Decentralized Planning for Non-Dedicated Agent Teams with Submodular Rewards in Uncertain Environments

Pritee Agrawal
Singapore Management University
Singapore 188065
priteea.2013@phdis.smu.edu.sg

Pradeep Varakantham
Singapore Management University
Singapore 188065
pradeepv@smu.edu.sg

William Yeoh
Washington University in St. Louis
St. Louis, MO 63130, USA
wyeoh@wustl.edu

Abstract

Decentralized planning under uncertainty for agent teams is a problem of interest in many domains including (but not limited to) disaster rescue, sensor networks and security patrolling. Decentralized MDPs, Dec-MDPs have traditionally been used to represent such decentralized planning under uncertainty problems. However, in many domains, agents may not be dedicated to the team for the entire time horizon. For instance, due to limited availability of resources, it is quite common for police personnel leaving patrolling teams to attend to accidents. Such non-dedication can arise due to the emergence of higher priority tasks or damage to existing agents. However, there is very limited literature dealing with handling of non-dedication in decentralized settings. To that end, we provide a general model to represent problems dealing with cooperative and decentralized planning for non-dedicated agent teams. We also provide two greedy approaches (an offline one and an offline-online one) that are able to deal with agents leaving the team in an effective and efficient way by exploiting the submodularity property. Finally, we demonstrate that our approaches are able to obtain more than 90% of optimal solution quality on benchmark problems from the literature.

1 INTRODUCTION

Decentralized planning for a team of agents is required in a wide variety of problems such as target tracking by a team of sensors [Nair *et al.*, 2005; Kumar and Zilberstein, 2011; Chapman and Varakantham, 2014], securing targets from unknown attackers using a team of defenders [Brown *et al.*, 2014; Shieh *et al.*, 2014; Varakantham *et al.*, 2013], rescuing of victims by a team of robots during disaster [Melo and Veloso, 2011; Varakantham *et al.*, 2009, 2014; Velagapudi *et al.*, 2011] and analysing underwater samples using a team of underwater vehicles [Yin and Tambe, 2011], etc.

These domains have the following common characteristics: (a) A decentralized team of agents (sensors, ambulances, fire-trucks, etc.) that coordinate plans to achieve a goal; (b) There is transition uncertainty in planning problems of individual agents, either due to travelling on roads (due to traffic) or due to physical constraints (sensors, robots, etc.) (c) The agents are independent and collaborate through a global reward (save victims, prevent attacks, etc.); and most importantly (d) The individual agents have a chance of leaving the team at any time step to address a higher priority task. For example, in the case of patrolling, agents (e.g., coast guard boats, traffic police) can be forced to leave their assignment to attend to an accident or incident (e.g., incursion, smuggling, accident).

We are interested in application problems with the above mentioned characteristics and specifically teams in which agents are non-dedicated and may leave the team due to higher priority tasks or damage to agents. Non-dedication has been explored by Agrawal and Varakantham [2017] for centralized planning. Further, Shieh *et al.* [2014] considered non-dedicated teams in decentralized settings, but they provide an exhaustive offline approach that is not scalable. Our contributions differ in providing quick solutions by exploiting reward submodularity for decentralized planning such that remaining agents can reconfigure their policies to attend to tasks of leaving agents.

Submodularity has been exploited by Kumar and Zilberstein [2009]; Satsangi *et al.* [2015] for centralized planning model while Kumar *et al.* [2017] focus on decentralized planning in cooperative teams. Our contributions differ from this line of work in considering non dedicated agent teams with multiple agent exits from the team while still considering joint submodular reward functions for

decentralized planning. Another closely related thread of research is on adaptive submodularity [Golovin and Krause, 2011], where a sequence of decisions are taken by accounting for the observations of past decisions. Our work differs from this thread because we consider a multi-stage submodular problem (which introduces a partition matroid constraint) where at every decision epoch, we have a new submodular problem[1] with fewer number of agents.

To that end, we provide a general model to represent the class of problems dealing with a team of independently collaborating non-dedicated agents. A key contribution of our work lies in establishing connections between non-dedicated agent teams and submodularity. We show that with monotone submodular reward functions subject to the matroid constraint, greedy solutions computed at every decision epoch are still submodular with fewer number of agents and provide an a priori guarantee of at least 50% from the optimal and much better posterior guarantees. Another main contribution includes our two greedy approaches to efficiently deal with agent exits before the end of horizon. In our first approach, we exploit lazy greedy to obtain a unique offline policy for every agent irrespective of the agent exits from the team. The second approach is an offline-online approach where the offline phase creates a fixed number of joint policies to be used in the online phase. Finally, our experiments demonstrate the improved performance of our approaches on benchmark problems from literature.

