Exploiting Fast Decaying and Locality in Multi-Agent MDP with Tree Dependence Structure

Guannan Qu, Na Li

***Abstract*— This paper considers a multi-agent Markov Decision Process (MDP), where there are n agents and each agent i is associated with a state s_i and action a_i taking values from a finite set. Though the global state space size and action space size are exponential in n, we impose local dependence structures and focus on local policies that only depend on local states, and we propose a method that finds nearly optimal local policies in polynomial time (in n) when the dependence structure is a one directional tree. The algorithm builds on approximated reward functions which are evaluated using locally truncated Markov process. Further, under some special conditions, we prove that the gap between the approximated reward function and the true reward function is decaying exponentially fast as the length of the truncated Markov process gets longer. The intuition behind this is that under some assumptions, the effect of agent interactions decays exponentially in the distance between agents, which we term "fast decaying property". Results in this paper are our preliminary steps towards designing efficient reinforcement learning algorithms with optimality guarantees in large multi-agent MDP problems whose state (action) space size is exponentially large in n.**

I. INTRODUCTION

Multi-agent Markov Decision Processes (MDP) have found many applications such as robot swarms, game play, queuing networks, and cyber-physcial systems [1]–[4]. In a typical multi-agent MDP, there are n agents and each agent i has a state s_i and an action a_i, both taking values from finite sets. Further, each agent is associated with stage reward r_i that is a function of s_i (and/or a_i), and the total stage reward is the summation of r_i. The goal is to find decision policies such that the average total reward is maximized.

A fundamental difficulty in solving multi-agent MDP is that even if individual state and action spaces are small, the entire state profile $(s_1, \ldots, s_n)$ and the action profile $(a_1, \ldots, a_n)$ can take values from a set of size exponentially large in n. Such curse of dimensionality renders the problem almost intractable to solve. In fact, it is even difficult to just specify the transition probability matrix of the problem. Another challenge is that even if an optimal policy can be found to map a global state $(s_1, \ldots, s_n)$ profile to an action profile $(a_1, \ldots, a_n)$, it is usually impractical to implement such a policy for real-world systems because of the limited information and communication among agents. For example, it may be costly for agent i obtain states other than s_i.

The above challenges motivate us to focus on Multi-agent MDPs with *local dependence structures* and *local policies*. Specifically, we associate the agents with a underlying dependence graph $\mathcal{G}$, and assume that the distribution of $s_i(t+1)$ only depend on the current states of the local neighborhood of i as well as the local $a_i(t)$. Further, we restrict to the class of local polices, where agent i takes action a_i based on its own local state s_i. Such structures can be found in many real networks, e.g. epidemic spreading network [5], opinion dynamics in social networks [6], [7], communication [8], networking [9].

However, challenges remain. Despite the local dependence structures, these structures can not be directly utilized by most dynamic programming algorithms like value iteration, policy iteration [10] or reinforcement learning algorithms like Q-learning, actor-critic methods [11], [12]. Though there are approximate dynamic programming methods using function approximation to reduce the computational burden caused by the large state/action space [13], it is not immediately clear how to choose function approximators (or basis vectors in linear function approximators) to achieve both computational efficiency and provable (near-)optimalitiy. This brings the following question of this paper: *can the dependence structure of the MDP be utilized to develop efficient algorithms that provably find a (near-)optimal local policy?*

Our Contributions. In this paper, we partially answer this question when the dependence graph is a one-directional tree. We propose a Locality-based Local Policy Search (LLPS) method to search for (near-)optimal local policies where individual agent i make the decision a_i based on the local state s_i. The method builds on approximately reward functions which are evaluated using locally truncated Markov processes, where for each agent i we consider a "truncated" model consisting of agent i and its k-hop local parents, where influence beyond the k-hop parent is ignored. For small k, these approximated reward functions can be computed and optimized efficiently. Further, we show that, under some conditions, the gap between the approximated reward function and the true reward function *decays exponentially in* k, meaning the resulting local policies are near-optimal with small k (whose computation is efficient). This result is based on what we call the "fast decaying property", which means the effect of node i's action on node j decays exponentially fast when the graph distance between i and j increases. We will provide both theoretic and empirical studies that show the fast decaying property holds under some assumptions.

We would like to clarify that while our current framework is model-based and requires full knowledge of the model, the framework in this paper sets ground work towards designing efficient model-free reinforcement learning algorithms with optimality guarantees in large multi-agent MDP problems whose state (action) space size is exponentially large in n.

A. Related Work

Partially-Observable MDPs and Stochastic Games. By restricting a_i to depend on only the local state s_i, the problem for each agent can be viewed as a Partially-Observable MDP (POMDP) [14, Chapter 5] if every other agent's policy

Guannan Qu and Na Li are affiliated with John A. Paulson School of Engineering and Applied Sciences at Harvard University. Email: gqu@g.harvard.edu, nali@seas.harvard.edu. The work was supported by NSF 1608509, NSF CAREER 1553407, AFOSR YIP, and ARPA-E through the NODES program.

is fixed. By allowing every agent to explore their optimal strategy, our problem shares many similarities to stochastic games [15], [16]. However, we would like to clarify that we only focus on the cooperative setting where agents cooperatively trying to minimize the total rewards. We also only consider the local policies where $a_i(t)$ is only based on *current* $s_i(t)$. The uncooperative setting and history-based local policies are left for future interest.

Computational Complexity Results. The problem we consider is related to a series of work on the computational complexity of stochastic control where the state space is exponentially large but the problem can be succinctly described. For a overview, see [17, Section 5.2]. As pointed out in [17], apart from rare examples like the multi-armed bandit MDP problem [18], such problems are in general intractable, as demonstrated by restless bandit problems [19] and queuing network control problems [9]. Compared to these work, our model is more structured. In the context of [17], our contribution can be viewed as that we provide another type of succinctly described MDPs and we identify some conditions under which the problem is tractable.

Multi-Agent and Distributed Reinforcement Learning. Our work is also related to the literature on multi-agent and distributed reinforcement learning which has been actively studied [4]. For example, [20]–[23] employ a game-theoretic framework to study multi-agent reinforcement learning. Paper [24] considers a setting where there is a global state which is accessible to every agent and proposes a variant of Q-learning that avoids the use of a exponentially large Q table on the joint action; [25] considers local policy gradient search by modeling each individual agent as a partial observed MDP. A recent line of work [26]–[30] considers the distributed reinforcement learning problem. However, most of them assume there is a global state s which is accessible to all agents; whereas we consider each agent has an individual state s_i and it is hard to access the global state $(s_1, \ldots, s_n)$.

Factored MDP. Another related line of work is factored MDP, see e.g. [31]. In a factored MDP, states follow similar local dependence structure as ours, and the reward is assumed to be additive. In [31], a class of linear function approximators are proposed that are similar in spirit to our "truncation" technique, and policy-iteration and linear programming methods based on the approximator are proposed. However, the theoretic guarantee in [31] is based on a certain projection error, which is not guaranteed to be small, whereas our optimality gap is guaranteed to be small, though we make stronger assumptions compared to [31].

Epidemic Model. Our network model share similarities to epidemic networks [5], [32], [33]; see cf. [5] for a comprehensive review. In a typical epidemic model, each state $s_i(t)$ means node i is infected or not, and an infected node can cause neighboring nodes to be infected with a certain probability. Despite the similarity in dependence structure, our work use very different analysis tools and have different focuses.

Glauber dynamics. Another related line of work is Glauber dynamics [34], [35] in the statistical physics. In a Glauber Dynamics, there is a underlying graph and state $s_i(t+1)$'s distribution depends on the states of i's local neighborhood. Dynamic belief propagation methods (also known as dynamic message passing, dynamic cavity methods) have been proposed to find the marginalized stationary distribution of such models [36]–[40]. However, most of these methods rely on certain ansatz (simplifying assumptions) and there is a lack of theoretic guarantee even when the graph is a tree.

