

$$\begin{aligned}
&\tilde{V} = \Pi \mathcal{T}^{\pi_G} \tilde{V} \iff \phi \cdot \underline{V} = \Pi \mathcal{T}^{\pi_G} \left(\phi \cdot \underline{V}\right) \\
\iff &\phi \cdot \underline{V} = \phi \cdot \omega \cdot \left(R^{\pi_G} + \gamma \cdot T^{\pi_G} \cdot \phi \cdot \underline{V}\right) \\
\iff &\underline{V} = \omega \cdot R^{\pi_G} + \gamma \cdot \omega \cdot T^{\pi_G} \cdot \phi \cdot \underline{V} \\
\iff &\underline{V} = R_A^{\pi_A} + \gamma \cdot T_A^{\pi_A} \underline{V} \iff \underline{V} = \mathcal{T}^{\pi_A} \underline{V}
\end{aligned}$$

Those equivalences imply the wanted equality and therefore on the property. □

As we proved here that the solution of the projected Bellman equation is the optimal value function of an abstract MDP, we now study the complexity of iterating a projected Bellman operator.

Iterations of Projected Bellman Operators In this part, we prove the convergence of any sequence of value functions (or Q value function) on which we iterate any projected Bellman operator.

Proposition 3. *1. Let $Q_0 \in \mathbb{R}^{\mathcal{S} \times \mathcal{A}}$ be an arbitrary Q-value function and let the iteration $Q_{t+1} \leftarrow \Pi \mathcal{T}_Q^* Q_t$ be. Then the series $(Q_t)_{t \in \mathbb{N}}$ converges to the unique solution to the projected optimal Bellman equation $Q = \Pi \mathcal{T}_Q^* Q$.*

2. Let $\pi \in \mathcal{A}^{\mathcal{S}}$ be an arbitrary policy. Let $V_0 \in \mathbb{R}^{\mathcal{S}}$ be any value function, and let the iteration $V_{t+1} \leftarrow \Pi \mathcal{T}^\pi V_t$ be. Then (V_t) converges to the unique solution to the projected Bellman equation $V = \Pi \mathcal{T}^\pi V$.

Bertsekas (2018) proved the contraction property of the operator $\Pi \mathcal{T}^*$. We generalize it to $\Pi \mathcal{T}_Q^*$ and $\Pi \mathcal{T}^\pi$ for any policy π to prove Proposition 3.

Proposition 4. *The operators $\Pi \mathcal{T}_Q^*$ and $\Pi \mathcal{T}^\pi$ for any policy π are contracting.*

Proof. The proof relies on the contraction induced by the Bellman operator and on the following inequality true for any $\mathcal{T} \in \{\mathcal{T}_Q^*, \mathcal{T}^\pi\}$:

$$\|\phi \cdot \omega \cdot (\mathcal{T}V - \mathcal{T}V')\|_\infty \leq \|\mathcal{T}V - \mathcal{T}V'\|_\infty$$

as ϕ simply repeats the entries in any vector $V \in \mathbb{R}^{\mathcal{S}}$. This property can be applied to $\Pi \mathcal{T}_Q^*$ and $\Pi \mathcal{T}^\pi$ for any policy π to conclude the proof. □

From now on, iterating any of the operator $\mathcal{T}^*$, $\Pi \mathcal{T}^*$, $\Pi \mathcal{T}_Q^*$ or $\Pi \mathcal{T}^\pi$ makes any value function converge to a unique piecewise constant final value function. In the following, we will be interested in the complexity of the computation of the solution of the projected Bellman equation and will propose a bound on the error to the optimal value function depending on the specific aggregation and on the current value.

Projected Operator Complexity Now, we will consider the complexity of computing the projected Bellman operators $\Pi \mathcal{T}^*$, $\Pi \mathcal{T}_Q^*$ and $\Pi \mathcal{T}^{\pi_G}$ for any piecewise constant policy π_G.

