CLUSTER-BASED CONTROL OF TRANSITION-INDEPENDENT MDPs

A PREPRINT

Carmel Fiscko$^1$, Soummya Kar$^1$, and Bruno Sinopoli$^{2*}$

November 11, 2022

ABSTRACT

This work studies efficient solution methods for cluster-based control of transition-independent Markov decision processes (TI-MDPs). We focus on the application of control of multi-agent systems, whereby a central planner influences agents to select target joint strategies. Under mild assumptions, this can be modeled as a TI-MDP where agents are partitioned into disjoint clusters such that each cluster can receive a unique control. To efficiently find a policy in the exponentially expanded action space, we present a clustered Bellman operator that optimizes over the action space for one cluster at any evaluation. We present Clustered Value Iteration (CVI), which uses this operator to iteratively perform “round robin” optimization across the clusters. CVI converges exponentially faster than standard value iteration (VI), and can find policies that closely approximate the MDP’s true optimal value. A special class of TI-MDPs with separable reward functions are investigated, and it is shown that CVI will find optimal policies. Finally, the optimal clustering assignment problem is explored. The value functions of separable TI-MDPs are shown to be submodular functions, and notions of submodularity are used to analyze an iterative greedy cluster splitting algorithm. The values of this clustering technique are shown to form a monotonic, submodular lower bound of the values of the optimal clustering assignment. Finally, these control ideas are demonstrated on simulated examples.

1 Introduction

Social media users, robot swarms, and financial transactions are examples of system behavior driven by the decisions of individuals. A person’s posts online reflect their own opinions, yet they are influenced by their friends, trends, and current events. These decisions, in turn, affect the path of future posts and trends. Within cooperative robotics, a central objective must be completed, yet each robot’s path must consider those of its peers. On the competitive side, businesses weigh their rivals’ actions to likewise position themselves to an advantage. Each of these examples show the importance of each agent’s goals, peers, and reactions to each other. Understanding the long-term group behavior can thus achieved by understanding the individuals. Any regulatory power that wishes to achieve control objectives on the system must therefore must understand the agents’ behavioral processes.

Consider a traffic control example. Drivers query tools such as Google Maps to get the fastest routes, but as a central controller Google Maps could optimize the collective travel time of all their users. As an adversarial example, the navigation app Waze collects crowd-sourced information on road blockages and police presence. Reports have accused angry homeowners of reporting false blockages to reduce traffic on neighborhood streets [1], and have accused reports of false police vehicles of reducing the integrity of the app’s functions [2]. From a more technical viewpoint, multi agent system theory has been used to propose traffic signal control [3] and scheduling of taxis [4].

$^1$Carmel Fiscko and Soummya Kar are with the Dept. of Electrical and Computer Engineering at Carnegie Mellon University in Pittsburgh, PA. cfiscko@andrew.cmu.edu, soummyak@andrew.cmu.edu

$^2*$Bruno Sinopoli is with the Dept. of Electrical and Systems Engineering at Washington University in St. Louis, MO. bsinopoli@wustl.edu
This objective of global control of heterogeneous agents has been widely explored in domain-specific application. In power systems distribution network management has been approached via multi-agent reinforcement learning (MARL) [5] and deep reinforcement learning (RL) [6]. In economic applications, MARL has been used to analyze market makers providing liquidity to markets [7], and creating tax policies that balance equality and productivity [8]. A few social objectives that have been considered include the issues of autonomous vehicle adoption [9] and responses to climate change [10].

1.1 Multi-Agent Systems

Such groups of agents are often modeled as multi-agent systems (MAS), which widely appear in distributed learning applications. In this paper, we adopt the definition of agent from [11]: “we consider an agent to be an entity with goals, actions, and domain knowledge, situated in an environment. The way it acts is called its ‘behavior.’” These agents can model applications such as computer networks, smart grids, and cyberphysical systems, in addition to the aforementioned models of social media, robots, and finance [12]. Natural questions of these topics include control, security, coordination, and scalability.

In engineered models of MAS, a common question is how should each agent act optimally to achieve a personal objective within the interconnected system. In a game theoretic model, an individual is endowed with a utility function that rates the goodness of a potential action in the current world [13]. Armed with these utilities, the agent chooses their next action according to a personal rule. The collectively played action across the system, known as the joint action or joint strategy, thus evolves stochastically as agents consider their goals and their peers. The outcome of this game may be described with optimal actions and equilibria, in which agents self-reinforce game-play at the same action.

MARL was born from the field of Markov or stochastic games [14], which extends the idea of stochastic repeated play games for agents to learn within a probabilistic environment, as presented in the seminal work [15]. Many works have investigated how agents may learn policies in this setting under a variety of assumptions, structures, and guarantees of optimality. A brief overview is provided here for context. The objective of the agents can generally be described as cooperative [16] [17], competitive [15], or general-sum [18]. Various agent communication structures have been proposed, ranging from fully centralized [19] to decentralized [20]. Techniques for agents to learn policies generally fall under value-based methods where agents estimate q-functions [21], or policy-based methods such as fictitious play [22], counterfactual regret minimization [23], and Thompson sampling [24]. Additional information may be found in the survey papers [25], [26], [27].

In this work, we consider a MAS with agents who have learned policies. These policies may be modeled formally via game theory or MARL, or they may simply be learned via frequentist estimation of observed agent behavior. Depending on the guarantees of the found policies, if any exist, the long-term behavior of the MAS may result in an equilibrium at one state, cyclical behavior between a subset of states, or some stationary distribution over the entire state space.

The objective in this work is to characterize the ability of a central planner (CP), i.e. a “superplayer,” to change this long-term behavior exhibited by the agents. The difference between this work and MARL is that we consider a centralized third-party entity who controls the system given the agents’ a priori learned policies; MARL solves the question of how agents should learn the policies. The CP takes advantage of these learned policies in that these policies are dependent on the actions of the agents, any environmental states, and the CP signal; the CP can thus select controls to increase the probability that the agents sample “good” future actions. There are, in essence, two levels of decision-making: each agent chooses actions to accomplish the agent’s goal, and the controller influences the agents by making certain actions seem “better” and thus more likely to be chosen. The goal of the superplayer may be aligned with, or completely different from the goals of the agents.

Under assumptions or a construction of Markovian dynamics, we model the controlled MAS as a Markov decision problem (MDP) [25]. The controller acts to achieve some personal goal, often stated as maximizing the expected sum of rewards received from visiting states. This formulation is similar to a MARL model where \( N - 1 \) agents have fixed (possibly non-optimal) policies and the \( N^{th} \) agent learns an optimal policy, under global visibility properties for the last agent. Assuming fixed policies of the other agents allows the superplayer to solve for their policy via a time-homogeneous MDP and infer additional control properties. This formulation gives great flexibility as the CP is agnostic to the specific process by which each agent learned their policy: agents may be competitive/cooperative/general-sum, acting optimally or sub-optimally, highly/sparsely connected, using policy-based or value-based methods, or a heterogeneous mix of all of the above, and this formulation may be used if the time-homogeneity and Markovian assumptions on the learned agent policies are fulfilled. It is precisely this flexibility that warrants defining a MDP from the superplayer’s perspective and analyzing the resulting control properties.
1.2 Modeling Agent Behavior with Transition-Independent MDPs

In this work, the superplayer’s MDP is defined such that the state space is the cartesian product of the individual agent state-action spaces. The MDP reward function can thus be defined for each joint state-action, and the resulting policy will steer the system towards highly rewarded agent behavior.

The downside to this approach is that the size of the state space will grow exponentially with respect to the number of agents. Techniques to handle the enlarged state space include the factored MDP [29] or transition-independent MDPs (TI-MDP) [30], in which state transition probabilities may be factored into independent components, such as one per agent. For example, consider from competitive game theory the best response dynamics, whereby an agent chooses a new action that would have been optimal at the last game instance. The state-to-state transition is the composition of the changes for each agent, and is thus factorable. Solutions for this formulation include approximate policy iteration [31], graphical methods [32], as well as approximate transition independence [33]. Efforts to reduce the scope of policy search include efficient policy sampling methods [34], relaxations to deterministic policies [35], and hierarchical methods [36]. Methods for basis function representations of value functions [37], and linear programming approaches [38] have been investigated.

1.3 Cluster-Based Control Policies

In this work we come to focus on clustered control techniques for MAS. Clustering techniques [39] are of importance in many engineering fields such as wireless sensor networks [40], and in particular, clustering for the purpose of control [41]. Robotic swarms provide a rich domain for application, where algorithms commonly aim to mimic animals’ innate abilities for self-grouping [42] [43]. Clustering in social media presents new challenges for analysis; the unpredictability of the users has led to data-driven techniques in machine learning [44].

Clustering has been extensively studied in the machine learning literature, such as in nearest neighbor search algorithms for unsupervised classification. Here we emphasize that we take an agent or node-based view of the MAS rather than an edge-based focus such as [45]. The set of agents is assumed to be partitioned, such that agents in the same partition receive the same control from central command, yet different clusters may receive different controls. The intuition is twofold: first, heterogeneous agents may still fall into a fewer number of similar types, and a practitioner may want to personalize controls for each type in accordance to system goals and constraints. Secondly, more degrees of freedom in control can yield better outcomes, even amongst agents for whom no clear partitions exist. Introducing some flexibility in the controls can improve performance of the outcome, as we show in section 2.2. The question stands as how to optimally choose the set partitions for the control objective. The TI-MDP MAS setup differs in that the objective does not necessarily care about closeness of the agents; the objective is concerned with the resulting value achieved by a cluster configuration. Our focus turns to the study of optimization of set partitions.

While the general optimization problem is known to be NP hard, approximations have been studied across many applications. Variants of this setup have been studied in the multiway partition problem (MPP) [46], but these are generally concerned with minimization with non-negativity constraints preventing analysis of maximization problems. Equivalently, the problem may be stated as an objective subject to a matroid constraint, for which a celebrated 1/2 approximate algorithm [47] for submodular optimization was then improved to a (1 − 1/e) approximation in [48] [49].

1.4 Our Contributions

In this work study the cluster-based TI-MDP control problem, both in terms of policy and the clustering assignment. We outline the problem formulation in Section 2 and show that a cluster-based control policy improves the optimal value of a MDP due to the greater degrees of freedom in the control. Solving for an optimal cluster-based policy, however, is difficult due to the high dimensionality of the new action space. This motivates the main results; in Section 3 we introduce Clustered Value Iteration (CVI), an approximate “round robin” method for solving a MDP whose per-iteration complexity is independent of the number of clusters. Convergence properties of CVI are established, along with complexity and error analyses. These results build upon the work we presented in [50], providing additional error analysis and optimality results. In Section 4 explores a deep dive into TI-MDPs with separable reward functions; CVI is shown to find optimal policies for this class of problems.

Section 5 investigates the optimal clustering assignment problem of the agents. While the full discrete optimization problem is NP-hard, we show that the value function is submodular for certain types of MDPs, thus allowing properties of submodularity to help find an approximate solution. We offer an approximate approach that iteratively builds clusters through greedy splitting. The optimal values of these suboptimal clustering assignments are shown to form a submodular lower bound for the optimal values of the optimal clustering assignments. Overall this method gives a structured search to the clustering problem with provable improvement in value.
1.5 Comparison to Other Behavior Control Paradigms

The objective of changing the behavior of a MAS has been studied in other domains beyond RL. Unsurprisingly, this is a common goal in economic applications, where a business may desire an optimal outcome, or a regulator may desire maximal welfare. This is studied in mechanism design [51], [52], where an economist may design agent interaction properties like information and available agent actions to induce a desired outcome. This work is related to, but distinct from, incentive design [53] where rewards are offered to agents to encourage them to select different actions. Incentive design can be connected with Stackelberg games [54], whereby a leader offers rewards and the agents react accordingly.

Another related paradigm, known as interventions, controls networked agents to induce a desired effect. For example, [55] considers removing an agent from a system to optimally improve aggregate network activity. Similar network-based interventions aim to maximize system welfare via designed incentives [56], and subject to random graphs [57].

In this work, any network structure can be encoded into the transition matrix of the MDP, and the proposed control methods are largely agnostic to graphical structure. In addition, the techniques we present are not constrained to a specific structure of network influence, i.e. aggregation, and we do not presuppose a method for superplayer influence beyond changing probabilities, i.e. we do not assume a specific incentive model. The scenario we consider is best described with the superplayer as an a posteriori regulator; someone who desires to induce change, but is bounded by the current agent identities, existing system infrastructure, and inherited behavioral problems.

