Solving Markov decision processes via state space decomposition and time aggregation

Rodrigo e Alvim Alexandre[a], Marcelo D. Fragoso[b], Virgílio J. M. Ferreira Filho[a], Edilson F. Arruda[c,*]

[a]*Programa de Engenharia de Produção, Instituto Alberto Luiz Coimbra de Pós Graduação e Pesquisa de Engenharia, Universidade Federal do Rio de Janeiro, Av. Horácio Macedo, 2030, Prédio do Centro de Tecnologia, Bloco F, Sala F-103, Rio de Janeiro, 21941-972, Rio de Janeiro, Brazil*

[b]*Laboratório Nacional de Computação Científica, Av. Getúlio Vargas, 333, Petrópolis, 25651-075, Rio de Janeiro, Brazil*

[c]*Department of Decision Analytics and Risk, Southampton Business School, University of Southampton, Boldrewood Innovation Campus, Burgess Road, Southamton, SO16 7QF, Hampshire, UK*

Abstract

Although there are techniques to address large scale Markov decision processes (MDP), a computationally adequate solution of the so-called curse of dimensionality still eludes, in many aspects, a satisfactory treatment. In this paper, we advance in this issue by introducing a novel multi-subset partitioning scheme to allow for a distributed evaluation of the MDP, aiming to accelerating convergence and enable distributed policy improvement across the state space, whereby the value function and the policy improvement step can be performed independently, one subset at a time. The scheme's innovation hinges on a design that induces communication properties that allow us to evaluate time aggregated trajectories via absorption analysis, thereby limiting the computational effort. The paper introduces and proves the convergence of a class of distributed time aggregation algorithms that combine the partitioning scheme with two-phase time aggregation to distribute the computations and accelerate convergence. In addition, we make use of Foster's sufficient conditions for stochastic stability to develop a new theoretical result which underpins a partition design that guarantees that large regions of the state space are rarely visited and have a marginal effect on the system's performance. This enables the design of approximate algorithms to find near-optimal solutions to large scale systems by focusing on the most visited regions of the state space. We validate the approach in a series of experiments featuring production and inventory and queuing applications. The results highlight the potential of the proposed algorithms to rapidly approach the optimal solution under different problem settings.

Keywords: Markov processes, Dynamic programming, Markov decision processes, time aggregation, Foster's stochastic stability conditions

1. Introduction

Simple and powerful algorithms such as value iteration and policy iteration are very efficient to solve small scale Markov decision processes (MDP) (Puterman, 2005). Moreover, a number of algorithmic variants were developed

*Corresponding author

Email addresses: alvim.rodrigo@pep.ufrj.br (Rodrigo e Alvim Alexandre), frag@lncc.br (Marcelo D. Fragoso), virgilio@ufrj.br (Virgílio J. M. Ferreira Filho), e.f.arruda@southampton.ac.uk (Edilson F. Arruda)

such as Modified Policy Iteration (Puterman & Shin, 1982; Puterman, 2005), Prioritized Sweeping (Moore & Atkeson, 1993), Topological Value Iteration (Dai & Goldsmith, 2007) and General Accelerated Value Iteration (GAVI) (Shlakhter et al., 2010). In Modified Policy Iteration (Puterman, 2005), the policy evaluation step is not solved exactly. Instead, it is tackled via a number of iterations of successive approximations, with a fixed decision rule for updating the policy. In contrast, prioritised sweeping algorithms update the states' value function in a prioritised order by using prioritisation metrics such as the Bellman error or others based, for example, on transition probabilities (Mohagheghi et al., 2020) or mean first passage time (Debnath et al., 2018). Analogously, topological value iteration utilises the graphical structure of an MDP by dividing it into strongly-connected components, and updates the states' value function estimates in a topological order. Finally, General Accelerated Value Iteration (GAVI) (Shlakhter et al., 2010) utilises a class of modified Bellman operators within classical value iteration to accelerate convergence. However, the exponential increase in the number of states and actions in complex real-world problems renders all such algorithms impractical for a myriad of real-world applications (Powell, 2011, 2019).

To circumvent this issue, one can employ a number of different techniques. For example, heuristic solutions for specific problem classes (Malekipirbazari, 2025), approximate dynamic programming (Powell, 2011), reinforcement learning (Sutton & Barto, 2018; Xu et al., 2024), state aggregation (Bertsekas et al., 2000) and time aggregation (Cao et al., 2002) aim to find a satisfactory solution within a reasonable computational time whilst championing different analytical techniques.

In practice, to deal with *large state space problems*, approximate dynamic programming (ADP) algorithms often exploit an approximate form of the solution (e.g., Jiang & Powell, 2015; Hu et al., 2020) or resort to approximated models (e.g., Arruda et al., 2013) while reinforcement learning (RL) algorithms rely on sampling the system's dynamics (e.g., Watkins, 1989; Kamanchi et al., 2019; John et al., 2020). Analogously, local policy and value iteration adaptive dynamic programming (Wei et al., 2017, 2018) rely on updating the value function and control law of a subset of the states at a time. The algorithms, however, were designed for deterministic discrete-time nonlinear systems with continuous state and action spaces, and require that all states be updated infinitely often to ensure convergence.

To tackle large scale MDPs, state aggregation (e.g., Xue et al., 2022) employs an approximate low-dimensional model that arbitrarily merges subsets of the original state space. A downside of this approach is that it fails to preserve the Markov property (Cao et al., 2002). To address such drawback, Cao et al. (2002) introduced the time aggregation approach, which preserves the Markov property by iterating on a carefully designed embedded stochastic process of reduced dimension. The rationale is to divide the state space into two disjoint subsets: a small subset of controllable or more important states, and a large subset that comprises all the remaining states (Arruda & Fragoso, 2011). To find the optimal solution, one must iterate in a semi-Markov decision process (SMDP) with equivalent performance measure and embedded within the controllable states, either via policy-iteration (Cao et al., 2002) or value-iteration-based algorithms (Sun et al., 2007; Arruda & Fragoso, 2011). Time aggregation reaches the optimal solution when the larger subset is comprised of non-controllable states. Otherwise, it will reach a sub-optimal solution by optimising within an arbitrary region of the state space whilst applying a fixed ad-hoc control policy in the remaining states

(Arruda & Fragoso, 2011, 2015).

To ensure that an optimal policy can always be reached via time aggregation, Arruda & Fragoso (2015) proposed a two-phase approach. The first phase seeks a sub-optimal policy by optimising within a small subset F of the state space S whilst selecting an ad-hoc policy for all the remaining states in F^c. The second phase uses properties of the embedded SMDP to derive the value function across all states and find an improved control policy in F^c, which will become the fixed ad-hoc policy at the next step. The two phases are alternated up to convergence. Two-phase time aggregation, however, requires the evaluation of trajectories leaving and returning to the embedded domain. This becomes computationally intensive as the cardinality of the state space increases. To alleviate computations, Arruda et al. (2019) introduced a partition scheme with multiple subsets. Derived for Markov chains, the approach provides only a partial solution to the computational issues, as it does not include control and therefore is only suitable for policy evaluation. In this paper, we will extend the multi-subset partitioning approach to Markov decision processes by deriving a novel scheme that exploits the connectivity properties of each state under all possible control actions.

An innovation of this work is the design of a partitioning scheme that allows us to embed the original MDP into an equivalent SMDP via simple absorption analysis within each individual subset of the partition. This is equivalent to evaluating *outbound* trajectories within each subset, which implies a computational effort that depends on the cardinality of the subset. Hence, by designing the partitioning scheme and its subsets, one will be able to control and limit the overall computational effort. This provides increased flexibility and a clear improvement over two-phase time aggregation (Arruda & Fragoso, 2015), which requires the evaluation of *inbound* trajectories that evolve within a very large subset F^c whose cardinality is fixed and analogous to that of the whole state space. One can argue that the proposed partitioning scheme is akin to Topological Value Iteration (TVI) (Dai & Goldsmith, 2007), as it exploits the MDP's connectivity properties. However, as TVI decomposes the state space into strongly connected components, its performance is tied to the MDP's topological properties; indeed, TVI's performance has been reported to degrade for MDP's with large strongly connected components (Dai & Goldsmith, 2007). The partitioning scheme proposed here avoids such an issue, as the communication properties within the subsets are ascertained by design to allow a distributed computation of the value function, whereby the value function and the policy improvement step can be performed independently, one subset at a time.

Relying upon the evaluation semi-regenerative trajectories, our approach bears some conceptual similarity with rollout algorithms (Powell, 2011; Bertsekas, 2021; Cakir et al., 2023), as the latter evaluate the value function based on simulated fixed-length trajectories starting from the current state. The complexity of the trajectories, however, grows with the length of the look-ahead horizon as do the number of possible destinations, thus limiting the algorithm's performance. In contrast, our approach relies on time aggregation to give rise to semi-regenerative trajectories of random duration but with a fixed set of destination states established by design. Additionally, we propose a partitioning scheme with multiple subsets and communication properties established by design that limit the length of the semi-regenerative trajectories, as we prove that each trajectory will be constrained to a single subset.

To guide the choice of the partition subsets, we prove that Foster's sufficient conditions for stochastic stability

(Foster, 1953; Brémaud, 2020) imply an exponentially decreasing steady state probability with respect to the Foster-Lyapunov function outside of a subset of the state space S. We utilise this result to propose a partition configuration based on the magnitude of the Foster-Lyapunov function, which enables the design of efficient approximate algorithms for large scale systems by focusing on the most visited states and applying ad-hoc approximations in the remote regions of the state space, as these will be rarely visited and therefore contribute marginally to the long-term performance.

This work proposes a class of distributed two-phase time aggregation algorithms that make use of the proposed partitioning scheme and prove their convergence to the optimal solution. We validate the approach by exploring some promising implementations in the light of a set of inventory and queue management problems. The results show an interesting behaviour, namely that the proposed algorithms tend to reach the vicinity of the optimal solution significantly faster than policy iteration whilst significantly outperforming both value iteration and the original two-phase time aggregation approach of Arruda & Fragoso (2015). The results illustrate the potential of the method to tackle large scale average cost MDP problems by using a distributed computation scheme that takes advantage of a carefully designed partitioning scheme that gives rise to favourable communication properties.

This paper is organised as follows. Section 2 introduces the studied problem and conveys properties of semi-regenerative trajectories within the time aggregation approach. In Section 3, we propose a novel design for a partitioning scheme that is based on the communication properties of the states of the MDP. The section also investigates the properties that such design induces into the semi-regenerative trajectories, and proposes a configuration of the partition subsets based on Foster's stochastic stability conditions (Foster, 1953; Brémaud, 2020). Section 4 utilises the communication properties of the semi-regenerative trajectories under the proposed partitioning design to derive a class of distributed two-phase time aggregation algorithms. The section also proves that an algorithm of such a class converges to the MDP's optimal solution in finite time. Sections 5, 6 and 7 present numerical experiments that illustrate the efficiency of the proposed approach in the light of production and inventory and queuing management examples. The proposed class of algorithms significantly outperforms traditional policy and value iteration, as well as the original two-phase time aggregation approach, which highlights its enormous potential. Finally, Section 8 concludes the paper.