Proposition 5. *The complexity of the computation of the projected Bellman operators $\Pi \mathcal{T}_Q^*$ and $\Pi \mathcal{T}^{\pi_G}$ for any piecewise constant policy π_G are respectively $O(K^3\,|\mathcal{A}|)$ and $O(K^3)$.*

Proof. As $\Pi\mathcal{T}^*_Q$ and $\Pi\mathcal{T}^{\pi_G}$ can be viewed as the Bellman operators $\mathcal{T}^*_{\underline{\mathrm{Q}}}$ and $\mathcal{T}^{\pi_A}$, then theirs complexities can be computed from the abstract MDP point of view. We therefore deduce it from the matrix computations

$$\mathcal{T}^*_{\underline{\mathrm{Q}}}(\underline{\mathrm{Q}}) = R_A + \gamma \cdot T_A \cdot \max_{a\in\mathcal{A}}(\underline{\mathrm{Q}})$$

and

$$\mathcal{T}^{\pi_A}(\underline{\mathrm{V}}) = R_A^{\pi_A} + \gamma \cdot T_A^{\pi_A} \cdot \underline{\mathrm{V}}$$

knowing that the complexity of the product $M \cdot N$, with $M \in \mathbb{R}^{l\times m}$, and $N \in \mathbb{R}^{m\times n}$ is equal to $l \cdot m \cdot n$. □

We consider now the computation complexity of the projected optimal Bellman operator $\Pi\mathcal{T}^*$ that we will iterate.

Proposition 6. *For any piecewise constant value function $\tilde{V}$, the number of operations to compute $\Pi\mathcal{T}^*\tilde{V}$ is $O(|\mathcal{S}|^2\,K\,|\mathcal{A}|)$.*

Proof. Considering

$$\Pi\mathcal{T}^*\left(\phi\cdot\underline{\mathrm{V}}\right) = \phi\cdot\omega\max_{a\in\mathcal{A}}\left(R + \gamma\cdot T\cdot\phi\cdot\underline{\mathrm{V}}\right)$$

the precomputation of the matrix product $T\cdot\phi \in \mathbb{R}^{|\mathcal{S}|\times\mathcal{A}\times K}$ allows the matrix product $(T\cdot\phi)\cdot\underline{\mathrm{V}}$ to have a complexity of $O(|\mathcal{S}|^2\,K\,|\mathcal{A}|)$. □

The complexities of $O(|\mathcal{S}|^2\,K\,|\mathcal{A}|)$ for $\Pi\mathcal{T}^*$, $O(K^3\,|\mathcal{A}|)$ for $\Pi\mathcal{T}^*_Q$, and $O(K^3)$ for $\Pi\mathcal{T}^\pi$ are much smaller than the $O(|\mathcal{S}|^3\,|\mathcal{A}|)$ complexity for $\mathcal{T}^*$. Having established that computing projected operators is more straightforward than traditional ones, we introduce an algorithm to systematically disaggregate regions into smaller ones, facilitating the evolution of a piecewise constant value function.

Progressive Disaggregation Process

In this section, we first establish a bound between a given piecewise constant value function and the optimal value function of any MDP. This bound depends on the aggregation quality (*i.e.* the capacity to aggregate states with the same value function) and the projected Bellman residual $\tilde{V} - \Pi\mathcal{T}\tilde{V}$ but does not use the optimal value function. We then provide the Progressive Disaggregation algorithm which is based on this bound: we iteratively improve the aggregation quality (reducing one term of the bound of Theorem 1) and decrease the projected Bellman residual by applying the projected Bellman operator.

Theorem 1 (Optimal Error Bound with Arbitrary Partition)**.** *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:*

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

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

Proof. We mainly use the classical inequality:

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

and also the following one:

$$\forall V \in \mathbb{R}^{\mathcal{S}},\ \|V - \Pi V\|_\infty \leq \max_{1\leq k\leq K}\mathrm{Span}_{S_k}(V)\,.$$

Concatenating inequalities, we get:

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

□

We furthermore note that $\max_{1\leq k\leq K}\mathrm{Span}_{S_k}\left(\mathcal{T}^*\tilde{V}\right)$ 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.

Corollary 1. *Inequality 2 can also be formulated using $\Pi\mathcal{T}^*_Q$ and $\Pi\mathcal{T}^{\pi_G}$ for any piecewise constant policy π_G.*

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 value iteration. Moreover, note that 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.

Proposed Algorithms

In this section, we provide the pseudocode for the algorithm that we described previously. We give its adaptation to Q-Value Iteration and Modified Policy Iteration. We then prove the convergence of the algorithm and lead a complexity analysis to conclude on its performance condition.