2 Problem Statement

2.1 Multi-Agent MDPs

Consider a multi-agent system modeled as a Markov Decision Process (MDP) \( \mathcal{M} = (\mathcal{X}, \mathcal{A}, R, T, \gamma) \) consisting of the state space of the system \( \mathcal{X} \), the action space of the superplayer \( \mathcal{A} \), a reward function of the superplayer \( R : \mathcal{X} \times \mathcal{A} \rightarrow \mathbb{R} \), a probabilistic state-action to state transition function \( T : \mathcal{X} \times \mathcal{A} \rightarrow \Delta(\mathcal{X}) \), and a discount parameter \( \gamma \in (0, 1) \).

The MDP must next be related to the MAS. Consider a finite set of agents \( \mathcal{N} = \{1, \ldots, N\} \). At any instance, each agent \( n \in \mathcal{N} \) is associated with some substate \( x_n \in \mathcal{X}_n \) describing the agent’s behavior. The composition of \( x_n \) will depend on the application, but will include an agent action \( a_n \in A_n \), and may include an optional agent environmental state \( s_n \in S_n \). All sets \( A_n, S_n \), and thus \( \mathcal{X}_n \) and \( \mathcal{X} = \bigotimes_{n \in \mathcal{N}} \mathcal{X}_n \) are assumed to be finite.

The collective state \( x = [x_1, \ldots, x_N] \) describes the full behavior of the system. In general, a state written with a set subscript such as \( x_B \) refers to the actions realized in state \( x \) by the agents in \( B \), i.e. \( x_B = \{x_b|b \in B\} \), and the notation \( -n \) will refer to the set \( \{n|n \in \mathcal{N}, n \neq n\} \).

In this work we consider a clustered formulation, in which the agents are partitioned into \( C \) disjoint clusters and controls are transmitted to each cluster. A clustering assignment of the agents is the set of sets \( C = \{C_1, \ldots, C_C\} \) such that \( C_u \cap C_v = \emptyset \forall u, v \in [1, C], u \neq v \) and \( \bigcup_{c \in C} = \mathcal{N} \).

In this setup, the action from the central controller is the vector \( \alpha = [\alpha_1, \ldots, \alpha_C] \) with action space \( \mathcal{A} = \bigotimes_{c \in C} \mathcal{A}_c \) where \( \alpha_c \in \mathcal{A}_c \) is a finite set. Bold \( \alpha \) will always refer to the vector, and \( \alpha \) will refer to an element within the vector. By design, the agents in some cluster \( c \) only see the action assigned to their cluster, \( \alpha_c \). For ease of notation, the action seen by agent \( n \) will be denoted by \( \alpha_n \) instead of \( \alpha_{c(n)} \).

The next element of the MDP is the state transition function, which defines the state-action to state densities in the form \( T(x, \alpha, x') = p(x'|x, \alpha) \). As the system-wide transitions are dependent on the decision processes of the individual agents, it is important to investigate the structure of the transition function.

Definition 1. An agent policy \( \omega_n : \mathcal{X} \times \mathcal{A}_n \rightarrow \Delta(A_n) \) describes the decision-making process of an agent by defining a distribution over the agent’s next actions given the current system state and superplayer’s signal to the agent.

Definition 2. If there exist environmental states, such as in the MARL framework, then the agent environmental transition \( \psi_n : \mathcal{X} \times \mathcal{A}_n \rightarrow \Delta(S_n) \) defines a distribution over the agent’s next environmental state given the current system state and the superplayer’s signal to the agent.

Definition 3. The agent behavior \( \phi_n : \mathcal{X} \times \mathcal{A}_n \rightarrow \Delta(X_n) \) defines the system state-action to agent state transitions. If there are no environmental states, then the agent behavior is equivalent to the agent policy. If there are environmental
states, then the agent behavior is the product of the agent’s policy and its environmental transition:

\[
p(x_{n+1}^t|x^t, \alpha_n^t) = p(s_{n+1}^t, a_{n+1}^t|s^t, a^t, \alpha_n^t) = p(a_{n+1}^t|\alpha_n^t)p(s_{n+1}^t|s^t, a^t, \alpha_n^t)
\]

(1)

**Assumption 1.** The agents’ policies, environmental transitions, and behaviors are Markovian and time-homogeneous:

\[
p(x_{n+1}^t|x^t, \ldots, x^0, \alpha_n^t, \ldots, \alpha_0^t) = p(x_{n+1}^t|x^t, \alpha_n^t), \forall \alpha_n \in \mathcal{A}_n, x_n \in X_n, n \in \mathcal{N}, x \in \mathcal{X}, t \geq 0.
\]

(2)

Furthermore, each agent’s policies, environmental transitions, and behaviors are independent of the superplayer actions assigned to the other agents:

\[
p(x_{n+1}^t|x, \alpha_n) = p(x_{n+1}^t|x, \alpha_n), \forall \alpha_n \in \mathcal{A}_n, x_n \in X_n, n \in \mathcal{N}, x \in \mathcal{X}, t \geq 0.
\]

(3)

These decision making processes are assumed to follow the standard Markov property of depending only on the current state \(x\) and the influence \(\alpha_n\) assigned to the agent. Furthermore, the time homogeneity property means that the agents have learned their decision processes \(a\ priori\), be it from MARL, game theory, or another paradigm. The superplayer is agnostic to the learning processes used by the agents as long as they satisfy the Markov and time-homogeneity assumptions. In particular, no specific agent learning process needs to be specified, as a trusted pre-learned empirical model of agent behavior would suffice.

These are the key assumptions enabling the superplayer to take advantage of the agents’ learned behaviors. While the set of agent behavioral distributions are fixed over time, the superplayer changes \(\alpha\) to change the distributions from which the agents actually sample, thus also changing the sequence of realized \(x\). In this way the agents do not learn new behavior, but react to the superplayer with their existing behavior.

Given Assumption 1, the MDP has structure imbued into its transition kernel.

**Definition 4.** A transition independent MDP (TI-MDP) is a MDP whose state transition probabilities may be expressed as the product of independent transition probabilities,

\[
p(x'|x, \alpha) = \prod_{i \in \mathcal{I}} p(x'|x, \alpha_i).
\]

(4)

The TI-MDP is “agent-independent” if \(\mathcal{I}\) is equivalent to the set of agents, and “cluster-independent” if \(\mathcal{I}\) is equivalent to the set of clusters. Note that factorization across the agents implies factorization across the clusters.

TI-MDPs are closely related to factored MDPs, which also have a structure such as (4) assumed on their transition matrix \[37, 38\]. Factored MDPs are usually derived as the result of a dynamic Bayesian network (DBN) such that:

\[
p(x'|s, \alpha) = \prod_{n \in \mathcal{N}} p(x_{n+1}'|x_{u(n)}', \alpha_n),
\]

(5)

where \(u(n)\) is the set of parents of agent \(n\) in the graph. In addition, factored MDPs defines a separable reward as the sum of local reward functions scoped by the same graph \(r(s, \alpha) = \sum_{n \in \mathcal{N}} r_n(x_{u(n)}', \alpha_n)\).

In the rest of Section 2 and Section 3 we use the TI-MDP structure as in Definition 4 with no assumptions on the structure of the reward function. Section 4 will investigate what further improvements may be found when the separability of the reward function can be assumed.

Definition 4 describes MAS where each agent makes their decision independently after observing the current state and control. This includes general non-cooperative games \[13\]. For example, in best response style game-play, each player \(n\) chooses an action that maximizes their utility function \(u\) based on the last round of game-play, i.e. \(a_n' \in \arg \max a_n u(a, a_n, a_n')\). Knowledge of the players’ utilities allows the controller to model each agent’s decision process as a probability function \(p(a_n'|a, \alpha_n)\). Therefore, any overall action transition \(a \to a'\) can be described as the product of \(N\) independent factors that are each conditioned on \(a\) and \(\alpha\). In comparison, communication between agents after observing \((a, \alpha)\) and before setting a probability distribution over \(a_n'\) may introduce dependence between agents; however, disjoint communication sets may still enable transition independence between clusters.

Finally, the reward function of the MDP must be defined.

**Assumption 2.** The reward function \(r(x, \alpha)\) is non-negative, deterministic, and bounded for all \(x \in \mathcal{X}, \alpha \in \mathcal{A}\).

The conditions in Assumption 2 follow from the motivation that the superplayer has constructed a goal \(a\ priori\), and can encode this known objective into the reward function. For example, an indicator reward can place a reward of one on desirable states.
Finally, the superplayer may solve the MDP for some policy \( \Pi : \mathcal{X} \rightarrow \mathcal{A} \). In this work we consider the goodness of a policy to be its expected infinite horizon discounted sum of rewards, known as the value function:

\[
V^\Pi(x) = \mathbb{E} \left[ \sum_{k=0}^{\infty} \gamma^k r(x_k, \alpha_k) | x_0 = x, \alpha_k \sim \Pi(x_k), x_{k+1} | x_k, \alpha_k \sim T \right].
\]

(6)

For brevity, the notation \( V \in \mathbb{R}^{\mathcal{X}} \) will refer to the vector of values \( V(x) \) \( \forall x \in \mathcal{X} \). The policy notation refers to the tuple \( \Pi(x) = \{ \pi_1(x), \ldots, \pi_C(x) \} \) where \( \Pi(x) \in \mathcal{A} \) and \( \pi_i(x) \in \mathcal{A}_i \). An optimal policy \( \Pi^* \) is one that maximizes the value function,

\[
\Pi^*(x) \in \arg \max_{\Pi} V^\Pi(x).
\]

(7)

The optimal value \( V^\ast(x) \) is known to be unique, and for finite stationary MPDs there exists an optimal stationary deterministic policy \([59]\). The optimal value function will also satisfy the Bellman operator.

**Definition 5.** The Bellman operator \( T \) applied to the value function \( V(x) \) is,

\[
TV(x) = \max_{\alpha \in \mathcal{A}} \mathbb{E} \left[ r(x, \alpha) + \gamma V(x') | x, \alpha \sim T \right],
\]

(8)

\[
= \max_{\alpha \in \mathcal{A}} \sum_{x'} p(x'|x, \alpha) (r(x, \alpha) + \gamma V(x')).
\]

(9)

The optimal value function satisfies,

\[
TV^\ast(x) = V^\ast(x).
\]

(10)

### 2.2 Cluster-Based Control

An immediate result of the clustered MDP setup is that a better optimal value is attainable if there are several clusters, as this increases the degrees of freedom for the control. This idea is formalized in the following lemma.

**Lemma 1.** Consider a MDP whose agents have been partitioned into \( C \) clusters of arbitrary agent assignment. The optimal values achieved from \( C = 1 \) and \( C = N \) are the lower and upper bounds on the optimal values achieved by arbitrary \( C \).

\[
V^\ast_{C=1}(x) \leq V^\ast_{1 < C < N}(x) \leq V^\ast_{C=N}(x).
\]

(11)

**Proof.** See \([50]\). \qed

This initial result shows that any arbitrary clustering assignment can improve the optimal value of the MDP. This finding motivates further investigation into the clustered formulation, namely to find good clustered control policies, to characterize the benefit of such policies, and to find optimal clustering assignments.

### 3 Solving for a Clustered Control Policy

The main goal is to attain the improved value enabled by the clustered-based control, but this can only be done if an acceptable policy is found. Cluster-based control \( \alpha \) introduces more degrees of freedom than scalar \( \alpha \), but this method suffers in computation time. Standard solution techniques like VI and PI rely on the Bellman operator \([9]\), which maximizes over the entire action space. The cluster-based policy formulation dramatically increases the size of the action space, rendering the computation time intractable. For example, let each \( |\mathcal{A}_i| = M \); then the control \( \alpha \) has \( |\mathcal{A}| = M^C \) possible options. PI has a complexity of \( O(|\mathcal{A}| |\mathcal{X}|^2 + |\mathcal{X}|^3) \) per iteration, and VI has a complexity of \( O(|\mathcal{A}| |\mathcal{X}|^2) \) per iteration \([60]\). This causes the computation time for either method to grow exponentially with the number of clusters. This problem is exacerbated in regimes when the model is unavailable, as standard estimated methods like Q-learning and SARSA only guarantee optimality with infinite observations of all state-action pairs. The conclusion is that an alternate method for solving for \( \Pi^* \) in the clustered policy regime is needed.

In this work we present a clustered Bellman operator that only optimizes the action for one cluster, rather than across the entire joint action space. This clustered Bellman operator is used in an algorithm called Clustered Value Iteration (CVI) as shown in Algorithm \([1]\). This algorithm differs from standard VI in that it takes a “round-robin” approach to optimize the actions across all the clusters. While one the action for one cluster is being optimized, the actions
across the other clusters are held constant until their turn to be improved at future iterations. The intuition is that the controller can focus on making improvements cluster-by-cluster, and perform policy improvement without searching over the whole action space at any one iteration.

