2023 62nd IEEE Conference on Decision and Control (CDC)
December 13-15, 2023. Marina Bay Sands, Singapore

Maximizing Reachability in Factored MDPs via Near-Optimal Clustering with Applications to Control of Multi-Agent Systems

Carmel Fiscko[1†], Soummya Kar[2], and Bruno Sinopoli[1]

Abstract— **We consider cluster-based control of agents modeled as a transition-independent Markov decision process (MDP), and the objective of assigning agents to clusters to maximize the size of the reachable state space. This goal is relevant to applications for which the same MDP model may be used to compute policies for different reward functions. The system controller wishes to define clusters to maximize flexibility within the attainable outcomes. Under the transition-independent MDP formulation, we first show that the size of the reachable state space is a submodular function. While maximizing the reachable state space subject to a desired number of clusters is a hard problem, properties of submodular optimization can be leveraged to propose approximate clustering techniques. We next demonstrate that a greedy clustering approach is a viable approximate solution and has a bounded optimality gap. We compare the performance in terms of value and computation complexity in using the flexibility-optimized clustering assignment versus a clustering assignment optimized for a specific reward function; there will be a loss in value at a savings in complexity. Finally, we demonstrate the utility of the flexibility-optimized clustering assignment in simulation on the same MDP model with various reward functions.**

I. Introduction

Finding an optimal control policy by leveraging a system model is a well-understood goal. An advantage of model-based techniques is that they enable generalized understanding of a system, as the same model can be re-used to find policies to achieve different control objectives.

In the context of reinforcement learning (RL), building models of the state-action to state transition matrix is difficult due to the large size of the state and action spaces. In particular, modeling multi-agent systems is a combinatorial problem, and the "curse of dimensionality" means that it takes longer and longer to find models and policies as the number of agents becomes large. It is therefore important to investigate methods that leveraging systems' internal structure to reduce the scale of the problem.

One such method to handle complex state spaces is the factored or transition-independent MDP [1], in which the state and/or action spaces are expressed as the Cartesian product of some relevant smaller set. This structure extends into the transition matrix, in which each transition probability may be expressed as a product. This structure reduces the scope of the state and policy spaces, therefore accelerating the learning process and aiding efficient exploration [2]. Factored MDPs in particular have been used to model multi-agent system applications [3], [4].

The factored MDP formulation may used to find policies for centralized control of a multi-agent system. We consider problems where a central planner (CP) transmits signals to the agents with the purpose of controlling the system to a desirable set of states. The goals of the CP and the agents are arbitrary and are not assumed to be necessarily cooperative or competitive.

Again, the trade-off between control performance and computation tractability must be discussed, and can be addressed via structural considerations. One strategy is to partition agents into disjoint clusters and design the CP's policy per-cluster instead of per-agent. In summary, this technique reduces the problem scope by restricting to a subset of the action space. For some given number of clusters, the clustering assignment that maximizes the resulting policy's value function can thus be chosen.

While the proposed clustering routine achieves good performance for a specific task, the efficiency breaks down if the goal of the controller changes. Must the CP re-optimize over the entire space of clustering assignments before computing a policy for this new objective? For example, consider a traffic example where, under normal operation, travelers may be grouped by category such as car, bus, and pedestrian. In an emergency scenario, however, it is more natural to classify these agents as first-responder or non-first-responder. This shows that depending on the control objective, the best clustering assignment of the agents can change.

In this paper, instead of optimizing clusters for a specific task, we find the clustering configuration that maximizes the size of the model's reachable space. This method enables the CP to solve for the clustering assignment once, then re-use the MDP model and the fixed clusters with new reward functions. The logic is as follows: if a state is reachable, then there will exist a policy that will take the system to that state at some time with non-zero probability. If the CP's objective is based on controlling the system to specific states, then designing clusters based on this reachability criterion thus maximizes the CP's flexibility to change their defined reward function. While this method does not optimize for the value of a specific task, it may be used as a starting point to find valid policies without having to re-optimize over the space

[1]Carmel Fiscko and Bruno Sinopoli are with the Dept. of Electrical and Systems Engineering at Washington University in St. Louis, MO. {carmel, bsinopoli}@wustl.edu

[2]Soummya Kar is with the Dept. of Electrical and Computer Engineering at Carnegie Mellon University in Pittsburgh, PA. soummyak@andrew.cmu.edu

[†]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. Additional support provided by the Hsu Chang Memorial Fellowship in ECE.

of clusters when the reward function is changed.