Formulation In Algorithm 1, we describe the *Progressive Disaggregation Value Iteration* (**PDVI**) process. It can be summarized into “While the bound of the Theorem 1 is not lower than ϵ, alternate between dividing heterogeneous regions and updating the piecewise constant value function $\tilde{V}$”. To fully benefit from the disaggregation process, it is

```
Algorithm 1: Progressive Disaggregation Value Iteration
Input: $\mathcal{M} = \langle \mathcal{S}, \mathcal{A}, T, R, \gamma \rangle$, $\epsilon > 0$
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
        $\|\underline{V}_t - \Pi\mathcal{T}^*\underline{V}_t\|_\infty + \max_{1 \le k \le K} \mathrm{Span}_{S_k}(\mathcal{T}^*\underline{V}_t) > 2\epsilon$
    do
 3:     $V_{t+1} := \mathcal{T}^*(\phi \cdot \underline{V}_t)$
 4:     if $\max_{1 \le k \le K} \mathrm{Span}_{S_k}(V_{t+1}) > \epsilon$ then
 5:         $(S_k)_k = \mathrm{UpdateRegion}(k, V_k, (S_k)_k, \epsilon)$
 6:     end if
 7:     while $\|\underline{V}_t - \Pi\mathcal{T}^*\underline{V}_t\|_\infty > \epsilon$ do
 8:         $\underline{V}_{t+1} \leftarrow \Pi\mathcal{T}^*\underline{V}_t$
 9:     end while
10: end while
11: return $(\underline{V}, (S_k)_k)$
```

```
Algorithm 2: UpdateRegions
Input: $V$, $(S_k)_k$, $\epsilon$
Output: Updated partition $(S'_k)_k$
for $l \in [\![1 ; K]\!]$ do
    if $\mathrm{Span}_{S_l}(V) > \epsilon$ then
        $(S_k)_k := (S_k)_k \backslash S_l$
        for $p \in [\![0 ; \lceil \frac{1}{\epsilon}(\max V_{|S_l} - \min V_{|S_l}) \rceil]\!]$ do
            $I_p := [\min V_{|S_l} + p.\epsilon, \min V_{|S_l} + (p+1).\epsilon]$
            $S_p := \{s \in S_l, V(s) \in I_p\}$
            if $S_p \neq \emptyset$ then
                $(S_k)_k := (S_k)_k \sqcup S_p$
            end if
        end for
    end if
end for
return Updated partition $\{S_k\}$
```

possible to implement the UpdateRegion function ensuring a $O(|\mathcal{S}|)$ complexity.

As this algorithm consists in iterating $\Pi\mathcal{T}^*$ and dividing regions along $\mathcal{T}^*V$, we generalize it to the Q-value process by applying $\Pi\mathcal{T}_Q^*$ and divide the regions along $\mathcal{T}_Q^*\tilde{Q}$. We name this new algorithm *Progressive Disaggregation Q-Value Iteration* (**PDQVI**). In the region division step, we ensure for any region k

$$\mathrm{Span}_{S_k}\left(\mathcal{T}_Q^*\tilde{Q}\right) := \max_{S_k \times \mathcal{A}} \mathcal{T}_Q^*\tilde{Q} - \min_{S_k \times \mathcal{A}} \mathcal{T}_Q^*\tilde{Q} \le \epsilon.$$

Moreover, PDQVI provides a state abstraction gathering states with close optimal Q-value.

We also propose a policy-based algorithm named *Progressive Disaggregation Policy Iteration* (**PDPI**) based on Modified Policy Iteration (MPI). In MPI, we start from an arbitrary given policy. We then iteratively evaluate its value function V^π (being the solution of $V^\pi = \mathcal{T}^\pi V^\pi$) and update the policy using V^π. In PDPI, we changed the Policy Evaluation part into a disaggregation process, progressively dividing regions and evaluating the solution of $V = \Pi\mathcal{T}^\pi V$ at the same time.

Convergence Property In this part, we state a guarantee of convergence for PDVI, PDQVI and PDPI and characterize the aggregation provided by the algorithms.