This investigation into clustered systems is thematically related to state aggregation methods, but the technical formulations are distinct. Both methods aim to reduce the dimensionality of the problem by solving smaller but related MDPs. In a state aggregation method, similar states are grouped together to form a new system with a reduced state space. In comparison, the clustered formulation is designed for MDPs whose state space is the Cartesian product across a set of substates; the clustering assignment partitions the substates, resulting in Cartesian products of a smaller size. In aggregate methods the new transition matrix is a weighted sum of the original transition probabilities; in the clustered formulation, the transitions are decomposed by their factored structure. These two methods could be used in tandem to enjoy further dimensionality reduction, whereby a clustered formulation could use aggregation techniques on the set of substates.

3.1 Algorithm Statement

Before detailing the algorithm, the following notation is introduced.

There exists an optimal stationary deterministic policy for this MDP, so our focus is constrained to deterministic policies \( \Pi \). Let \( \pi^c_k(x) = \alpha^c_k \) denote the action to cluster \( c \) at iteration \( k \) for state \( x \). The vector across all clusters is \( \Pi^k(x) = \alpha^k \). With some new notation, the Bellman operator can be defined with respect to one element of \( \alpha \).

**Definition 6.** Let \( \Pi_{-c}(x) = \alpha_{-c} \) be the tuple of \( \alpha \) without the element specified by the subscript.

\[
\Pi_{-c}(x) = \{\pi_1(x), \ldots, \pi_{c-1}(x), \pi_{c+1}(x), \ldots, \pi_C(x)\}, \quad (12)
\]

\[
\alpha_{-c} = \{\alpha_1, \ldots, \alpha_{c-1}, \alpha_{c+1}, \ldots, \alpha_C\}. \quad (13)
\]

**Definition 7.** For some \( \Pi(x), V \in \mathbb{R}^{|X|} \), and \( x \in X \), define \( T^c_{\Pi} : \mathbb{R}^{|X|} \rightarrow \mathbb{R} \) as the clustered Bellman operator that optimizes \( \alpha_c \in A_c \) for fixed \( \Pi_{-c}(x) \).

\[
T^c_{\Pi}V(x) \triangleq \max_{\alpha_c \in A_c} \mathbb{E} \left[ r(x, \{\Pi_{-c}(x), \alpha_c\}) + \gamma \mathbb{E} \left[ V(x') | x, \{\Pi_{-c}(x), \alpha_c\} \right] \right],
\]

\[
= \max_{\alpha_c \in A_c} \sum_{x' \in X} p(x' | x, \{\Pi_{-c}(x), \alpha_c\}) (r(x, \{\Pi_{-c}(x), \alpha_c\}) + \gamma V(x')). \quad (14)
\]

**Definition 8.** For some \( \Pi(x), V \in \mathbb{R}^{|X|} \), and \( x \in X \), define \( T_{\Pi} : \mathbb{R}^{|X|} \rightarrow \mathbb{R} \) as the clustered Bellman evaluation.

\[
T_{\Pi}V(x) \triangleq \mathbb{E} \left[ r(x, \Pi(x)) + \mathbb{E} \left[ V(x') | x, \Pi(x) \right] \right],
\]

\[
= \sum_{x' \in X} p(x' | x, \Pi(x)) (r(x, \Pi(x)) + \gamma V(x')). \quad (15)
\]

With this notation, a solution for finding a cluster-based control policy can now be presented. To enable the round-robin optimization, the order of the cluster optimization must be defined. It will generally be assumed in this paper that the following statement holds.

**Assumption 3.** The cluster optimization order \( \Omega \) is a permutation of \( \{1, \ldots, C\} \).

The proposed algorithm begins like standard VI by initializing the value of each state to zero. For each following iteration \( k \), the algorithm selects a cluster according to the optimization order \( c_k \leftarrow \Omega(k) \). Next, the clustered Bellman operator \( T^c_{\Pi_k} \) is applied to the current value estimate \( V_k(x) \). In the clustered Bellman operator, all clusters except for \( c_k \) follow the fixed policy \( \Pi^k_{-c_k} \), while the action for \( c_k \) is optimized. The policy for \( c_k \) is updated accordingly. A tie-breaking rule for actions is assumed for when \( \arg \max_{\alpha_c} T^c_{\Pi_k}V_k(x) \) has more than one element, such as choosing the \( \alpha_{c_k} \) with the smallest index. The algorithm terminates when the difference in value update is appropriately small.

The cluster-based approach of CVI improves computation complexity versus standard VI as the optimization step searches over a significantly smaller space. Again, let each \( \alpha_c \) have \( M \) possible actions. Standard VI on cluster-based controls has a complexity of \( O(M^C |X|^2) \) per iteration; in comparison, CVI iterations have a complexity of \( O(M |X|^2) \), thus eliminating the per-iteration dependence on the number of clusters. While there will be an increase in the number of clusters, the savings per iteration will make up for the overall computation time as explored in simulation in Section 6.

The following sections will examine the computation of this algorithm, develop its convergence properties, and discuss the optimality of the output policy.
Algorithm 1: Clustered Value Iteration (CVI)

1. $V_0(x) \leftarrow 0$, $\forall x \in \mathcal{X}$;
2. Initialize policy guess $\Pi^0(x) = \{\pi^0_0(x), \ldots, \pi^0_C(x)\}$, $\forall x \in \mathcal{X}$;
3. Choose cluster optimization order $\Omega$;
4. $k = -1$;
5. while $\|V_k - V_{k-1}\|_\infty > \epsilon$ do
   6. $k = k + 1$;
   7. $c_k \leftarrow \Omega(k \mod C)$;
   8. $V_{k+1}(x) \leftarrow T^{c_k}_{\Pi} V_k(x)$, $\forall x \in \mathcal{X}$;
   9. $\pi^{k+1}_{c_k}(x) \leftarrow \arg \max_{\alpha_{c_k}} T^{c_k}_{\Pi} V_k(x)$, $\forall x \in \mathcal{X}$;
10. $\Pi_{\leq c_k}^{k+1}(x) \leftarrow \Pi_{\leq c_k}^k(x)$;
end

3.2 Convergence

This section studies the properties of the CVI algorithm. The first main theorem below establishes convergence.

**Theorem 1. Convergence of CVI.** Consider the CVI Algorithm (Algorithm 1) for a MDP such that Assumptions 1, 2 and 3 are fulfilled. Then, there exists $\hat{V} \in \mathbb{R}^{|\mathcal{X}|}$ such that $V_k \to \hat{V}$ as $k \to \infty$.

The following lemmas will be used to prove Theorem 1. The first result will show boundedness of the value estimates.

**Lemma 2. Boundedness of Values.** The CVI iterates satisfy $\sup_{k \geq 0} \|V_k\|_\infty < \infty$.

**Proof.** Let $r = \sup_{x \in \mathcal{X}, \alpha \in A} r(x, \alpha)$.

\[
V_{k+1}(x) = \max_{\alpha_k} \sum_{x'} p(x'|x, \{\Pi_{\leq c_k}(x), \alpha_{c_k}\})(r(x, \{\Pi_{\leq c_k}(x), \alpha_{c_k}\}) + \gamma V_k(x')),
\]

\[
\leq r + \gamma \|V_k\|_\infty.
\]

Therefore $\|V_k\|_\infty \leq r \sum_{i=0}^{k-1} \gamma^i + \gamma^k \|V_0\|_\infty$. As $\gamma < 1$, $\lim_{k \to \infty} \|V_k\|_\infty \leq r/(1 - \gamma)$.

**Lemma 3. Monotonicity of the Clustered Bellman Operator.** $T_{\Pi}$ is monotone in the sense that if $V_1(x) \geq V_2(x)$ then $T_{\Pi} V_1(x) \geq T_{\Pi} V_2(x)$.

**Proof.** Note that the action here is fixed. Result follows from [59] Volume II, Lemma 1.1.1.

Now we proceed with the proof of Theorem 1.

**Proof.** First, induction will show that the sequence $\{V_k(x)\}_{k \geq 0}$ increases monotonically for all $x$. The base case inequality $V_0(x) \leq V_1(x)$ will shown first. The estimated values are initialized to $V_0(x) = 0$. The next value is evaluated as $V_1(x) = \max_{\alpha_c} \sum_{x'} p(x'|x, \{\Pi_{c}(x), \alpha_c\})(r(x, \{\Pi_{c}(x), \alpha_c\}) + \gamma V_0(x')) = \max_{\alpha_c} r(x, \{\Pi_{c}(x), \alpha_c\}) \geq V_0(x)$, hence the conclusion for any $c$ and $x$.

Next we will show that $V_{k+1}(x) \geq V_k(x)$. Note that preforming the optimization $T^{c_k}_{\Pi} V_k(x) = V_{k+1}(x)$ yields new action $\alpha_{c_k+1}$ for $x$. Letting $\Pi^{k+1} = \alpha_{c_k}$ in Eq. (15) means that $T_{\Pi^{k+1}} V_k(x) = V_{k+1}(x)$.

Consider $V_{k+1}(x) = T^{c_k}_{\Pi} V_k(x)$ with action maximizer $\alpha_{c_k+1}$ for $x$. If $\alpha_{c_k+1} = \alpha_c$, then this implies that $\alpha^{k+1} = \alpha^k$ and $V_{k+1}(x) = T_{\Pi} V_k(x)$. Since $V_k(s) = T_{\Pi} V_{k-1}(x)$, then by Lemma 3, $T_{\Pi} V_k(x) \geq T_{\Pi} V_{k-1}(x)$ for $V_k(x) \geq V_{k-1}(x)$ for all $x$. 


However if $\alpha_{c_k}^{k+1} \neq \alpha_{c_k}^k$, then the value can be bounded:

$$V_{k+1}(x) = T_{\Pi_{k+1}} V_k(x) = T_{\Pi_k}^c V_k(x),$$

$$= \max_{\alpha_c \in \mathcal{A}_c} \sum_{x' \in \mathcal{X}} p(x'|x, \{\Pi_{c_k}(x), \alpha_c\}) \left( r(x, \{\Pi_{c_k}(x), \alpha_c\}) + \gamma V_k(x') \right),$$

$$> \sum_{x' \in \mathcal{X}} p(x'|x, \Pi_k(x)) \left( r(x, \Pi_k(x)) + \gamma V_k(x) \right) = T_{\Pi_k} V_k(x).$$

Since $T_{\Pi_k} V_k(x) \geq T_{\Pi_{k-1}} V_{k-1}(x)$, then $T_{\Pi_{k+1}} V_k(x) \geq T_{\Pi_k} V_{k-1}(x)$ for all $x$.

By the boundedness property established by Lemma 4 and the above monotonicity of the sequence $\{V_k(x)\}_{k \geq 0}$, for all $x \in \mathcal{X}$, we conclude the convergence of $\{V_k\}$.

**Lemma 4. Contraction.** The clustered Bellman operator satisfies,

$$\max_{x \in \mathcal{X}} |T_{\Pi_1}^c V_1(x) - T_{\Pi_2}^c V_2(x)| \leq \gamma \max_{x \in \mathcal{X}} |V_1(x) - V_2(x)|.$$  \hspace{1cm} (16)

Note that Lemma 4 can be used to find a convergence rate for CVI, but it does not guarantee uniqueness of the fixed point $\lim_{k \to \infty} (T_{\Pi}^c)^k V(x)$ unless $c_1 = c_2$ and $\Pi_1 = \Pi_2$ for all $k$.

**Proof.** Note that:

$$\max_x f(x, y, z) - \min_x g(x, y, z) \leq \max_{x, y, z} |f(x, y, z) - g(x, y, z)|.$$ \hspace{1cm} (17)

Then,

$$|T_{\Pi_1}^c V_1(x) - T_{\Pi_2}^c V_2(x)|$$

$$= \left| \max_{\alpha_c \in \mathcal{A}_c} \sum_{x' \in \mathcal{X}} p(x'|x, \{\Pi_{c-1}(x), \alpha_c\}) \left( r(x, \{\Pi_{c-1}(x), \alpha_c\}) + \gamma V_1(x') \right) \right|$$

$$- \left| \max_{\alpha_c \in \mathcal{A}_c} \sum_{x' \in \mathcal{X}} p(x'|x, \{\Pi_{c-1}(x), \alpha_c\}) \left( r(x, \{\Pi_{c-1}(x), \alpha_c\}) + \gamma V_2(x') \right) \right|$$

$$\leq \left| \max_{\alpha_c \in \mathcal{A}_c} \sum_{x' \in \mathcal{X}} p(x'|x, \alpha_c) \left( r(x, \alpha_c) + \gamma V_1(x') \right) - \sum_{x' \in \mathcal{X}} p(x'|x, \alpha_c) \left( r(x, \alpha_c) + \gamma V_2(x') \right) \right|$$

$$= \max_{\alpha_c \in \mathcal{A}_c} \sum_{x' \in \mathcal{X}} p(x'|x, \alpha_c) \left| \gamma V_1(x') - \gamma V_2(x') \right|$$

$$\leq \max_{x \in \mathcal{X}} \gamma |V_1(x') - V_2(x'|)$$

where (18) uses (17).