2 BACKGROUND

2.1 Monotone Submodularity and Matroids

Definition 1 *Given a finite set, Π, a **submodular** function is a set function, $g : 2^{\Pi} \rightarrow \mathbb{R}$, where 2^{Π} is the power set corresponding to Π. More importantly, $\forall X, Y \subseteq \Pi$ with $X \subseteq Y$ and for every $i \in \Pi \setminus Y$, we have:*

$$g(X \cup i) - g(X) \geq g(Y \cup i) - g(Y)$$

*A submodular function g is **monotone** if $g(Y) \geq g(X)$ for $X \subseteq Y$.*

Monotone submodular functions are interesting because maximizing a submodular function to pick a fixed number of elements (say k) from the finite set (Π) while difficult can be approximated efficiently with a strong quality guarantee. Specifically, a greedy algorithm that incrementally generates the solution set by maximizing marginal utility provides solutions that are at least 63% $(1 - \frac{1}{e})$ of the optimal solution.

[1]This is unlike in adaptive submodularity, where there is one submodular problem with updated information on sensor state.

If we have a submodular function under a specific constraint on the finite set (Π) and the elements that are picked, the constraint is specified using a partition matroid. In this paper, we are also interested in maximizing a submodular function, however, under a specific constraint on the finite set (Π) and the elements that are picked. Specifically, the constraint is specified using a partition matroid. We provide the formal definitions below:

Definition 2 *For a finite ground set Π, let $\mathcal{P}$ be a non-empty collection of subsets of Π. The system $\Gamma = (\Pi, \mathcal{P})$ is a matroid if it satisfies the following two properties:*

- *The hereditary property: $\mathcal{P}_1 \in \mathcal{P} \wedge \mathcal{P}_2 \subset \mathcal{P}_1 \implies \mathcal{P}_2 \in \mathcal{P}$. In other words, all the subsets of $\mathcal{P}_1$ must be in $\mathcal{P}$.*
- *The exchange property: $\forall \mathcal{P}_1, \mathcal{P}_2 \in \mathcal{P} : |\mathcal{P}_1| < |\mathcal{P}_2| \implies \exists x \in \mathcal{P}_2 \setminus \mathcal{P}_1; \mathcal{P}_1 \cup x \in \mathcal{P}$.*

We are specifically interested in a ground set that is partitioned as $\Pi = \Pi_1 \cup \Pi_2 \cup \ldots \cup \Pi_k$. The family of subsets, $\mathcal{P} = \{P \subseteq \Pi : \forall i, |P \cap \Pi_i| \leq 1\}$ forms a matroid called a partition matroid. This family of subsets denotes that any solution can include at most one element from each ground set partition where the ground set partitions represent the policy space of each agent and exactly one policy must be picked for each agent.

2.2 Submodular TI-Dec-MDP

Submodular Transition Independent Decentralized Markov Decision Process (TI-Dec-MDP) model [Kumar *et al.*, 2017] is characterized by the tuple: $\left\langle \mathcal{A}g, S, A, \{P_i\}_{i \in \mathcal{A}g}, R, H, \alpha \right\rangle$, where

- $\mathcal{A}g$ is the set of agents.
- S is the factored joint state space. $S = S_1 \times S_2 \ldots S_{|\mathcal{A}g|}$, where S_i is the state space corresponding to each individual agent i. We can also have a global unaffected state feature S_u.
- A is the joint action space. $A = \times_{i \in \mathcal{A}g} A_i$, where A_i is the action space corresponding to each agent i.
- P_i is the individual agent transition function. $P_i(s_i'|s_i, a_i)$ indicates the transition probability of moving from s_i to s_i' on taking action a_i.
- R is the monotone submodular joint reward, with $R(s, a)$ representing the reward for taking joint action a in joint state s. In security domains [Shieh *et al.*, 2014], reward is both monotonically increasing and submodular. It is defined as follows:

$$R(s, a) = \sum_{\tau} y_\tau \cdot f_\tau(\sigma(s, a, \tau)) \qquad (1)$$

y_τ indicates the value of target τ and hence is a non-

negative number. $f_\tau(.)$ is a monotone submodular function referred to as the effectiveness of patrolling a target τ. Effectiveness of patrols at a target τ depends on the number of agents patrolling the target. $\sigma(s, a, \tau)$ counts the number of agents at target τ if the current joint state is s and joint action is a. Let ϵ $(0 < \epsilon \leq 1)$ represent the effectiveness of one agent visiting a target. Then, the effectiveness of σ agents visiting the same target τ in the joint state s is given by the usual definition of $f(.)$ for effectiveness parameter ϵ is $f(\sigma) = 1 - (1 - \epsilon)^\sigma$.