In contrast, re-using the clustering assignment found for a particular goal as the starting point for a new goal may not always be a valid strategy. The clusters designed for objective A may have no relation to the rewarded states for a new objective B. This means that under the clustering structure of A, there are no guarantees that a policy can be found that takes the system to a rewarded state for B. As computation is a particular concern, it is not desirable to pose a reward function for which no policy may exist. We note that designing clusters for the purpose of reachability is only relevant for applications with *highly sparse* transition matrices, as dense transitions naturally imply that all pairwise combinations of states are reachable.

Reachability has been studied within the context of general RL problems as it can influence decisions on which algorithm to implement and the resulting performance and guarantees [5], [6]. For example, ideas stemming from reachability have been used to propose stopping methods for value iteration [7] and as guidance for exploration in learning [8].

The problem of clustering agents (factors) for control in a MDP model has been studied in the context of sensor placement [9], autonomous vehicles [10], and event detection [11]. These works share the common method of posing the problem as optimization of a submodular objective function subject to some constraint set. In particular, submodularity is a useful property to establish for set-valued optimization functions as it provides useful properties to quantify iterative improvement [12], [13]. Useful submodular functions for reinforcement learning include the value function of factored MDPs [14][15] and random walk times [16].

The main contribution in this paper is proving that the size of the reachable state space is submodular with respect to clustering assignments of the agents, and thus a clustering assignment can be found by adapting techniques from submodular optimization. We are not aware of another work that proposes clustering agents explicitly for the purpose of maximizing reachable sets.

Section II discusses the preliminaries of the multi-agent MDP model and defines reachability. Section III describes the optimization problem. Section IV introduces the definition of submodularity and establishes the main theorem. An algorithm to approximately solve the presented optimization problem is shown in Section V, and simulations are shown in Section VI.

II. Problem Setup

A. Preliminaries

Consider a set of agents (factors) $\mathcal{N} = \{1, \dots, N\}$. Let there be a factored MDP described by the tuple $\mathcal{M} = (\mathcal{S}, \mathcal{A}, r, T, \gamma)$; each element will be defined in this section. The state space factors across the agents as $\mathcal{S} = S_1 \times \cdots \times S_N$ where one state describes one combination of substates across all agents $s = [s_1, \dots, s_N]$. Any state with a set subscript as in $s_{\mathcal{X}}$ references the elements of the state s indexed by the set $\mathcal{X}$; for example, s_n means the substate of agent n within the greater state.

The controller can assign an action $\alpha \in \mathcal{A}_1$ to each agent where $\mathcal{A}_1$ is a finite set of choices. This yields the overall joint action space $\mathcal{A} = \bigotimes_{n=1}^{N} \mathcal{A}_1$. (Unique action spaces for each agent may be considered, but this paper will consider identical agent action spaces.) While best control performance of the MDP can be achieved by assigning each agent a possibly unique action at any moment in time, computing controls at that fine granularity can be expensive in terms of computation and storage. For these reasons, the controller may instead wish to group agents by common characteristics and assign controls per cluster instead of per agent.

Definition 1: A *clustering* or a *partition* of a set $\mathcal{N}$ is a collection of subsets $\mathcal{C} = \{c_1, \dots, c_C\}$ such that $c_1 \cup \cdots \cup c_C = \mathcal{N}$ and $c_i \cap c_j = \emptyset$ for all i, j. Let Ω denote the set of all possible clusterings.

Definition 2: The *size* of a clustering is the number of clusters, i.e. the number of subsets $|\mathcal{C}|$.

Given a clustering assignment $\mathcal{C}$, let $\mathcal{C}(n)$ denote cluster assignment of agent n. Let $\alpha_{\mathcal{C}(n)}$ denote the action assigned to agent n's cluster and thus to agent n. The action space given a particular $\mathcal{C}$ is thus,

$$\mathcal{A}_{\mathcal{C}} = \{\{\alpha_{\mathcal{C}(1)}, \dots, \alpha_{\mathcal{C}(N)}\} \mid \alpha_{c(n)} \in \mathcal{A}_1,\ n \in \mathcal{N}\}. \quad (1)$$

The size of the action space is $|\mathcal{A}_1|^{|\mathcal{C}|}$. If unique controls were given to each agent, the size would be $|\mathcal{A}_1|^{|\mathcal{N}|}$.

The rest of the factored MDP definition is quite standard. The reward function is defined to give a bounded deterministic scalar reward for each state-action pair. We assume the reward function satisfies the separable structure $r(s, \boldsymbol{\alpha}) = \sum_{n \in \mathcal{N}} r_n(s_n, \alpha_{\mathcal{C}(n)})$, which corresponds to a summation across agent-specific rewards.