With this convergence property established, it is clear that the CVI Algorithm may be used to find a usable control policy. In the next section, the performance of the found policy will be investigated.

### 3.3 Error Analysis

In this section we discuss the performance of the policy output by CVI. While Theorem 1 establishes convergence of CVI, it does not guarantee uniqueness or optimality of the found policy. In fact, the fixed point of the algorithm will depend on the initial policy guess and the cluster optimization order. This section will describe the performance of any fixed point found by the CVI algorithm.

**Theorem 2. Consistency.** The true optimal value $V^*$ is a fixed point of $T_{\Pi}^c$, for any $c \in \mathcal{C}$ and optimal policy $\Pi^*$.

**Proof.** Let $\Pi^*(x) = \alpha^*$ and $V^*(x)$ be a true optimal policy, optimal control for $x$, and optimal value of $x$ for $\mathcal{M}$. Now consider one iteration of CVI that optimizes with respect to cluster $c$. The next choice of $\alpha_c$ is $\alpha_c' = \arg \max_{\alpha_c \in \mathcal{A}_c} \sum_{x' \in \mathcal{X}} p(x'|x, \{\Pi^*(x), \alpha_c\}) \left( r(x, \alpha_c) + \gamma V^*(x) \right)$. If $\alpha_c' \neq \alpha^*_c$, then the choice $\alpha' = \{\alpha^*_1, \ldots, \alpha^*_c, \ldots, \alpha^*_n\}$ would yield a higher value than $\alpha^* = \{\alpha^*_1, \ldots, \alpha^*_c, \ldots, \alpha^*_n\}$. Therefore $\Pi^*(x) = \alpha^*$ is not a true optimal policy and control. This contradicts the assumption that $V^*(x)$ is the optimal value and that $\Pi^*(x) = \alpha^*$ are optimal policies and controls. This means that $T_{\Pi}^c V^*(x) = V^*(x)$.
While the consistency result shows that CVI is capable of finding the optimal answer, the next theorem provides a general performance bound that holds for every fixed point of the algorithm.

**Theorem 3. Suboptimality of Approximate Policy.** Consider the CVI Algorithm (Algorithm 1) for a MDP such that Assumptions 1, 2, and 3 are fulfilled. Let \( V^* \) be the true optimal value under optimal policy \( \Pi^* \), and let \( \hat{V} \) be a fixed point of CVI. Then,

\[
(UB) \| V^* - \hat{V} \| \leq \frac{1}{1 - \gamma} \| \hat{V} - T_{\Pi^*} \hat{V} \|,
\]

\[
(LB) \| V^* - \hat{V} \| \geq \frac{1}{1 + \gamma} \| \hat{V} - T_{\Pi^*} \hat{V} \|.
\]

**Proof.**

\[
\| V^* - \hat{V} \|_\infty = \| T_{\Pi^*} V^* - T_{\Pi^*} \hat{V} + T_{\Pi^*} \hat{V} - \hat{V} \|_\infty \\
\leq \| T_{\Pi^*} V^* - T_{\Pi^*} \hat{V} \|_\infty + \| T_{\Pi^*} \hat{V} - \hat{V} \|_\infty \\
\leq \gamma \| V^* - \hat{V} \|_\infty + \| T_{\Pi^*} \hat{V} - \hat{V} \|_\infty \\
\Rightarrow (1 - \gamma) \| V^* - \hat{V} \|_\infty \leq \| T_{\Pi^*} \hat{V} - \hat{V} \|_\infty \\
\Rightarrow \| V^* - \hat{V} \|_\infty \leq \frac{1}{1 - \gamma} \| T_{\Pi^*} \hat{V} - \hat{V} \|_\infty
\]

\[
\| \hat{V} - V^* \|_\infty = \| \hat{V} - T_{\Pi^*} \hat{V} + T_{\Pi^*} \hat{V} - V^* \|_\infty \\
\geq \| \hat{V} - T_{\Pi^*} \hat{V} \|_\infty - \| T_{\Pi^*} \hat{V} - T_{\Pi^*} V^* \|_\infty \\
\geq \| \hat{V} - T_{\Pi^*} \hat{V} \|_\infty - \gamma \| \hat{V} - V^* \|_\infty \\
\Rightarrow (1 + \gamma) \| V^* - \hat{V} \|_\infty \geq \| T_{\Pi^*} \hat{V} - \hat{V} \|_\infty \\
\Rightarrow \| V^* - \hat{V} \|_\infty \geq \frac{1}{1 + \gamma} \| T_{\Pi^*} \hat{V} - \hat{V} \|_\infty
\]

The right hand side of equations (19), (20) depend on the term \( \| \hat{V} - T^* \hat{V} \| \), which is a measure of the possible improvement under the full Bellman operator. Upon reaching a fixed point of CVI, a practitioner can either perform one evaluation of the full Bellman operator to check the bound, or use the fixed point of CVI as an initial guess for standard VI. Note that CVI can be considered as a type of approximate VI, and Theorem 3 matches the general results of approximate VI [59].

The structures of (19), (20) imply that a hybrid approach of combining CVI and VI style updates may be used to find optimal \( V^* \). This idea will be further explored in the next section.

### 3.4 Hybrid Approach

The structures of (19) and (20) imply that a hybrid approach may be used to find optimal \( V^* \) but with the computational savings of CVI; the error introduced by the computationally cheap CVI updates may be rectified with an occasional expensive VI update. This proposed method is outlined in Algorithm 2. From any value guess \( \hat{V}_k \), CVI updates are used to compute \( \hat{V}_{k+1} \). The subsequent VI update corrects the policy guess, enabling the next iteration to use policy that escapes any local optimum found by CVI.

One particularity here is how to choose the stopping conditions \( \delta \) and \( \epsilon \) such that the fewest number of full Bellman updates are run as possible. There is a risk that the true Bellman operator will repeatedly output the same policy; i.e. CVI had found the correct policy, but \( \epsilon \) was not chosen small enough to satisfy the convergence condition for \( \delta \). To avoid this scenario, we suggest choosing \( \epsilon \) an order of magnitude less than \( \delta \), and monitoring \( \Pi_k \) versus \( \Pi_{k+1} \) to check for improvement.
An alternate convergence criteria is to use \cite{19} to use $\hat{V}_{k+1}$ and $V_{k+1}$ to bound $\|\hat{V}_{k+1} - V^*\|$. If the distance of $\hat{V}$ to the optimum is sufficiently small, then the user may choose to terminate the Algorithm and not run anymore evaluations of the full Bellman operator.

**Algorithm 2: Hybrid CVI/VI**

1. Initialize values $V_0(x) \leftarrow 0, \forall x \in \mathcal{X}$;
2. Initialize policy guess $\Pi^0(x), \forall x \in \mathcal{X}$;
3. $k = -1$;
4. while $\|V_k - V_{k-1}\|_\infty > \delta$ do
   5. $k = k + 1$;
   6. $\hat{V}_{k+1}(x) \leftarrow \text{CVI}(V_k(x), \Pi_k(x), \epsilon), \forall x \in \mathcal{X}$;
   7. $V_{k+1}(x) \leftarrow T\hat{V}_{k+1}(x), \forall x \in \mathcal{X}$;
   8. $\Pi_{k+1}(x) \leftarrow \arg \max_\alpha T\hat{V}_{k+1}(x)$;
5. end

**Theorem 4. Convergence and Optimality of Hybrid CVI/VI.** Consider the Hybrid CVI/VI Algorithm (Algorithm 2) for a MDP such that Assumptions 2 and 3 are fulfilled. Then $V_k \to V^*$ as $k \to \infty$.

**Proof.** Convergence may once again be shown through monotonicity and boundedness of updates. $V_k(x) \leq \hat{V}_{k+1}(x)$ for all $x \in \mathcal{X}$ is already known for CVI updates. The next improvement $\hat{V}_{k+1}(x) \leq V_{k+1}(x)$ for all $x \in \mathcal{X}$ may be determined as the $T$ operator searches over the entire action space and will perform at least as well as a CVI update.

To see optimality, note that for any evaluation $T\hat{V}_k(x) = V_k(x)$, if $\hat{V}_k(x) = V_k(x)$ then the fixed point $V_k(x) = V^*(x)$ of $T$ has been achieved; else there will be monotonic improvement $V_k(x) \geq \hat{V}_k(x)$. As $V^*(x) \geq \hat{V}_k$ for all $k$ (due to $r(x, \alpha) \geq 0$ and $V_0(x) = 0$), $V_k(x) \to V^*(x)$ as $k \to \infty$. \hfill \Box

### 3.5 Extensions & Variants

The Bellman operator may be substituted with the clustered Bellman operator in a variety of dynamic programming approaches and variants given the clustered setting. Occasional VI updates may still be desirable to ensure optimal convergence, so a balance may set to achieve fast, accurate performance. In this section, we recap a few of the most immediate extensions and suggest how the clustered Bellman operator could be utilized.

**Value Function Approximation (VFA):** Recall that CVI was motivated to handle the expansion of the action space of the system. This may be combined with existing techniques to handle the combinatorial nature of the agent-based formulation, in particular, the scale of the state space.

Consider the VFA approach \cite{59}, which approximates the value function via some parametric class, such as a linear basis function. The value function may be approximated as $V = \Phi \theta$, where the parameter vector may be obtained by considering the fixed point $\Phi \theta = \Pi T(\Phi \theta)$ where $\Pi$ is the projection operator.

For a selected VFA construction, CVI-style updates \cite{14} may be used for the application of the Bellman operator. The resulting error between the fixed point $\hat{V}$ after both VFA and CVI and the optimal point after VFA $\overline{V}^*$ may be analyzed as,

$$\|\overline{V}^* - \hat{V}\| \leq \|\overline{V}^* - V^*\| + \|V^* - \hat{V}\| + \|\hat{V} - \overline{V}\|.$$  \hspace{1cm} (21)

Note that the three terms in equation (21) may be interpreted as the combined error from VFA and CVI. Terms 1 and 3 are the error from the projection operator, and term 2 may be found from \cite{19}. The resulting algorithm will preserve the computational savings from both VFA and CVI.

**Clustered Policy Iteration:** The PI algorithm may also make use of CVI updates to reduce computation in the policy improvement step. While monotonicity will still give convergence, the computation savings may not be significant. Recall that the per-iteration complexity of PI is $O(|A||X|^2 + |X|^3)$; the computation here will likely be dominated by the matrix inversion term $|X|^3$, meaning that the clustered Bellman operator may be used, but may not be the most impactful step for speed.
Approximate PI (API): In practice, API [59] is often used in lieu of PI or VI. Rather than alternating between full policy evaluation and full policy improvement, API only computes these steps up to some allowed error bounds. The results of Theorem 3 are similar to those developed for API as both techniques perform an approximate policy improvement step. The difference is that API has an additional step of approximately evaluating each proposed policy with \( V_k \approx V_{n_k} \). In CVI, policy evaluation is only performed after the found policy converges. The two techniques are similar in concept, but CVI is designed specifically for structure inherent to a cluster-based control policy.

The policy improvement step of API can be modified to be the cluster-wise maximization as in CVI. This means that API policy improvement bound \( \|T_{\Pi_k}V_k - TV^*\| \leq \epsilon \) measures the goodness of the maximization approximation in CVI.

Asynchronous Methods: General VI may be adapted such that value updates across the states are performed asynchronously. These techniques generally yield faster convergence and enable parallel computation. In particular, the Gauss-Seidel method is an adaptation of VI whereby the value for only one state is updated at each iteration. CVI-style updates may be directly used in conjunction with an asynchronous method. The clustered structure lends itself to distributed computation, as the optimization step across each entry \( \alpha_c \) may be performed by a different machine.

4 Separable Systems

The next section of this paper will focus on a special class of TI-MDPs that satisfy an additional structural property of separability in the reward function. It will be shown that this class is of particular interest because the reward structure guarantees that the CVI algorithm will find optimal policies.