2. Problem Statement

Consider a Markov decision process (MDP) whose evolution is described by a Markov Chain $X_t, t \geq 0$ with a finite state space S of cardinality $|S|$. Let $A(i) \subset \mathbb{N}$ be the finite set of feasible control actions at state $i \in S$ and $A = \bigcup_{i \in S} A(i)$ be the finite set of feasible actions across the state space. At each period $t \geq 0$, an agent takes an action $a \in A$ according to a stationary control policy $\mathcal{L} : S \to A$ among the set of all feasible stationary control policies, $\mathbb{L}$. If $X_t = i$ at time t, then $X_{t+1} = j \in S$ with probability p_{ij}^a. Figure 1 illustrates a Markov decision process. Let $f : S \times A \to \mathbb{R}_+$ be the instantaneous cost function, with $f(i, a)$ denoting the single-period cost of selecting action

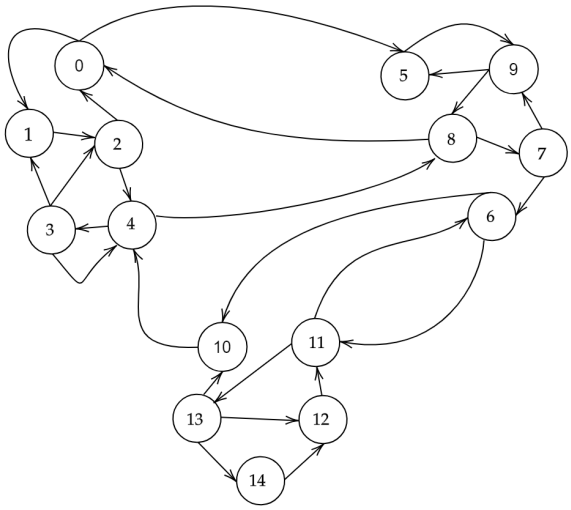

Figure 1: A Markov decision process. The arrows represent all transitions that may occur with positive probability under some control action.

$a \in A(i)$ at state $i \in S$, where $\mathbb{R}_+$ denotes the set of non-negative real numbers. Therefore, each stationary control policy $\mathcal{L} \in \mathbb{L}$ has an associated long-term average cost defined as:

$$\eta^{\mathcal{L}} = \lim_{N\to\infty} \frac{1}{N} E\left[\sum_{t=0}^{N} f(X_t, \mathcal{L}(X_t))\right]. \tag{1}$$

Following (Cao et al., 2002), we assume that the controlled chain is ergodic under any feasible policy $\mathcal{L} \in \mathbb{L}$, which makes the long-term average cost independent of the initial state.

The decision maker seeks an optimal stationary control policy $\mathcal{L}^* \in \mathbb{L}$ such that:

$$\eta^* = \eta^{\mathcal{L}^*} \leq \eta^{\mathcal{L}}, \forall \mathcal{L} \in \mathbb{L}. \tag{2}$$

Eq. (2) can be solved via the classical Relative Value Iteration algorithm (e.g., Puterman, 2005). Classical MDP theory yields that, for any policy $\mathcal{L} \in \mathbb{L}$, the solution to (1) can be obtained by finding a pair $(V^{\mathcal{L}}, \eta^{\mathcal{L}})$ that solve the Poisson equation:

$$V^{\mathcal{L}}(i) = f(i, \mathcal{L}(i)) - \eta^{\mathcal{L}} + \sum_{j\in S} p_{ij}^{\mathcal{L}(i)} V^{\mathcal{L}}(j),\ i \in S, \tag{3}$$

where $V^{\mathcal{L}} : S \to \mathbb{R}$ is a real-valued function in the space $\mathcal{V}$ of real-valued functions in S. Furthermore, Corollary 2

from (Arruda & Fragoso, 2015) yields that, for any finite stopping time τ such that $P(\tau < \infty) = 1$:

$$V^{\mathcal{L}}(i) = E\left\{\sum_{t=0}^{\tau-1}\left(f(X_t, \mathcal{L}(X_t)) - \eta^{\mathcal{L}}\right) + V^{\mathcal{L}}(X_\tau)\right\},\ i \in S. \tag{4}$$

2.1. Constrained Markov decision processes via time aggregation

When the state space of the original MDP is very large, one can define a small subset $F \subset S$, set up a fixed policy d in the remainder of the state space, that is, $d : F^c \to A$, $d(i) \in A(i)\,\forall i \in F^c$, where F^c is the complement of F ($F^c = S \setminus F$), and use time aggregation to find a suboptimal solution to the original problem which solves the following optimisation problem (Arruda & Fragoso, 2015):

$$\begin{aligned} \min \quad & \eta = \eta^{\mathcal{L}^*} \\ \text{subject to} \quad & \eta \geq \eta^{\mathcal{L}},\ \mathcal{L} \in \mathbb{L},\ \mathcal{L}_{\text{out}|_{F^c}} = d, \end{aligned} \tag{5}$$

where $\mathcal{L}_{\text{out}|_{F^c}} = \{\mathcal{L}(i), i \in F^c\}$, $\mathcal{L}_{\text{out}|_{F^c}} \in \mathbb{L}_{\text{out}}$ with $\mathbb{L}_{\text{out}}$ standing for the set of feasible stationary policies constrained to the set F^c. In the remainder of this subsection, we provide some results that will be germane to the following sections.

The following theorem is a restatement of (Arruda & Fragoso, 2015, Theorem 1) and establishes a connection between the time-aggregated and the original formulations, to be utilised in the new approach we propose in this paper.

Theorem 1. *Let $\mathcal{L}^*$ be a solution to Eq.* (2) *and let $\mathcal{L}_{out|_{F^c}}(i) = d(i) = \mathcal{L}^*(i)$, $\forall i \in F^c$ in Problem* (5)*. Then, the solutions of problems* (2) *and* (5) *are identical.*

To solve (5), we need to define a set of stopping times τ_k, $k \geq 0$ such that $\tau_0 = 0$ and $\tau_{k+1} = \{\min t > \tau_k : X_t \in F\}$. For any policy $\mathcal{L} : \mathcal{L}_{\text{out}|_{F^c}} = d$, the average cost $\eta^{\mathcal{L}}$ coincides with the average cost of the semi-Markov process (Puterman, 2005, Chapter 11) $Y_k = X_{\tau_k}$, $k \geq 0$ which evolves in the subset F and is comprised of the following elements (Cao et al., 2002; Arruda & Fragoso, 2011):

- an action space $A_y = \bigcup_{i \in F} A_y(i)$, where $A_y(i) = A(i)\,\forall i \in F$, i.e. the control actions in the subset F coincide with those of the original MDP;
- a cost function $h_f : F \times A \to \mathbb{R}_+$ that satisfies:

$$h_f(i,a) = f(i,a) + E\left[\sum_{t=1}^{\tau_1-1} f(X_k, \mathcal{L}(X_k))\,|X_0 = i\right],\ i \in F,\ a \in A(i); \tag{6}$$

- inter-transition times defined as:

$$h_1(i,a) = h_f(i,a),\ \text{for } f(j,a) \equiv 1\ \forall j \in S,\ a \in A(j); \tag{7}$$

- and embedded transition probabilities:

$$\tilde{p}_{ij}^{a} = p_{ij}^{a} + \sum_{k \in F^c} p_{ik}^{a} p_{kj}^{d}(\tau_1),\ p_{kj}^{d}(\tau_1) = P^{\mathcal{L}}(X_{\tau_1} = j | X_0 = k). \tag{8}$$

If η_d^* solves (5) and $\mathcal{L}_d^*$ is an optimal policy for problem (5), then there is a pair (V_d^*, η_d^*), where $V_d^* : F \to \mathbb{R}$ is a real-valued function in F, that satisfy:

$$V_d^*(i) = \min_{a \in A(i)} \left\{ \left(h_f(i,a) - \eta_d^* h_1(i,a) \right) + \sum_{j \in F} \tilde{p}_{ij}^{a} V_d^*(j) \right\}, i \in F \tag{9}$$

where η_d^* is also the solution to (1) under policy $\mathcal{L}_d^*$, with:

$$\mathcal{L}_d^*(i) = \begin{cases} \arg\min_{a \in A(i)} \left\{ \left(h_f(i,a) - \eta_d^* h_1(i,a) \right) + \sum_{j \in F} \tilde{p}_{ij}^{a} V_d^*(j) \right\}, & \text{if } i \in F, \\ d(i), & \text{if } i \in F^c. \end{cases} \tag{10}$$

The following auxiliary result will play a pivotal role in the proof of a key result that illustrates the power of the partitioning scheme proposed here and allows us to implement an effective distributed policy improvement.

Lemma 2. *Let η_d^* solve* (5) *and assume $\mathcal{L}_d^*$ is an optimal policy for problem* (5)*, then the value function of policy $\mathcal{L}_d^*$ in the original MDP is given by:*

$$V^{\mathcal{L}_d^*}(i) = \begin{cases} V_d^*(i), & \text{if } i \in F, \\ E\left\{ \sum_{t=0}^{\tau_1 - 1} \left(f(X_t, \mathcal{L}_d^*(X_t)) - \eta_d^* \right) + V_d^*(X_{\tau_1}) \right\}, & \text{if } i \in F^c. \end{cases} \tag{11}$$

Proof. The first part of Eq. (11), for states $i \in F$, is a direct application of (Arruda & Fragoso, 2015, Theorem 2). In contrast, the part referring to states $i \in F^c$ follows directly by applying Eq. (4) to policy $\mathcal{L}_d^*$ with stopping time τ_1, and replacing $V^{\mathcal{L}_d^*}(X_{\tau_1})$ by $V_d^*(X_{\tau_1})$. The replacement is possible because $\tau_1 = \min\{t > 0 : X_t \in F\}$, and this implies that $X_{\tau_1} \in F$.

2.1.1. Semi-regenerative trajectories

Albeit the time aggregated solution via Eq. (9) iterates solely on the subset F, the components must be evaluated via Eq. (6)-(8). These quantify total costs and transition probabilities of semi-regenerative trajectories through the subset F^c, up to the first return of X_t, $t \geq 0$ to the subset F. To evaluate these trajectories we need to define $O_{F^c}^d := [p^d(i,j)]$, $i, j \in F^c$, where p_{ij}^d is the transition probability between states $i \in F^c$ and $j \in F^c$ under policy $\mathcal{L}_{\text{out}|_{F^c}} = d$, $d \in \mathbb{L}_{\text{out}}$ in the subset F^c, and

$$E_{F^c} = \left(I - O_{F^c}^d \right)^{-1}.$$

Standard Markov chain results (e.g., Brémaud, 2020) yield that element $e_{F^c}[i,j]$ of matrix E_{F^c} is the expected number of visits a trajectory starting from state $i \in F^c$ will pay to state $j \in F^c$ before the controlled process X_t, $t \geq 0$ reaches

subset F. Standard absorption analysis also yields that (e.g., Cao et al., 2002; Arruda & Fragoso, 2015):

$$\begin{aligned} h_f(i,a) &= f(i,a) + \sum_{k\in F^c} p_{ik}^a \sum_{j\in F^c} e_{F^c}[k,j] f(j,d(j)),\ i \in F, \\ \tilde{p}_{ij}^a &= p_{ij}^a + \sum_{k\in F^c} p_{ik}^a \sum_{m\in F^c} e_{F^c}[k,m] p_{mj}^d,\ i\,,\, j \in F. \end{aligned} \tag{12}$$

It is worth noticing that Eq. (12) has summations on both F and F^c. Indeed, (Arruda & Fragoso, 2015, Remark 1) report a complexity of the order $O(|F|\cdot|A|+|F|^3+|F^c|^3)$, which is dominated by the maximum cardinality between sets F and F^c. Hence, as time aggregation was proposed for the case where $|F| << |F^c|$, limiting the computational effort on the summations within the subset F^c is a straightforward way to improve the overall efficiency of the approach. In the next session we will introduce a novel partitioning approach designed to do just that by inducing a specific communication structure between the subsets in the partition.

3. A novel multi-subset partitioning approach

In this section we set forth a state-space partitioning approach which is a centrepiece of this paper and is tailored in such a way that allows a great variety of partitioning schemes. The approach gives rise to different algorithms and sets the stage for possible advancements in previous techniques of recent vintage. A peculiar feature of this partitioning approach is that, when combined with time aggregation (Cao et al., 2002), it allows one to churn out efficient algorithms, as it will be clear latter on.