- H is the time horizon and α is the starting state distribution.

The goal is to obtain a joint policy $\pi^* = \langle \pi_1, \pi_2, \ldots, \rangle$ (with one policy, π_i for each agent i) that maximizes expected reward or value defined as follows:

$$V(\pi) = \sum_s \alpha(s) \cdot V^H(s, \pi) \tag{2}$$

$$V^t(s, \pi) = R\Big(s, \big\langle \pi_1^t(s_1), \ldots, \pi_{|\mathcal{A}g|}^t(s_{|\mathcal{A}g|}) \big\rangle\Big) + \sum_{s'} \Big[\prod_{i \in \mathcal{A}g} P_i\Big(s_i' | \pi_i^t(s_i), s_i\Big)\Big] \cdot V^{t-1}(s', \pi) \tag{3}$$

3 SUBMODULAR ND-TI-Dec-MDP

We extend Submodular TI-Dec-MDPs to Non-Dedicated TI-Dec-MDPs (ND-TI-Dec-MDPs) in order to model non-dedicated teams. The model is characterised by the following tuple:

$$\langle \mathcal{A}g, \{\Delta_i\}_{i \in \mathcal{A}g}, S, A, \{P_i\}_{i \in \mathcal{A}g}, R, H, \alpha \rangle$$

The main change to the Submodular TI-Dec-MDP is Δ_i. Δ_i is the vector of probabilities for agent i leaving the system at different times. Specifically, Δ_i^t represents the probability of agent i leaving the team at time t and $\sum_t \Delta_i^t = 1$. We use the global state S_u to represent the dead state (i.e., the state that agents enter when they move out of the system). The individual agent transition function $P_i^t(s_i'|s_i, a_i)$ is modified to $P_i^t(s_i'|s_i, a_i, \Delta_i)$ and is described as following:

$$P_i^t(s_i'|s_i, a_i, \Delta_i) = P_i^t(s_i'|s_i, a_i) \cdot (1 - \Delta_i^t) \tag{4}$$

$$P_i^t(S_u|s_i, a_i, \Delta_i) = \Delta_i^t \tag{5}$$

If $\Delta_i^t = 0$, it implies that the agent transitions to the expected state according to its transition probability $P_i^t(s_i'|s_i, a_i)$. Otherwise, if $\Delta_i^t \neq 0$, the transitions depend on the agent's probability of staying in the system (i.e., $1 - \Delta_i^t$). Furthermore, an agent transitions to the dead state from any other state with probability Δ_i^t. Note that once an agent transitions to the dead state S_u, it stays there until the end of horizon (i.e., $P_i^t(S_u|S_u, a_i, \Delta_i) = 1$) irrespective of the action taken. The joint reward function $R(s, a)$ however remains unchanged since the computation of reward only requires the count of agents present in the joint state s. In addition, there is no reward associated with agents present in S_u and we simply have $R(S_u, a) = 0$. The goal of submodular ND-TI-Dec-MDP is to obtain a joint policy π that maximizes the expected value $V^t(s, \pi)$ over all agents with an additional constraint that the agents may leave the team.

3.1 Properties of ND-TI-Dec-MDP

We now describe the important properties of ND-TI-Dec-MDPs with a joint reward function that is monotonically increasing and submodular. Let us first consider the case of a dedicated team where no agent leaves the system (represented as $\Delta_i^H = 1$ for all agents). In this case, the state of the system is fixed (i.e., no agents leaving) and already known to the decision maker, and hence, the policy of every agent can be determined in advance. However, in a non-dedicated agent team, agents may leave the team midway requiring reconfiguration of the remaining agent policies. The timestep at which an agent leaves the team is referred to as observation timestep, t' and the set of agents leaving the system at t' represent the observation ψ. All the agents that have left until t' constitute the observation set $\psi_{t'}$. The joint policy for a ND-TI-Dec-MDP is a concatenated policy which is formally defined for one observation timestep as following.