Proposition 7 (Convergence Guarantee for PDVI)**.** *Let* $(\underline{V}, (S_k)_k)$ *denote the value and the abstraction computed by PDVI. Then, the following properties hold.*

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

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

where the precision ϵ *is an input of the algorithm. Moreover, for any region* $k \in [\![1 ; K]\!]$*,*

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

We stated here that PDVI converges with an accuracy similar to VI and aggregates states having close optimal values.

Proof. 1. At first, let us prove that Algorithm 1 stops within $|\mathcal{S}| + 1$ steps by contradiction. Let us assume that

$$\|\underline{V}_t - \Pi\mathcal{T}^*\underline{V}_t\|_\infty + \max_{1 \le k \le K} \mathrm{Span}_{S_k}(\mathcal{T}^*\underline{V}_t) > 2\epsilon$$

after $|\mathcal{S}| + 2$ steps. As at the end of each step $t \in [\![0 ; |\mathcal{S}| + 2]\!]$, $\|\underline{V}_t - \Pi\mathcal{T}^*\underline{V}_t\|_\infty \le \epsilon$ (due to the lines 7-9 condition), then it follows that $\max_{1 \le k \le K} \mathrm{Span}_{S_k}(\mathcal{T}^*\underline{V}_t) > \epsilon$ as the *while* condition is not fulfilled. We therefore deduce that for each of the steps $t \in [\![1 ; |\mathcal{S}| + 1]\!]$, a disaggregation step has occurred. Given that for each disaggregation step, the number of regions strictly increases, we deduce that at step $t = |\mathcal{S}| + 1$, the state aggregation is made of $|\mathcal{S}| + 1$ regions which is impossible. We then conclude on the finite number of step of PDVI.

2. The final precision condition

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

is ensured by the while loop condition and Theorem 1. Finally, let us show that the regions $(S_k)_k$ group states having close optimal value. Let $k \in [\![1 ; K]\!]$ be any region and $s, s' \in S_k$ be states. We observe that

$$\begin{aligned} &|V^*(s) - V^*(s')| \\ &\quad \le \left|V^*(s) - \tilde{V}(s)\right| + \left|\tilde{V}(s) - V^*(s')\right| \\ &\quad = \left|V^*(s) - \tilde{V}(s)\right| + \left|\tilde{V}(s') - V^*(s')\right| \le \frac{4\epsilon}{1-\gamma} \end{aligned}$$

Second step comes from $\tilde{V}(s) = \tilde{V}(s')$ as s and s' are in the same region. The last inequality can be stated using the final precision of the algorithm.

□

We also mention that PDQVI aggregates states having close optimal Q-value and PDPI also groups states having close optimal value V^*. Both converge and provide optimal Q-value and optimal policies following the same steps for the proof. We add that the proof of the policy-based disaggregation algorithm convergence contains some subtlety and precise that we keep the value function V^π from one policy evaluation to the next one. We finally state that PDPI still converges in the expected total-reward criterion ($\gamma = 1$). This convergence result can only be checked with the assumption $R \geq 0$ as we use the convergence properties of PIM in the expected total-reward case (Puterman 2014).

Complexity Analysis

We provide here a complexity analysis for PDVI, PDQVI and PDPI and see that the worst-case complexity of those algorithms can be higher than the traditional ones. This characterization will also explain why our disaggregation method can be outperformed by traditional ones on some specific models that we identify. Hence, to prove the bounds, one considers an unichain model (Puterman 2014).

Proposition 8 (Disaggregation Algorithm Complexity)**.** *Let us assume that any VI-like algorithm takes n steps to reach an ϵ-close optimal value. To reach an ϵ-optimal value function, our algorithms require the following number of operations.*

Algorithm	PDVI	PDQVI	PDPI
Operations	$O(n\,\lvert\mathcal{S}\rvert^4\,\lvert\mathcal{A}\rvert)$	$O(n\,\lvert\mathcal{S}\rvert^4\,\lvert\mathcal{A}\rvert)$	$O(n\,\lvert\mathcal{S}\rvert^4)$

Proof. Let us evaluate the complexity of $PDVI$. We first claim that there exists a model for which solving with this algorithm requires a maximum number of disaggregation steps ($|\mathcal{S}|$ steps). For each visited aggregation, we then count the operations necessary to approximate the solution of the projected optimal Bellman equation $\tilde{V} = \Pi\mathcal{T}^*\tilde{V}$. Finally, we sum the operations required at each visited aggregation to deduce the overall computational complexity.