II. Problem Formulation

Consider n agents $\mathcal{N} = \{1, \ldots, n\}$ and each agent is associated with state $s_i \in \{0, 1\}$ and action $a_i \in \{0, 1\}$.[1] The global state is denoted as $s = (s_1, \ldots, s_n) \in \{0, 1\}^n$, and the global action is $a = (a_1, \ldots, a_n) \in \{0, 1\}^n$.

Dependence Graph. The n agents are associated with an underlying one-directional tree $\mathcal{G}$ rooted at node 1. Each node i has 1-hop parent δ_i^1, k-hop parent δ_i^k, and set of children $c_i \subset \mathcal{N}$. We will also use Δ_i^k to denote the path from i to i's k-hop parent, $(i, \delta_i^1, \ldots, \delta_i^k)$, and $\tilde{\Delta}_i^k$ to denote the path from i'th 1-hop parent to k-hop parent, $(\delta_i^1, \ldots, \delta_i^k)$. See Fig. 1 for an illustration. When node i does not have a k-hop parent, then δ_i^k means empty set, Δ_i^k means the path from i to the root node, and $\tilde{\Delta}_i^k$ means the path from i's 1-hop parent to the root node.

Transition Probabilities. We assume that conditioned on the current global state $s(t)$ and global action $a(t)$, the next state $\{s_i(t+1)\}_{i=1}^n$ is generated independently, and the distribution of $s_i(t+1)$ (the next state at i) only depends on $s_i(t)$ (the current state state at i), $a_i(t)$ (the current action at state i), and $s_{\delta_i^1}(t)$ (the current state of the 1-hop parent of i) when i is not the root node. Mathematically, this means the transition probability factorizes in the following manner,

$$P(s(t+1)|s(t), a(t)) = \prod_{i=1}^{n} P_i(s_i(t+1)|s_i(t), s_{\delta_i^1}(t), a_i(t)). \tag{1}$$

Reward. Each agent i is associated with a reward function $r_i(s_i)$ that depends on agent i's state.[2] We will also interpret r_i as a two-dimensional vector in $\mathbb{R}^2$ and we assume that all rewards r_i are bounded above by $\bar{r}$. The total stage reward $r(s)$ for $s = (s_1, \ldots, s_n)$ is the summation of individual reward functions,

$$r(s) = \sum_{i=1}^{n} r_i(s_i). \tag{2}$$

Local Policy. In this paper, we focus on local polices. At node i, a local policy $\zeta_i : \{0, 1\} \to \{0, 1\}$ is a deterministic decision rule that determines the action $a_i(t)$ based on the current state $s_i(t)$ at node i. The set of all possible deterministic local policies at node i is denoted as Γ_i, and $|\Gamma_i| = 4$ as there are 4 distinct maps from $s_i \in \{0, 1\}$ to $a_i \in \{0, 1\}$. We collectively write all agents' local policies in the tuple $\zeta = (\zeta_1, \ldots, \zeta_n) \in \Gamma_1 \times \cdots \times \Gamma_n \triangleq \Gamma$.

Stationary Distribution. In Proposition 1, we show that if the individual transition P_i is ergodic in some sense, then the entire MDP is ergodic for localized policies. Due to space limit, the proof of Proposition 1 is postponed to our online report [41, Appendix-B]. Throughout this paper, we assume the conditions in Proposition 1 hold and the stationary distribution exists and is unique.

[1] In this paper we focus on binary state and action for ease of exposition.

[2] Without too much difficulty, our work can be generalized to the case r_i can depend on $s_i(t), a_i(t), s_i(t+1)$.

Proposition 1. *If for each i, $\forall s_{\delta_i^1} \in \{0,1\}, \zeta_i \in \Gamma_i$, if the Markov Chain on state s_i determined by the transition matrix $P_i(s_i(t+1)|s_i(t), s_{\delta_i^1}(t) = s_{\delta_i^1}, a_i(t) = \zeta_i(s_i(t)))$ is ergodic, then for the transition matrix* (1) *and for all local policies $\zeta \in \Gamma$, the induced Markov chain is ergodic and has a unique stationary distribution π.*

Average Reward. Given a local policy $\zeta = (\zeta_1, \ldots, \zeta_n) \in \Gamma$, the stationary distribution of the states is determined by Proposition 1, which we denote as $\pi(\zeta) \in \mathbb{R}^{2^n}$, where highlight the dependence of the stationary distribution on the policy ζ. We also define $\pi_i(\zeta)$ to be the marginalized stationary distribution for state s_i under policy ζ. Hence, the average reward $R(\zeta)$ under policy ζ (for any initial state $s(0)$) is

$$R(\zeta) = \lim_{T\to\infty} \frac{1}{T} \mathop{\mathbb{E}}_{a(t)=\zeta(s(t))} \left[\sum_{t=0}^{T} r(s(t)) | s(0) \right] = \mathop{\mathbb{E}}_{s\sim\pi(\zeta)} r(s) = \sum_{i=1}^{n} \mathop{\mathbb{E}}_{s_i\sim\pi_i(\zeta)} r_i(s_i) = \sum_{i=1}^{n} r_i^T \pi_i(\zeta). \quad (3)$$

We will also define

$$R_i(\zeta) = \mathop{\mathbb{E}}_{s_i\sim\pi_i(\zeta)} r_i(s_i) = r_i^T \pi_i(\zeta)$$

to be the average reward at node i under policy ζ.

Reward Maximization. We consider the problem of finding a local policy $\zeta = (\zeta_1, \ldots, \zeta_n) \in \Gamma$ such that the average reward is maximized.

$$\max_{\zeta\in\Gamma} R(\zeta) \quad (4)$$

We define tuple $\mathfrak{M} = (n, \mathcal{G}, \{P_i, r_i\}_{i=1}^n)$ as a problem instance, where n is the number of agents, $\mathcal{G}$ is the directed tree graph, P_i is the local transition probability, and r_i is the local reward.

Since $\Gamma = \Gamma_1 \times \Gamma_2 \times \cdots \times \Gamma_n$, we have $|\Gamma| = \prod_{i=1}^n |\Gamma_i| = 4^n$, indicating the search space is exponential in n. Despite this, in the next two chapters, we exploit the structures in the problem as exhibited in the factored transition probability (1) and the additive reward (2) to develop a framework that finds near optimal solutions to (4) in time polynomial in n under some assumptions on the problem instance.

Notation Conventions. s_C for a tuple $C \subset \mathcal{N}$ means the state tuple $(s_i)_{i\in C}$. In particular, we will use $s_{i:j}$ to denote the state tuple $(s_i, s_{i+1}, \ldots, s_j)$; we use $s_{\Delta_i^k}$ for the state tuple on the path from i to i's k-hop parent, $s_{\tilde{\Delta}_i^k}$ for the state tuple on the path from i's 1-hop parent to k-hop parent. We use the same convention for action a, policy ζ and set of policies Γ. For stationary distribution, we use $\pi_{1:n}$, or simply π to denote the stationary distribution for the tuple of all states $(s_1, \ldots, s_n)$, we use π_C where $C \subset \mathcal{N}$ to denote the marginalized stationary distribution for $(s_i)_{i\in C}$. In particular, π_i is the marginalized stationary distribution for node i; $\pi_{i:j}$ is the marginalized stationary distribution for nodes $(i, i+1, \ldots, j)$.

III. Locality-based Local Policy Search

Given problem instance $\mathfrak{M} = (n, \mathcal{G}, \{P_i, r_i\}_{i=1}^n)$, we propose a Locality-based Local Policy Search (LLPS) method to approximately find a local policy $\zeta \in \Gamma$ that solves (4) approximately.