Definition 3 *Policy Concatenation: Let π_{ψ_0} be the joint policy over all agents until the first observation at time t' and $\pi_{\psi_{t'}}$ be the joint policy with observation set $\psi_{t'}$. The concatenated policy $\hat{\pi}$ is represented as:*

$$\hat{\pi} = [\pi_{\psi_0}]_{t=0}^{t<t'} + [\pi_{\psi_{t'}}]_{t=t'}^{t=H}$$

Proposition 1 *[Kumar* et al.*, 2017]: For a TI-Dec-MDP, $V^H(s, \pi)$ is monotonically increasing and submodular if the joint reward, R is monotonically increasing and submodular.*

At $t = 0$, ND-TI-Dec-MDP is similar to TI-Dec-MDP and is solved for $|\mathcal{A}g|$ agents and H timesteps. The value function, $V^H(s, \pi)$ is a monotone and submodular being the case of dedicated agent team. Similarly, for every observation timestep t', ND-TI-Dec-MDP is solved as a new TI-Dec-MDP problem with $\mathcal{A}g \setminus \psi_{t'}$ agents and $H - t'$ timesteps where $\psi_{t'}$ represents the set of agents that have left until t'. The value function, $V^{H-t'}(s, \pi)$ at t' is also monotonically increasing and submodular. Hence, for a single observation $\psi_{t'}$, the joint policy comprises of two components (as per definition 3) where the second

component is guaranteed to be submodular but not the first component. This is because submodularity of the value function $V^H(\pi)$ holds for $[t]_0^H$ but for ND-TI-Dec-MDP, we consider only timesteps $[t]_0^{t'}$ for the first component. Hence, the value function $V^H(\hat{\pi})$ for ND-TI-Dec-MDP is not guaranteed to be submodular for $\hat{\pi}$, however, it is submodular for every TI-Dec-MDP sub-problem.

The goal in ND-TI-Dec-MDPs is to maximize the expected value by obtaining a correct joint policy (i.e., exactly one policy per agent). Formally, the goal is to maximize $V^H(\pi)$ for every individual TI-Dec-MDP problem given the partition matroid $\Gamma = (\Pi, I)$ where $I = \{X \subseteq \Pi : |X \cap \Pi_i| = 1\}$. Intuitively, the partition matroid enforces that we can only have one policy for each agent.

Proposition 2 *[Fisher* et al.*, 1978]: Greedy algorithm for maximizing a monotone submodular function subject to a partition matroid yields solutions that are at least 50% of the optimal solution.*

For a non-dedicated agent team, the a priori bounds for every TI-Dec-MDP sub-problem at any t' is guaranteed to be at least 50% of optimal in the worst case. However, these bounds are quite loose since the solution provided by greedy is much better in most cases. Therefore, we compute online bounds by adding the marginal value of the best policy for every agent in the solution set to provide a tighter upper bound on the optimum. The online bound for a monotonically increasing and submodular value function is represented as below:

Proposition 3 *[Kumar* et al.*, 2017]: For any joint policy, π:*

$$V(\pi^*) \leq V(\pi) + \sum_{i \in Ag} \delta_i(\pi)$$

where $\quad \delta_i(\pi) = \max_{\pi_i \in \Pi_i} V(\pi \cup \pi_i) - V(\pi)$

Here, π^* is the optimal joint policy with optimal individual policies for every agent. For any joint policy π, we get an upper bound on the value of the optimal policy by adding the individual policies, π_i that yield best marginal values for each agent. In the context of ND-TI-Dec-MDP, at every observation timestep t', we solve a new TI-Dec-MDP problem with $\mathcal{A}g \setminus \psi_{t'}$ agents and $H - t'$ timesteps where any policy $\pi_{\psi_{t'}}$ provides an upper bound on the optimal policy $\pi^*_{\psi_{t'}}$. However, any concatenated policy $\hat{\pi}$ is not guaranteed to provide an upper bound on the optimal concatenated policy $\hat{\pi}^*$ since submodularity may not hold for π_{ψ_0}. We still compute the online bound for the concatenated policy as following.

$$V(\hat{\pi}^*) \leq \left[V(\pi_{\psi_0}) + \sum_{i \in \mathcal{A}g} \delta_i(\pi_{\psi_0}) \right]_{t=0}^{t<t'} + \left[V(\pi_{\psi_{t'}}) + \sum_{i \in \mathcal{A}g \setminus \psi_{t'}} \delta_i(\pi_{\psi_{t'}}) \right]_{t=t'}^{t=H} \quad (6)$$

Algorithm 1 ND-GREEDY $(\mathcal{A}g, S, A, P, R, H - t', \alpha, \psi_{t'})$

```
Z ← ∅
π*_i ← ∅, ∀i ∈ Ag \ ψ_{t'}
repeat
    for all i ∈ Ag \ {ψ_{t'} ∪ Z} do
        π*_i ← max_{π_i} V_i(π_i, α_i^{t'} | π*_Z)
    ⟨i*, V_{i*}⟩ ← max_{i ∈ Ag\ψ_{t'}∪Z} V_i(π*_i, α_i^{t'} | π*_Z)
    Z ← Z ∪ {i*}
until Ag \ {ψ_{t'} ∪ Z} = ∅
return {Z, π* ← {π*_i}_{i ∈ Ag\ψ_{t'}}}
```

where $\delta_i(\pi_{\psi_t}) = \max_{\pi_i \in \Pi_i} V(\pi_{\psi_t} \cup \pi_i) - V(\pi_{\psi_t}),\ t \in \{0, t'\}$