Definition 1 (The multi-cluster partition scheme). *A multi-cluster partition scheme $\mathcal{P}(S)$ is a partition of the state space S into n disjoint subsets $\{B_1, B_2, \ldots, B_n\}$, satisfying $\bigcup_{l=1}^{n} B_l = S$, $B_l \cap B_m = \emptyset$, $\forall\ l,m \in \{1,\ldots,n\} : l \neq m$. For partition scheme $\mathcal{P}(S)$, define the following sets of frontier (F_l, F) and interior (I_l, I) states:*

$$F_l \triangleq \{j \in B_l : \exists\, i \notin B_l \text{ and } \exists\, a \in A(i) \text{ such that } p_{ij}^a > 0\}, \tag{13}$$

$$F = \bigcup_{l=1}^{n} F_l, \quad I_l = B_l \setminus F_l,\ 1 \le l \le n, \quad F^c = S \setminus F = \bigcup_{l=1}^{n} I_l. \tag{14}$$

The novelty of the multi-cluster partitioning scheme in Definition 1 is that it gives rise to a topology that concentrates the information within a small subset of states that can be accessed from different subsets within the partition: the set F in Eq. (14). This, in turn, allows us to simplify, distribute and reduce the complexity of the evaluations of the trajectories between successive visits to subset F, as detailed below.

The scheme is illustrated in Figure 2 for the Markov decision process of Figure 1. For each subset B_l in the partition, the states in grey belong to the set $F_l \subset B_l$ of *frontier* states, for which Eq. (13) holds. Note that F_l comprises the states that can be directly accessed from different subsets in the partition. Conversely, the states in white in Fig. 2 belong to the set $I_l = B_l \setminus F_l$ of *interior states*, which satisfy Eq. (14) and can only be reached from the subset B_l. Observe that this guarantees that a semi-regenerative trajectory starting at a given subset I_l will never reach another interior subset I_m, $m \neq l$ before returning to subset F.

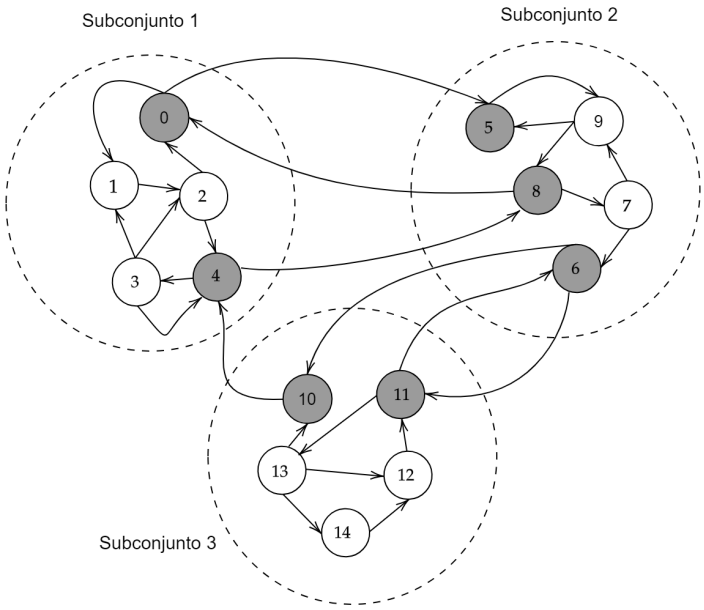

Figure 2: An example of a state space partition

To finish the set up of our approach, we apply the partitioning scheme in Definition 1, with subsets F and F^c that satisfy (14), to the time-aggregated formulation in Eq. (5). It is noteworthy that the subset F^c as defined in (14) comprises all the *interior* subsets in the partitioning scheme. This means that the policy $\mathcal{L}_{\text{out}|_{F^c}} = d$ in the time aggregated formulation establishes a fixed control action $d(i) \in A(i)$ for each *interior* state $i \in F^c$, while optimising in the set of frontier states F.

3.1. Properties of the proposed multi-cluster partitioning approach

This subsection establishes some key communication properties of the *frontier* states belonging to set F and the *interior* states in F^c. Assume a partition $\mathcal{P}(S)$ as in Definition 1. In addition, assume that the controlled process X_t, $t \geq 0$ is irreducible under any feasible policy $\mathcal{L} \in \mathbb{L}$. Then, it follows that:

i) An interior state from any given subset I_l connects directly only to states in $F \cup I_l$:

$$p_{ij}^a = 0, \ \forall i, j \in F^c, \ i \in I_l \text{ and } j \in I_m, \ m \neq l, \ l, m \in \{1, \ldots, , n\}. \tag{15}$$

ii) Any state $i \in F_l$ can access the subset F^c only via states $j \in I_l$:

$$p_{ij}^a = 0, \forall j \in F^c \setminus I_l, \text{ and } a \in A(i), \ i \in F_l, \ F_l \subset F, \ 1 \leq l \leq n. \tag{16}$$

That means that a state in the frontier set of a given subset B_l will never directly access an interior subset I_m of another subset B_m of the partition. This property will allow us to rewrite Eq. (12) only as a function of the interior subset I_l for all frontier states in F_l, as will be detailed in the next subsection.

iii) The frontier subset F is reachable from any interior subset I_l under any control policy $\mathcal{L} \in \mathbb{L}$:

$$\sum_{j \in F} p_{ij}^{\mathcal{L}(i)} > 0, \text{ for some } i \in I_l,\ I_l \subset B_l,\ 1 \leq l \leq n,\ \mathcal{L} \in \mathbb{L}. \tag{17}$$

iv) A semi-regenerative trajectory that leaves subset F via some state $j \in I_l$ does not leave subset I_l before returning to subset F:

$$\sum_{j \in F^c} p_{ij}^{\mathcal{L}(i)} = \sum_{j \in I_l} p_{ij}^{\mathcal{L}(i)},\ \forall i \in I_l,\ I_l \subset B_l,\ 1 \leq l \leq n,\ \mathcal{L} \in \mathbb{L}. \tag{18}$$

3.2. Semi-regenerative trajectories under the proposed partitioning scheme

In this subsection we combine the partitioning $\mathcal{P}(S)$ from Definition 1 and Lemma 2 to derive a result that enables a distributed computation of the value function across the different partition subsets.

We assume that $\mathcal{L}_{\text{out}|_{F^c}} = d$ is the control policy applied in subset F^c, as in Eq. (5) and let $p_{ij}^d = p_{ij}^{d(i)}$, $i \in F^c$ be the transition probability from state $i \in F^c$ to state $j \in S$. Starting from any state $i \in I_l$, standard Markov chain analysis (e.g., Brémaud, 2020) yields that the expected number of visits to state $j \in I_l$ before leaving subset I_l is the element $e_l^d[i, j]$ of matrix E_l^d, given by:

$$E_l^d = (I - O_l^d)^{-1}, \quad l = 1, \cdots, n, \tag{19}$$

where $O_l^d := [p_{ij}^d]$, $i, j \in I_l, l = 1, ..., n$. In contrast, element $a_l^d[i, j]$ of matrix A_l^d defined below denotes the probability that an outbound trajectory starting in $i \in I_l$ will reach frontier state $j \in F$ upon leaving I_l. Let $R_l^d := [p_{ij}^d]$, $i \in I_l,\ j \in F$. Standard absorption analysis yields (Brémaud, 2020):

$$A_l^d = E_l^d \cdot R_l^d, \quad l = 1, \cdots, n, \tag{20}$$

Recalling that X_t, $t \geq 0$ is the controlled trajectory and defining $\tau = \{\min t > 0 : X_t \in F\}$, it follows that

$$a_l^d[i, j] = P(X_\tau = j | X_0 = i, \mathcal{L}_{\text{out}|_{F^c}} = d).$$

Defined in Eq. (6), the cumulative cost of a trajectory starting at frontier state $i \in F_l$ under control $a \in A(i)$ until returning to subset F, can be rewritten as:

$$h_f(i, a) = f(i, a) + \sum_{k \in I_l} p_{ik}^a \sum_{j \in I_l} e_l^d[k, j] f(j, d(j)), \tag{21}$$

where the summation accounts for the total cost incurred within the subset I_l before reaching the next frontier state. The summation over set I_l suffices due to Eq. (16), which shows that a semi-regenerative trajectory started in F_l can only access interior states in I_l. Similarly, the length of the trajectory from the frontier state $i \in F_l$ to the next frontier state is given by:

$$h_1(i,a) = 1 + \sum_{k \in I_l} p_{ik}^a \sum_{j \in I_l} e_l^d[k,j], \tag{22}$$

which is simply an application of (21) with $f(i,a) = 1, \forall (i,a) \in S \times A$. Finally, the transition probability of the embedded semi-Markov decision process within the subset of frontier states F is:

$$\tilde{p}_{ij}^a = p_{ij}^a + \sum_{k \in I_l} p_{ik}^a a_l^d[k,j],\ i \in F_l,\ j \in F. \tag{23}$$

Under the proposed partitioning scheme, we can rewrite the results of Lemma 2 as follows.

Lemma 3. *Assume partition $\mathcal{P}(S)$ as in Definition 1. Let also (V_d^*, η_d^*) be the solution to* (9) *for a time aggregated formulation* (5) *with $\mathcal{L}_{out|_{F^c}} = d$ under the proposed partitioning scheme. In addition, let policy $\mathcal{L}_d^*$ be an optimal policy satisfying* (10)*. Then, the value function of the original MDP under policy $\mathcal{L}_d^*$ satisfies:*

$$V^{\mathcal{L}_d^*}(i) = \begin{cases} V_d^*(i), & \text{if } i \in F, \\ \sum\limits_{k \in I_l} e_l^d[i,k] \left(f(k, d(k)) - \eta_d^* \right) + \sum\limits_{j \in F} a_l^d[i,j] V_d^*(j), & \text{if } i \in F^c. \end{cases} \tag{24}$$

Proof. The first expression in the right-hand side of Eq. (24) is a direct application of Lemma 2. Let us assume $i \in I_l$, then according to Lemma 2

$$V^{\mathcal{L}_d^*}(i) = \sum_{t=0}^{\tau_1 - 1} \left(f(X_t, \mathcal{L}_d^*(X_t)) - \eta_d^* \right) + V_d^*(X_{\tau_1}),\ \tau_1 = \min\{t > 0 : X_t \in F\}, \tag{25}$$

as $i \in I_l$ implies that $i \in F^c$. Now, if $p_{ij}^{d(i)} > 0$ for some $j \notin I_l$, then by definition $j \in F_m$ for some $m \in \{1, \ldots, n\}$ and therefore $j \in F$. Consequently, it also follows that $\tau_1 = \min\{t > 0 : X_t \notin I_l\}$ as any state within I_l can only immediately access either states in I_l or states in F. Hence, standard Markov chain results yield that:

$$E\left[\sum_{t=0}^{\tau_1 - 1} \left(f(X_t, \mathcal{L}_d^*(X_t)) - \eta_d^* \right) \right] = \sum_{j \in I_l} e_l^d[i,j] \left(f(j, d(j)) - \eta_d^* \right),$$

where $E_l^d = e_l^d[i,j]$, $k, j \in I_l$ is defined in (19), as this is the cumulative cost at subset I_l before leaving the subset under cost function ($f(i, d(i) - \eta_d^*$) for each state $i \in I_l$. Standard absorption analysis yields that

$$E\left(V(X_{\tau_1})\right) = \sum_{j \in F} a_l^d[i,j] V_d^*(j),$$

where $A_l^d = a_d^d[i,j]$, $i \in I_l$, $j \in F$ is defined in (20) and defines the probability that a trajectory states at $i \in I_l$ will reach state $j \in F$ immediately upon leaving the interior subset I_l.

To conclude the proof, it suffices to substitute the latter couple of expressions in Eq. (25), which leads to the value function expression for $i \in I_l$ in (24).

The result in Lemma 3 illustrates the power of the proposed partitioning scheme, as it shows that to determine the value function of the states within any subset I_l we only need to store the value function within the subset F, which

can be obtained from the embedded semi-Markov formulation in F - Eq. (5).