Overview. Recall that the objective function R is a summation of the R_i's (cf. (3)). For each i, the average reward $R_i(\zeta_1, \zeta_2, \ldots, \zeta_n)$ is a function of the policies of all the nodes and even optimizing R_i is difficult due to the exponentially large search space. In this section, we will show that despite the dependence on the local policies of all nodes, $R_i(\cdot)$ has special structure that makes optimization tractable. Firstly in Section III-A, we define an approximated reward function $\hat{R}^k$ by replacing R_i by an approximated local reward function $\hat{R}_i^k$ that only depends on the local policy of i and i's upto k-hop parents, reducing the search space to $O(4^k)$, much more efficient for small $k \ll n$. Secondly in Section III-B, we bound the gap between the true reward function and the approximated reward function. Lastly in Section III-C, we optimize the approximated reward function.

A. Definition of Approximated Reward

For each i, we consider a truncated MDP by looking at a MDP consisting of i and i's upto k-hop parents, while truncating the influence from nodes beyond the k-hop parent. In details, the truncated MDP consists of the path from node i to i's k-hop parent. The set of nodes on the path is $\Delta_i^k = (i, \delta_i^1, \ldots, \delta_i^k)$. See Figure 1 for an illustration. The local transition probabilities of node $(i, \delta_i^1, \delta_i^2, \ldots, \delta_i^{k-1})$ will be the same as the original model; however for the k-hop parent δ_i^k, we will simply draw $s_{\delta_i^k}(t)$ uniformly from $\{0,1\}$ for all t, independent from everything else. More formally, the transition probability for state tuple $s_{\Delta_i^k}$ under action $a_{\Delta_i^k}$ is given by

$$\begin{aligned} &P(s_{\Delta_i^k}(t+1) | s_{\Delta_i^k}(t), a_{\Delta_i^k}(t)) \\ &= P_i(s_i(t+1) | s_i(t), s_{\delta_i^1}(t), a_i(t)) \\ &\quad \times P_{\delta_i^1}(s_{\delta_i^1}(t+1) | s_{\delta_i^1}(t), s_{\delta_i^2}(t), a_{\delta_i^1}(t)) \\ &\quad \times \cdots \times P_{\delta_i^{k-1}}(s_{\delta_i^{k-1}}(t+1) | s_{\delta_i^{k-1}}(t), s_{\delta_i^k}(t), a_{\delta_i^{k-1}}(t)) \\ &\quad \times Uniform(s_{\delta_i^k}(t+1)) \end{aligned}$$

which factorizes in the same way as the original model, except that the last factor $Uniform(s_{\delta_i^k}(t+1))$ is the uniform probability mass function on $\{0,1\}$, showing $s_{\delta_i^k}(t+1)$ is drawn from the uniform distribution independent of everything else. By forcing $s_{\delta_i^k}(t)$ to be drawn independently from a uniform distribution, we are effectively truncating the influence from all upper-stream nodes, and focus on a MDP of $k+1$ nodes. We will refer to the truncated model as $\hat{\mathfrak{M}}_i^k$.

For the truncated MDP $\hat{\mathfrak{M}}_i^k$, given local policy $(\zeta_i, \zeta_{\delta_i^1}, \ldots, \zeta_{\delta_i^k}) = \zeta_{\Delta_i^k}$, the stationary distribution for state tuple $s_{\Delta_i^k}$ is determined, and we define the marginalized stationary distribution at node i as $\hat{\pi}_i^k(\zeta_{\Delta_i^k})$, which is a function of the policies of nodes in Δ_i^k. Further, given $\hat{\pi}_i^k(\zeta_{\Delta_i^k})$, we define the the average reward at i in the truncated model as the following,

$$\hat{R}_i^k(\zeta_{\Delta_i^k}) = \mathop{\mathbb{E}}_{s_i\sim\hat{\pi}_i^k(\zeta_{\Delta_i^k})} r_i(s_i) = r_i^T \hat{\pi}_i^k(\zeta_{\Delta_i^k})$$

The *approximated reward function* $\hat{R}^k(\zeta_1, \ldots, \zeta_n)$ is defined to be the summation of the $\hat{R}_i^k(\zeta_{\Delta_i^k})$.

$$\hat{R}^k(\zeta_1, \ldots, \zeta_n) = \sum_{i=1}^{n} \hat{R}_i^k(\zeta_{\Delta_i^k}) \quad (5)$$

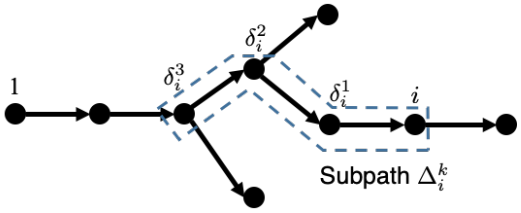

Fig. 1: Illustration of truncated model at node i for $k = 3$.

B. Approximation Error

The idea behind the above truncated model and approximated reward is that, intuitively, the policy at a node far away from i should not have that much effect on i. Therefore, by looking at the truncated model instead of the full model, we should get a good approximation of R_i. In the following, the above intuition will be formally stated as "fast decaying property" in Definition 1, and we show in Lemma 2 that under the fast decaying property, the approximation error will decay exponentially in k. In this section, we will assume the fast decaying property holds. We will study under what conditions the fast decaying property holds in Section IV.

Definition 1. *A Multi-Agent Markov Model $\mathfrak{M}$ has (C, ρ)-fast decaying property if for any i and any local policy $\zeta = (\zeta_1, \ldots, \zeta_n) \in \Gamma$,*

$$\|\pi_i(\zeta) - \hat{\pi}_i^k(\zeta_{\Delta_i^k})\|_1 \le C\rho^k$$

The fast decaying property directly leads to the following bound on the approximation error.

Lemma 2. *If the Multi-Agent Markov Model $\mathfrak{M}$ has (C, ρ)-fast decaying property, then we have for all $\zeta = (\zeta_1, \ldots, \zeta_n)$,*

$$\frac{1}{n}|R(\zeta) - \hat{R}^k(\zeta)| \le C\bar{r}\rho^k$$

Proof. Fixing $\zeta = (\zeta_1, \ldots, \zeta_n)$, then by the definition of $\hat{R}_i^k(\zeta_{\Delta_i^k})$, we have

$$\begin{aligned}|\hat{R}_i^k(\zeta_{\Delta_i^k}) - R_i(\zeta)| &= |r_i^T(\pi_i(\zeta) - \hat{\pi}_i^k(\zeta_{\Delta_i^k}))| \\ &\le \|r_i\|_\infty \|\pi_i(\zeta) - \hat{\pi}_i^k(\zeta_{\Delta_i^k})\|_1 \le \bar{r}C\rho^k\end{aligned}$$

Therefore,

$$\frac{1}{n}|R(\zeta) - \hat{R}^k(\zeta)| \le \frac{1}{n}\sum_{i=1}^n |\hat{R}_i^k(\zeta_{\Delta_i^k}) - R_i(\zeta)| \le C\bar{r}\rho^k$$

□

C. Optimizing Approximated Reward Function

The approximated reward function (5) is a summation of individual $\hat{R}_i^k(\cdot)$, each of which only depend on policies at $k+1$ nodes. Maximization of such functions is equivalent to the MAP (Maximum a Posteriori) problem in graphical models with tree structures [42], and can be efficiently solved by many (mutually-equivalent) methods, like belief propagation, max-product, elimination method, or dynamic programming, cf. [42, Sec. 2.5] for a review. Here we provide a dynamic programming based algorithm for maximizing the approximated reward function.

The algorithm LLPS is provided in Algorithm 1. In the algorithm, BFS means Breadth-First Search (BFS) Ordering [43] of the nodes, in which the root node comes first and parents must precede children, and leaf nodes come at last; reverse BFS means the reverse of a BFS ordering, where children always come before parents.

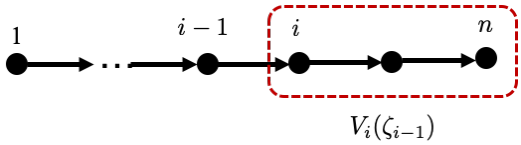

Fig. 2: The case when the graph is a line.