Next, the transition kernel $T : \mathcal{S} \times \mathcal{A}_C \to \Delta(\mathcal{S})$ defines the state-action to state transition probabilities, where $\Delta(\mathcal{S})$ refers to the probability simplex over $\mathcal{S}$. The transition assumed to be stationary and satisfy the factored Markovian form $P(s'|s, \boldsymbol{\alpha}) = \prod_{n \in \mathcal{N}} P(s'_n|s, \alpha_{\mathcal{C}(n)})$. This factorization structure means that each agent has an independent transition when conditioned on the current state and action, and each agent is only dependent on the action assigned to it from the central controller. The final component $\gamma \in (0, 1)$ is a scalar discount factor that diminishes future rewards.

The goal of an MDP is to solve for a control policy $\pi : \mathcal{S} \to \Delta(\mathcal{A}_{\mathcal{C}})$. The *value* of a policy π is the expected reward,

$$V_H^{\pi}(s) \triangleq \mathbb{E}\Bigg[\sum_{t=0}^{H} \gamma^t r(s_t, \boldsymbol{\alpha}_t) \Big| s_0 = s, \boldsymbol{\alpha}_t \sim \pi(s_t), s_{t+1}|s_t, \boldsymbol{\alpha}_t \sim T\Bigg], \quad (2)$$

where H is a time horizon. The value for an infinite time horizon is attained by $V^{\pi}(s) \triangleq \lim_{H \to \infty} V_H^{\pi}(s)$. An optimal policy for an infinite time horizon, therefore, is a maximizer $\pi^* \in \text{argmax}_{\pi} V^{\pi}(s)$. As we consider a factored MDP, the policy can be independently factored across the agents given a state [17].

B. Reachability

Given a MDP model $\mathcal{M}$, the feasibility of control objectives may be assessed by considering their reachability.

Definition 3: We say that a state $j \in \mathcal{S}$ is *reachable* from state $i \in \mathcal{S}$ if there exists an integer t and some $\{\boldsymbol{\alpha}_t\}_{t\geq 0}$ such that $P(s_t = j | s_0 = i, \boldsymbol{\alpha}_0, \ldots, \boldsymbol{\alpha}_t) > 0$.

The pair-wise reachability between any two states can be determined according to the following criterion.

Definition 4: Define the composite adjacency matrix $\widetilde{T}$ as,

$$\widetilde{T} = \mathcal{B}\left(T(\boldsymbol{\alpha})\right), \tag{3}$$

where $T(\boldsymbol{\alpha})$ is the $|\mathcal{S}| \times |\mathcal{S}|$ matrix where the element $[T(\boldsymbol{\alpha})]_{ij} = P(s' = j | s = i, \boldsymbol{\alpha})$, and the operation $\mathcal{B}$ is defined element-wise as $[\mathcal{B}(X)]_{ij} = 1$ if $X_{ij} > 0$ and zero otherwise.

Let the matrix $R(\mathcal{A}_\mathcal{C})$ be defined as,

$$R(\mathcal{A}_\mathcal{C}) = \mathcal{B}\left[\sum_{t=1}^{|\mathcal{S}|-1} \Big(\sum_{\boldsymbol{\alpha}\in\mathcal{A}_\mathcal{C}} \widetilde{T}(\boldsymbol{\alpha}) \Big)^t \right]. \tag{4}$$

According to [18], a state j is reachable from i if and only if $[R(\mathcal{A}_\mathcal{C})]_{ij} > 0$. Furthermore if a state is reachable, then a policy exists such that the system can be controlled to go to that state in finite time.

The overall size of the reachable set given a clustering configuration $|R(\mathcal{A}_\mathcal{C})|$ can thus be calculated as,

$$|R(\mathcal{A}_\mathcal{C})| = \mathbf{1}^\top R(\mathcal{A}_\mathcal{C})\mathbf{1}, \tag{5}$$

where $\mathbf{1}$ is a vector of 1s of the appropriate dimension.

III. Objective

The goal in this paper is to find a clustering assignment $\mathcal{C}$ that satisfies,

$$\begin{aligned} &\max_{\mathcal{C}} f(\mathcal{C}), \\ &\text{s.t. } |\mathcal{C}| \leq C, \end{aligned} \tag{6}$$

where $f(\mathcal{C}) \triangleq |R(\mathcal{A}_\mathcal{C})|$ and C is a maximal number of clusters. Intuitively, this objective corresponds to designing internal structure of the action space to achieve generalization within the transition matrices.

Unfortunately, (6) is difficult in general and within the class of NP-hard problems [19]. For submodular objective functions, however, this problem can be made tractable by leveraging useful results of submodularity to find approximate solutions [12]. The general version of this problem can be posed as optimization of a submodular function subject to a partition matroid constraint [13].

In this paper, we will establish that the desired objective function, the size of the reachable state space, is submodular with respect to the clustering assignment of the agents, and use this fact to formulate approximate solutions to (6).