Remark 1. *Observe that in* (21)-(23)*, the inversion of the* $|F^c|-$*dimensional matrix in Eq.* (12) *is substituted by a distributed computation scheme that applies* (19) *to invert an* $|F_l|-$*dimensional matrix for each subset* $F_l, 1 \leq l \leq n$*. Thus, the computational effort of* $O(|F^c|^3)$ *in the original subset is substituted by an effort of* $O(\sum_{l=1}^{n} |I_l|^3)$*, recalling that* $|F^c| = \sum_{l=1}^{n} |I_l|$*. This has the potential to significantly reduce the effort in the case when* $|I_l| << |F^c| \approx |S|$*.*

Remark 2. *The idea of dividing the states into* frontier *and* interior *states makes it possible to solve the system via absorption analysis (*(21)-(23)*), as a function of the sojourn time within each individual subset* I_l*. This is due to the underlying communication properties induced by the design of the partitioning scheme, summarised in Eq.* (15) *to* (18)*.*

Whilst partitioning scheme $\mathcal{P}(S)$ in Definition 1 induces favourable topological properties due to the carefully tailored concepts of frontier and interior states, see Remarks 1 and 2, the design of the subsets B_1 to B_n, as well as the total number of subsets in the partition, still bear some consideration in order to drill down further the possibility of its application in high-dimensional problems. The next subsection addresses these issues by exploiting Foster's sufficient conditions for stochastic stability (Foster, 1953) to derive a subset configuration for $\mathcal{P}(S)$ based on the underlying Foster-Lyapunov function, that allows us to address large scale problems.

3.3. On the number and configuration of the partition's subsets

In this section, we will introduce a partitioning design that utilises Foster's sufficient conditions for positive recurrence, see for example (Foster, 1953) and (Brémaud, 2020, Theorem 7.1.1, page 227). The rationale is to set a partitioning scheme that makes use of the long-term distribution of the controlled Markov chain which is induced by Foster's stability conditions. The detailed technical results that underpin the proposed scheme can be found in Appendix A.

Our results rely on Assumption 1 (Assumption (A.1) in Appendix A), which suffices to ensure the controlled chain is ergodic under all feasible policies (Brémaud, 2020; Foster, 1953), even if the state space's cardinality is infinite.

Assumption 1 (Foster's condition). *For any feasible policy* $\mathcal{L} \in \mathbb{L}$*, we assume that the Markov chain is irreducible and that there exists a function* $g : S \to \mathbb{Z}_+$*, where* $\mathbb{Z}_+$ *is the set of non-negative integers, such that:*

$$\begin{aligned} &\sum_{j \in S} p_{ij}^{\mathcal{L}} g(j) < \infty, i \in S_1 \\ &\sum_{j \in S} p_{ij}^{\mathcal{L}} g(j) \leq g(i) - \epsilon, i \notin S_1, \\ &\max_{i \in S_1} g(i) < \min_{j \notin S_1} g(j), \end{aligned} \tag{26}$$

for some finite set $S_1 \subset S$ *and scalar* $\epsilon > 0$*, where* $g(\cdot)$ *is usually known as the Foster-Lyapunov function and* (26) *is Foster's condition.*

Theorem 7 of Appendix A proves that, under Assumption 1, the steady state probability that $g(X_t)$ exceeds a given threshold $\bar{\sigma}$ decreases exponentially in $\bar{\sigma}$ outside of region S_1. In what follows, we propose a partitioning scheme

with subsets of increasing cardinality as a function of $g(\cdot)$, which can be applied to problems with very large state spaces, as follows:

Definition 2 (Foster-Lypaunov based partition). *Let $\mathcal{P}(S)$ be a partitioning scheme that satisfies Definition 1. We say that $\mathcal{P}(S)$ is a Foster-Lypaunov based partition if:*

$$n = \left\lceil \ln\left(\max_{j\in S} g(j)\right)\right\rceil, \quad B_1 = \{j \in S : Cg(j) < e^1\},$$

and

$$B_l = \{j \in S \text{ such that } j \notin \bigcup_{m=1}^{l-1} B_m \text{ and } Cg(j) < e^l\},\ l = 2, \ldots, n,$$

where $0 < C < \infty$ is an arbitrary constant.

To link the conditions in Assumption 1 with partition scheme $\mathcal{P}(S)$ in Definition 2, let $S_1 \subseteq \bigcup_{l=1}^{\bar{m}} B_l$, where $\bar{m} < n$. One can expect the cardinality of B_l to increase in l, which is counterbalanced by the exponential decrease in the steady state probability that $g(\cdot)$ falls out of the subset $\bigcup_{m=1}^{l} B_m$, see Theorem 7 of Appendix A. The second expression in Eq. (26) implies that states outside S_1 will tend to have an access to a limited vicinity in terms of the value of the Lyapunov function. This implies that the frontier states will tend to be close to the boundary of subset B_l, therefore for subsets B_l of large cardinality, one can expect the cardinality of $F_l \subset B_l$ to be much smaller than that of B_l, thereby rendering viable the solution of the embedded formulation in Step 6 of Algorithm 1.

Furthermore, in view of Theorem 7, we know that the importance of a subset B_l to the long-term average decreases exponentially in l. Hence, as the cardinality of the interior set I_l increases, one can apply an approximate policy with limited effect on the long-term performance.

Having established a partitioning scheme $\mathcal{P}(S)$ with special topological properties derived by design, and defined the number and configuration of the partition subsets based on Foster-Lyapunov stochastic stability, we are now ready to introduce an algorithm which generalises the two-phase time aggregation approach by allowing a distributed computation of the policy evaluation step, one subset at a time.

Furthermore, since the Foster-Lyapunov approach ensures that subsets B_l with larger cardinality are seldom visited, one can approximate the policy evaluation within these subsets with limited effect on the overall performance. This will be further explored in the numerical results of Sections 6 and 7. The algorithm, which will be proposed in the next section, is general. Therefore, it allows for variations in the manner in which the two phases, namely time aggregated formulation and policy evaluation for interior states, are evaluated. Such variations will give rise to algorithm variations that will be explored to improve computational efficient. These will be introduced in the next section and implemented in our case studies in Sections 5, 6 and 7.

4. Distributed time aggregation algorithms for Markov decision processes

As previously stated, we will now utilise the properties of the *the partitioning scheme* $\mathcal{P}(S)$ in Definition 1 to frame the essential basis to propose an algorithm that will utilise the distributed computation scheme in Eq. (21)-(23)

to solve the time-aggregated formulation. It will rely on Lemma 3 to implement a distributed policy improvement step that iterates at a single interior subset I_l, $1 \leq l \leq n$ at a time. Each iteration solves Eq. (9) for a distinct formulation in Eq. (5). The procedure is presented in Algorithm 1 below.

Algorithm 1 Distributed Two-phase Time Aggregation (DTPTA)

1: Partition the state space in n disjoint subsets $\{B_1, \cdots, B_n\}$
2: For each $l \in \{1, 2, \ldots, n\}$, set $F_l = \{j \in B_l : \exists\, i \notin B_l \text{ and } \exists\, a \in A(i) \text{ such that } p_{ij}^a > 0\}$, and make $I_l = B_l \setminus F_l$.
3: Set an arbitrary initial policy $\mathcal{L}_{\text{out}} = d_0$, $d_0 \in \mathbb{L}_{\text{out}}$ for the subset F^c and a tolerance ϵ
4: $k \leftarrow 0$, $er \leftarrow \infty$, $\eta_0 \leftarrow \infty$, $\bar{d}_0 = d_0$, $\eta_k^l = \infty$, $1 \leq l \leq n$, $\text{Best}_\eta \leftarrow \infty$
5: **while** $|er| > \epsilon$ **do**
6: $d_{k+1} = \bar{d}_k$, $\eta_{k+1} \leftarrow \text{Best}_\eta$, $k \leftarrow k + 1$
7: Sweep = $\emptyset$
8: **while** Sweep $\neq \{1, \ldots, n\}$ **do**
9: Select l from the set $\{1, \ldots, n\}$ and make Sweep = Sweep $\cup\, l$
10: Apply Eq. (9) to find a pair (V_d^*, η_d^*) that solves the time aggregated formulation (5), with $d = \bar{d}_k$. Make $\eta_k^l = \eta_d^*$ and $V_k^l(i) = V_d^*(i)$, $\forall i \in F$
11: $\text{Best}_\eta = \min\{\text{Best}_\eta, \eta_k^l\}$
12: Apply Lemma 3 to find the value function $V_k^l(i) = V^{\mathcal{L}_{\bar{d}_k}}(i)$, $i \in I_l$ of the original MDP
13: Apply a policy improvement step by making:

$$\bar{d}_k(i) = \arg\min_{a \in A(i)} \left\{ f(i,a) + \sum_{j \in I_l \cup F} p_{ij}^a V_k^l(i) \right\}, \forall i \in I_l, \tag{27}$$

and keeping $\bar{d}_k(i) = d_k(i)$ whenever possible in (27)
14: Use the new improved policy in I_l to generate a new time aggregated formulation with $\mathcal{L}_{\text{out}|_{F^c}} = \bar{d}_k$. This means updating (21)-(23) to all state-action pairs (i,a), $i \in F_l$, $a \in A(i)$ and keeping $h_f(i,a)$, $h_1(i,a)$, and $\tilde{p}_{ij}^a$ unaltered for all remaining state-action pairs, i.e., for all pairs $(i,a) : i \notin F_l$.
15: **end while**
16: $er \leftarrow \eta_k^n - \eta_k$
17: **end while**

Note that Step 8 of Algorithm 1 makes sure that at least one policy improvement step is performed for the interior states in each subset of the partition, while also allowing a flexible exploration of the subsets. One could, for example, perform more frequent local searches at subsets I_l that are visited more frequently with the incumbent policy, as these would have potentially more impact over the long-term cost. Or one could apply other heuristic strategies used in reinforcement learning, such as prioritised sweeping (Moore & Atkeson, 1993).

In the following subsection, we will firstly consider variations of Algorithm 1 that consider different strategies for updating the subsets in Step 8, or possible approximations for the policy evaluation within large cardinality subsets B_l which are seldom visited and therefore contribute little to the overall performance. We will then prove the convergence of Algorithm 1.

4.1. Variations of the two-phase time aggregation algorithm

Note that Step 10 of Algorithm 1 requires the solution of a time aggregated problem. To find this solution, one can employ different dynamic programming alternatives, such as value iteration (VI) or policy iteration (PI). In this paper, we utilise value iteration up to convergence for all variants of Algorithm 1.

The flexible design of the search strategy in Step 8 of Algorithm 1 can give rise to multiple implementations. We implemented two versions in our experiments: the *distributed two phase time aggregation (DTPTA)* includes one subset at a time in increasing order of the label, hence B_1 is firstly included, then B_2, and so on, until we reach B_n. A second variant called *multi cluster two phase time aggregation (MCTPTA)* searches all the subsets at the same time,

in a single step. While it can be argued that this is an application of the original two phase time aggregation algorithm (Arruda & Fragoso, 2015), it is worth pointing out that it is the partitioning scheme from Definition 1 that will give rise to the set F of frontier states.

Finally, Step 13 of Algorithm 1 can be approximated - for example via Reinforcement Learning, Approximate Dynamic Programming or Monte Carlo simulation of an ad-hoc policy for large l, or even truncation of the state space. For the sake of simplicity, we utilised the latter approach to speed up convergence to a near-optimal policy. For both DTPTA and MCTPTA, we solved increasingly accurate approximate problems by firstly truncating the state space to B_1 and B_2 and then progressively including the remaining subsets in the partition, one at a time.

Finally, for large scale problems that become intractable in our computational setting, we truncated the state space to keep the problem computationally tractable and utilised an ad-hoc policy for the rarely visited states outside the truncation. This strategy is explored in Sections 7 for large scale and difficult to solve queuing problems. As expected, the experiments show the limited effect of the subsets B_l with larger cardinality to the overall performance and validate the stochastic stability results in Appendix A.