Algorithm 1 LLPS: Locality-based Local Policy Search

for i traverses the tree in reverse BFS ordering **do**
  Collect $V_j(\zeta_{\tilde{\Delta}_j^k})$ for all children $j \in c_i$.
  for $\zeta_{\tilde{\Delta}_i^k} \in \Gamma_{\tilde{\Delta}_i^k}$ **do**
    $V_i(\tilde{\Delta}_i^k) \leftarrow \max_{\zeta_i \in \Gamma_i} \hat{R}_i^k(\zeta_i, \zeta_{\tilde{\Delta}_i^k}) + \sum_{j \in c_i} V_j(\zeta_i, \zeta_{\tilde{\Delta}_i^{k-1}})$
  end for
end for
$\zeta_1^* \leftarrow \arg\max_{\zeta_1 \in \Gamma_1} \hat{R}_1^k(\zeta_1) + \sum_{j \in c_1} V_j(\zeta_1)$
for i traverses the tree except the root in BFS ordering **do**
  $\zeta_i^* \leftarrow \arg\max_{\zeta_i \in \Gamma_i} \hat{R}_i^k(\zeta_i, \zeta_{\tilde{\Delta}_i^k}^*) + \sum_{j \in c_i} V_j(\zeta_i, \zeta_{\tilde{\Delta}_i^{k-1}}^*)$
end for
Output $\zeta^* = (\zeta_1^*, \ldots, \zeta_n^*)$.

Illustration on a line. For ease of presentation and explanation, we provide the special case of the algorithm for a line graph with $k = 1$, where the nodes are labelled in the natural ordering with the root node being 1, and the parent of i is $i-1$, as illustrated in Figure 2. In this case, the approximated reward function can be written as,

$$\hat{R}^1(\zeta) = \hat{R}_1^1(\zeta_1) + \hat{R}_2^1(\zeta_1, \zeta_2) + \ldots \hat{R}_n^1(\zeta_{n-1}, \zeta_n) \qquad (6)$$

and we provide the algorithm below, in two steps.
Step 1. For $i = n$ to 2, construct the following table $V_i(\cdot)$. For every $\zeta_{i-1} \in \Gamma_{i-1}$, set $V_i(\zeta_{i-1}) = \max_{\zeta_i \in \Gamma_i} \hat{R}_i^1(\zeta_{i-1}, \zeta_i) + V_{i+1}(\zeta_i)$, where V_{i+1} is the table constructed for node $i+1$ and $V_{n+1}(\cdot)$ is treated as 0.
Step 2. Compute $\zeta_1^* = \arg\max_{\zeta_i \in \Gamma_i} \hat{R}_1^1(\zeta_1) + V_2(\zeta_1)$. Then, for $i = 2$ to n, compute $\zeta_i^* = \arg\max_{\zeta_i \in \Gamma_i} \hat{R}_i^k(\zeta_{i-1}^*, \zeta_i) + V_{i+1}(\zeta_i)$.

The algorithm is similar to the backtracking method in dynamic programming [10], except here we do induction on a spatial quantity i instead of time. The key to understanding the algorithm is $V_i(\zeta_{i-1})$. In fact, $V_i(\zeta_{i-1})$ is the optimal reward value of the path from i to the leaf node n, if the policy at $i-1$ is ζ_{i-1}. More formally, V_i satisfies the following Lemma.

Lemma 3. *In LLPS for $k = 1$ on a line graph, for $i \ge 2$,*

$$V_i(\zeta_{i-1}) = \max_{\zeta_i, \ldots, \zeta_n} \hat{R}_i^1(\zeta_{i-1}, \zeta_i) + \ldots + \hat{R}_n^1(\zeta_{n-1}, \zeta_n)$$

Proof. The lemma is true for $i = n$ by the definition of $V_n(\cdot)$, and it is true for $i < n$ by a simple induction argument. □

With this lemma, we can easily show that

$$\zeta_1^* = \arg\max_{\zeta_1} \max_{\zeta_2,\ldots,\zeta_n} \hat{R}_1^1(\zeta_1) + \hat{R}_2^1(\zeta_1,\zeta_2) + \ldots \hat{R}_n^1(\zeta_{n-1},\zeta_n)$$
$$= \arg\max_{\zeta_1} \max_{\zeta_2,\ldots,\zeta_n} \hat{R}^1(\zeta_1,\ldots,\zeta_n)$$

and

$$(\zeta_1^*,\ldots,\zeta_i^*) = \arg\max_{(\zeta_1,\ldots,\zeta_i)} \max_{\zeta_{i+1},\ldots,\zeta_n} \hat{R}^1(\zeta_1,\ldots,\zeta_i,\zeta_{i+1},\ldots,\zeta_n)$$

showing $(\zeta_1^*,\ldots,\zeta_n^*)$ is indeed a maximizer of $\hat{R}^1$.

The above arguments also extends to the general case in Algorithm 1, except that the traversing order is (reverse) Breadth First Search, the children of node i may not be $i+1$, and each V_i may depend on the states on i's 1-hop to k-hop parents $\hat{\Delta}_i^k$, not just 1-hop parent. As a result of these reasoning, the algorithm LLPS finds a maximizer of $\hat{R}^k$ for the general tree-graph case. Now we discuss the computational complexity of implementing LLPS.

Complexity of evaluating $\hat{R}_i^k$. By definition, $\hat{R}_i^k(\zeta_{\Delta_i^k}) = r_i^T \hat{\pi}_i^k(\zeta_{\Delta_i^k})$. Evaluation of $\hat{R}_i^k(\zeta_{\Delta_i^k})$ requires computing the stationary distribution of the truncated model, whose state space size is 2^{k+1}, and then marginalize it to $\hat{\pi}_i^k(\zeta_{\Delta_i^k})$, and finally do a simple inner product. The bulk of the computation is calculating the stationary distribution, which takes at most cubic time in state-space size. Therefore, the evaluation of $\hat{R}_i^k$ takes time $O(2^{3k})$.

Time complexity of LLPS. For each $\zeta_{\tilde{\Delta}_i^k}$, the calculation of $V_i(\zeta_{\tilde{\Delta}_i^k})$ takes time $O(2^{3k}+d)$, where d is the maximum degree of the graph. As a result, the implementation of the algorithm takes time $O(n4^k(2^{3k}+d))$, which is linear in n for fixed k and d.

Space complexity of LLPS. For each i, a table $V_i(\cdot)$ of size $O(4^k)$ needs to be maintained. Therefore the space complexity is $O(n4^k)$.

Wrapping up the above results, as well as the approximation gap in Lemma 2, we have the following straightforward proposition.

Proposition 4. *LLPS finds a maximizer $\zeta^* = (\zeta_1^*,\ldots,\zeta_n^*)$ of $\hat{R}^k(\zeta_1,\ldots,\zeta_n)$ within time $O(n4^k(2^{3k}+d))$. If the problem instance has (C,ρ)-fast decaying property, then ζ^* is a $C\rho^k\bar{r}$ approximate optimal solution in the sense that,*

$$\frac{1}{n}\big[\max_{\zeta\in\Gamma} R(\zeta) - R(\zeta^*)\big] \le C\bar{r}\rho^k.$$

Tradeoff between computation and accuracy. It can be seen in Proposition 4 that computational complexity increases exponentially in k, whereas the optimality gap of the solution decays exponentially k. This suggests a tradeoff between computational complexity and desired accuracy level of the solution. In Section V, we will discuss the tradeoff in more detail through numerical analysis.