IV. Submodularity

In this section, we will introduce the concept of submodularity and show that size of the reachable state space is a submodular function with respect to the clustering assignment of the agents.

Submodularity is a property of set-valued functions, related to convexity, where the marginal benefit to evaluating the function on more elements decreases as the base set grows larger. This notion is formalized in the following definition.

Definition 5: A function $f : 2^\Omega \to \mathbb{R}$ is *submodular* if for every $A \subseteq B \subseteq V$ and $e \in V \setminus B$ it holds that,

$$f(A \cup \{e\}) - f(A) \geq f(B \cup \{e\}) - f(B). \tag{7}$$

Equation (7) displays the "diminishing returns" property of submodular functions.

The submodularity property can be used to help approximate optimal solutions to (6). The performance gap between two potential guesses, A and B as in Definition 5, may be bounded, and thus a sequence of monotonically improving approximate solutions may be constructed.

A. Submodularity of Reachable Space

In this section, we will lay out the necessary definitions and notation to establish submodularity of the size of the reachable state space.

Definition 6: A clustering $\mathcal{C}_1$ is a subset of $\mathcal{C}_2$ if for every $c \in \mathcal{C}_1$, there exists a set of clusters $\{c'_1, \ldots, c'_c\} \subseteq \mathcal{C}_2$ such that $c = \{c'_1, \ldots, c'_c\}$.

Example 1: For $\mathcal{N} = \{1, 2, 3, 4\}$, the clustering $\mathcal{C}_1 = \{\{1, 2\}, \{3, 4\}\}$ is a subset of $\mathcal{C}_2 = \{\{1, 2\}, \{3\}, \{4\}\}$, but not a subset of $\mathcal{C}_3 = \{\{1\}, \{2, 3, 4\}\}$.

Definition 7: We define the *union* of two clustering assignments $\mathcal{C}_1$ and $\mathcal{C}_2$ to be,

$$\mathcal{C}_1 \cup \mathcal{C}_2 = \{c_1 \cap c_2 | \ \forall \ c_1 \in \mathcal{C}_1, c_2 \in \mathcal{C}_2\}. \tag{8}$$

Example 2: For $\mathcal{N} = \{1, 2, 3, 4\}$, $\mathcal{C}_1 = \{\{1, 2\}, \{3, 4\}\}$, and $\mathcal{C}_3 = \{\{1\}, \{2, 3, 4\}\}$, the union $\mathcal{C}_1 \cup \mathcal{C}_3 = \{\{1\}, \{2\}, \{3, 4\}\}$.

Lemma 1: If $\mathcal{C}_1 \subseteq \mathcal{C}_2$ then $\mathcal{C}_1 \cup \mathcal{C}_3 = \mathcal{C}_2 \cup \mathcal{C}_3$.

Proof: In the definition of the union $C_1 \cup \mathcal{C}_3 = \{c_1 \cap c_3 | \forall \ c_1 \in \mathcal{C}_1, c_3 \in \mathcal{C}_3\}$, substitute the definition of cluster subset $\{\{c'_1, \ldots, c'_c\}_1 \cap c_3\}$. Then note this set of sets is equal to $\{c_2 \cap c_3 | \forall \ c_2 \in \mathcal{C}_2, c_3 \in \mathcal{C}_3\} = \mathcal{C}_2 \cup \mathcal{C}_3$ as there exists $c_2 = \{c'_1, \ldots, c'_c\}_1$ across all possible $c_2 \in \mathcal{C}_2$. ■

Lemma 2: If $\mathcal{C}_1 \subseteq \mathcal{C}_2$, then $|R(\mathcal{A}_{C_1})| \leq |R(\mathcal{A}_{C_2})|$.

Proof: As $\mathcal{C}_1 \subseteq \mathcal{C}_2$, $k \leq C$. Recall that $\widetilde{T}(\boldsymbol{\alpha}) \geq 0$ for all entries. Then according to the definition of clustering subsets and (1), the action space of $\mathcal{C}_1$ will be a subset of the action space of $\mathcal{C}_2$. Thus,

$$\begin{aligned} \sum_{\boldsymbol{\alpha}\in\mathcal{A}_{\mathcal{C}_2}} \widetilde{T}(\boldsymbol{\alpha}) &= \sum_{\boldsymbol{\alpha}\in\mathcal{A}_{\mathcal{C}_2}\setminus\mathcal{A}_{\mathcal{C}_1}} \widetilde{T}(\boldsymbol{\alpha}) + \sum_{\boldsymbol{\alpha}\in\mathcal{A}_{\mathcal{C}_1}} \widetilde{T}(\boldsymbol{\alpha}), \\ &\geq \sum_{\boldsymbol{\alpha}\in\mathcal{A}_{\mathcal{C}_1}} \widetilde{T}(\boldsymbol{\alpha}). \end{aligned}$$