Definition 9. The TI-MDP is called a separable system if the rewards are additive with respect to the cluster assignments:

\[
r(x, \alpha) = \sum_{c=1}^{C} r_c(x_c, \alpha_c)
\]

(22)

A more general version of this definition is additive rewards with respect to the agents, i.e. \( r(x, \alpha) = \sum_{n=1}^{N} r_n(x_n, \alpha) \). Note that rewards separable by agent are also separable by cluster. The rest of this section will presuppose that the system of interest can be modeled as a separable TI-MDP.

This category of MDPs occurs naturally arises in clustered control applications. For example, agents may fall into a customer type, and the central planner may define a reward based on the type. The reward per cluster may be the number of agents showing desirable behavior, i.e. \( r_c(x_c, \alpha_c) = \sum_{n \in D_c} \mathbb{1}(x_n \in D_c) \) where \( D_c \) is a set of desirable actions for agents of type \( c \). If the agent types are not apparent \textit{a priori}, the reward may represent the proportion of agents representing general good behavior, i.e. \( r(x, \alpha) = \frac{1}{N} \sum_{n \in N} \mathbb{1}(x_n \in D_h) \). This structure further lends itself to a distributed setting, where a full state need not be known to assign an agent reward.

Reward functions that are not separable depend on the whole state. For example, zero-sum games and common toy non-cooperative games such as the prisoner’s dilemma and chicken cannot be represented with separable reward functions, as the reward for one player can only be granted if the states for both players are known. Rewarding desirable states, i.e. \( r(x, \alpha) = \mathbb{1}(x \in D) \), will not be separable for arbitrary \( D \); however, structured \( D \) as previously described may be separable.

The next results discuss the performance of CVI on this special class of TI-MDPs. The first finding asserts that value functions of separable TI-MDPs are themselves separable.

Lemma 5. Separability of Bellman Operator on Separable Systems. Consider solving a separable TI-MDP with VI. At some iteration \( k \), the full Bellman operator can be expressed as the sum of \( C \) modified Bellman operators that each maximize over the action space for one cluster.

\[
V_{k+1}(x) = TV_k(x) = \sum_{c=1}^{C} T^c V_k(x) = \sum_{c=1}^{C} V_{k+1}(x_c).
\]

(23)

Where the local cluster values are defined iteratively as \( V_0(x_c) = 0 \) and,

\[
T^c V_k(x) = \max_{\alpha_c \in A_c} \mathbb{E} \left[ r_c(x_c, \alpha_c) + \gamma V_k(x'_c) \middle| x_c, \alpha_c \sim T_c \right],
\]

(24)

where \( T_c = p(x'_c | x, \alpha_c) \) is one of the factors of the factored transition matrix.
Note that any evaluation of the clustered Bellman operator requires a full state $x$ for proper conditioning, thus the local cluster value equation (24) is a function of $x$.

**Proof.** The proof will show by induction that each iteration of standard VI can be split into independent optimizations over the action space for each cluster.

**Base case.** Initialize $V_0(x) = 0$ for all $x$. Then $V_1(x) = r(x, \alpha) = \sum_{c=1}^{C} r_c(x_c, \alpha_c) = \sum_{c=1}^{C} V_1(x_c)$. Then,

$$V_2(x) = TV_1(x)$$

$$= \max_{\alpha \in A} \sum_{x' \in X} p(x'_1|x, \alpha_1) \ldots p(x'_C|x, \alpha_C) \left[ \sum_{c=1}^{C} r_c(x_c, \alpha_c) + \gamma r(x', \alpha_{k=1}) \right]$$

$$= \max_{\alpha \in A} \left[ \sum_{x'_1 \in X_1} p(x'_1|x, \alpha_1) \sum_{x'_2 \in X_2} p(x'_2|x, \alpha_2) \ldots \sum_{x'_C \in X_C} p(x'_C|x, \alpha_C) \left[ \sum_{c=1}^{C} r_c(x_c, \alpha_c) + \gamma r(x', \alpha_{k=1}) \right] \right]$$

$$= \max_{\alpha} \left[ E_{x'_1|\alpha_1} \ldots E_{x'_C|\alpha_C} \left[ \sum_{c=1}^{C} r_c(x_c, \alpha_c) + \gamma r(x', \alpha_{k=1}) \right] \right]$$

$$= \max_{\alpha} \left[ \sum_{c=1}^{C} \left[ E_{x'_1|\alpha_1} \ldots E_{x'_C|\alpha_C} \left[ r_c(x_c, \alpha_c) + \gamma r(x', \alpha_{k=1}) \right] \right] \right]$$

Notice that $r_c(x'_c)$ is independent of $x'_c$, where $c' \neq c$. The last line means that the resulting $V$ is decomposable over $c$, and that each element of $\alpha$ may be optimized independently.

**General case.**

$$TV_k(x)$$

$$= \max_{\alpha \in A} \sum_{x'} p(x'_1|x, \alpha_1) \ldots p(x'_C|x, \alpha_C) \left[ \sum_{c=1}^{C} r_c(x_c, \alpha_c) + \gamma V_k(x') \right]$$

$$= \max_{\alpha \in A} \left[ E_{x'_1|\alpha_1} \ldots E_{x'_C|\alpha_C} \left[ \sum_{c=1}^{C} r_c(x_c, \alpha_c) + \gamma \sum_{c=1}^{C} V_k(x'_c) \right] \right]$$

$$= \max_{\alpha} \left[ \sum_{c=1}^{C} \left[ E_{x'_1|\alpha_1} \ldots E_{x'_C|\alpha_C} \left[ r_c(x_c, \alpha_c) + \gamma V_k(x'_c) \right] \right] \right]$$

The last line shows that $V$ may again be decomposed for general $k$ and that $\alpha$ may be optimized over each element independently. Therefore, each step of general VI may be expressed as $C$ CVI evaluations of the current value. □
Lemma 5 shows that each Bellman update may be written as a sum over clusters, where each addend is only dependent on one element of the control vector, $\alpha_c$. With this decomposition, the next theorem establishes that CVI applied to separable systems will converge to the true optimal value.

**Theorem 5. Optimality of CVI on Separable Systems.** Consider the CVI Algorithm for a separable TI-MDP. The CVI algorithm will converge to the true optimal value, $\hat{V} = V^*$.

**Proof.** First it will be shown that local updates on separable systems as in (24) converge to the unique optimal value, and then it will be shown that (24) performs the same optimization as CVI.

As shown in the proof of Lemma 5, a VI update for a separable system can be written as $TV_k(x) = \sum_{c=1}^C T^c V_k(x)$. Therefore,

$$V^*(x) = \lim_{k \to \infty} TV_k(x) = \lim_{k \to \infty} T^k V_0(x) = \lim_{k \to \infty} \sum_{c=1}^C (T^c)^k V_0(x) = \sum_{c=1}^C V^*(x_c).$$

(25)

Next, it will be shown that (24) performs the same optimization as CVI.

Consider a sequence of local cluster values $\{U(x_c)\}_{k \geq 0}$ for some cluster $c$, except now instead of being optimized at each time step as in (24), it will be optimized one time out of every $C$ time steps as in CVI.

Let us construct $\{U(x_c)\}_{k \geq 0}$. Let $b_c$ be the index of cluster $c$ in $\Omega$; i.e. $c$ is optimized whenever $k \mod C = b_c$. Then, the value function $V(x)$ at time $b_c$ is $U_{b_c} = \sum_{c=1}^C \overline{T}^c U_0(x)$ which is $b_c - 1$ evaluations of the Bellman operator with initial guess policy $\pi$.

Next, define the operator $\overline{T}^c U(x) = \overline{T}_c \overline{T}_{c-1} \overline{T}_c U(x)$ where $\overline{T}^c$ performs the maximization step and produces candidate $\alpha_c$, the value that evaluated $C-1$ times on the resulting cluster value. Note that $\overline{T}^c$ is a contraction (with parameter $\gamma C$), so taking the limit of $k \to \infty$ of $(\overline{T}^c)^k U(x)$ will converge to a unique fixed point. Any additional evaluations of $\overline{T}_c$ after convergence of $\overline{T}^c$ will leave the final value unchanged, and the $b_c - 1$ applications of $\overline{T}_c$ to $U_{b_c}$ (with arbitrary $\pi$) before $\overline{T}$ will change the initial guess to $U_{b_c}$, but will not change the location of the convergence point. Thus,

$$\lim_{k \to \infty} \sum_{c=1}^C (\overline{T}^c)^k U_0(x) = \lim_{k \to \infty} \sum_{c=1}^C (\overline{T}^c)^k U_{b_c}(x) = \sum_{c=1}^C U^*(x_c) = \sum_{c=1}^C V^*(x_c).$$

(26)

This key result shows that an optimal cluster-based policy for any separable TI-MDP may be found efficiently with CVI, enjoying all the complexity savings. While in general the clustered Bellman operator and CVI is suboptimal, optimality is guaranteed for the class of separable TI-MDPs, also known as factored MDPs. For these problems, the user may enjoy both the value improvement from the clustered control policy and the complexity savings of CVI without sacrificing loss in performance of the found policy.

## 5 Choosing Cluster Assignments

Leading up to this section, it has been assumed that the cluster assignments are known a priori. This is a reasonable assumption for applications where classes of agents are apparent or existing; i.e. category of customer or type of robot. As seen in the definition of separability, classes of agents may be defined in terms of similarity of central control objectives. For example, clusters may be defined such that their reward functions are separable, thus enabling optimality of CVI.

For general multi-agent MDPs, however, the clustered control policy may be used simply to increase the received value without relying on intuition about existing agent classes or objectives. The objective in this section is to find a clustering assignment that yields good value when none is suggested a priori.

**Definition 10.** An optimal $C$ clustering assignment for a TI-MDP is $C^*$ such that the resulting optimal value is maximal;

$$C^* \in \arg \max_C V^*(x)$$

s.t. $|C| = C, C_1 \cup \cdots \cup C_C = \mathcal{N}, C_u \cap C_v = \emptyset \forall u \neq v, u, v \in [1, C]$.  

(27)
Note that (27) is a discrete optimization problem that requires $O(C^N)$ evaluations if solved by enumeration. In general, this problem is known to be NP-hard [48].

In this section, an approximate iterative solution to the optimization problem (27) will be explored based on ideas of submodularity. Let $V(C_k)$ be a shorthand to mean the vector of values associated with clustering assignment $C$ that consists of $k$ clusters. Let $C^*_k$ be the optimal clustering assignment of $k$ clusters, and let $\hat{C}_k$ be the clustering assignment of $k$ clusters as found by an approximate algorithm. All discussion will be for separable TI-MDPs, both to take advantage of the additional structure of the reward function, and because CVI is optimal for this class of problems.

The first useful definition is for submodular functions, which is a notion often used in discrete optimization.

Definition 11. Given a finite set $E$, a submodular function $f : 2^E \rightarrow \mathbb{R}$ is such that for every $A \subseteq B \subseteq E$ and $i \in E \setminus B$,

$$f(A \cup \{i\}) - f(A) \geq f(B \cup \{i\}) - f(B).$$  \hspace{1cm} (28)

Furthermore, $f$ is monotone if $f(B) \geq f(A)$.

Submodularity is a useful property for set optimization as it means the benefit of adding more elements to a set decreases as the set grows in size; i.e. diminishing returns. Maximizing a submodular function subject to a dimensionality constraint, for example, means that the possible improvement between the $k$ and $k + 1$ indices is upper bounded.

5.1 Submodularity of Value Functions

It can be shown that the value function for separable TI-MDPs is a submodular function. With the following result, the optimization problem (27) may be approached with techniques developed for submodular function.

Lemma 6. Submodularity of TI-MDP. Consider a TI-MDP with $r(x, \alpha)$ submodular with respect to the number of agents. Then the value function $V(x)$ is also submodular with respect to the number of agents. More specifically, for sets of agents $A \subseteq B \subseteq N$ and agent $n \in N \setminus B$,

$$V^{A \cup \{n\}}(x) - V^A(x) \geq V^{B \cup \{n\}}(x) - V^B(x),$$ \hspace{1cm} (29)

where $V^D(x)$ is the value function of the MDP comprised of the agents in set $D$.

Proof. $V(s) = \lim_{k \to \infty} T^k V_0(x)$, so it suffices to show that (29) holds for any $k$th application of the Bellman operator.

$$V^A_1(x) = r(x_A, \alpha); \quad V^{A \cup \{n\}}_1(x) = r(x_{A \cup \{n\}}, \alpha);$$

$$V^B_1(x) = r(x_B, \alpha); \quad V^{B \cup \{n\}}_1(x) = r(x_{B \cup \{n\}}, \alpha).$$

As $r(x, \alpha)$ is given to be submodular, the claim will hold for $k = 1$.  