The expression in the first square bracket bounds the value of the optimal concatenated policy $V^H(\hat{\pi}^*)$ from $t = 0$ to $t \leq t'$ for the policy π_{ψ_0} (however, it is not a guaranteed online bound), while the second expression provides a guaranteed online bound on the value of the optimal concatenated policy from $t \geq t'$ to $t = H$. For our experiments, we compute online bounds for ND-TI-Dec-MDP using Equation 6.

4 APPROACHES

In this section, we provide enhancements to the existing approaches in literature along with an offline and an offline-online approach for solving ND-TI-Dec-MDPs. We extend the existing lazy greedy algorithm for TI-Dec-MDPs to provide solutions for non-dedicated agent teams. We further provide a lazy greedy extension for the benchmark heuristics in non-dedicated teams [Agrawal and Varakantham, 2017] to provide bounds on the solution quality of ND-TI-Dec-MDPs.

4.1 Greedy and Lazy Greedy

For dedicated agent teams, greedy has been well explored in the context of Dec-MDPs [Shieh *et al.*, 2014; Agrawal *et al.*, 2016; Kumar *et al.*, 2017] while for non-dedicated agent teams, it has been explored only in centralized settings [Agrawal and Varakantham, 2017]. Therefore, we extend the previous work by [Kumar *et al.*, 2017] to provide a lazy greedy extension for non-dedicated teams in decentralized settings.

Algorithm 1 provides the pseudocode for a non-dedicated greedy algorithm that is solved at every observation timestep, t' where $|\psi_{t'}|$ agents leave the team and $H - t'$ timesteps are remaining. The algorithm is initially invoked at the starting timestep (i.e., $t = t' = 0$ and $\psi_{t'=0} = \emptyset$) after which it is invoked only for timesteps where $\psi_{t'} \neq \emptyset$. ND-Greedy builds the solution set by incrementally adding a policy for every agent that has not

been assigned a policy. Initially, we start with an empty solution set Z (line 1). At every iteration, for each agent in the set of remaining agents, $\mathcal{A}g \setminus \psi_{t'}$ that has not been assigned a policy (line 4), we compute a policy with the highest marginal value given the current solution set (line 5) by constructing and solving an MDP (similar to the TI-Dec-MDP. Among those highest marginal value policies, we choose the one with the highest value and add it to the solution set (lines 6-7). This process is repeated until all $\mathcal{A}g \setminus \psi_{t'}$ agents have been assigned a policy to collectively provide the joint policy π^* (Every agent is assigned exactly one policy with the help of partition matroid constraint). Finally, the agents in Z are present in decreasing order of their marginal values. We refer this solution set Z as **selection order** of the agents.

ND-Greedy evaluates the marginal value for all the agents at every iteration, thereby affecting the scalability of the algorithm with increasing agents. Interestingly, submodularity of the value function $V^H()$ can be exploited to implement an accelerated version of classical greedy algorithm, otherwise known as Lazy Greedy [Minoux, 1978]. Instead of computing the marginal gain for all agents, lazy greedy allows a lazy evaluation of marginal benefits by storing the upper bounds $\mu(i)$ on the marginal gain for all agents $i \in \mathcal{A}g$ sorted in descending order. This reduces the marginal gain computation as the submodularity of value/objective function guarantees that the marginal gain for an agent is always equal to or lower than the previous iteration. Intuitively, for each iteration, lazy greedy evaluates the agent on the top of the list, say i, and updates its upper bound, $\mu(i)$. If $\mu(i) \geq \mu(i'), \forall i' \neq i$, submodularity guarantees that agent i has the highest marginal gain. Therefore, lazy greedy leads to significant reduction in running times compared to the classical greedy.

Why is the new policy recomputation needed: The recomputation of a new joint policy at every observation timestep t' is important because the contribution of rewards by agents at every timestep may vary. This means that an agent may have higher rewards at earlier timesteps compared to later timesteps. In security games, if the remaining agents continue with their initial policies even after few agents leave, the coverage of important targets may be missed, making the system vulnerable to attacks. This creates an urgency for policy recomputation and therefore, we use lazy greedy to obtain a new selection order for agents by considering the reward contributions from the current timestep to the end of planning horizon. For example, let the selection order of agents at $t = 0$ be $[A_2(555), A_3(545), A_1(500), A_4(490)]$ with the reward values for agents specified alongside. Let a_1 leave the system at $t = 1$. The total value for agents at $t = 1$ could be $[A_2(500), A_3(505), A_4(490)]$ on recomputation of reward for the remaining agents. This creates a change in order of selection of the agents because the contribution at $t = 0$ dominated the contribution over remaining timesteps for agents A_2 and A_3. Hence, the change in order contributes to the change in marginal gain, and therefore, agents must rearrange their policies to adapt to the change in system.