With substitution,

$$R(\mathcal{A}_{\mathcal{C}_2}) = \mathcal{B}\left[\sum_{t=1}^{|\mathcal{S}|-1}\Big(\sum_{\boldsymbol{\alpha}\in\mathcal{A}_{\mathcal{C}_2}}\widetilde{T}(\boldsymbol{\alpha})\Big)^t\right],$$
$$\geq \mathcal{B}\left[\sum_{t=1}^{|\mathcal{S}|-1}\Big(\sum_{\boldsymbol{\alpha}\in\mathcal{A}_{\mathcal{C}_1}}\widetilde{T}(\boldsymbol{\alpha})\Big)^t\right] = R(\mathcal{A}_{\mathcal{C}_1}),$$

where again the $\geq$ operator is evaluated element-wise. Thus,

$$\mathbf{1}^\top R(\mathcal{A}_{\mathcal{C}_2})\mathbf{1} \geq \mathbf{1}^\top R(\mathcal{A}_{\mathcal{C}_1})\mathbf{1}, \tag{9}$$
$$\Rightarrow |R(\mathcal{A}_{\mathcal{C}_2})| \geq |R(\mathcal{A}_{\mathcal{C}_1})|. \tag{10}$$

■

The combination of these lemmas and properties leads to the main theorem.

Theorem 1: **(Submodularity)** The size of the reachable state space $|R(\mathcal{C})|$ is submodular with respect to $\mathcal{C}$.

Proof: Note that a particular clustering assignment maps to a particular action space, which in turn maps to a reachable set. Thus, $|R(\mathcal{C})| : 2^\Omega \to \mathbb{R}$.

Next, consider two clusterings $\mathcal{C}_1 \subseteq \mathcal{C}_2$ and a third clustering $\mathcal{C}_3 \in \Omega \setminus (\mathcal{C}_2 \cup \mathcal{C}_0)$ where $|\mathcal{C}_3| \geq |\mathcal{C}_2|$ and $\mathcal{C}_0$ is the set of all possible subsets of $\mathcal{C}_2$. Let $\mathcal{C}_4 = \mathcal{C}_1 \cup \mathcal{C}_3$. It holds that $|\mathcal{C}_1| \leq |\mathcal{C}_2| \leq |\mathcal{C}_3| \leq |\mathcal{C}_4|$. Then,

$$|R(\mathcal{A}_{\mathcal{C}_1\cup\mathcal{C}_3})| - |R(\mathcal{A}_{\mathcal{C}_1})| = |R(\mathcal{A}_{C_4})| - |R(\mathcal{A}_{\mathcal{C}_1})|. \tag{11}$$

Then by Lemma 1,

$$|R(\mathcal{A}_{\mathcal{C}_2\cup\mathcal{C}_3})| - |R(\mathcal{A}_{\mathcal{C}_2})| = |R(\mathcal{A}_{\mathcal{C}_4})| - |R(\mathcal{A}_{\mathcal{C}_2})|. \tag{12}$$

By Lemma 2, (11) will be greater than (12), thus satisfying (7) and completing the proof. ■

Corollary 1: The size of the reachable set from an initial state i is submodular with respect to $\mathcal{C}$.

Proof: This result follows from (9) by replacing the $\mathbf{1}$ vectors with vectors of the form $[0,\ldots,0,1,0,\ldots,0]^\top$ where the entry with 1 entry is the index i. ■

V. Algorithm

We can find an approximate solution to (6) in an efficient manner by leveraging the established submodularity property of the objective function. It is well known in submodular optimization that a greedy method can yield good results and is simple to evaluate. To implement the approximate reachability maximization routine, the following algorithm can be proposed.

In general, C will refer to the desired final number of clusters and k will be the number clusters at an intermediate step. The initialization defines one cluster of all the agents. At each subsequent query, the algorithm searches for an existing cluster of agents U and a split of U into two new clusters $\{U \setminus X, X\}$ that provides maximum improvement to $|R|$. This is repeated until C clusters are achieved and the final size $|R|$ is returned. The entire process is shown in Algorithm 1.

Using known properties of submodularity, it is possible to evaluate the resulting performance of this solution technique.