15
Now show for general $k$.

\[
TV^{B,n}_k(x) - TV^B_k(x) \\
= \sum_{x_{B\cup\{n\}}} p(x'_{B\cup\{n\}} | x, \alpha) [r(x_{B\cup\{n\}}, \alpha) + \gamma V^{B,n}_{k-1}(x')] - \sum_{x'_B} p(x'_B | x, \alpha) [r(x_B, \alpha) + \gamma V^B_{k-1}(x')]
\]

= \sum_{x'_B} [\sum_{x_{B\cup\{n\}}} p(x'_{B\cup\{n\}} | x, \alpha) [r(x_{B\cup\{n\}}, \alpha) + \gamma V^{B,n}_{k-1}(x')] - [r(x_B, \alpha) + \gamma V^B_{k-1}(x')]]

= \sum_{x'_B} \sum_{x'_{A\cup\{n\}}} p(x'_{A\cup\{n\}} | x, \alpha) [r(x_{A\cup\{n\}}, \alpha) - r(x_B, \alpha) + \gamma (V^{A,n}_{k-1}(x') - V^A_{k-1}(x'))]

\leq \sum_{x'_B} \sum_{x'_{A\cup\{n\}}} p(x'_{A\cup\{n\}} | x, \alpha) \sum_{x'_{B\cup\{n\}}} p(x'_{B\cup\{n\}} | x, \alpha) [r(x_{A\cup\{n\}}, \alpha) - r(x_B, \alpha) + \gamma (V^{A,n}_{k-1}(x') - V^A_{k-1}(x'))]

= TV^{A,n}_k(x) - TV^A_k(x)

where the inequality holds due to submodularity of the reward function and the induction hypothesis.

By induction, \(TV^{A,n}_k(x) - TV^A_k(x) \geq TV^{B,n}_k(x) - TV^B_k(x)\) for all $k$, so take the limit as $k \to \infty$ to show that the claim holds.

The reward function for an agent-separable TI-MDP can be written in the form $r(x, \alpha) = \sum_n r_n(x_n, \alpha_n)$. Linear functions are modular (submodular), so $r(x, \alpha)$ is submodular and therefore will satisfy the requirements for Lemma 6.

**Lemma 7. Monotonicity of TI-MDPs:** Consider a TI-MDP with $r(x, \alpha)$ monotone with respect to the number of agents. Then the value function $V(x)$ is also monotone with respect to the number of agents.

**Proof.** This again can be shown via induction.

\[
V^B_k(x) - V^A_k(x) = r(x_B, \alpha) - r(x_A, \alpha) \geq 0
\]

which holds by the given monotonicity.

Induction on $k$:

\[
TV^B_k(x) - TV^A_k(x) \\
= \sum_{x'_A} p(x'_{B\setminus A} | x, \alpha) \sum_{x'_{B\cup\{n\}}} p(x'_{B\cup\{n\}} | x, \alpha) [r(x_{B\cup\{n\}}, \alpha) + \gamma V^{B,n}_{k-1}(x')] - r(x_B, \alpha) + \gamma V^A_{k-1}(x')]
\]

= \sum_{x'_A} p(x'_{B\setminus A} | x, \alpha) \sum_{x'_{B\cup\{n\}}} p(x'_{B\cup\{n\}} | x, \alpha) [r(x_{B\cup\{n\}}, \alpha) - r(x_B, \alpha) + \gamma (V^{B,n}_{k-1}(x') - V^A_{k-1}(x'))] \geq 0

By induction, $TV^B_k \geq TV^A_k$ for all $k$, so therefore $\lim_{k \to \infty} V^B_k(x) \geq \lim_{k \to \infty} V^A_k(x)$ and $V^B(x) \geq V^A(x)$.

The results in this section establish that properties of submodularity may be used to analyze separable TI-MDPs. This structure will help find methods of solving the clustering assignment problem with theoretical guarantees.
5.2 Greedy Clustering

With the notions of submodularity established, we return to the optimal clustering problem. The optimization problem can be restated as,

$$\max_C \sum_{c=1}^{C} V^*(x_c),$$

s.t. $|C| = C, C_1 \cup \cdots \cup C_C = \mathcal{N}, C_u \cap C_v = \emptyset \forall u \neq v, u, v \in [1, C]$.

As discussed in the introduction, (30) can equivalently be represented as an optimization problem subject to a partition matroid constraint, for which there exists a $(1 - 1/e)$ approximate algorithm [48]. The presented method creates a continuous approximation of the discrete objective function via random sampling, and then rounds the solution of the continuous problem to a feasible point in the matroid. While this method provides a close approximation to the optimal, it was not formulated specifically for MDP value functions and requires a complicated implementation for the approximation subroutines. In this section we will explore a more simple construction that builds upon the optimality of CVI queries on separable systems.

In [46], Zhao et al. suggest a Greedy Splitting Algorithm (GSA) to solve (30) for the minimization case that achieves an approximation $f(\hat{C}) \leq (2 - \frac{2}{e}) f(C)$ for monotonone submodular $f$ in $O(kN^3\theta)$ time where $\theta$ is the time to query $f$. This approach begins with all the agents as one cluster, and then takes for some desired number of clusters $k$, the best possible refinement of the clusters determined for $k-1$. Note that this method only requires computation of $V$ for some clustering configuration, which is a perfect use case for CVI. However, adapting GSA to a maximization problem loses the error bound guarantees due to non-negativity constraints. In general, the maximization of submodular functions subject to a cardinality constraint is a much harder problem than minimization [48], and includes famous problems such as the maximum coverage problem.

Nevertheless, in this paper we suggest adapting GSA for value maximization of cluster assignments because it guarantees value improvement with each split, and because it only requires calls to a CVI subprocess. Recall the original motivations for cluster-based control: (1) a priori constraints or agent types, and (2) to achieve better value from more sophisticated controls. This section focuses on the second motivation; a simple method like greedy splitting is easy to implement, only needing the existing resources to calculate $V$, and gives monotonically increasing values for increasing $k$. A maximization version of GSA is presented in Algorithm 3, and it will be shown that the algorithm finds clustering assignments whose values form a submodular lower bound of the optimal values of (30).

In general, the term $C$ will be used to refer to the desired final number of clusters, and $k$ will refer to the number clusters at an intermediate step. The initialization defines one cluster of all the agents. At each subsequent query for new clusters, the algorithm searches for some existing cluster $U$ and a split $\{U - X, X\}$ that provides the most value improvement. This is repeated until $C$ clusters are achieved and the final value is returned. The number of possible sets formed by splitting $n$ elements split into $k$ disjoint subsets may found by the Stirling number of the second kind $S(n, k)$. Therefore the number of possible splits across some clustering assignment is $\sum_{C \in \hat{C}} S(|\hat{C}|, 2)$.

**Algorithm 3:** Greedy Splitting Algorithm - Reward

1. $\hat{C}_1 = \{N\}$;
2. $\hat{V}_1 = \text{CVI}(\hat{C}_1)$;
3. for $k \in \{2, \ldots, C\}$ do
   4. $(X_k, U_{k-1}) \leftarrow \text{argmax} \{V(X) + V(U - X) - V(U) \mid \emptyset \subset X \subset U, U \in \hat{C}_{k-1}\}$;
   5. $\hat{C}_k \leftarrow \{\hat{C}_{k-1} - U_{k-1}\} \cup \{X_k, U_{k-1} - X_k\}$;
   6. $\hat{V}_k = \text{CVI}(\hat{C}_k)$;
7. end

**Remark 1.** The optimal values for an optimal $k$-clustering is lower bounded by the optimal value for a clustering found by GSA-R.

$$V^*(\hat{C}^*_k) \geq V^*(\hat{C}_k)$$

The clustering assignments found by GSA-R are optimal for $k = 1, 2, N$, i.e. $\hat{C}_1 = C_1, \hat{C}_2 = C_2, \hat{C}_N = C_N$. For the case $k = 2$, this can be seen due to the fact that optimizing over the set of 2-cluster assignments is equivalent to optimizing over one split.
**Lemma 8. Monotonicity of Splitting.** Consider a TI-MDP with a $k$-clustering assignment. The optimal value $V^*(\mathcal{C}_{k+1})$ will at least as much as $V^*(\mathcal{C}_k)$ if the $k + 1$ cluster is formed by splitting one of the existing $k$ clusters.

**Proof.** Proof by induction.

Base case: Assume there are $k = 2$ clusters, $c_1$ and $c_2$. The value of a state under $\pi^*$ is,

$$V^*_{k=2}(x) = \max_{\alpha_1, \alpha_2} \sum_{x' \in X} p(x'|x, \{\alpha_1, \alpha_2\})(r(x, \{\alpha_1, \alpha_2\}) + \gamma V^*(x')).$$

If $\alpha_1 = \alpha_2$, then this is equivalent to the one-cluster case and the same value is recovered $V^*_{k=2}(x) = V^*_{k=1}(x)$.

Else if $\alpha_1 \neq \alpha_2$ then,

$$V^*_{k=2}(x) = \max_{\alpha_1, \alpha_2} \sum_{x' \in X} p(x'|x, \{\alpha_1, \alpha_2\})(r(x, \{\alpha_1, \alpha_2\}) + \gamma V^*(x')) \geq \max_{\alpha} \sum_{x} p(x'|x, \{\alpha\})(r(x, \{\alpha\}) + \gamma V^*(x')) = V^*_{k=1}(x).$$

Inductive step: Say there are $k'$ clusters and that the controller uses policy $\Pi^*$. Take cluster $c'$ and split it into two clusters, $\{c'_1, c'_2\}$ where $c'_1 \cup c'_2 = \emptyset$ and $c'_1 \cap c'_2 = c'$. This forms $k'+1$ clusters. Again if $\alpha_{c'_1} = \alpha_{c'}$ and $\alpha_{c'_2} = \alpha_{c'}$, then we recover the same value $V^*_{k'}(x) = V^*_{k'+1}(x)$. However if $\alpha_{c'_1} \neq \alpha_{c'}$ or $\alpha_{c'_2} \neq \alpha_{c'}$, then,

$$V^*_{k'+1}(x) = \max_{\alpha_1, \ldots, \alpha_{c_1}, \alpha_{c_2}} \sum_{x'} p(x'|x, \{\alpha_1, \ldots, \alpha_{c'_1}, \alpha_{c'_2}\}) \times (r(x, \{\alpha_1, \ldots, \alpha_{c'_1}, \alpha_{c'_2}\}) + \gamma V^*(x')) \geq \max_{\alpha_1, \ldots, \alpha_{c'_1}, \alpha_{c'_2}} \sum_{x'} p(x'|x, \{\alpha_1, \ldots, \alpha_{c'_1}, \alpha_{c'_2}\}) \times (r(x, \{\alpha_1, \ldots, \alpha_{c'_1}, \alpha_{c'_2}\}) + \gamma V^*(x')) = V^*_{k'}(x).$$

This result on the splittings further shows that the resulting values for the optimal clustering problem monotonically improve for increasing $k$.

**Lemma 9. Monotonicity of Optimal Clusterings.** Consider a TI-MDP. The series of optimal values for each optimal clustering assignment is a monotonically increasing sequence:

$$V^*(\mathcal{C}_1^*) \leq \cdots \leq V^*(\mathcal{C}_k^*) \leq \cdots \leq V^*(\mathcal{C}_{n}^*). \quad (31)$$

**Proof.** By Lemma 8, $V^*(\mathcal{C}_k^*) \leq V^*(\mathcal{C}_{k+1})$ where $\mathcal{C}_{k+1}$ is a split of $\mathcal{C}_k$. Then $V^*(\mathcal{C}_{k+1}) \leq V^*(\mathcal{C}_{k+1})$ and the result holds.

Note that Lemmas 8 and 9 hold for non-separable TI-MDPs. For the separable case, we can find better performance guarantees in that the values have diminishing returns for increased numbers of clusters.

**Theorem 6. Performance of GSA-R:** Consider a TI-MDP with $r(x, \alpha)$ submodular with respect to the number of agents. The sequence of $\mathcal{C}_k$ found by GSA-R satisfies,

$$V^*(\mathcal{C}_{k+1}) - V^*(\mathcal{C}_k) \leq V^*(\mathcal{C}_k) - V^*(\mathcal{C}_{k-1}).$$

**Proof.** In this proof, all value functions are assumed to be evaluated for the optimal policy, but the superscript notation is omitted.

**Base Case:** We need to show $V(\mathcal{C}_2) - V(\mathcal{C}_1) \leq V(\mathcal{C}_1) - V(\mathcal{C}_0)$, but for zero clusters $V(\mathcal{C}_0) = 0$, so we need to show $V(\mathcal{C}_2) - V(\mathcal{C}_1) \leq V(\mathcal{C}_1)$. Denote the set $\mathcal{C}_2 = \{U, W\}$. By Lemma 4, $V(\mathcal{C}_1) \geq V(U)$ and $V(\mathcal{C}_1) \geq V(W)$. Thus $V(\mathcal{C}_2) = V(U) + V(W) \leq 2V(\mathcal{C}_1)$ as desired.