4.2 Benchmarking Heuristics

The existing benchmark heuristics for non dedicated agent teams [Agrawal and Varakantham, 2017] are centralized approaches and incapable of computing joint policy and joint reward for the agent team. Hence, we provide a lazy greedy extension for the existing benchmarks to be able to solve ND-TI-Dec-MDPs.

Ignore the leaving agent, Dec-ILA: We start with a lazy greedy solution for the dedicated team and whenever agents leave the team, the remaining agents continue with the execution of their existing policies. However, due to the presence of joint reward for the system, we recompute the joint reward over the remaining agents whenever agents leave. This provides a good lower bound on solution quality that has to be achieved. For example, in security games domain, ignoring the targets covered by leaving defender agent is not the best choice since the leaving agent may be protecting a target of high importance. Hence, it is important for the remaining agents to modify their policies to provide an improved coverage to the targets that would become vulnerable to attacks. Similarly, in sensor domain, the sensors in the vicinity of a spoilt sensor should be able to change their policies and sense the target locations assigned to the spolit sensor for better observation of any spatial phenomenon.

Offline Optimal, Dec-OPT: This heuristic assumes that the sample information (details of agents leaving the system) is received beforehand. Mixed integer program provides an optimal solution, but is not a suitable approach for finding the joint policy and the joint reward computation for a decentralized team of heterogeneous agents. Hence, we use lazy greedy for finding the agent policies where the agents are selected sequentially in the decreasing order of their values. Since the agents leaving the system have a shorter timespan compared to non-leaving agents, the marginal gain for such agents will be lowest. Hence, non-leaving agents are provided least preference in the selection process by greedy. Although not an exactly optimal approach, this heuristic provides a good upper bound on the solution quality.

Online Revamp, Dec-O-Rev: Similar to Dec-ILA, for this heuristic, we start with the initial lazy greedy solution until one or more agents leave the system. At the observation timestep t', the problem is solved again for

Algorithm 2 OFFLINE-GREEDY ($\xi, \mathcal{A}g, W$)

1: $Z \leftarrow \emptyset, O \leftarrow \emptyset$
2: $V_i \leftarrow 0, \forall i \in \mathcal{A}g$
3: **for all** $\xi^k \in \xi$ **do**
4: $V^k \leftarrow$ Dec-OPT($\mathcal{A}g, S, A, P, R, H, \alpha, \xi^k$)
5: $V_i \leftarrow V_i + W^k \cdot V_i^k$
6: **for all** $i \in \mathcal{A}g$ **do**
7: $V_{i^*} \leftarrow \max_{i \in \mathcal{A}g \setminus O} V_i$
8: $O \leftarrow O \cup i^*$
9: **for all** $o \in O$ **do**
10: $\langle \pi_o^*, V_o^* \rangle \leftarrow V_o(\pi_o^*, \alpha_o^0 | \pi_Z^*)$
11: $Z \leftarrow Z \cup \{o\}$
12: $\pi^* \leftarrow \{\pi_o^*\}_{o \in O}$
13: **return** $\langle \pi^*, O \rangle$

the remaining agents $\mathcal{A}g \setminus \psi_{t'}$ and remaining timesteps $H - t'$. The starting distribution of the remaining agents is recomputed at t' and is input to the lazy greedy algorithm along with the information of leaving agents, $\psi_{t'}$. The new joint policy obtained for the remaining agents is executed by the agent team until there is a change in the system (i.e., an agent leaves the system). Dec-O-Rev provides a good upper bound on the desired performance for our proposed approaches but suffers from some limitations. Although the running time reduction due to lazy greedy is significant compared to classical greedy, the total number of function evaluations with lazy greedy cannot be predicted beforehand to provide the exact running cost. This makes the complete recomputation of selection order at observation timesteps time consuming and difficult to be evaluated on the fly. Secondly, if there is a requirement of recomputation at every timestep t, revamp would become infeasible since at least $\mathcal{A}g \setminus \psi_t$ rounds of sequential computation for agents will be required.