Let us consider an unichain model made of $|\mathcal{S}|$ states, assuming the exit is on the first state s_0. The PDVI process initially distinguishes the exit from others states ($\{s_0\} \sqcup \{s_1 \cdots, s_{|\mathcal{S}|}\}$) and then discovers the immediately connected states ($\{s_0\} \sqcup \{s_1\} \sqcup \{s_2 \cdots, s_{|\mathcal{S}|}\}$) and so on, which results in $|\mathcal{S}|$ disaggregation steps.

Considering the execution of $|\mathcal{S}|$ iterations of the $\Pi\mathcal{T}^*$ operator and according to Proposition 6, each iteration of $\Pi\mathcal{T}^*$ takes $O(K\,|\mathcal{S}|^2\,|\mathcal{A}|)$ operations. We assume it takes n operations to approximate the solution of the projected optimal Bellman equation $\tilde{V} = \Pi\mathcal{T}^*\tilde{V}$ with an accuracy ϵ. The total number of operations through the $|\mathcal{S}|$ iterations can be estimated as $O(|\mathcal{S}|^2\,|\mathcal{A}|\sum_{K=1}^{|\mathcal{S}|} K) = O(|\mathcal{S}|^4\,|\mathcal{A}|)$.

The same kind of argument allows evaluating the complexity of Progressive Disaggregation Q-Value Iteration and Progressive Disaggregation Policy Iteration. □

Let us note that Value Iteration algorithm takes at most $O(n\,|\mathcal{S}|^3\,|\mathcal{A}|)$ operations (Feinberg and He 2020) and Policy Iteration requires $O(|\mathcal{S}|^3)$ (Littman, Dean, and Kaelbling 1995). We obviously lose in complexity during disaggregation in some specific cases that we detailed here. Nevertheless, we claim (based on numerical experiments performed later) that these worst-case bounds are not reached in practice, and that the average complexity of our algorithm is much better.

Numerical Results

We conducted a benchmark of our approach on three scalable models. We compared PDVI, PDQVI and PDPI to usual VI, Modified PI, as well as Bertsekas' approach (Bertsekas, Castanon et al. 1988) and Chen's Adaptive Aggregation (Chen et al. 2022)[1]. These two methods leverage aggregation-disaggregation processes to accelerate dynamic programming updates. However, unlike our approach, they do not gather information on the MDP throughout the process. Our comparison with a diverse set of solving methods shows that our disaggregation algorithms outperform other methods on most of the models.

We selected configurable models with variable state space and action space sizes. We evaluated MDPs on a randomly generated transition matrix, a toy model (four rooms), and a real-life problem. The Random MDPs are commonly used in the literature to benchmark solvers (Archibald, McKinnon, and Thomas 1995; Bhatnagar et al. 2009). The Four Rooms model is a stochastic shortest path model (Hengst 2012; Sutton and Barto 2018) that we scaled to explore the state space size impact. Finally, we faced a real-life queue management situation with scalable servers and queue sizes (Ohno and Ichiki 1987; Tournaire et al. 2022) already used for benchmark (Puterman 2014).

We ran our experiments on one thread of a CPU Intel Xeon @ 3.00GHz, using Python with `numpy` and `scipy` with at most 16GB of RAM. As Chen Adaptive Aggregation method was behind the other value-based method by at least a factor 2, we did not keep it in the numerical results.

Random Models Our slicing strategy gave its best on random models. We drew random distributions $T\,(s, a, .)$ on $\mathcal{S}$ for any $(s, a) \in \mathcal{S} \times \mathcal{A}$. We also build R with random coefficients in $[0, 1]$. 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 1, our disaggregation methods demonstrate their advantages for both value and policy-based approaches. Small transition matrices densities induce independent states while higher densities of T smooth the optimal value function.