**Induction:** The goal is to show that $V(\mathcal{C}_{k+1}) - V(\mathcal{C}_k) \leq V(\mathcal{C}_k) - V(\mathcal{C}_{k-1})$. 

18
Let $U_k \in \hat{C}_k$ be the cluster that is split into $X_{k+1}$ and $Y_{k+1}$ to form $\hat{C}_{k+1}$, i.e. $X_{k+1} \cup Y_{k+1} = U_k$, $X_{k+1} \cap Y_{k+1} = \emptyset$. There are two cases: either $U_k \subset U_{k-1}$ or $U_k \not\subset U_{k-1}$.

\[ V(\hat{C}_{k+1}) - V(\hat{C}_k) = V(X_{k+1}) + V(Y_{k+1}) - V(U_k) \]

If $U_k \not\subset U_{k-1}$ then optimally splitting $U_{k-1}$ provided a bigger value increase than optimally splitting $U_k$. This is because both sets were available to select to split at stage $k - 1$, but the greedy algorithm selected $U_{k-1}$. Thus $V(X_k) + V(Y_k) - V(U_{k-1}) \geq V(X_{k+1}) + V(Y_{k+1}) - V(U_k)$ and therefore $V(\hat{C}_k) - V(\hat{C}_{k-1}) \geq V(\hat{C}_{k+1}) - V(\hat{C}_k)$.

In the other case, $U_{k-1}$ is split, and then $X_k$ or $Y_k$ are selected as the next $U_k$. Say that $X_k = U_{k-1}$. Need to show:

\[ V(X_{k+1}) + V(Y_{k+1}) - V(X_{k+1} \cup Y_{k+1}) \leq V(X_k) + V(Y_k) - V(X_k \cup Y_k) = V(X_{k+1} \cup Y_{k+1}) + V(Y_k) - V(X_{k+1} \cup Y_{k+1} \cup Y_k) \]

By submodularity,

\[ V(X_{k+1} \cup Y_{k+1}) - V(X_{k+1}) - V(Y_{k+1}) \geq V(X_{k+1} \cup Y_{k+1} \cup Y_k) - V(X_{k+1} \cup Y_k) - V(Y_{k+1}) \]

Therefore to claim (32), need,

\[ V(X_{k+1} \cup Y_{k+1}) + V(Y_k) \geq V(X_{k+1} \cup Y_k) + V(Y_{k+1}) \]

Recall that $X_{k+1} \cup Y_{k+1} \cup Y_k = U_{k-1}$. The above equation shows the value of two possible splittings of $U_{k-1}$: the LHS is the split \{ $X_{k+1} \cup Y_{k+1}, Y_k$ \} and the RHS is the split \{ $X_{k+1} \cup Y_k, Y_{k+1}$ \}. As the algorithm selects the best splits in a greedy fashion, the LHS is a better split and thus has a greater value. Thus (32) holds, and so does the original claim.

Theorem 6 shows that GSA-R finds a series of clustering assignments whose optimal values are a submodular lower bound of the optimal values of optimal clustering assignments. In addition, this bound is tight at the endpoints. This bound justifies the search method of GSA-R; it shows that each iteration of $k \rightarrow k + 1$ clusters improves the value, and that the amount of improvement will decrease at each subsequent iteration.

**Remark 2.** This diminishing returns property may be used as a stopping criterion when a desired final number of clusters is not known a priori. For example, consider a clustering $\hat{C}_k$ with known $V^*(\hat{C}_k)$. The clusterings at the previous iterations are known, so $V^*(\hat{C}_{k-1})$ is known and thus $V^*(\hat{C}_k) - V^*(\hat{C}_{k-1}) = \delta_k$ is known. By Theorem 6, $\delta_k \geq \delta_{k+1}$. Therefore, if $\delta_k$ is sufficiently small, the user may decide to terminate the algorithm and not compute $\hat{C}_{k+1}$.

### 6 Examples

In this section the CVI and GSA-R algorithms are explored in simulation. In the first example, the results and computation times of VI versus CVI are studied, as well as how CVI performs within our established error bound. The second example looks at the GSA-R technique, and demonstrates its computation savings versus a naive approach. Finally, an applied example where agents assign themselves to channels subject to bandwidth and costs constraints is presented.

#### 6.1 VI vs CVI Examples

Figures 1 and 2 give the results of the CVI algorithm as compared to VI on non-separable and separable systems, respectively. In both cases, a system of $N = 7$ agents was considered, where each agent selected between binary choices at each time step. With the MDP states defined as the joint strategies of the system, the overall state space had a size of $2^7 = 128$. The controller was given $M = 3$ actions that may be assigned to any particular cluster, leading to an action space of size $3^C$ where $C$ is the number of clusters. Appropriate transition matrices were randomly generated, as well as deterministic non-separable and separable state reward functions of the form $r(x, \alpha) \equiv r(x)$.

The graphs show the value $\|V^*\|$ averaged over every possible clustering assignment for a fixed $C$ and normalized by the maximum possible value when $C = 7$. The basic case, where $C = 1$, is plotted across each $C$ to show the increase in value attained by using cluster-based control. The black dot on each bar shows the $\|\hat{V}\|$ achieved via CVI, again averaged across all clustering assignments and normalized by the maximum optimal value. The CVI value is shown with the minimum and maximum error bounds as found by (19) and (20).
Figure 1: Performance of cluster-based control policies on a non-separable system for different $C$. Note that cluster-based control improves the attained value, and CVI is able to approximate the true optimal value with significant computation time savings.

Figure 2: Performance of cluster-based control policies on a separable system for different $C$. Note that in for TI-MDPs with separable reward functions, CVI recovers the true optimal value exponentially faster than standard VI.
These results exemplify the performance of CVI as compared to VI. Clearly Figure 2 shows that CVI recovers the same values as VI for separable systems. In the general case, Figure 1 shows that $\|V^*\|$ is consistently in the best side of its error range, being close to the true $\|V^*\|$.

The red squares show the ratio of the wall clock computation time for VI and CVI. The simulations were written in MATLAB, with the only difference in implementation being that VI optimized over the whole action space at each iteration, whereas CVI optimized over a specified subset of actions. These results confirm that the computation time of VI scales exponentially with the number of clusters, which is an increase not seen with CVI. For example, consider the non-separable case where $C = 7$. CVI converged about 620x faster than VI. While the computational savings is primarily reflected in the per-iteration time, it is reasonable to consider that higher $C$ means that more total iterations are needed. In fact, CVI for $C = 7$ was only about 1.2x slower than CVI for $C = 1$, whereas VI for $C = 7$ was about 825x slower than VI for $C = 1$.

Finally, the blue squares show the ratio of the wall clock computation for VI and hybrid CVI/VI, which recovers the true optimal value. The stopping conditions used were $\delta = 1e-5$, $\epsilon = 1e-6$. The average number of VI calls used in the hybrid implementation is shown in Figure 3. Note that including even a few evaluations of the true Bellman operator significantly slows the computation time versus standalone CVI. The wall clock time of hybrid CVI/VI scaled approximately polynomially with respect to the number of clusters, whereas CVI scaled quadratically. A practitioner can therefore recover the optimal value of a non-separable TI-MDP with some computational time savings with that hybrid CVI/VI approach.

### 6.2 Greedy Clustering

This next example explores the utility of selecting clustering assignments with the GSA-R algorithm. Figure 4 shows the results from a randomly generated separable TI-MDP constructed with $N = 10$ agents for a state space of size $2^{10} = 1024$. The bars show $\|V^*\|$ (again normalized by the maximum where $C = 10$) produced by clustering assignments selected by the greedy splitting technique. Note that the values display the diminishing returns property as more clusters are considered, thus demonstrating the results of Theorem 6.

The black dots here demonstrate the ratio of the number of clustering assignments checked by a naive brute-force method versus using the greedy splitting technique. In the naive method, every possible assignment of $N$ agents split into $C$ clusters is evaluated, which leads to $S(N, C)$ possible combinations. For GSA-R, only refinements of the previous clustering assignment are considered; for $C = 3, \ldots, 8$ this leads to a reduction in the number of evaluations. For the $C = 9, 10$ cases, however, solving directly via the naive method is more efficient as $S(N, C)$ is small and GSA-R requires iterating from $k = 1, 2, \ldots, C$. The resulting curve corroborates intuition that the greatest number of possible clustering assignments is achieved at $C = N/2$, where in this case GSA-R evaluated about 70x fewer cases yet achieved more than 99% of the maximum value. The results of this simulation imply that for applications where using $C = N$ is not feasible either computationally or in deployment, GSA-R is an effective technique for finding a clustering assignment that achieves a good optimal value.

### 6.3 Channel Assignment Example

In this section we consider a scenario where agents attempt to assign themselves to different channels subject to bandwidth constraints and costs. This scenario will exemplify how a MAS, described as a game, can be abstracted to a MDP framework and controlled with CVI and GSA-R techniques.

Let there be three channels of low, medium, and high bandwidths (20, 50, 100). Each agent in the set of $N = 6$ agents selects a channel to use at each time step. The agents evaluate each channel with a utility function of the form,

$$u_n(x) = \frac{b(x)}{1 + \sum_{n' \neq n} I(x_{n'} = x)} - \beta_n \nu(x)$$

where $b(x)$ is the bandwidth of channel $x$, $\nu(x)$ is the cost of using the channel, and $\beta_n$ is a scaling factor for each $n$ chosen randomly from $[0, 1]$. This utility function can be interpreted as the effective bandwidth for $n$ had they chosen $x$ minus a scaled cost to use the channel. All agents are assumed to use a best-response style decision process, leading
to a large sparse transition matrix describing 729 states. The cost structure, as chosen by the controller, can be one of the following structures:

| $\nu(x, \alpha)$ | Low BW | Med BW | High BW |
|-------------------|--------|--------|---------|
| $\alpha_1$        | 0      | 10     | 30      |
| $\alpha_2$        | 1      | 15     | 50      |
| $\alpha_3$        | 10     | 50     | 100     |

Consider two scenarios for the controller.

**Scenario 1: Maximum Revenue** Here the controller aims to maximize their received payments from the agents. The overall reward to the controller is thus $r(x, \alpha) = \sum_{n \in N} \nu(x_n, \alpha_n)$ which is clearly agent-separable. Note that this reward function depends on both $\alpha$.

**Scenario 2: Desired Configuration** This scenario represents the case where the controller wants a specific action configuration. In this example, the controller wants all the agents to choose the medium bandwidth channel. Their reward function is thus $r(x, \alpha) = \sum_{n \in N} 1(x_n = \text{medium})$, which again is agent-separable.

These scenarios were solved and the results are shown in Figure 5. Note that each row is normalized by the maximum value achieved when $C = 6$. First, we immediately notice that introducing clusters improves the performance. As the reward structure is separable, we can use CVI to solve for optimal control policies given a clustering assignment chosen by GSA-R. Second, these examples demonstrate scenarios in which agent-separable reward functions enable desirable outcomes. Finally, note that GSA-R implies a natural stopping criterion. In practice the controller can observe the negligible value improvement from two to three clusters (in either case), and then from the diminishing returns property conclude that the policy for three clusters achieved near-optimal performance.

|\|\|\|\|\|\|
|----------------|--------|--------|--------|--------|--------|
|\|\|\|\|\|\|

Figure 5: Results of the channel assignment example using CVI and GSA-R on two different separable reward functions.

7 Conclusion

In this work we examined cluster-based control techniques on a multi-agent system to achieve behavioral objectives. A collective of agents, whose behavior may be learned or abstracted from a known model, can be represented as a
transition-independent Markov decision process under mild assumptions. While the classic setup has the controller assign an action to the system, better performance may be achieved by partitioning agents into groups and giving each group its own action. This cluster-based approach can greatly increase the attained value, however solving through standard techniques such as value iteration and policy iteration become prohibitively slow due to huge action space. This work details a solution in clustered value iteration, which takes a “round-robin” approach to policy optimization. This algorithm’s convergence properties are studied, and it is shown that the true optimal may be attained in systems whose reward functions have a separable quality.

The second half of this work examined techniques for optimally clustering agents. An iterative greedy splitting technique is proposed as a desirable approach, as it provides monotonic, submodular improvement in value, and requires significantly less computation than a naive approach. Examples of both clustered value iteration and the greedy splitting clustering are explored in simulation under a variety of scenarios.