4.3 Offline-Greedy Approach

Offline-Greedy is a sampling-based approach that computes an offline selection order, O and a single joint policy π^* over multiple scenarios of agent availability. Since it is impossible to consider all the samples of agent availability on larger problems, we choose a smaller training set for the joint policy computation. The sample set is represented as ξ and has $|K|$ samples. Due to repetition of samples, we assign frequency-specific weights $W^k, \forall k \in K$ and select 20 best samples in decreasing order of weights. Every sample of agent availability, ξ^k is generated by sampling from a biased coin with probability p_i independently for every agent i. At every timestep t, the coin is tossed to decide whether agent i leaves or stays in the team depending on the value of associated probability in Δ_i. Hence, for every sample ξ^k, we know the available horizon $\xi^k(i)$ for every agent i.

Algorithm 2 provides the pseudocode for Offline-Greedy

Algorithm 3 OFFLINE-ONLINE ($\mathcal{A}g, N$)

1: **for all** $n \in N$ **do**
2: $Z \leftarrow \emptyset$
3: $\pi_i^n \leftarrow \emptyset, \forall i \in \mathcal{A}g$
4: **repeat**
5: $r_i \leftarrow Random(\mathcal{A}g \setminus Z)$
6: $\pi_{r_i}^n \leftarrow V_{r_i}^n(\pi_{r_i}, \alpha_{r_i}^0 | \pi_Z^n)$
7: $Z \leftarrow Z \cup \{r_i\}$
8: **until** $\mathcal{A}g \setminus Z = \emptyset$
9: $\pi^n \leftarrow \{\pi_i^n\}_{i \in \mathcal{A}g}$
10: $\Pi \leftarrow \Pi \cup \{\pi^n\}$
11: **return** Π

with the training set ξ, the agent set $\mathcal{A}g$ and the vector of frequency weights over all samples W as inputs. The agent selection set, Z and the selection order O are initialized as empty sets and the total value of every agent over all samples V_i is set to 0 (line 1-2). For every sample ξ^k in the training set, the available horizon of every agent is already known, and therefore, we use Dec-OPT heuristic to obtain the total value, V^k for every $\xi^k \in \xi$ (line 4). The total value for every agent V_i is computed as the weighted sum of values over the sample set (i.e., $W^k \cdot V_i^k$) (line 5). The selection order O is computed by sorting the agents in decreasing order of their values V_i (line 6-9) such that the agents with higher probability of staying in the system are added before the agents with higher probability of leaving. For all the agents in the selection order, highest marginal value policy for an agent given the current solution set (line 10) is computed by constructing and solving an MDP (similar to TI-Dec-MDP) and the computed agent is then added to the solution set (line 11). Finally, we return the best selection order O and the offline joint policy π^* over all agents and all training samples.

For every test sample, the agents are assigned their individual policies from the offline joint policy π^*. However, irrespective of the observations obtained at different observation timesteps, the agents continue with their preassigned policies while the joint reward is recomputed for the remaining agents. This approach saves the online recomputation of policy at observation timesteps but with a compromise in the solution quality.

4.4 Offline-Online Approach

In this section, we present our Offline-Online algorithm which is a randomized greedy algorithm with an offline and an online phase. The offline phase focuses on the generation of multiple agent(s) selection orders to handle the different possibilities of scenarios, while the online phase focuses on choosing the best selection order for remaining agents depending on the current observation (availability of agents). We note that having multiple selection orders is better than having one fixed selection

order for all scenarios (as present in Offline-Greedy) because the total value of a selection order can change at different observation timesteps due to the dominance of rewards in previous timesteps (explained in details in section 4.1). We generate a fixed number of selection orders for the agent set since the total number of orderings possible with $|\mathcal{A}g|$ agents is $|\mathcal{A}g|!$ orders which is difficult to maintain with increasing agents. At every decision stage, we choose the best/closest selection order such that the position of the leaving agent is towards the end of the selection order, thereby, avoiding the recomputations. The time complexity of the offline phase is linear in the number of agents (or $O(|\mathcal{A}g|)$) while it takes constant time for the online phase. The main difference with respect to lazy greedy (used in all the above approaches) is that instead of choosing the agent with highest marginal gain at every iteration, we randomly pick an agent and add it to the selection set. However, due to the joint reward computation and the presence of submodular rewards, the total utility always improves with addition of agents iteratively.

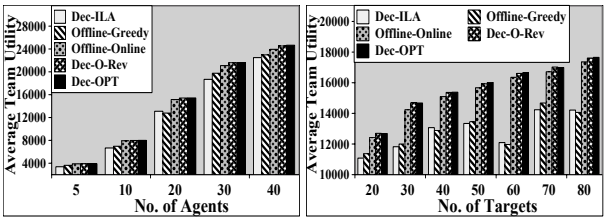
Algorithm 3 shows the offline phase of Offline-Online algorithm where the input to the algorithm is the agent count $|\mathcal{A}g|$, and the number of selection orders to be generated (N). For computing every order n, we start with an empty agent selection set Z and add one agent at a time by randomly selecting agents from the set of remaining agents $\mathcal{A}g \setminus Z$. The policy and value of every agent is obtained by solving an MDP and is stored in π^n. Finally, we return Π that represents the set of policies for all the N selection orders.