Algorithm 1: Greedy Splitting Algorithm

1 $C_1 = \{\mathcal{N}\}$;
2 $f_1 = f(\mathcal{C}_1)$;
3 **for** $k \in \{2,\ldots,C\}$ **do**
4 $\quad (X_k, U_{k-1}) \leftarrow \text{argmax}\{f(X) + f(U \setminus X) - f(U) \mid \emptyset \subset X \subset U,\ U \in \mathcal{C}_{k-1}\}$;
5 $\quad \mathcal{C}_k \leftarrow \{\mathcal{C}_{k-1} \setminus U_{k-1}\} \cup \{X_k, U_{k-1} \setminus X_k\}$;
6 $\quad f_k = f(\mathcal{C}_k)$;
7 **end**

Theorem 2: **(Optimality gap)** The clustering assignment $\widehat{\mathcal{C}}$ found via greedy splitting in Algorithm 1 satisfies,

$$|R(\widehat{\mathcal{C}})| \geq \frac{1}{2}|R(\mathcal{C}^*)|, \tag{13}$$

where $\mathcal{C}^*$ is a solution to (6).

Proof: Given Theorem 1, see [20]. ■

It is known for problems of the form (6), a $(1 - 1/e)$ approximation is optimal. The authors of [13] propose an algorithm to find solution that satisfies this optimal approximation (in expectation); however, the implementation's running time is costly (the authors estimate $\tilde{O}(n^8)$). For this reason, we instead use the greedy implementation with the $(1 - 1/2)$ approximation.

A. Complexity

To evaluate the computation complexity of Algorithm 1, we separate analysis of the outer loop iterations, specifically the number of times the $\arg\max$ step must be evaluated in 4, and inner looop oracle queries to f.

As a comparison method, the same procedure Algorithm 1 can be used to compute clusters for specific factored MDPs with submodular reward functions by defining $f \triangleq V$. This holds because submodular reward functions yield submodular value functions [17]. Clearly, the number of outer loop iterations will be on the same order when computing clusters for reachability or for V. In addition, if k value functions can be considered, then the savings in terms of outer loop complexity will on the order of $1/k$ because the reachability-optimized clusters will only need to be computed one time. The main difference in computation time will thus be from oracle calls to the chosen f. In terms of inner loop complexity, evaluating $|R(\mathcal{C})|$ will be faster than computing $V^*(\mathcal{C})$. The composite adjacency matrix $\widetilde{T}$ can be pre-computed, so the product of $|\mathcal{S}| - 1$ square matrices of size $|\mathcal{S}|$ will need to be evaluated to calculate the reachability criterion in equation (4). As it is assumed these matrices are highly sparse, libraries optimized for operations on sparse matrices can further speed computation time. In particular, the multiplication of sparse matrices with at most d nonzero elements can be achieved with a complexity of $O(d|\mathcal{S}|)$ [21].

In comparison, computation of the optimal value function requires solving the entire MDP for each proposed cluster assignment. To name two standard techniques, policy iteration has a per-iteration complexity of $O(|\mathcal{A}_1|^C|\mathcal{S}|^2 + |\mathcal{S}|^3)$ and value iteration has a per-iteration complexity of

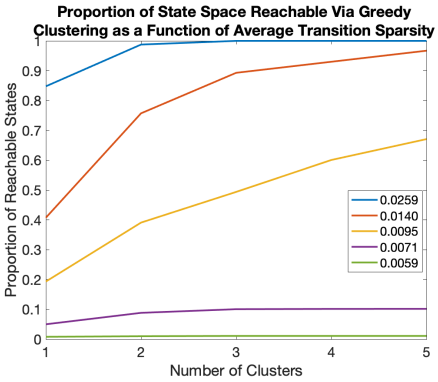

Fig. 1. Size of the reachable state space as a result of Algorithm 1 as a function of the average sparsity of the transition matrices. Note that adding clusters leads to submodular improvement, and the sparseness correlates with worse reachability.

$O(|\mathcal{A}_1|^C|\mathcal{S}|^2)$ which can be improved to $O(|\mathcal{A}_1||\mathcal{S}|^2)$ for factored MDPs [14]. Both of these methods are contractions and theoretically need an infinite number of iterations to converge. In practice, the number of iterations will be a function of the approved error tolerance of the application.

VI. Experiments

In this section, simulations will demonstrate how reachability of the MDP model changes with respect to clustering assignments and how the proposed reachability-optimized clusters perform for specific tasks.

A. Submodularity of Reachable Space

This first experiment demonstrates how the agents may be clustered to maximize the size of the reachable state space via Algorithm 1. Consider a system with $N = 5$ agents, where each agent has a subspace of size $|S_n| = 3$ and the CP can, at most, select between $|\mathcal{A}_1| = 3$ actions to transmit to each agent/cluster. The transition factors for each agent were randomly generated but where biased to be highly sparse.