While inspired by multi-agent systems, this formulation has relevance to other reinforcement learning problems, specifically in the case of complicated action spaces. These techniques may be layered with approaches like value function approximation for the combined benefits for handling large state and action spaces. Future work can study scalability of both spaces subject to large systems of agents, as well as more sophisticated techniques to choose cluster configurations.

Acknowledgment

This material is based upon work supported by the National Science Foundation Graduate Research Fellowship Program under Grant No. DGE1745016. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation. In addition, the authors would like to acknowledge support from the Hsu Chang Memorial Fellowship in ECE.

References

[1] S. Hendrix. (2016) Traffic-weary homeowners and waze are at war, again. guess who’s winning? [Online]. Available: https://www.washingtonpost.com/local/traffic-weary-homeowners-and-waze-are-at-war-again-guess-whos-winning/2016/06/05/c466df46-299d-11e6-b989-4e5479715b54_story.html

[2] L. Plaugic. (2015) Miami cops are falsely reporting their locations on waze to trick drivers. [Online]. Available: https://www.theverge.com/2015/2/12/8025209/waze-miami-cops-false-reporting-google-traffic-app

[3] I. Arel, C. Liu, T. Urbanik, and A. G. Kohls, “Reinforcement learning-based multi-agent system for network traffic signal control,” IET Intelligent Transport Systems, vol. 4, no. 2, pp. 128–135, 2010.

[4] A. Glašchenko, A. Ivashchenko, G. Rzevski, and P. Skobelev, “Multi-agent real time scheduling system for taxi companies,” in 8th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2009), Budapest, Hungary, 2009, pp. 29–36.

[5] Q. Yang, G. Wang, A. Sadeghi, G. B. Giannakis, and J. Sun, “Two-timescale voltage control in distribution grids using deep reinforcement learning,” IEEE Transactions on Smart Grid, vol. 11, no. 3, pp. 2313–2323, 2019.

[6] Y. Gao, W. Wang, and N. Yu, “Consensus multi-agent reinforcement learning for volt-var control in power distribution networks,” IEEE Transactions on Smart Grid, vol. 12, no. 4, pp. 3594–3604, 2021.

[7] S. Ganesh, N. Vadori, M. Xu, H. Zheng, P. Reddy, and M. Veloso, “Reinforcement learning for market making in a multi-agent dealer market,” arXiv preprint arXiv:1911.05892, 2019.

[8] S. Zheng, A. Trott, S. Srinivasa, N. Naik, M. Gruenebeck, D. C. Parkes, and R. Socher, “The ai economist: Improving equality and productivity with ai-driven tax policies,” arXiv preprint arXiv:2004.13332, 2020.

[9] J.-F. Bonnefon, A. Shariff, and I. Rahwan, “The social dilemma of autonomous vehicles,” Science, vol. 352, no. 6293, pp. 1573–1576, 2016.

[10] M. Janssen and B. De Vries, “The battle of perspectives: a multi-agent model with adaptive responses to climate change,” Ecological economics, vol. 26, no. 1, pp. 43–65, 1998.

[11] P. Stone and M. Veloso, “Multiagent systems: A survey from a machine learning perspective,” Autonomous Robots, vol. 8, no. 3, pp. 345–383, 2000.

[12] A. Dorri, S. S. Kanhere, and R. Jurdak, “Multi-agent systems: A survey,” Ieee Access, vol. 6, pp. 28 573–28 593, 2018.
[13] M. J. Osborne and A. Rubinstein, *A course in game theory*. MIT press, 1994.
[14] L. S. Shapley, “Stochastic games,” *Proceedings of the national academy of sciences*, vol. 39, no. 10, pp. 1095–1100, 1953.
[15] M. L. Littman, “Markov games as a framework for multi-agent reinforcement learning,” in *Machine learning proceedings 1994*. Elsevier, 1994, pp. 157–163.
[16] C. Boutilier, “Planning, learning and coordination in multiagent decision processes,” in *TARK*, vol. 96, 1996, pp. 195–210.
[17] M. Tan, “Multi-agent reinforcement learning: Independent vs. cooperative agents,” in *Proceedings of the tenth international conference on machine learning*, 1993, pp. 330–337.
[18] M. L. Littman, “Value-function reinforcement learning in markov games,” *Cognitive systems research*, vol. 2, no. 1, pp. 55–66, 2001.
[19] T. Chen, K. Zhang, G. B. Giannakis, and T. Başar, “Communication-efficient policy gradient methods for distributed reinforcement learning,” *IEEE Transactions on Control of Network Systems*, vol. 9, no. 2, pp. 917–929, 2021.
[20] S. Kar, J. M. Moura, and H. V. Poor, “Qd-learning: A collaborative distributed strategy for multi-agent reinforcement learning through consensus+ innovations,” *IEEE Transactions on Signal Processing*, vol. 61, no. 7, pp. 1848–1862, 2013.
[21] J. Hu and M. P. Wellman, “Nash q-learning for general-sum stochastic games,” *Journal of machine learning research*, vol. 4, no. Nov, pp. 1039–1069, 2003.
[22] G. W. Brown, “Iterative solution of games by fictitious play,” *Act. Anal. Prod Allocation*, vol. 13, no. 1, p. 374, 1951.
[23] M. Zinkevich, M. Johanson, M. Bowling, and C. Piccione, “Regret minimization in games with incomplete information,” *Advances in neural information processing systems*, vol. 20, 2007.
[24] A. DiGiovanni and A. Tewari, “Thompson sampling for markov games with piecewise stationary opponent policies,” in *Uncertainty in Artificial Intelligence*. PMLR, 2021, pp. 738–748.
[25] L. Buşoniu, R. Babuška, and B. D. Schutter, “Multi-agent reinforcement learning: An overview,” *Innovations in multi-agent systems and applications-1*, pp. 183–221, 2010.
[26] Y. Yang and J. Wang, “An overview of multi-agent reinforcement learning from game theoretical perspective,” *arXiv preprint arXiv:2011.00583*, 2020.
[27] K. Zhang, Z. Yang, and T. Başar, “Multi-agent reinforcement learning: A selective overview of theories and algorithms,” *Handbook of Reinforcement Learning and Control*, pp. 321–384, 2021.
[28] C. Fiscko, B. Swenson, S. Kar, and B. Sinopoli, “Control of parametric games,” *2019 18th European Control Conference (ECC)*, pp. 1036–1042, 2019.
[29] C. Guestrin, D. Koller, and R. Parr, “Multiagent planning with factored mdps.” in *NIPS*, vol. 1, 2001, pp. 1523–1530.
[30] R. Becker, S. Zilberstein, V. Lesser, and C. V. Goldman, “Transition-independent decentralized markov decision processes,” in *Proceedings of the second international joint conference on Autonomous agents and multiagent systems*, 2003, pp. 41–48.
[31] D. Koller and R. Parr, “Policy iteration for factored mdps,” *arXiv preprint arXiv:1301.3869*, 2013.
[32] B. Sallans and G. E. Hinton, “Reinforcement learning with factored states and actions,” *The Journal of Machine Learning Research*, vol. 5, pp. 1063–1088, 2004.
[33] D. Sahabandu, L. Niu, A. Clark, and R. Poovendran, “Scalable planning in multi-agent mdps,” *arXiv preprint arXiv:2103.15894*, 2021.
[34] A. Y. Ng and M. I. Jordan, “Pegasus: A policy search method for large mdps and pomdps,” *arXiv preprint arXiv:1301.3878*, 2013.
[35] T. G. Dietterich, “Hierarchical reinforcement learning with the maxq value function decomposition,” *Journal of artificial intelligence research*, vol. 13, pp. 227–303, 2000.
[36] D. Koller and R. Parr, “Computing factored value functions for policies in structured mdps,” in *IJCAI*, vol. 99, 1999, pp. 1332–1339.
[37] C. Guestrin, D. Koller, R. Parr, and S. Venkatakrishnan, “Efficient solution algorithms for factored mdps,” *Journal of Artificial Intelligence Research*, vol. 19, pp. 399–468, 2003.
[38] S. Chaimontree, K. Atkinson, and F. Coenen, “Best clustering configuration metrics: towards multiagent based clustering,” in *International Conference on Advanced Data Mining and Applications*. Springer, 2010, pp. 48–59.

[39] A. A. Abbasi and M. Younis, “A survey on clustering algorithms for wireless sensor networks,” *Computer communications*, vol. 30, no. 14-15, pp. 2826–2841, 2007.

[40] A. S. Rostami, M. Badkoobe, F. Mohanna, A. A. R. Hosseinabadi, A. K. Sangaiah *et al.*, “Survey on clustering in heterogeneous and homogeneous wireless sensor networks,” *The Journal of Supercomputing*, vol. 74, no. 1, pp. 277–323, 2018.

[41] A. Krause, R. Rajagopal, A. Gupta, and C. Guestrin, “Simultaneous placement and scheduling of sensors,” in *2009 International Conference on Information Processing in Sensor Networks*. IEEE, 2009, pp. 181–192.

[42] D. S. Dos Santos and A. L. Bazzan, “Distributed clustering for group formation and task allocation in multiagent systems: A swarm intelligence approach,” *Applied Soft Computing*, vol. 12, no. 8, pp. 2123–2131, 2012.

[43] M. Brambilla, E. Ferrante, M. Birattari, and M. Dorigo, “Swarm robotics: a review from the swarm engineering perspective,” *Swarm Intelligence*, vol. 7, no. 1, pp. 1–41, 2013.

[44] E. Ferrara, M. JafariAshbagh, O. Varol, V. Qazvinian, F. Menczer, and A. Flammini, “Clustering memes in social media,” in *2013 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM 2013)*. IEEE, 2013, pp. 548–555.

[45] F. D. Malliaros and M. Vazirgiannis, “Clustering and community detection in directed networks: A survey,” *Mathematical Programming*, vol. 102, no. 1, pp. 167–183, 2005.

[46] L. Zhao, H. Nagamochi, and T. Ibaraki, “Greedy splitting algorithms for approximating multiway partition problems,” *Mathematical Programming*, vol. 53, no. 4, pp. 95–142, 2003.

[47] G. L. Nemhauser, L. A. Wolsey, and M. L. Fisher, “An analysis of approximations for maximizing submodular set functions—I,” *Mathematical programming*, vol. 14, no. 1, pp. 265–294, 1978.

[48] G. Calinescu, C. Chekuri, M. Pál, and J. Vondrák, “Maximizing a submodular set function subject to a matroid constraint,” in *International Conference on Integer Programming and Combinatorial Optimization*. Springer, 2007, pp. 182–196.

[49] L. Fleischer, M. X. Goemans, V. S. Mirrokni, and M. Sviridenko, “Tight approximation algorithms for maximum general assignment problems,” in *Proceedings of the seventeenth annual ACM-SIAM symposium on Discrete algorithm*, 2006, pp. 611–620.

[50] C. Fiscko, S. Kar, and B. Sinopoli, “Efficient solutions for targeted control of multi-agent mdps,” in *2021 American Control Conference (ACC)*. IEEE, 2021, pp. 690–696.

[51] T. Groves, “Incentives in teams,” *Econometrica: Journal of the Econometric Society*, pp. 617–631, 1973.

[52] T. Roughgarden, “Algorithmic game theory,” *Communications of the ACM*, vol. 53, no. 7, pp. 78–86, 2010.

[53] T. Başar and G. J. Olsder, *Dynamic noncooperative game theory*. SIAM, 1998.

[54] S. Sanjari, S. Bose, and T. Başar, “Incentive designs for stackelberg games with a large number of followers and their mean-field limits,” *arXiv preprint arXiv:2207.10611*, 2022.

[55] C. Ballester, A. Calvó-Armengol, and Y. Zenou, “Who’s who in networks. wanted: The key player,” *Econometrica*, vol. 74, no. 5, pp. 1403–1417, 2006.

[56] A. Galeotti, B. Golub, and S. Goyal, “Targeting interventions in networks,” *Econometrica*, vol. 88, no. 6, pp. 2445–2471, 2020.

[57] F. Parise and A. Ozdaglar, “Graphon games,” in *Proceedings of the 2019 ACM Conference on Economics and Computation*, 2019, pp. 457–458.

[58] I. Osband and B. Van Roy, “Near-optimal reinforcement learning in factored mdps,” *Advances in Neural Information Processing Systems*, vol. 27, 2014.

[59] D. P. Bertsekas, D. P. Bertsekas, D. P. Bertsekas, and D. P. Bertsekas, *Dynamic programming and optimal control*. Athena scientific Belmont, MA, 1995, vol. 1, no. 2.

[60] M. L. Littman, T. L. Dean, and L. P. Kaelbling, “On the complexity of solving markov decision problems,” *arXiv preprint arXiv:1302.4971*, 2013.