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

Density	VI	PDVI	PDQVI	MPI	PDPI	Bertsekas
1%	113.3 ± 1.0	6.6 ± 0.5	8.0 ± 0.4	3.0 ± 1.25	$\mathbf{1.09 \pm 0.23}$	2.8 ± 0.6
10%	300.3 ± 10.9	7.5 ± 0.1	15.2 ± 0.3	1.65 ± 0.46	$\mathbf{1.57 \pm 0.45}$	2.5 ± 0.3
25%	751.7 ± 16.0	6.2 ± 0.6	24.1 ± 0.8	1.17 ± 0.08	$\mathbf{0.72 \pm 0.11}$	1.5 ± 0.4
45%	1397.7 ± 23.7	7.6 ± 1.3	36.3 ± 1.7	1.83 ± 0.32	$\mathbf{0.61 \pm 0.21}$	2.0 ± 0.2
65%	1915.4 ± 54.2	6.7 ± 0.4	50.3 ± 3.6	2.86 ± 1.03	$\mathbf{1.57 \pm 0.74}$	3.3 ± 0.7

Table 1: Random MDPs mean solving time (s). $\mathcal{S} = 500$, $\mathcal{A} = 50$, $\gamma = 0.99$, $\varepsilon = 10^{-2}$, 10 experiments.

$\lvert\mathcal{S}\rvert$	VI	PDVI	PDQVI	MPI	PDPI	Bertsekas
8100	12.1 ± 0.5	$\mathbf{8.0 \pm 1.3}$	15.3 ± 0.7	1442.5 ± 39.2	267.5 ± 5.6	1626.1 ± 13.4
12544	41.5 ± 0.8	$\mathbf{18.8 \pm 1.8}$	35.3 ± 1.6	4211.0 ± 63.1	994.7 ± 6.3	3577.2 ± 14.8

Table 2: Tandem Queues model mean solving time (s). $\lvert\mathcal{S}\rvert \in \{8100, 12544\}$, $\lvert\mathcal{A}\rvert = 3$, $\gamma = 0.99$, $\varepsilon = 10^{-2}$, 10 experiments.

$\lvert\mathcal{S}\rvert$	VI	PDVI	PDQVI	MPI	PDPI	Bertsekas
36	2.72 ± 0.0	7.46 ± 0.4	103.28 ± 0.7	2 ± 1	$\mathbf{1 \pm 0.1}$	1 ± 0.5
100	3.63 ± 0.1	6.77 ± 1.7	267.63 ± 2.6	18 ± 3	$\mathbf{2 \pm 0.7}$	19 ± 0.9
196	3.57 ± 0.4	9.25 ± 2.7	276.04 ± 2.5	29 ± 4	$\mathbf{3 \pm 0.4}$	29 ± 0.9
324	10.25 ± 0.8	14.16 ± 5.0	456.31 ± 7.9	47 ± 7	$\mathbf{10 \pm 1.2}$	47 ± 0.6

Table 3: Four Rooms model mean solving time (s). $\lvert\mathcal{S}\rvert \in \{36, 100, 196, 324\}$, $\lvert\mathcal{A}\rvert = 4$, $\gamma = 0.999$, $\varepsilon = 10^{-3}$, 10 experiments.

Real Model 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. We present the results for $\lvert\mathcal{S}\rvert \in \{8100, 12544\}$. According to common hypothesis in this domain, we chose a queue size (15 and 16) greater than the server size (6 and 7) inspired by Tournaire et al. (2022).

In the results shown in Table 2, the disaggregation method still outperforms MPI, VI, and Bertsekas' approach. This model is particularly quickly solved by value-based methods, and the PDVI process benefits from region updates instead of updates on single states.

Classical Model We finally considered the grid model four rooms (Hengst 2012). It is made of four rooms (5×5 states for each) with doors connecting adjacent rooms. The goal is to reach a given square and each action (N, S, E or W) leads to the adjacent state with a probability .8 (if it is accessible) and otherwise leads to stay in the same state. When reaching the exit, the agent returns to the starting room. We scaled the model size in order to make it more complex. The model is very sparse with a density of $2/\lvert\mathcal{S}\rvert$, which implies a sparsity of at least 98% for our instance. The slicing algorithm performs better as the state space dimension increases for the policy-based version. We increased the discount factor up to 0.999 and the final precision to 10^{-3} so that the value-based methods are outperformed by policy-based ones. The results are shown in Table 3. Note that the partition found by our method gathers states which are equidistant from the exit.

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 and aggregation methods to accelerate traditional dynamic programming algorithms.