The results of this experiment are shown in Figure 1. The different lines on the plot show the results for transition matrices of varying sparseness, where average sparsity is defined as the number of non-zero elements in the transition matrix divided by the matrix size and averaged across the action space. Note that the graph clearly displays the "diminishing returns" property; as more clusters are added to the definition of the action space, the marginal improvement to the size of the reachable state space decreases. This suggests that the greedy splitting approach can be used as a structured method for designing the action space for the purpose of maximal reachabilit. In addition, these results confirm that as the transition matrices become more sparse, it becomes harder to guarantee that all pair-wise combinations of states will be reachable.

B. Performance of Reachability-Optimized Clusters

The next experiment investigates how the reachability-optimized clustering configurations perform for general tasks and can out-perform clusters optimized for a specific task. We simulate a system of $N = 5$ agents, where each agent has a subspace of size $|S_n| = 3$ and the CP can, at most, select between $|\mathcal{A}_1| = 3$ actions to transmit to each agent/cluster. We consider two different reward functions, $r_A(s) = \sum_{n\in\mathcal{N}} \mathbb{1}(s_n = 1)$ and $r_B(s) = \sum_{n\in\mathcal{N}} \mathbb{1}(s_n = 2)$. Note that $r_A(s)$ and $r_B(s)$ are identically distributed, so differences in the value function will arise due to the policy and transitions. Higher values will be correlated with reachability of more highly rewarded states. Again, sparse transition factors for each agent were randomly generated.

Next, the clustering assignment was found according to three different metrics: maximizing reachability, maximizing the value function of system $\mathcal{M}_A$ defined with submodular reward function r_A, and maximizing the value function of system $\mathcal{M}_B$ defined with submodular reward function r_B where $r_B \neq r_A$. As submodular reward functions yield submodular value functions [17], Algorithm 1 was used to find clustering assignments that optimize for the value function.

The results of this experiment are shown in Figures 2 and 3. Each chart shows the value of the corresponding system as produced by the different clustering assignments. Note that the values for $C = 1$ and $C = N$ are equal as there is only one possible clustering assignment for these number of clusters. As expected, the best performance is found by the clusters optimized for the specific value functions of the system. In second place is the reachability-optimized clustering assignment, and the worst performance is found by clusters for the opposite system. These results are logical, as the clusters optimized for $\mathcal{M}_A$ should have arbitrary performance for system $\mathcal{M}_B$ because r_A and r_B are not related. In comparison, good value can be attained via the reachability-optimized clusters for both $\mathcal{M}_A$ and $\mathcal{M}_B$ because this criteria ensures that policies that control the system to reachable states will exist. It is not guaranteed, however, for the reachability-optimized clusters to yield higher value. In terms of computation complexity, only two reward functions were considered so the outer-loop complexity savings was a factor of $1/2$. If k reward functions were considered, this factor would be further reduced to $1/k$.

This example shows that if the CP's objective can change from r_A to r_B, they would achieve better performance both terms of value and in terms of computation complexity by using the reachability-optimized clusters.

VII. Conclusion

In this work we consider control of a multi-agent system modeled as a factored MDPs where scale is addressed by partitioning agents into disjoint clusters and exerting controls per-cluster instead of per-agent. Instead of optimizing the cluster assignments for a particular reward function, we instead propose designing clusters to maximize the size of the reachable state space. The resulting clusters ensure that valid policies can be found across the state space; this allows the controller to use the same cluster assignment while generalizing to different goals. Solving for the reachability-optimized clusters one time is a more efficient strategy than

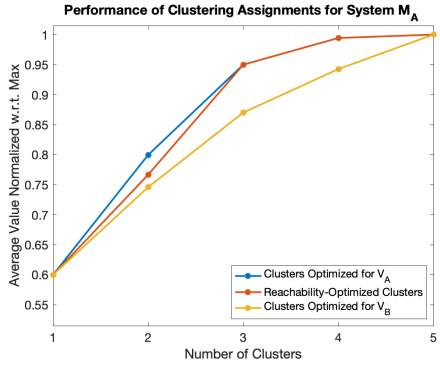

Fig. 2. Comparison on system $\mathcal{M}_A$ of clustering assignments optimized for V_A^*, $|R(\mathcal{C})|$, and V_B^*. Note that as $r_A \neq r_B$, the clusters designed for system $\mathcal{M}_B$ do not have meaning for $\mathcal{M}_A$, leading to worse performance. The reachability-optimized clusters help ensure that policies exist that attain rewarded states for both $\mathcal{M}_A$ and $\mathcal{M}_B$.