4.2. Convergence proof of Algorithm 1

We will now proceed to prove the convergence of Algorithm 1 to the solution of the original problem - Eq. (2). We begin by proving some auxiliary results.

Lemma 4. *Let η_k^l, $1 \leq l \leq n$ be the value obtained in Step 10 of Algorithm 1. Then*

$$\eta_k^l \leq Best_\eta,\ 1 \leq l < n.$$

Furthermore, if $\eta_k^l < Best_\eta$ then $\bar{d}_k(i) \neq d_k(i)$, for some $i \in I_l$.

Proof. The proof follows directly from (Cao, 1998, Lemma 2), which implies that if $\bar{d}_k(i) \neq d_k(i)$ for some $i \in I_l$ in the policy improvement step - Eq. (27), then $\eta_k^l < \text{Best}_\eta$, as $\eta_k^l = \eta_d^*$ in (5) with $d = \bar{d}_k$. On the other hand, if $\bar{d}_k(i) = d_k(i)$, $\forall i \in I_l$, then it is clear that $\eta_k^l = \text{Best}_\eta$ as in both cases we solve the same time aggregated problem.

Lemma 5. *Let η_k be the value obtained in Step 6 of Algorithm 1. Then it follows that:*

$$\eta_{k+1} \leq \eta_k,\ k \geq 0. \tag{28}$$

Proof. It follows directly from Lemma 4 that:

$$\eta_{k+1} = \text{Best}_\eta = \min_{1 \leq l \leq n} \eta_k^l,$$

when Sweep = $\{1, \ldots, n\}$. To conclude the proof, it suffices to notice that Step 6 of Algorithm 1 makes $d_k = \bar{d}_{k-1}$ at iteration k for the first subset selected in Step 9, where $\bar{d}_{k-1}$ is the policy in F^c following the last policy improvement step of iteration $k-1$; i.e., when Sweep = $\{1, \ldots, n\}$ in the *while* loop (Step 8 of the algorithm). Therefore, one can once again apply (Cao, 1998, Lemma 2) to verify that:

$$\eta_k \leq \text{Best}_\eta,$$

and that concludes the proof.

Theorem 6. *Algorithm 1 terminates at an optimal policy for problem* (2).

Proof. Lemma 5 establishes that Algorithm 1 monotonically decreases the long-term average cost. This implies that the algorithm converges in finite time, given that the state and action spaces are finite. Therefore there is a finite number of policies to choose from and the average cost cannot be improved forever.

To prove that the algorithm converges to the optimal policy, it suffices to verify that Step 8 necessarily sweeps every subset I_l, $l = \{1, \ldots, n\}$ and hence upon convergence at least one policy improvement step will have been attempted for all states $i \in F^c$, without changing the incumbent solution. Now, let d_k be the policy within subset F^c upon convergence. Suppose, for the sake of argument that $d_k \neq \mathcal{L}^*_{\text{out}|F^c}$, where

$$\mathcal{L}^*_{\text{out}|F^c}(i) = \mathcal{L}^*(i), \ \forall i \in F^c$$

for some policy $\mathcal{L}^*$ that solves (2). Then, it follows that $d_{k+1}(i) = \bar{d}_k(i) \neq d_k(i)$ for at least one $i \in F^c$ in (27), which contradicts the initial hypothesis, once convergence implies $d_{k+1} = d_k$. Hence, at convergence we must have $d_k = \mathcal{L}^*_{\text{out}|_{F^c}}$. As a result, Theorem 1 implies that η_k must be equal to the solution of (2), and that concludes the proof.

In what follows we apply the Foster-Lyapunov based partition in two applications, in order to give a glimpse at the use of this concept.

5. A production and inventory problem

To illustrate the proposed algorithm, this section replicates the production and inventory management example from (Arruda & Fragoso, 2015). The problem consists of 3 products and a single machine, which can produce only one product at a time and can alternate between products without any significant setup. Whenever a new demand arrives or the production of a new product is finished, the decision maker decides whether to continue the production of the same product, to change the production to another product or, alternatively, to halt production. If production is halted, the decision maker decides whether to keep it halted or to select one of the products to start production.

Let the stock level of each product be an integer variable in the range $\{-100, \ldots, 25\}$. That means that the maximal allowed backlog for each product is of 100 units, with no demand being accepted when this level of backlog is reached. On the other hand, the maximal available stock allowed for each product is 25 units. Hence the state space S is comprised of $126^3 \approx 2 \cdot 10^6$ elements, each corresponding to a possible stock/deficit combination of the three products.

We assume that the demands of products 1, 2 and 3 arrive according to Poisson processes with rates 3, 2 and 1, respectively. On the other hand, the production time follows an exponential distribution with rate 8. We also assume that each demand order is comprised of a single item and that the production facility can continue production even if there is no customer waiting. The stock/deficit cost is given by:

$$f(x) = |x_1| + 2|x_2| + 3|x_3|,$$

where x_1, x_2 and x_3 are the stock/deficit of products 1, 2 and 3, respectively. Since demands and production batches always comprise a single item, and the state space is finite, it is not difficult to see that the chain is irreducible and ergodic under all all feasible policies $\mathcal{L} \in \mathbb{L}$. The objective is to find the policy $\mathcal{L}^*$ which minimises the average cost

and satisfies (2). To run all experiments, we used a Laptop computer with Intel Core i3 2.30 GHz and 4 GB RAM, running on Windows 10. We used the C++ programming language (Prata, 2001) and a tolerance $\epsilon = 0.001$ in Step 5 of Algorithm 1.

5.1. Partitioning Scheme

To solve the example, we applied the partitioning scheme proposed in Section 3.3. Let $\bar{k} = \arg\min_{k\in\{1,2,3\}} (x_k)$, and let the Foster-Lyapunov function $g : S \to \mathbb{Z}_+$ be defined as:

$$g(x) = \begin{cases} |x_{\bar{k}} + x_{\hat{k}}|, & \text{if } |x_{\bar{k}}| = \max_{k\in\{1,2,3\}} |x_k|, \text{ and } |x_{\bar{k}} - x_{\hat{k}}| = 1 \text{ for some } \hat{k} \in \{1,2,3\} \setminus \{\bar{k}\}, \\ 2 \max_{k\in\{1,2,3\}} |x_k|, & \text{otherwise.} \end{cases} \tag{29}$$

We set up the partition scheme by making $C = \dfrac{1}{2}$ in Definition 2. Therefore, we have:

$$n = \left\lceil \ln\left(\frac{1}{2} \max_{j\in S} g(j)\right) \right\rceil = \lceil \ln 100 \rceil = 5.$$

Hence, the state space is divided in 5 subsets as follows:

$$\begin{aligned} B_1 &= \{j \in S : \frac{g(j)}{2} < e^1\} = \{j \in S : \frac{g(j)}{2} < 2,72\}; \\ B_2 &= \{j \in S : e^1 \leq \frac{g(j)}{2} < e^2\} = \{j \in S : 2,72 \leq \frac{g(j)}{2} < 7,39\}; \\ B_3 &= \{j \in S : e^2 \leq \frac{g(j)}{2} < e^3\} = \{j \in S : 7,39 \leq \frac{g(j)}{2} < 20,09\}; \\ B_4 &= \{j \in S : e^3 \leq \frac{g(j)}{2} < e^4\} = \{j \in S : 20,09 \leq \frac{g(j)}{2} < 54,60\}; \\ B_5 &= \{j \in S : e^4 \leq \frac{g(j)}{2} < e^5\} = \{j \in S : 54,60 \leq \frac{g(j)}{2} < 148,41\}. \end{aligned} \tag{30}$$

Finally, we assume that only actions $a \in A(i)$ for which $\sum_{j\in S} p_{ij}^a g(j) < g(i)$ are deemed feasible for all states $i \notin \bigcup_{l=1}^{3} B_l$. This implies that the problem satisfies Assumption 1 with $S_1 = \bigcup_{l=1}^{3} B_l$. Note that, since the production rate $\mu = 8$ is larger than the sum of the demand rates ($\lambda_1 + \lambda_2 + \lambda_3 = 6$), Eq. (26) can be met in subset $B_4 \cup B_5$ by producing (one of) the product(s) with the largest deficit if $x_{\bar{k}} < 0$ in (29), or not producing (any of) the product(s) with the largest inventory otherwise.

5.2. Results

To establish useful benchmarks for our problem, we solved the example to optimality using classical Policy Iteration (PI), Relative Value Iteration (VI) (Puterman, 2005) and Two Phase Time Aggregation (TPTA) (Arruda & Fragoso, 2015). For the original TPTA, we followed Arruda & Fragoso (2015) and selected the region F as the set of states whose stock/deficit levels are in the interval $[-10, 9]$, i.e, $F := \{x \in S : -10 \leq x_1 \leq 9, -10 \leq x_2 \leq 9, -10 \leq$

$x_3 \leq 9\}$. It is worth highlighting that the original TPTA defines the subset F arbitrarily, based on the problem domain, with no use of the concepts of frontier and interior states that we introduced in Definition 1.

The convergence times and number of iterations of all the benchmark algorithms are depicted in Table 1. Table 1 also shows the convergence times and number of iterations of the two variations of Algorithm 1 discussed in Section 4.1: distributed two phase time aggregation (DPTA), and multi cluster two phase time aggregation (MCTPTA). Recall that DPTA includes a single subset B_l, $l \in \{1, \ldots, n\}$ at a time in Step 8 of Algorithm 1, in increasing order of l. On the other hand, MCTPTA includes all subsets simultaneously.

Table 1: Computational Results

Algorithm	**Avg. Cost**	**Time**	**It.**
PI	3.4095	19,993 s	14
VI	3.4095	5,972 s	1,290
TPTA	3.4096	8,199 s	5
DTPTA	3.4095	8,548 s	4
MCTPTA	3.4095	4,573 s	4

Table 1 shows that MCTPTA was the fastest algorithm, followed by VI, TPTA, DTPTA and PI, respectively. It is noteworthy that DTPTA's convergence time ($8,548s$) was similar to that of TPTA ($8,199s$), albeit slightly larger. In contrast, MCTPTA converged much faster ($4,573s$). A possible explanation is that DTPTA performs more evaluations of the embedded chain in Step 10 of Algorithm 1, as it has to be evaluated every time a new subset l is updated in Step 8. In comparison, TPTA and MCTPTA will evaluate fewer embedded chains as they apply a single policy improvement step for all subsets.

Note, however, that MCTPTA was considerably more efficient than TPTA. This illustrates the benefits of utilising the concept of frontier states in subset F, which stems from our multi-subset partition, as frontier states have topological properties induced by design.

5.2.1. Evolution to the vicinity of the optimal solution

Let us now analyse how the proposed approach converges to the vicinity of the optimal solution in comparison to PI and TPTA. Table 2 depicts the average cost evolution at each iteration of the implemented algorithms.