We focused on three main aspects. Initially, we established a robust connection between the projected Bellman operator and the abstract MDP's Bellman operator, extending it to the Q-value case and introducing a policy-based version. Following that, we presented a bound on the distance to the optimal value function based on a given state abstraction, leading to a progressive disaggregation process to refine state partitions. Our algorithm tests demonstrated the effectiveness of this approach, particularly in solving MDPs with dense transition matrices. Compared to Modified Policy Iteration and other Adaptive Aggregation methods, the policy-based approach significantly outperformed in solving realistic MDP instances with well-known models.

However, further testing on MDP instances is needed. Algorithmic improvements should be implemented to switch to dynamic programming when partitioning is inefficient. Tailoring our approach specifically to challenges akin to the shortest path problem is crucial. Taking inspiration from the approximation of Tsitsiklis and Castañon, this method could also be combined with progressive disaggregation process. Future work should also investigate generalizations to the model-free Reinforcement Learning problem, incorporating not only state grouping but also the approximation of the state space using Deep Learning methods.

References

Abel, D. 2019. A theory of state abstraction for Reinforcement Learning. In *Proceedings of the Doctoral Consortium of the AAAI Conference on Artificial Intelligence*.

Abel, D.; Umbanhowar, N.; Khetarpal, K.; Arumugam, D.; Precup, D.; and Littman, M. 2020. Value preserving state-action abstractions. In *AISTATS*, 1639–1650. PMLR.

Abel, D.; et al. 2016. Near optimal behavior via approximate state abstraction. In *ICML*, 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.

Bean, J. C.; et al. 1987. Aggregation in Dynamic Programming. *Operations Research*, 35(2): 215–220.

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. *IEEE transactions on automatic control*.

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

Boucherie, R. J.; and van Dijk, N. M. 2017. *Markov Decision Processes in practice*, volume 248. Springer.

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 5th Workshop on Planning and Learning, ICAPS*.

Coulom, R. 2006. Efficient selectivity and backup operators in Monte-Carlo tree search. In *International conference on computers and games*, 72–83. Springer.

Dean, T.; and Givan, R. 1997. Model minimization in Markov Decision Processes. In *AAAI/IAAI*, 106–111.

Feinberg, E. A.; and He, G. 2020. Complexity bounds for approximately solving discounted MDPs by value iterations. *Operations Research Letters*, 48(5): 543–548.

Ferrer-Mestres, J.; Dietterich, T. G.; Buffet, O.; and Chades, I. 2020. Solving k-MDPs. In *ICAPS*, volume 30, 110–118.

Gopalan, N.; Littman, M.; MacGlashan, J.; Squire, S.; Tellex, S.; Winder, J.; Wong, L.; et al. 2017. Planning with abstract Markov Decision Processes. In *ICAPS*, volume 27, 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*, 293–323. Springer.

Jothimurugan, K.; et al. 2021. Abstract value iteration for hierarchical Reinforcement Learning. In *International Conference on Artificial Intelligence and Statistics*, 1162–1170. PMLR.

Li, L.; Walsh, T. J.; and Littman, M. L. 2006. Towards a unified theory of state abstraction for MDPs. *ISAIM*.

Littman, M. L.; Dean, T. L.; and Kaelbling, L. P. 1995. On the complexity of solving Markov decision problems. *11th UAI*, 394–402.

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

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

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

Puterman, M. L. 2014. *Markov Decision Processes: discrete stochastic Dynamic Programming*. John Wiley & Sons.

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, S.; et al. 2010. Reduced-rank hidden Markov models. In *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, 741–748. JMLR Workshop and Conference Proceedings.

Singh, S.; et al. 1995. Reinforcement Learning with soft state aggregation. *Advances in neural information processing systems*, 361–368.

Starre, R. A.; et al. 2022. Model-Based Reinforcement Learning with State Abstraction: A Survey. In *Benelux Conference on Artificial Intelligence*, 133–148. Springer.

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.

Tagorti, M.; Scherrer, B.; Buffet, O.; and Hoffmann, J. 2013. Abstraction Pathologies In Markov Decision Processes. In *Proceedings of the 8th Journées Francophones Planification, Décision, et Apprentissage (JFPDA-13)*.

Tournaire, T.; Jin, Y.; Aghasaryan, A.; Castel-Taleb, H.; and Hyon, E. 2022. Factored Reinforcement Learning for autoscaling in tandem queues. In *NOMS 2022-2022 IEEE/IFIP*, 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.