IV. Fast Decaying Property

In this section, we provide conditions under which the (C,ρ)-fast decaying holds. To state our theoretic results, we parameterize the transition probabilities as follows, where the ordering of the rows and columns is such that the first row (column) corresponds to state 0, and the second row (column) corresponds to state 1.

$$P_i(s_i(t+1)|s_i(t),s_{\delta_i^1}(t)=0,a_i(t)=0) = \begin{bmatrix} e_i & 1-e_i \\ f_i & 1-f_i \end{bmatrix}$$
$$P_i(s_i(t+1)|s_i(t),s_{\delta_i^1}(t)=1,a_i(t)=0) = \begin{bmatrix} g_i & 1-g_i \\ h_i & 1-h_i \end{bmatrix}$$
$$P_i(s_i(t+1)|s_i(t),s_{\delta_i^1}(t)=0,a_i(t)=1) = \begin{bmatrix} e_i' & 1-e_i' \\ f_i' & 1-f_i' \end{bmatrix}$$
$$P_i(s_i(t+1)|s_i(t),s_{\delta_i^1}(t)=1,a_i(t)=1) = \begin{bmatrix} g_i' & 1-g_i' \\ h_i' & 1-h_i' \end{bmatrix}$$

For the root node $i=1$ who has no parent, we will simply set $e_1=g_1$, $f_1=h_1$, $e_1'=g_1'$ and $f_1'=h_1'$. Our results can be stated as follows,

Lemma 5. *Given problem instance $\mathfrak{M}$. If for all i, the parameters satisfy,*

$$e_i - f_i = g_i - h_i \in (-1,1), \quad e_i - f_i' = g_i - h_i' \in (-1,1)$$
$$e_i' - f_i = g_i' - h_i \in (-1,1), \quad e_i' - f_i' = g_i' - h_i' \in (-1,1)$$

and further if

$$\max\left\{ \left|\frac{e_i-g_i}{1-(e_i-f_i)}\right|, \left|\frac{e_i-g_i}{1-(e_i-f_i')}\right|, \right.$$
$$\left. \left|\frac{e_i'-g_i'}{1-(e_i'-f_i)}\right|, \left|\frac{e_i'-g_i'}{1-(e_i'-f_i')}\right| \right\} \le \rho \in (0,1)$$

Then, the problem instance is $(2,\rho)$-fast decaying.

The reason we impose the assumption on the transition probability parameters is that it greatly simplifies the proof, which we will provide in Section IV-A. However, we conjecture that the fast decaying property holds for much general range of parameters. We will conduct numerical simulations in Section IV-B that certifies fast decaying property for randomly generated transition probability parameters.

A. Proof of Lemma 5

We notice that under the tree dependence structure, fixing any local policy $\zeta = (\zeta_1,\ldots,\zeta_n)$, the states along the path from the root to node i form a self-complete Markov chain, and the marginalized stationary distribution $\pi_i(\zeta)$ at node i only depends on the Markov Chain on the path. Further, the truncated model at node i is a subpath of the path from root to i, where the nodes beyond i'th k-hop parent are "truncated". Therefore, to study the fast decaying property it suffices to study the case where the dependence graph is a line rooted at 1 and with leaf node i. We label the nodes with the nature ordering on the line.

In the line structure, the parent of node j is $j-1$ (for $j \ge 2$). We fix a local policy $\zeta = (\zeta_1,\ldots,\zeta_i)$ and from now on, drop the dependence on policy. Let the local transition probabilities induced by the policy be,

$$P_j(s_j(t+1)|s_j(t),s_{j-1}(t)=0) = \begin{bmatrix} \alpha_j & 1-\alpha_j \\ \beta_j & 1-\beta_j \end{bmatrix} \quad (7a)$$
$$P_j(s_j(t+1)|s_j(t),s_{j-1}(t)=1) = \begin{bmatrix} \gamma_j & 1-\gamma_j \\ \omega_j & 1-\omega_j \end{bmatrix} \quad (7b)$$
$$P_1(s_1(t+1)|s_1(t)) = \begin{bmatrix} \alpha_1 & 1-\alpha_1 \\ \beta_1 & 1-\beta_1 \end{bmatrix} \quad (7c)$$

where we also introduce $\gamma_1 = \alpha_1$ and $\omega_1 = \beta_1$ for notational consistency. The value of $(\alpha_j,\beta_j,\gamma_j,\omega_j)$ depends on the local policy ζ_j and the local transition parameters $(e_j,f_j,g_j,h_j,e_j',f_j',g_j',h_j')$. For example, if ζ_j is such that

action $a_j = 0$ regardless of s_j, then $(\alpha_j, \beta_j, \gamma_j, \omega_j) = (e_j, f_j, g_j, h_j)$; if the ζ_j is such that $a_j = 0$ when $s_j = 0$, $a_j = 1$ when $s_j = 1$, then $(\alpha_j, \beta_j, \gamma_j, \omega_j) = (e_j, f'_j, g_j, h'_j)$. Under the assumptions in Lemma 5, it is easy to check that regardless of the policy γ_j, there exists some $\mu_j \in (-1, 1)$ s.t. $(\alpha_j, \beta_j, \gamma_j, \omega_j)$ satisfies,

$$\alpha_j - \beta_j = \gamma_j - \omega_j = \mu_j \tag{8}$$

$$\left|\frac{\alpha_j - \gamma_j}{1 - \mu_j}\right| \le \rho \tag{9}$$

We denote the stationary distribution of the model as $\pi_{1:i}$, and the marginalized distribution at node j as π_j for $1 \le j \le i$.

The statement of Lemma 5 bounds marginalized stationary distribution π_i and that of the truncated model. Recall that, the truncated model is on state tuple $s_{i-k:i}$, i.e. on the path from i to i's k-hop parent, node $i-k$. Let the local transition probabilities be $\hat{P}_j(\cdot)$ for $i-k \le j \le i$, and let them be parameterized by $(\hat{\alpha}_j, \hat{\beta}_j, \hat{\gamma}_j, \hat{\omega}_j)_{i-k \le j \le i}$ in the same fashion as (7). By the way the truncated model is constructed, we have $\hat{\alpha}_j = \alpha_j, \hat{\beta}_j = \beta_j, \hat{\gamma}_j = \gamma_j, \hat{\omega}_j = \omega_j$ for $i-k+1 \le j \le i$; and for $j = i-k$,

$$\hat{\alpha}_{i-k} = \hat{\beta}_{i-k} = \hat{\gamma}_{i-k} = \hat{\omega}_{i-k} = \frac{1}{2}.$$

Setting $\hat{\mu}_j = \mu_j$ for $i-k < j \le i$ and $\hat{\mu}_{i-k} = 0$, we have for $i-k \le j \le i$,

$$\hat{\alpha}_j - \hat{\beta}_j = \hat{\gamma}_j - \hat{\omega}_j = \hat{\mu}_j \in (-1, 1) \tag{10}$$

Let the stationary distribution of truncated model be $\hat{\pi}_{i-k:i}$, and let the marginalized stationary distribution at node j be $\hat{\pi}_j$ for $i-k \le j \le i$.

We now provide a recursive formula on the marginalized stationary distribution that works for both the original model and the truncated model.

Lemma 6. *For any Markov chain on state tuple $s_{1:i} = (s_1, \ldots, s_i)$ that factorizes in in the following way,*

$$P(s_{1:i}(t+1)|s_{1:i}(t)) = P_1(s_1(t+1)|s_1(t)) \prod_{j=2}^{i} P_j(s_j(t+1)|s_j(t), s_{j-1}(t))$$

where P_j is parameterized by,

$$P_j(s_j(t+1) = \cdot | s_j(t) = \cdot, s_{j-1}(t) = 0) = \begin{bmatrix} \alpha_j & 1-\alpha_j \\ \beta_j & 1-\beta_j \end{bmatrix}$$

$$P_j(s_j(t+1) = \cdot | s_j(t) = \cdot, s_{j-1}(t) = 1) = \begin{bmatrix} \gamma_j & 1-\gamma_j \\ \omega_j & 1-\omega_j \end{bmatrix}$$

$$P_1(s_1(t+1) = \cdot | s_1(t) = \cdot) = \begin{bmatrix} \alpha_1 & 1-\alpha_1 \\ \beta_1 & 1-\beta_1 \end{bmatrix}$$