One can notice in Table 2 that policy iteration converges more slowly. It takes 14 iterations and nearly $20,000s$ to converge and $18,604s$ to reach the vicinity of the optimal solution. DTPTA performs better and reaches the optimal solution vicinity in about $4,556s$, a considerable improvement over PI, which illustrates the efficiency of the distributed computation within the proposed approach, that combines a local search in subset F with distributed policy improvement steps within the subsets in F^c. However, TPTA performs even better than DTPTA and reaches the vicinity of the optimal solution in two iterations and $3,969s$, possibly due to the local search performed at the states in subset F and also due to the fewer embedded chains required to be solved in comparison to the DTPTA. Nonetheless, DTPTA reaches a good quality solution sooner (average of 3.78 in $1,389s$). In contrast, MCTPTA takes $2,556s$ to reach the vicinity of the optimal solution. This result highlights the benefits provided by utilising the frontier states of

Table 2: Average cost evolution

	Time in seconds (Average Cost)			
It.	**PI**	**TPTA**	**DTPTA**	**MCTPTA**
1	2,577 (60.8539)	2,332 (3.4887)	69 (40.8626)	53 (40.8626)
2	3,924 (52.0270)	3,969 (3.4138)	1,389 (3.78961)	773 (3.7896)
3	5,871 (41.5221)	5,465 (3.4126)	4,556 (3.4096)	2,556 (3.4096)
4	7,055 (35.9269)	6,906 (3.4096)	8,548 (3,4095)	4,573 (3.4095)
5	8,811 (26.5572)	8,198 (3.4096)	-	-
6	10,110 (22.7465)	-	-	-
7	11,664 (13.3264)	-	-	-
8	12,996 (11,6875)	-	-	-
9	14,563 (6.3609)	-	-	-
10	15,825 (5.8927)	-	-	-
11	17,302 (3.5493)	-	-	-
12	18,604 (3.4173)	-	-	-
13	19,723 (3.4095)	-	-	-
14	19,993 (3.4095)	-	-	-

the proposed partitioning scheme (Definition 1) in subset F, whose topological properties are enforced by design.

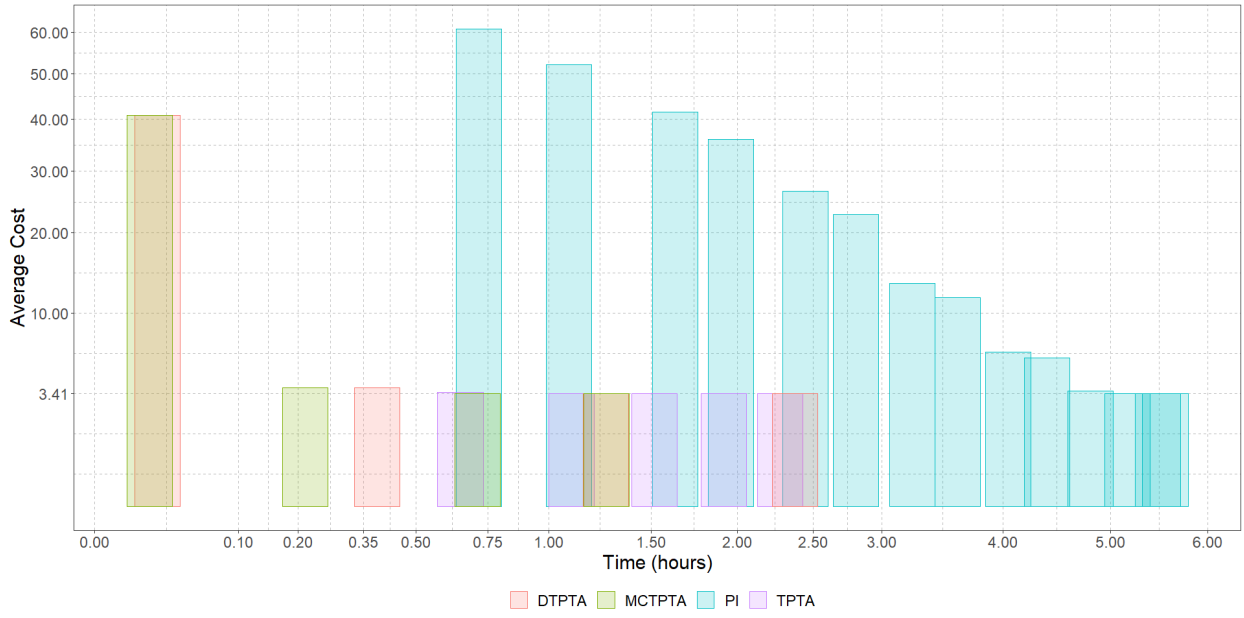

Figure 3: Cost and time evolution for the inventory example

Figure 3 introduces a graphical representation of the average cost evolution for each algorithm in time. It highlights

the rapid approach to the optimal solution by the proposed algorithms, DTPTA and MCTPTA. Note that both DTPTA and MCTPTA provide a good quality solutions faster than any of the benchmark algorithms, an evidence of the added benefits stemming from the topological properties of the proposed partitioning scheme (Definition 1), in combination with the exploitation of the Foster-Lyapunov stochastic stability conditions (Definition 2).

5.2.2. On the nature of frontier states

This subsection investigates how the frontier states are spread over the state space, to provide an insight into the topology generated by the proposed partitioning scheme. To carry out the analysis and facilitate visualisaton, we considered the same inventory problem described above but constrained to products 1 and 2. The partitioning is illustrated in Figure 4.

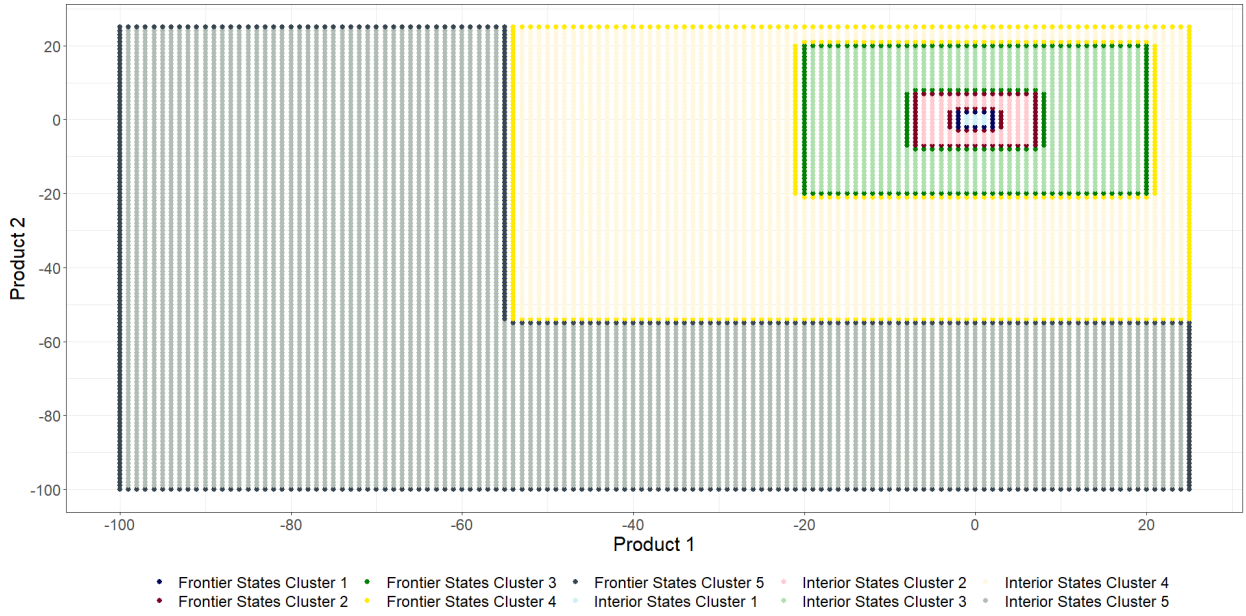

Figure 4: An example of a state space partition

It is clear that the frontier states connect the subsets. It is also apparent that the interior states are confined by the frontier states in each subset. These topological properties are ensured by design and allow us to perform policy evaluation in a distributed manner, one subset at a time, as the information from the interior states of one subset does not flow directly to the interior states in any other subset.

6. A queue management problem

This section features a new set of examples similar to the queuing management problems in (Veatch, 2001). In this problem, customers arrive according to a Poisson process with rate 1 and they are immediately routed to one of

the three servers, each with its own queue. Each time a new customer arrives, the decision maker decides to which server this customer will be routed.

Let the queue size of each server be an integer variable in the range $\{0, \ldots, 150\}$. That means that the maximum number of customers allowed in each queue is 150 and no additional customers are accepted once this level is reached. Therefore, the state space S has $151^3 \approx 3 \cdot 10^6$ elements, each corresponding to a possible queue size combination across the three servers. We assume that the service times of servers 1, 2 and 3 follow exponential distributions with rate 0.5 for servers 1 and 2, and rate 1.5 for server 3. We also assume that the cost of each state is given by:

$$f(x) = x_1 + 2x_2 + 4x_3,$$

where x_1, x_2 and x_3 are the queue sizes of servers 1, 2 and 3, respectively. Since demands and service batches always comprise a single item, and the state space is finite, it is not difficult to see that the chain is irreducible and ergodic under all all feasible policies $\mathcal{L} \in \mathbb{L}$. The objective is to find the policy $\mathcal{L}^*$ which minimises the average cost and satisfies (2). To run all experiments, we used a Laptop computer with Intel Core i3 processor with 2.30 GHz and 4 GB RAM, running on Windows 10. We used the C++ programming language (Prata, 2001) and a tolerance $\epsilon = 0.001$ in Step 5 of Algorithm 1.

6.1. Partitioning Scheme

To solve the queue management problem, we used again the partitioning scheme proposed in Section 3.3. Let $\bar{k} = \arg\max_{k\in\{1,2,3\}} |x_k|$ and let the Foster-Lyapunov function $g : S \to \mathbb{Z}_+$ be defined as:

$$g(x) = \begin{cases} |x_{\bar{k}} + x_{\hat{k}}|, & \text{if } |x_{\bar{k}} - x_{\hat{k}}| = 1 \text{ for some } \hat{k} \in \{1,2,3\} \setminus \{\bar{k}\}, \\ 2 \max\limits_{k\in\{1,2,3\}} |x_k|, & \text{otherwise.} \end{cases} \tag{31}$$

We set up the partition scheme by making $C = \dfrac{1}{2}$ in Definition 2. Therefore, we have:

$$n = \left\lceil \ln\left(\max_{j\in S} \frac{g(j)}{2}\right) \right\rceil = \lceil \ln 150 \rceil = 6.$$

Hence, the state space is divided in 6 subsets as follows:

$$
\begin{aligned}
B_1 &= \{j \in S : \frac{g(j)}{2} < e^1\} = \{j \in S : \frac{g(j)}{2} < 2,72\};\\
B_2 &= \{j \in S : e^1 \leq \frac{g(j)}{2} < e^2\} = \{j \in S : 2,72 \leq \frac{g(j)}{2} < 7,39\};\\
B_3 &= \{j \in S : e^2 \leq \frac{g(j)}{2} < e^3\} = \{j \in S : 7,39 \leq \frac{g(j)}{2} < 20,09\};\\
B_4 &= \{j \in S : e^3 \leq \frac{g(j)}{2} < e^4\} = \{j \in S : 20,09 \leq \frac{g(j)}{2} < 54,60\};\\
B_5 &= \{j \in S : e^4 \leq \frac{g(j)}{2} < e^5\} = \{j \in S : 54,60 \leq \frac{g(j)}{2} < 148,41\};\\
B_6 &= \{j \in S : e^5 \leq \frac{g(j)}{2} < e^6\} = \{j \in S : 148,41 \leq \frac{g(j)}{2} < 403,43\}.
\end{aligned}
\tag{32}
$$

Finally, we assume that only actions $a \in A(i)$ for which $\sum_{j\in S} p_{ij}^a g(j) < g(i)$ are deemed feasible for all states $i \notin \bigcup_{l=1}^{3} B_l$. This implies that the problem satisfies Assumption 1 with $S_1 = \bigcup_{l=1}^{3} B_l$. Since for all examples in this section, we have that the demand rate ($\lambda = 1$) is less than the sum of the service rates, Eq. (26) can be met in $B_4 \cup B_5 \cup B_6$ for example by routing a potential new arrival to (one of) the queue(s) with the lowest number of clients.

6.2. Results

For the sake of comparison, we solved the example to optimality using Policy Iteration (PI), Relative Value Iteration (VI) and Two Phase Time Aggregation (TPTA) (Arruda & Fragoso, 2015). For the Two-phase Time Aggregation algorithm, we selected the region F as the set of states whose queue sizes are in interval $[0, 9]$, i.e, $F := \{x \in S : 0 \leq x_1 \leq 9, 0 \leq x_2 \leq 9, 0 \leq x_3 \leq 9\}$.