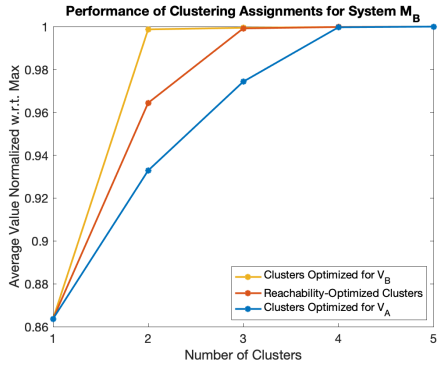

Fig. 3. Comparison on system $\mathcal{M}_B$ of clustering assignments optimized for V_B^*, $|R(\mathcal{C})|$, and V_A^*. Note that as $r_A \neq r_B$, the clusters designed for system $\mathcal{M}_A$ do not have meaning for $\mathcal{M}_B$, leading to worse performance. The reachability-optimized clusters help ensure that policies exist that attain rewarded states for both $\mathcal{M}_A$ and $\mathcal{M}_B$.

re-computing clusters for each possible reward function.

We show that the size of the reachable state space is submodular with respect to the clusters of the agents. Thus, ideas from submodular optimization may be applied to this problem to analyze how the reachable space evolves with the cluster definition. We demonstrate that the greedy method can be used to find $(1 - 1/2)$ approximate solutions that are easy to evaluate. Future work will be done to quantify performance of the reachability-optimized clusters and to expand these conceptual clustering ideas to the model-free setting via exploration.

REFERENCES

[1] C. Guestrin, D. Koller, R. Parr, and S. Venkataraman, "Efficient solution algorithms for factored mdps," *Journal of Artificial Intelligence Research*, vol. 19, pp. 399–468, 2003.

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

[3] C. Guestrin, D. Koller, and R. Parr, "Multiagent planning with factored mdps," *Advances in neural information processing systems*, vol. 14, 2001.

[4] Y. Yang and J. Wang, "An overview of multi-agent reinforcement learning from game theoretical perspective," *arXiv preprint arXiv:2011.00583*, 2020.

[5] P. Auer, T. Jaksch, and R. Ortner, "Near-optimal regret bounds for reinforcement learning," *Advances in neural information processing systems*, vol. 21, 2008.

[6] Y. Bai, C. Jin, and T. Yu, "Near-optimal reinforcement learning with self-play," *Advances in neural information processing systems*, vol. 33, pp. 2159–2170, 2020.

[7] S. Haddad and B. Monmege, "Reachability in mdps: Refining convergence of value iteration," in *Reachability Problems: 8th International Workshop, RP 2014, Oxford, UK, September 22-24, 2014. Proceedings 8*. Springer, 2014, pp. 125–137.

[8] K. Kawaguchi, "Bounded optimal exploration in mdp," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 30, no. 1, 2016.

[9] 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.

[10] A. Ma, M. Ouimet, and J. Cortés, "Hierarchical reinforcement learning via dynamic subspace search for multi-agent planning," *Autonomous Robots*, vol. 44, no. 3-4, pp. 485–503, 2020.

[11] A. Kumar and S. Zilberstein, "Event-detecting multi-agent mdps: Complexity and constant-factor approximation," 2009.

[12] A. Krause and D. Golovin, "Submodular function maximization." *Tractability*, vol. 3, pp. 71–104, 2014.

[13] G. Calinescu, C. Chekuri, M. Pal, and J. Vondrák, "Maximizing a monotone submodular function subject to a matroid constraint," *SIAM Journal on Computing*, vol. 40, no. 6, pp. 1740–1766, 2011.

[14] 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.

[15] R. Kumar, P. Varakantham, and A. Kumar, "Decentralized planning in stochastic environments with submodular rewards," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 31, no. 1, 2017.

[16] A. Clark, B. Alomair, L. Bushnell, and R. Poovendran, "On the structure and computation of random walk times in finite graphs," *IEEE Transactions on Automatic Control*, vol. 64, no. 11, pp. 4470–4483, 2019.

[17] C. Fiscko, S. Kar, and B. Sinopoli, "Cluster-based control of transition-independent mdps," *arXiv preprint arXiv:2207.05224*, 2022.

[18] C. Fiscko, B. Swenson, S. Kar, and B. Sinopoli, "Control of parametric games," in *2019 18th European Control Conference (ECC)*. IEEE, 2019, pp. 1036–1042.

[19] G. L. Nemhauser, L. A. Wolsey, and M. L. Fisher, "An analysis of approximations for maximizing submodular set functions—i," *Mathematical programming*, vol. 14, pp. 265–294, 1978.

[20] G. L. Nemhauser and L. A. Wolsey, "Best algorithms for approximating the maximum of a submodular set function," *Mathematics of operations research*, vol. 3, no. 3, pp. 177–188, 1978.

[21] R. Yuster and U. Zwick, "Fast sparse matrix multiplication," *ACM Transactions On Algorithms (TALG)*, vol. 1, no. 1, pp. 2–13, 2005.