Assume $\alpha_j - \beta_j = \gamma_j - \omega_j = \mu_j \in (-1, 1)$. For $1 \le j \le i$, let b_j be the probability of $s_j = 1$ under the stationary distribution. Then, for any $1 \le k \le i-1$,

$$b_i = b_{i-k} \prod_{j=i-k+1}^{i} \frac{\alpha_j - \gamma_j}{1 - \mu_j} + \sum_{j=i-k+1}^{i} \frac{1-\alpha_j}{1-\mu_j} \prod_{\ell=j+1}^{i} \frac{\alpha_\ell - \gamma_\ell}{1 - \mu_\ell}$$

Recall that π_j is the stationary distribution of s_j in the original model. We use the notation in Lemma 6 to write it as $\pi_j = [1 - b_j, b_j]$. Similarly, we write $\hat{\pi}_j = [1 - \hat{b}_j, \hat{b}_j]$ for the truncated model. Then Lemma 6 shows that for the original model,

$$b_i = b_{i-k} \prod_{j=i-k+1}^{i} \frac{\alpha_j - \gamma_j}{1 - \mu_j} + \sum_{j=i-k+1}^{i} \frac{1-\alpha_j}{1-\mu_j} \prod_{\ell=j+1}^{i} \frac{\alpha_\ell - \gamma_\ell}{1 - \mu_\ell}$$

and for the truncated model,

$$\hat{b}_i = \hat{b}_{i-k} \prod_{j=i-k+1}^{i} \frac{\hat{\alpha}_j - \hat{\gamma}_j}{1 - \hat{\mu}_j} + \sum_{j=i-k+1}^{i} \frac{1-\hat{\alpha}_j}{1-\hat{\mu}_j} \prod_{\ell=j+1}^{i} \frac{\hat{\alpha}_\ell - \hat{\gamma}_\ell}{1 - \hat{\mu}_\ell}$$

Since for $j \ge i-k+1$, $(\alpha_j, \beta_j, \gamma_j, \omega_j, \mu_j) = (\hat{\alpha}_j, \hat{\beta}_j, \hat{\gamma}_j, \hat{\omega}_j, \hat{\mu}_j)$, we have

$$|b_i - \hat{b}_i| = |b_{i-k} - \hat{b}_{i-k}| \prod_{j=i-k+1}^{i} \left|\frac{\alpha_j - \gamma_j}{1 - \mu_j}\right| \le \rho^k$$

where we have used (9). As a result, $\|\pi_i - \hat{\pi}_i\|_1 \le 2\rho^k$, which concluces the proof.

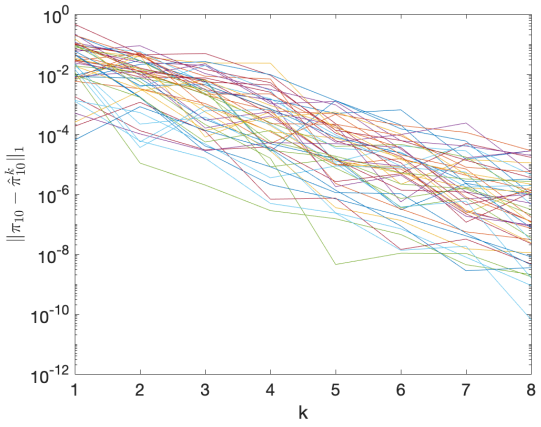

Fig. 3: Monte Carlo simulations showing $\|\pi_{10} - \hat{\pi}_{10}^k\|_1$ as a function of k.

B. An Empirical Study of Fast Decaying Property

We note that Lemma 5 requires the parameters $(e_i, f_i, g_i, h_i, e'_i, f'_i, g'_i, h'_i)$ to satisfy certain equality constraint. However, we conjecture that the fast decaying property holds for much more general parameters. To this end, we do Monte-Carlo simulations to test our conjecture. We fix the graph to be a line consisting of 10 nodes. For each run, we generate all parameters uniformly from $[0, 1]$, and also select ζ_i uniformly random from one of the 4 possible maps from $s_i \in \{0, 1\}$ to $a_i \in \{0, 1\}$. We then compute the marginalized stationary distribution at the leaf node π_{10}, as well the the marginal distribution of the truncated model $\hat{\pi}_{10}^k$, for $k = 1, \ldots, 8$, and plot the gap $\|\pi_{10} - \hat{\pi}_{10}^k\|_1$ as a function of k. We do 50 runs with random parameters and policies. The results are given in Figure 3. The figure confirms that the error $\|\pi_{10} - \hat{\pi}_{10}^k\|_1$ decays exponentially fast in k.

V. Case Studies

In this section, we conduct simulations on our algorithm. We let the dependence graph be the 9-node graph in Fig. 1. We draw the local transition probability parameters $(e_i, f_i, g_i, h_i, e'_i, f'_i, g'_i, h'_i)$ independently from the uniform distribution on $[0, 1]$, and we draw each component of the

TABLE I: Reward and running time results for exhaustive search and LLPS.

Algorithm	Reward	Optimality Gap	Running Time
Exhaustive Search	4.2578	0	1804.9s
LLPS $k = 1$	4.2123	0.0456	0.6049s
LLPS $k = 2$	4.2563	0.0016	0.2558s
LLPS $k = 3$	4.2578	0	1.6037s
LLPS $k = 4$	4.2578	0	5.9123s
LLPS $k = 5$	4.2578	0	17.9991s
LLPS $k = 6$	4.2578	0	46.1841s
LLPS $k = 7$	4.2578	0	47.3225s

reward vector r_i from the uniform distribution on $[0, 1]$. For a model of this size, we are able to find the exact optimal solution through exhaustive search. We then run our algorithm LLPS for $k = 1, 2, \ldots, 7$. In Table I, we provide the reward achieved by the optimal local policy (obtained through exhaustive search) and the reward achieved by the local policy output by LLPS for different k, as well as the optimality gap and the running time[3] of the respective algorithm. As shown in Table I, our algorithm is several magnitudes faster than the exhaustive search, and increasing k also causes our algorithm to be slower. Further, the reward achieves by our algorithm is very close to the optimal reward for $k = 1, 2$, and our algorithm achieves the exact optimal reward for $k \geq 3$. One possible explanation is that for large k, the gap between the approximate reward and the true reward, becomes smaller than the gap of the optimal reward and the reward of the second best policy. As a result, the optimizer of the approximate reward and the true reward function coincide.

VI. Conclusion

We study a multi-agent MDP problem with an exponentially large state and action space, and we aim to find the optimal local policy when the MDP has local dependence structure. We propose a Locality-based Local Policy Search method and show it can efficiently find a near-optimal local policy when the MDP has a "fast-decaying property". We show the fast decaying property holds when the dependence structure is a one-directional tree and the transition probabilities satisfy certain assumptions. However, our Monte-Carlo simulations in IV-B show that the fast decaying property appears to hold under very general conditions. Future work includes gaining deeper understanding on the fast decaying property and show the fast decaying property holds under more general conditions on the parameters and for more general dependence graphs. Further, we are interested in utilizing the fast decaying property to design model-free learning algorithms with provable near-optimal guarantees.

References

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

[2] I. Borovikov, Y. Zhao, A. Beirami, J. Harder, J. Kolen, J. Pestrak, J. Pinto, R. Pourabolghasem, H. Chaput, M. Sardari, L. Lin, N. Aghdaie, and K. Zaman, "Winning isn't everything: Training agents to playtest modern games," in *AAAI 2019 Workshop on Reinforcement Learning in Games*, 01 2019.

[3] Z. Wu, Q.-S. Jia, and X. Guan, "Optimal control of multiroom hvac system: An event-based approach," *IEEE Transactions on Control Systems Technology*, vol. 24, no. 2, pp. 662–669, 2016.

[3]All numerical simulations are done on MATLAB2018b using a MacBook Pro Early 2015 Model with 2.7GHz Intel Core i5.

[4] L. Bu, R. Babu, B. De Schutter *et al.*, "A comprehensive survey of multiagent reinforcement learning," *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, vol. 38, no. 2, pp. 156–172, 2008.