Table 3 displays the convergence times of the proposed algorithms (DTPTA and MCTPTA), as well as those of the benchmark algorithms. Note that VI is the fastest algorithm, followed by MCTPTA, TPTA, DTPTA and PI, respectively. Whilst the good performance of VI can be attributed to the low occupation rate of the servers, as the overall production rate for all servers is $2,5$ and the arrival rate is 1, we will show in Section 6.2.1 that our algorithms approach the optimal solution considerably faster. Once again, MCTPTA converged faster than both traditional TPTA and DTPTA.

Table 3: Computational Results

Algorithm	**Average Cost**	**Time**	**It.**
PI	3.0002	6,109 s	4
VI	3.0002	2,038 s	866
TPTA	3.0002	3,406 s	2
DTPTA	3.0002	5,354 s	5
MCTPTA	3.0002	2,570 s	5

6.2.1. Evolution to the vicinity of the optimal solution

Let us now analyse how the proposed approach converges to the optimal solution. Table 4 depicts the average cost evolution at each iteration of the implemented algorithms.

Table 4 shows that policy iteration converges more slowly. It takes 4 iterations and nearly $6,000s$ to converge and $5,000s$ to reach the vicinity of the optimal solution. TPTA performs better and reaches the optimal solution

Table 4: Average cost evolution

	Time in seconds (Average Cost)			
It.	**PI**	**TPTA**	**DTPTA**	**MCTPTA**
1	1,613 (3.3406)	1,702 (3.0002)	8 (3.0149)	7 (3.0149)
2	3,420 (3.0319)	3,403 (3.0002)	33 (3.0002)	22 (3.0002)
3	5,110 (3.0002)	-	106 (3.0002)	69 (3.0002)
4	6,109 (3.0002)	-	2,537 (3.0002)	1,359 (3.0002)
5	-	-	5,354 (3.0002)	2,570 (3.0002)

vicinity in about $1,700s$. However, DTPTA performs even better than TPTA and reaches the vicinity of the optimal solution in two iterations and $33s$, a considerable improvement over PI and TPTA, which illustrates the efficiency of the distributed computation within the proposed approach, that combines a local search in subset F with distributed policy improvement steps within the subsets in F^c. MCTPTA takes $22s$ to reach the vicinity of the optimal solution, which further highlights the benefits provided by utilising the frontier states in subset F, as opposed to the ad-hoc approach of TPTA.

In the next section, we will evaluate the efficiency of the approach as the occupation rate increases and approaches 1, thus rendering the problem more difficult to solve. We will also exploit the properties of the Foster-Lyapunov based partition (see Section 3.3), and how it can help find good approximate solutions for large-scale problems.

7. A new approximate algorithm using the Foster-Lyapunov function

Consider the same queue management problem of Section 6 and the partitioning scheme described in Eq. (32), but with arrival rates varying from $\lambda = 1$ to $\lambda = 2.25$. We will see later in the section that higher arrival rates will make the problem more difficult to solve and render value iteration less efficient.

To address the added complexity and to illustrate the effectiveness of the proposed partitioning scheme (Def. 1 and 2), we propose an approximate multicluster two-phase time aggregation algorithm (AMCTPTA). It consists in utilising the properties of the Foster-Lyapunov partition to build an approximate model and using its solution as an approximation to the original problem. To construct the approximate model, we truncate the state space by considering only states belonging to subsets B_1 to B_4, plus the frontier states in B_5 (subset F_5) from the partition in (32). For the remaining states, we fix an *ad-hoc* policy. In doing so, the AMCTPTA will sweep a state space with $56^3 \approx 1 \cdot 10^5$ elements, instead of the whole state space comprising $151^3 \approx 3 \cdot 10^6$ states. Note, however, that the results in Appendix A ensure that the states left out are seldom visited and hence their impact on the quality of the approximate solution will be limited, as demonstrated below.

Table 5 shows the average cost and time to convergence (in parenthesis) obtained by applying Relative Value Iteration (VI) and Approximate Multicluster Two-phase Time Aggregation (AMCTPTA) to solve the problem for different arrival rates. As expected, VI converges slower as the arrival rate increases. We can also see that proposed

algorithm (AMCTPTA) considerably outperforms VI in all instances, converging in a matter of seconds for the easier problems, and in about 25% of the VI time for $\lambda = 2.25$. Table 5 also shows that, as expected, the effect of the state space truncation is negligible in terms of the quality of the solution.

Table 5: Computation Results

	Average Cost (Time in Sec.)			
Algorithm	$\lambda = 1$	$\lambda = 1.5$	$\lambda = 2.0$	$\lambda = 2.25$
VI	3.00022 (2,038 s)	5.81106 (2,512 s)	12.6144 (6,211 s)	23.4402 (15,720 s)
AMCTPTA	3.00022 (68 s)	5.81106 (120 s)	12.6144 (843 s)	23.5147 (4,020 s)

To further explore the effect of the truncation and illustrate the potential of the Foster-Lyapunov stability results in Appendix A, we simulated the policy attained by the approximate (AMCTPTA) algorithm, using an ad hoc policy for all states outside of the truncation whereby the system routes a new arrival to the server with the smallest queue and breaks eventual ties by routing to the queues with the lower cost. We simulated the system for 1 million steps, starting from an empty queue. Table 6 depicts the empirical average cost obtained by the simulation for each input rate λ, and the number of times the process visited states outside the truncation (in brackets).

Table 6: Computational Results

	Average Cost (Number of visits outside truncation)			
Algorithm	$\lambda = 1$	$\lambda = 1.5$	$\lambda = 2.0$	$\lambda = 2.25$
Simulation	3.00368 [0]	5.79566 [0]	12.5574 [0]	23.0096 [0]

Table 6 shows that the system did not leave the truncated region within the simulation, obtaining an empirical average cost close to the optimal. This corroborates with the theoretical results from Theorem 7 in Appendix A and demonstrates that states with large values of the Foster-Lyapunov function $g(\cdot)$ hardly impact on the long-term average cost as they are seldom visited. It also illustrates the power of AMCTPTA to find near-optimal solutions to large-scale problems.

Naturally, the proximity to the optimal solution will depend on the degree of truncation, as well as the arrival rate. For larger problems, with more variables (e.g. servers), a greater degree of truncation may be necessary to make the solution computationally viable or to solve the problem in a shorter computational time. Alternatively, one can use approximate dynamic programming or reinforcement learning to find an approximate policy for states outside of the truncated region.

To further explore the application of AMCTPTA to large scale problems, we will next explore a similar queuing problem with 4 servers, which renders the problem intractable for classical approaches in the computational setup that we used for our experiments.

7.1. A larger scale queuing network example

Consider a queuing problem with four servers and a variable arrival rate $0 < \lambda < \infty$. Assume that the service time of the servers follows exponential distributions with rate 0.5 for servers 1 and 2, rate 1.5 for server 3 and rate 1 for

server 4. We also assume that the cost of each state is given by:

$$f(x) = x_1 + 2x_2 + 4x_3 + 5x_4,$$

where x_1, x_2, x_3 and x_4 are the queue size of servers 1, 2, 3 and 4, respectively. Since demands and service batches always comprise a single item, and the state space is finite, it is not difficult to see that the chain is irreducible and ergodic under all all feasible policies $\mathcal{L} \in \mathbb{L}$. The objective is to find the policy $\mathcal{L}^*$ which minimises the average cost and satisfies (2).

Let $\bar{k} = \arg\max_{k\in\{1,2,3,4\}} |x_k|$ and let the Foster-Lyapunov function $g : S \to \mathbb{Z}_+$ be defined as:

$$g(x) = \begin{cases} |x_{\bar{k}} + x_{\hat{k}}|, & \text{if } |x_{\bar{k}} - x_{\hat{k}}| = 1 \text{ for some } \hat{k} \in \{1,2,3,4\} \setminus \{\bar{k}\}, \\ 2 \max\limits_{k\in\{1,2,3\}} |x_k|, & \text{otherwise.} \end{cases} \tag{33}$$

We partitioned the state space according to Eq. (32). Like in the previous section, we assume that only actions $a \in A(i)$ for which $\sum_{j\in S} p_{ij}^a g(j) < g(i)$ are deemed feasible for all states $i \notin \bigcup_{l=1}^{3} B_l$. This implies that the problem satisfies Assumption 1 with $S_1 = \bigcup_{l=1}^{3} B_l$.

We applied AMCTPTA constrained to the states in subsets B_1, B_2, B_3, as well as the frontier states in B_4. The results are exposed in Table 7.

We set the same ad-hoc policy utilised in the previous section for the states outside of the truncation, and simulated the system for 1 million steps to obtain an approximate average cost, as well as an empirical probability of leaving the states within the truncation. Table 7 depicts the results, with the probability of leaving the truncation in brackets. Notice that only for the largest arrival rate the system ever leaves the truncated region, and even in that case the probability of leaving is very small $\left(\frac{1,544}{10^6}\right)$. This further demonstrates that the states left out of the truncation have very small effect on the overall performance.

Table 7: Computational Results

	Average Cost (Number of visits outside truncation)			
Algorithm	$\lambda = 1$	$\lambda = 2.0$	$\lambda = 3.0$	$\lambda = 3.25$
Simulation	2.87788 [0]	7.82091 [0]	24.2004 [0]	48.8414 [1,544]

The results in this section illustrate the potential of the proposed approach and highlight the power of the Foster-Lyapunov partitioning from Section 3.3 for large-scale Markov decision processes.

While this paper provides a theoretical underpinning to guide the use of state space partitioning combined with time aggregation to solve large scale Markov decision problems, which utilises not only the topological properties of the partitioning, but also hinge on Foster-Lyapunov stability results to guide the partitioning design, it provides interesting avenues for future research. For instance, further research is needed to investigate algorithmic approaches

to find suitable approximate policies in remote regions of the state space that might be left out of the truncated approximate problem. Possible alternatives include combining the proposed partitioning approach with approximate dynamic programming, Monte Carlo simulation starting from the frontier states that communicate with the truncated region, or reinforcement learning algorithms.

8. Concluding remarks

This paper explores and combines two concepts, namely distributed computation and local search, to derive a class of efficient algorithms to solve Markov decision processes under the average cost criterion. To allow distributed computation, the paper introduces a new partitioning scheme that induces desirable topological properties by design within each subset of the partition. These allow us to evaluate the time aggregated semi-regenerative trajectories by focusing on a single subset at a time, thereby limiting the computational effort. In addition, it also allows distributed policy improvement steps that focus on a single subset at a time, contributing to accelerate convergence.

The partitioning scheme allows a seamless design to two-phase time aggregation, as it automatically sets up the subset F where the embedded semi-Markov process evolves. By design, this subset comprises those states that can be accessed from different subsets in the partition, thus setting up a local search within the states from which information spreads across the state space. In contrast, classical two-phase time aggregation utilises an arbitrary design of the embedded subset, relying heavily on the designer's knowledge of the problem domain.

Additionally, we explore the concept of stochastic stability to set up the subsets within the partition via a Foster-Lyapunov function. This ensures that states with high values of the Lyapunov function will be seldom visited and therefore contribute little to the long-term average cost. Combined with the topological properties of the proposed partitioning scheme, this will provide a theoretical underpinning for approximate algorithms which can guide the design of algorithms that focus the computational effort on the regions of the state space that contribute the most to the long-term performance.

The computational experiments demonstrate the potential of the partitioning approach when combined with time aggregation to quickly approach the optimal solution, whilst also illustrating the potential to address large-scale problems by exploiting Foster-Lyapunov stability conditions. Potential avenues for future work include extensions to MDPs with discounted long-term cost criterion and the exploration of novel strategies to explore remote, seldom visited, regions of the state space. The design of the latter can make use of recent advances in reinforcement learning and approximate dynamic programming, whilst also making use of the topological properties induced by the partitioning design.