[5] W. Mei, S. Mohagheghi, S. Zampieri, and F. Bullo, "On the dynamics of deterministic epidemic propagation over networks," *Annual Reviews in Control*, vol. 44, pp. 116–128, 2017.

[6] D. Chakrabarti, Y. Wang, C. Wang, J. Leskovec, and C. Faloutsos, "Epidemic thresholds in real networks," *ACM Transactions on Information and System Security (TISSEC)*, vol. 10, no. 4, p. 1, 2008.

[7] M. Llas, P. M. Gleiser, J. M. López, and A. Díaz-Guilera, "Nonequilibrium phase transition in a model for the propagation of innovations among economic agents," *Physical Review E*, vol. 68, no. 6, p. 066101, 2003.

[8] W. Vogels, R. van Renesse, and K. Birman, "The power of epidemics: Robust communication for large-scale distributed systems," *SIGCOMM Comput. Commun. Rev.*, vol. 33, no. 1, pp. 131–135, Jan. 2003. [Online]. Available: http://doi.acm.org/10.1145/774763.774784

[9] C. H. Papadimitriou and J. N. Tsitsiklis, "The complexity of optimal queuing network control," *Mathematics of Operations Research*, vol. 24, no. 2, pp. 293–305, 1999.

[10] D. P. Bertsekas, *Dynamic Programming and Optimal Control, Vol. II*, 3rd ed. Athena Scientific, 2007.

[11] R. S. Sutton, A. G. Barto *et al.*, *Introduction to reinforcement learning*. MIT press Cambridge, 1998, vol. 135.

[12] V. R. Konda and J. N. Tsitsiklis, "Actor-critic algorithms," in *Advances in neural information processing systems*, 2000, pp. 1008–1014.

[13] J. N. Tsitsiklis and B. Van Roy, "Analysis of temporal-diffference learning with function approximation," in *Advances in neural information processing systems*, 1997, pp. 1075–1081.

[14] D. P. Bertsekas, *Dynamic programming and optimal control*, 3rd ed., ser. Athena Scientific optimization and computation series. Belmont, Mass.: Athena Scientific, 2005.

[15] L. S. Shapley, "Stochastic games," *Proceedings of the National Academy of Sciences*, vol. 39, no. 10, pp. 1095–1100, 1953.

[16]

[17] V. D. Blondel and J. N. Tsitsiklis, "A survey of computational complexity results in systems and control," *Automatica*, vol. 36, no. 9, pp. 1249–1274, 2000.

[18] J. Gittins, K. Glazebrook, and R. Weber, *Multi-armed bandit allocation indices*. John Wiley & Sons, 2011.

[19] P. Whittle, "Restless bandits: Activity allocation in a changing world," *Journal of applied probability*, vol. 25, no. A, pp. 287–298, 1988.

[20] M. L. Littman, "Markov games as a framework for multi-agent reinforcement learning," in *Machine learning proceedings 1994*. Elsevier, 1994, pp. 157–163.

[21] C. Claus and C. Boutilier, "The dynamics of reinforcement learning in cooperative multiagent systems," *AAAI/IAAI*, vol. 1998, pp. 746–752, 1998.

[22] M. L. Littman, "Value-function reinforcement learning in markov games," *Cognitive Systems Research*, vol. 2, no. 1, pp. 55–66, 2001.

[23] 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.

[24] M. Lauer and M. Riedmiller, "An algorithm for distributed reinforcement learning in cooperative multi-agent systems," in *In Proceedings of the Seventeenth International Conference on Machine Learning*. Citeseer, 2000.

[25] L. Peshkin, K.-E. Kim, N. Meuleau, and L. P. Kaelbling, "Learning to cooperate via policy search," in *Proceedings of the Sixteenth conference on Uncertainty in artificial intelligence*. Morgan Kaufmann Publishers Inc., 2000, pp. 489–496.

[26] K. Zhang, Z. Yang, H. Liu, T. Zhang, and T. Başar, "Fully decentralized multi-agent reinforcement learning with networked agents," *arXiv preprint arXiv:1802.08757*, 2018.

[27] 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.

[28] S. V. Macua, J. Chen, S. Zazo, and A. H. Sayed, "Distributed policy evaluation under multiple behavior strategies," *IEEE Transactions on Automatic Control*, vol. 60, no. 5, pp. 1260–1274, 2015.

[29] A. Mathkar and V. S. Borkar, "Distributed reinforcement learning via gossip," *IEEE Transactions on Automatic Control*, vol. 62, no. 3, pp. 1465–1470, 2017.

[30] H.-T. Wai, Z. Yang, Z. Wang, and M. Hong, "Multi-agent reinforcement learning via double averaging primal-dual optimization," *arXiv preprint arXiv:1806.00877*, 2018.

[31] 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.

[32] E. Cator and P. Van Mieghem, "Second-order mean-field susceptible-infected-susceptible epidemic threshold," *Physical review E*, vol. 85, no. 5, p. 056111, 2012.
[33] F. D. Sahneh, C. Scoglio, and P. Van Mieghem, "Generalized epidemic mean-field model for spreading processes over multilayer complex networks," *IEEE/ACM Transactions on Networking (TON)*, vol. 21, no. 5, pp. 1609–1620, 2013.
[34] A. Y. Lokhov, M. Mézard, and L. Zdeborová, "Dynamic message-passing equations for models with unidirectional dynamics," *Phys. Rev. E*, vol. 91, p. 012811, Jan 2015. [Online]. Available: https://link.aps.org/doi/10.1103/PhysRevE.91.012811
[35] M. Mezard and A. Montanari, *Information, physics, and computation*. Oxford University Press, 2009.
[36] A. Y. Lokhov, "Dynamic cavity method and problems on graphs," Theses, Université Paris Sud - Paris XI, Nov. 2014. [Online]. Available: https://tel.archives-ouvertes.fr/tel-01127103
[37] I. Neri and D. Bollé, "The cavity approach to parallel dynamics of ising spins on a graph," *Journal of Statistical Mechanics: Theory and Experiment*, vol. 2009, no. 08, p. P08009, 2009.
[38] Y. Kanoria, A. Montanari *et al.*, "Majority dynamics on trees and the dynamic cavity method," *The Annals of Applied Probability*, vol. 21, no. 5, pp. 1694–1748, 2011.
[39] E. Aurell and H. Mahmoudi, "Dynamic mean-field and cavity methods for diluted ising systems," *Physical Review E*, vol. 85, no. 3, p. 031119, 2012.
[40] F. Altarelli, A. Braunstein, L. DallAsta, and R. Zecchina, "Optimizing spread dynamics on graphs by message passing," *Journal of Statistical Mechanics: Theory and Experiment*, vol. 2013, no. 09, p. P09011, 2013.
[41] G. Qu and N. Li, "Exploiting fast decaying and locality in multi-agent mdp with tree dependence structure," https://scholar.harvard.edu/files/gqu/files/cdc2019full.pdf, accessed: 2019-03-15.
[42] M. J. Wainwright, M. I. Jordan *et al.*, "Graphical models, exponential families, and variational inference," *Foundations and Trends® in Machine Learning*, vol. 1, no. 1–2, pp. 1–305, 2008.
[43] S. S. Skiena, *Graph Traversal*. London: Springer London, 2008, pp. 145–190. [Online]. Available: https://doi.org/10.1007/978-1-84800-070-4_5

APPENDIX

A. Proof of Lemma 6

Firstly, we note that for any $1 \leq \ell \leq i$, the states $(s_1, \ldots, s_\ell)$ form a self-complete Markov chain, with transition matrix $P_{1:\ell}$ given by $P_{1:\ell}(s_{1:\ell}(t+1)|s_{1:\ell}(t)) = P_1(s_1(t+1)|s_1(t)) \prod_{j=2}^{\ell} P_j(s_j(t+1)|s_j(t), s_{j-1}(t))$ and we denote the stationary distribution as $\pi_{1:\ell} \in \mathbb{R}^{2^\ell}$.