Acknowledgements

We are indebted to the editor and the reviewers for their many comments and suggestions, which helped to significantly improve the paper.

This work was partially supported by the Brazilian national research council - CNPq, under grants #311075/2018-5 and #312119/2020-8, and by Coordenação de Aperfeiçoamento de Pessoal de Nível Superior—Brasil (CAPES) [Finance Code 001].

References

Arruda, E. F., & Fragoso, M. D. (2011). Time aggregated Markov decision processes via standard dynamic programming. *Operations Research Letters*, *39*, 193–197.

Arruda, E. F., & Fragoso, M. D. (2015). Solving average cost Markov decision processes by means of a two-phase time aggregation algorithm. *European Journal of Operational Research*, *240*, 697–705.

Arruda, E. F., Fragoso, M. D., & Ourique, F. (2019). A multi-cluster time aggregation approach for Markov chains. *Automatica*, *99*, 382–389.

Arruda, E. F., Ourique, F. O., LaCombe, J., & Almudevar, A. (2013). Accelerating the convergence of value iteration by using partial transition functions. *European Journal of Operational Research*, *229*, 190–198.

Bertsekas, D. (2021). *Rollout, policy iteration, and distributed reinforcement learning*. Athena Scientific.

Bertsekas, D. P. et al. (2000). *Dynamic programming and optimal control: Vol. 1*. Nashua: Athena scientific Belmont.

Brémaud, P. (2020). *Gibbs fields, Monte Carlo simulation, and queues*. Number 31 in Texts in Applied Mathematics (2nd ed.). Gewerbestrasse: Springer-Verlag.

Cakir, F., Thomas, B. W., & Street, W. N. (2023). Rollout-based routing strategies with embedded prediction: A fish trawling application. *Computers & Operations Research*, *150*, 106055.

Cao, X.-R. (1998). The relations among potentials, perturbation analysis, and Markov decision processes. *Discrete Event Dynamic Systems*, *8*, 71–87.

Cao, X.-R., Ren, Z., Bhatnagar, S., Fu, M., & Marcus, S. (2002). A time aggregation approach to Markov decision processes. *Automatica*, *38*, 929–943.

Ciucu, F., Poloczek, F., & Rizk, A. (2019). Queue and loss distributions in finite-buffer queues. *Proc. ACM Meas. Anal. Comput. Syst.*, *3*. doi:10.1145/3341617.3326146.

Dai, P., & Goldsmith, J. (2007). Topological value iteration algorithm for Markov decision processes. In *IJCAI* (pp. 1860–1865).

Debnath, S., Liu, L., & Sukhatme, G. (2018). Solving Markov decision processes with reachability characterization from mean first passage times. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* (pp. 7063–7070). IEEE.

Foster, F. G. (1953). On the stochastic matrices associated with certain queuing processes. *The Annals of Mathematical Statistics*, *24*, 355–360.

Hu, B., Liu, D., Zang, C., Wang, Y., Li, W., Liu, Y., & Zeng, P. (2020). Coordinate dispatch of combined heat and power system based on approximate dynamic programming. In *2020 IEEE 4th Conference on Energy Internet and Energy System Integration (EI2)* (pp. 3772–3778). IEEE.

Jiang, D. R., & Powell, W. B. (2015). An approximate dynamic programming algorithm for monotone value functions. *Operations Research*, *63*, 1489–1511.

John, I., Kamanchi, C., & Bhatnagar, S. (2020). Generalized Speedy Q-Learning. *IEEE Control Systems Letters*, *4*, 524–529.

Kamanchi, C., Diddigi, R. B., & Bhatnagar, S. (2019). Successive Over-Relaxation Q-Learning. *IEEE Control Systems Letters*, *4*, 55–60.

Malekipirbazari, M. (2025). Optimizing sequential decision-making under risk: Strategic allocation with switching penalties. *European Journal of Operational Research*, *321*, 160–176. doi:https://doi.org/10.1016/j.ejor.2024.09.023.

Mohagheghi, M., Karimpour, J., & Isazadeh, A. (2020). Prioritizing methods to accelerate probabilistic model checking of discrete-time Markov models. *The Computer Journal*, *63*, 105–122.

Moore, A. W., & Atkeson, C. G. (1993). Prioritized sweeping: Reinforcement learning with less data and less time. *Machine Learning*, *13*, 103–130.

Powell, W. B. (2011). *Approximate Dynamic Programming: Solving the Curses of Dimensionality*. New Jersey: John Wiley & Sons.

Powell, W. B. (2019). A unified framework for stochastic optimization. *European Journal of Operational Research*, *275*, 795–821. doi:https://doi.org/10.1016/j.ejor.2018.07.014.

Prata, S. (2001). *C++ Primer Plus*. (4th ed.). USA: Sams.

Puterman, M. L. (2005). *Markov decision processes: discrete stochastic dynamic programming*. New Jersey: John Wiley & Sons.

Puterman, M. L., & Shin, M. C. (1982). Action elimination procedures for modified policy iteration algorithms. *Operations Research*, *30*, 301–318.

Shlakhter, O., Lee, C.-G., Khmelev, D., & Jaber, N. (2010). Acceleration operators in the value iteration algorithms for Markov decision processes. *Operations Research*, *58*, 193–202.

Sun, T., Zhao, Q., & Luh, P. B. (2007). Incremental value iteration for time-aggregated Markov-decision processes. *IEEE Transactions on Automatic Control*, *52*, 2177–2182.

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

Veatch, M. H. (2001). Fluid analysis of arrival routing. *IEEE Transactions on Automatic Control*, *46*, 1254–1257.

Watkins, C. J. C. H. (1989). *Learning from delayed rewards*. Ph.D. dissertation King's College Cambridge, United Kingdom.

Wei, Q., Lewis, F. L., Liu, D., Song, R., & Lin, H. (2018). Discrete-time local value iteration adaptive dynamic programming: Convergence analysis. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, *48*, 875–891.

Wei, Q., Liu, D., Lin, Q., & Song, R. (2017). Discrete-time optimal control via local policy iteration adaptive dynamic programming. *IEEE Transactions on Cybernetics*, *47*, 3367–3379.

Xu, J., Liu, B., Zhao, X., & Wang, X. (2024). Online reinforcement learning for condition-based group maintenance using factored markov decision processes. *European Journal of Operational Research*, *315*, 176–190. doi:https://doi.org/10.1016/j.ejor.2023.11.039.

Xue, X., Ai, X., Fang, J., Yao, W., & Wen, J. (2022). Real-time schedule of integrated heat and power system: A multi-dimensional stochastic approximate dynamic programming approach. *International Journal of Electrical Power & Energy Systems*, *134*, 107427.

Appendix A. The choice of the partition subsets

To guide our partition scheme for large-scale problems, we will make use of the powerful sufficient conditions for positive recurrence (ergodicity) in Foster's theorem (Brémaud, 2020, Theorem 7.1.1, page 227); see also (Foster, 1953). Assume that for any feasible policy $\mathcal{L} \in \mathbb{L}$, the Markov chain is irreducible and there exists a Foster-Lyapunov function $g : S \to \mathbb{Z}_+$, where $\mathbb{Z}_+$ is the set of non-negative integers, such that:

$$\begin{aligned} &\sum_{j\in S} p_{ij}^{\mathcal{L}} g(j) < \infty, i \in S_1 \\ &\sum_{j\in S} p_{ij}^{\mathcal{L}} g(j) \le g(i) - \epsilon, i \notin S_1, \\ &\max_{i\in S_1} g(i) < \min_{j\notin S_1} g(j), \end{aligned} \tag{A.1}$$

for some finite set $S_1 \subset S$ and scalar $\epsilon > 0$. As per Foster's theorem, this is a sufficient condition to ensure that the controlled chain is ergodic under all feasible policies, as assumed in Section 2, even if the state space's cardinality is countably infinite.

Now, to guide our choice of the partition scheme, let us define a sequence of stopping times $\tau_0 = 0$ and $\tau_k = \min\{t > \tau_{k-1} : X_t \in S_1^c\}$, $k = 1, 2, \ldots$, where $S_1^c = S \setminus S_1$ is the complement of subset S_1. The embedded Markov chain $Y_k \triangleq X_{\tau_k}$, $k \ge 0$ represents the trajectories of the controlled process X_t, $t \ge 0$ in S_1^c. Define also:

$$\bar{\beta} \triangleq \max_{j\in S_1^c} \left\{ E^{\mathcal{L}} \left[\max\left(g(X_t) - g(X_{t+1}) \right) \mid X_t = j \right] \right\},$$

which gives the maximum expected one-step decrease in the Lyapunov function over all states $j \in S_1^c$, under policy $\mathcal{L}$. And let β be a random variable such that:

$$P(\beta = c) = \begin{cases} \dfrac{\bar{\beta}}{\lceil \bar{\beta} \rceil}, & \text{if } c = \lceil \bar{\beta} \rceil, \\ 1 - \dfrac{\bar{\beta}}{\lceil \bar{\beta} \rceil}, & \text{if } c = 0. \end{cases} \tag{A.2}$$

One can easily see that $E(\beta) = \bar{\beta}$, where β can assume one of two integer values: either c or 0. We can now represent the accumulated increase in the Lyapunov function $g : S \to \mathbb{Z}_+$ within the set S_1^c by process:

$$Z_{t+1} = \max\{0, Z_t + a_t - \beta\}, \; Z_0 = 0, \tag{A.3}$$

where

$$a_t \triangleq r(Y_t) = \beta + (\, g(X_1) - g(X_0)\,) \,|\, X_0 = Y_t,$$

is an integer random variable that is modulated by process Y_t, $t \geq 0$. One can also easily see that process Z_t, $t \geq 0$ is stable as $E(a_t) < E(\beta)$, $\forall t \geq 0$. Process Z_t, $t \geq 0$ as defined above is equivalent to a Markov modulated process (Ciucu et al., 2019, Eq. (2)), with an unbounded queue ($K \to \infty$), and a random packet size is either c or zero, with probabilities defined in Eq. (A.2). This fits the definition of a Markov modulated process with random packet size that admits a Martingale envelop (Ciucu et al., 2019, Section 6.2). This, in turn, signifies that (Ciucu et al., 2019, Theorem 3) holds for process Z_t, $t \geq 0$, which implies:

$$\lim_{t\to\infty} P(Z_t > \sigma) \leq Me^{-\theta\sigma}, \tag{A.4}$$

for some scalars $0 < M < \infty$ and $\theta > 0$. Eq. (A.4) corresponds to the third expression in (Ciucu et al., 2019, Theorem 3), for the case of an unlimited queue ($K \to \infty$).

The value of θ depends on the intensity of the drift of process Z_t, $t \geq 0$, which is by definition equal to the drift in S_1^c in Eq. (A.1). The main result of this section follows in Theorem 7.

Theorem 7. *Let $g : S \to \mathbb{Z}_+$ satisfy Foster's stability condition in* (A.1) *and $D = \{j \in S_1^c : \exists i \in S_1$ such that $p_{ij}^{\mathcal{L}} > 0\}$ represent the states in S_1^c that can be directly reached from S_1. Assume the controlled Markov process X_t, $t \geq 0$ is operating under policy $\pi \in \Pi$. Then, it follows that:*

$$\lim_{t\to\infty} P\,(\, g(X_t) > \bar{g} + \sigma\,) \leq Me^{-\theta\sigma}, \qquad \bar{g} = \max_{j\in D} g(j). \tag{A.5}$$

Proof. Eq. (A.5) follows directly from (A.4), by noticing that $\bar{g}$ is the largest possible value of $g(Y_t)$ at the outset of a trajectory in S_1^c, when $Z_t = 0$. This is straightforward, since Z_t, $t \geq 0$ represents the cumulative increment of process Y_t, $t \geq 0$ in the current sojourn within F_1^c, and considering that the sojourn must have started with $g(Y_t) \leq \bar{g}$.