For any $1 \leq \ell \leq i$, the rows and columns of matrix $P_{1:\ell} \in \mathbb{R}^{2^\ell \times 2^\ell}$, as well as the entries of the stationary distribution $\pi_{1:\ell}$, are indexed by state tuple $(s_1, s_2, \ldots, s_\ell) \in \{0,1\}^\ell$. We use the following ordering that maps each state tuple $(s_1, s_2, \ldots, s_\ell)$ to a row (column) number in $\{1, \ldots, 2^\ell\}$.

$$\{0,1\}^\ell \ni (s_1, s_2, \ldots, s_\ell) \mapsto 1 + \sum_{j=1}^{\ell} 2^{j-1} s_j \in \{1, \ldots, 2^\ell\}$$

The above ordering means that, for the 2^ℓ rows (columns), designate the upper (left) half to $s_\ell = 0$ and the lower (right) half to $s_\ell = 1$; within each of the halves, designate the upper (left) quarter to $s_{\ell-1} = 0$ and the lower (right) quarter to $s_{\ell-1} = 0$. Continue this procedure we can uniquely identify each row (column) with a state tuple $(s_1, \ldots, s_\ell)$.

We divide $P_{1:i-1}$ into the upper half and lower half as in the following.

$$P_{1:i-1} = \begin{bmatrix} P^-_{1:i-1} \\ P^+_{1:i-1} \end{bmatrix}$$

It is easy to check,

$$P_{1:i} = \begin{bmatrix} \alpha_i P^-_{1:i-1} & (1-\alpha_i) P^-_{1:i-1} \\ \gamma_i P^+_{1:i-1} & (1-\gamma_i) P^+_{1:i-1} \\ \beta_i P^-_{1:i-1} & (1-\beta_i) P^-_{1:i-1} \\ \omega_i P^+_{1:i-1} & (1-\omega_i) P^+_{1:i-1} \end{bmatrix}.$$

We do the following similarity transform.

$$\begin{aligned} P_{1:i} &= \overbrace{\begin{bmatrix} I & 0 \\ I & I \end{bmatrix}}^{:=T} \begin{bmatrix} \begin{bmatrix} P^-_{1:i-1} \\ P^+_{1:i-1} \end{bmatrix} & \begin{bmatrix} (1-\alpha_i) P^-_{1:i-1} \\ (1-\gamma_i) P^+_{1:i-1} \end{bmatrix} \\ 0 & \begin{bmatrix} (\alpha_i - \beta_i) P^-_{1:i-1} \\ (\gamma_i - \omega_i) P^+_{1:i-1} \end{bmatrix} \end{bmatrix} \\ &\quad \times \begin{bmatrix} I & 0 \\ -I & I \end{bmatrix} \\ &= T \overbrace{\begin{bmatrix} P_{1:i-1} & \begin{bmatrix} (1-\alpha_i) P^-_{1:i-1} \\ (1-\gamma_i) P^+_{1:i-1} \end{bmatrix} \\ 0 & \mu_i P_{1:i-1} \end{bmatrix}}^{:=\tilde{P}_{1:i}} T^{-1} = T \tilde{P}_{1:i} T^{-1} \end{aligned}$$

As such, $P_{1:i}$ is similar to $\tilde{P}_{1:i}$. If row vector π is an left eigenvector of $\tilde{P}_{1:i}$ with eigenvalue λ, then $\pi T^{-1} P_{1:i} = \pi T^{-1} T \tilde{P}_{1:i} T^{-1} = \lambda \pi T^{-1}$, i.e. πT^{-1} is an left eigenvector of $P_{1:i}$ with eigenvalue λ. With this relation, we can study the stationary distribution of $P_{1:i}$ through the eigenvector of $\tilde{P}_{1:i}$. Let $\pi_{1:i-1}$ be the stationary distribution of $P_{1:i-1}$. Let

$$\pi^+_{1:i} = \pi_{1:i-1} \begin{bmatrix} (1-\alpha_i) P^-_{1:i-1} \\ (1-\gamma_i) P^+_{1:i-1} \end{bmatrix} (I - \mu_i P_{1:i-1})^{-1} \in \mathbb{R}^{2^{i-1}}$$

Then, let $\tilde{\pi}_{1:i} = [\pi_{1:i-1}, \pi^+_{1:i}] \in \mathbb{R}^{2^i}$. We can verify

$$\begin{aligned} \tilde{\pi}_{1:i} \tilde{P}_{1:i} &= [\pi_{1:i-1}, \pi^+_{1:i}] \begin{bmatrix} P_{1:i-1} & \begin{bmatrix} (1-\alpha_i) P^-_{1:i-1} \\ (1-\gamma_i) P^+_{1:i-1} \end{bmatrix} \\ 0 & \mu_i P_{1:i-1} \end{bmatrix} \\ &= [\pi_{1:i-1}, \pi^+_{1:i}(I - \mu_i P_{1:i-1}) + \pi^+_{1:i} \mu_i P_{1:i-1}] \\ &= \tilde{\pi}^T_{1:i} \end{aligned}$$

Therefore, $\tilde{\pi}_{1:i}$ is the left eigenvector of $\tilde{P}_{1:i}$ corresponding to eigenvalue 1, and as a result $\pi_{1:i} = \tilde{\pi}_{1:i} T^{-1} = [\pi_{1:i-1} - \pi^+_{1:i}, \pi^+_{1:i}]$ is the stationary distribution under $P_{1:i}$.

Notice that b_i, the probability of $s_i = 1$ under the stationary distribution, is $b_i = \pi^+_{1:i}\mathbf{1}$ (where $\mathbf{1}$ is the all one column vector of appropriate dimension), the summation of the right half of $\pi_{1:i}$ per the way that the states are ordered. Therefore,

$$\begin{aligned} b_i &= \pi^+_{1:i}\mathbf{1} \\ &\overset{(a)}{=} \pi_{1:i-1} \begin{bmatrix} (1-\alpha_i) P^-_{1:i-1} \\ (1-\gamma_i) P^+_{1:i-1} \end{bmatrix} (I - \mu_i P_{1:i-1})^{-1} \mathbf{1} \\ &\overset{(b)}{=} \frac{1}{1-\mu_i} \pi_{1:i-1} \begin{bmatrix} (1-\alpha_i) P^-_{1:i-1} \\ (1-\gamma_i) P^+_{1:i-1} \end{bmatrix} \mathbf{1} \\ &\overset{(c)}{=} \frac{1}{1-\mu_i} \pi_{1:i-1} \begin{bmatrix} (1-\alpha_i)\mathbf{1} \\ (1-\gamma_i)\mathbf{1} \end{bmatrix} \\ &\overset{(d)}{=} \frac{1-\alpha_i}{1-\mu_i}(1 - b_{i-1}) + \frac{1-\gamma_i}{1-\mu_i} b_{i-1} \\ &= \frac{1-\alpha_i}{1-\mu_i} + \frac{\alpha_i - \gamma_i}{1-\mu_i} b_{i-1} \end{aligned} \tag{11}$$

where (a) is due to the definition of $\pi^+_{1:i}$, (c) is due to that each row of $P^-_{1:i-1}$ and $P^+_{1:i-1}$ sum up to 1; (d) is due to the definition of b_{i-1}. For (b), it is due to, since $\mu_i \in (-1, 1)$,

$$(I - \mu_i P_{1:i-1})^{-1} \mathbf{1} = \mathbf{1} + \sum_{\tau=1}^{\infty} \mu_i^\tau P^\tau_{1:i-1} \mathbf{1} = \frac{1}{1-\mu_i} \mathbf{1}.$$

We can expand (11) recursively and get the desired result.

$$b_i = b_{i-k} \prod_{j=i-k+1}^{i} \frac{\alpha_j - \gamma_j}{1-\mu_j} + \sum_{j=i-k+1}^{i} \frac{1-\alpha_j}{1-\mu_j} \prod_{\ell=j+1}^{i} \frac{\alpha_\ell - \gamma_\ell}{1-\mu_\ell}$$