The online phase of our algorithm does not require any computation and only reacts to a situation by choosing the best order from the set of offline orders for the remaining agents and providing a new policy for every agent from the observation timestep t' until the end of horizon. The selection criteria for choosing the best order for the defender team whenever any agent leaves the team depends on the number of exact matching and closest matching selection orders. For example, let us assume that there are 4 agents in the system $\{a_1, a_2, a_3, a_4\}$ and the available set of selection order contains three orders, $O_1 = \{3, 2, 1, 4\}$, $O_2 = \{4, 2, 3, 1\}$ and $O_3 = \{3, 2, 4, 1\}$ with total utility of $\{200, 150, 100\}$ for the orders respectively. Let us consider two case studies:

- **Agent a_1 leaves the system:** In this case, O_2 and O_3 are the best suitable orders since they require no re-evaluation but the order with highest utility is given preference and hence, O_2 is chosen.

- **Agent a_3 leaves the system:** In this case, none of the matches are exact and therefore, we find the closest match. We choose O_2 to assign policies to the remaining agents since it requires minimal updates to agent policies. At the observation timestep, the previous policy of a_1 is replaced by the existing policy of a_3, but after considering the change in state distribution of the agents since a_1 and a_3 are not guaranteed to be in the same state at the considered timestep. However, due to the replacement of agent policies, a_1 would now become the third agent in the system, assuming the presence of two agents. Policy recomputation is not required because the offline joint policy (of every selection order) computes the $V^t(s, \pi)$ values for all states at all intermediate timesteps (i.e., joint value after selection of every agent in the selection order). Furthermore, no reward recomputation is required since the joint reward considers only the count of agents (and not the identity of agents) at any state due to the monotone submodular reward structure for the joint reward.

In this manner, the online phase improves the value of solution roughly the same as Dec-O-Rev, but very quickly.



(a) $\tau = 40, H = 20, \epsilon = 0.7$ (b) $|\mathcal{A}g| = 20, H = 20, \epsilon = 0.7$

Figure 1: Quality Comparison w.r.t. (a) Agents and (b) Targets

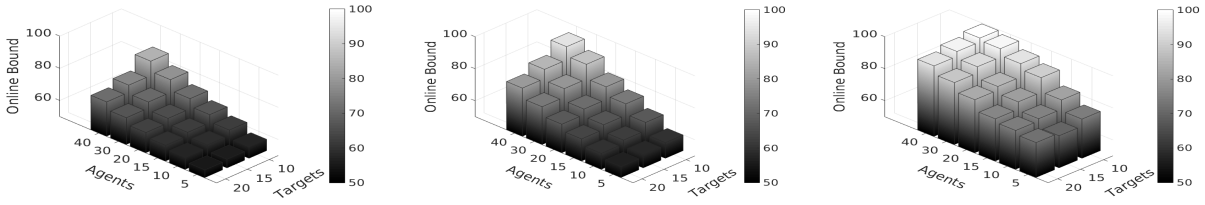
5 EXPERIMENTS

We evaluate[2] the performance of our greedy approaches and compare them with the benchmark approaches mentioned in section 4.2 on the security games domain provided by Shieh *et al.* [2014] and the sensor network domain provided by Kumar *et al.* [2017]. The performance is evaluated on the following metrics: (a) solution quality; (b) runtime; (c) quality of online bounds. We generate 1500 samples of agent availability (defenders in security domain and sensors in sensor domain) and divide it into training and testing sets of 1000 and 500 samples, respectively. To obtain a fair comparison over all approaches, we compare the solutions on the same test set.

5.1 Security Games Domain

In this domain, there are a set of targets (train stations) on the metro rail network which must be defended by a set of decentralized (yet cooperative) defenders in the presence

[2]All our optimization problems are run on CPLEX v12.7



(a) $H = 10, \epsilon = 0.5$ (b) $H = 10, \epsilon = 0.7$ (c) $H = 10, \epsilon = 0.9$

Figure 2: Comparison of Online Bound w.r.t. Effectiveness

of transition uncertainty. We constructed the metro rail graphs by connecting the stations together in lines of length 5 and then randomly adding $|\tau|/2$ edges between targets, to resemble train systems in the real world with complex loops. The reward is a joint reward which is a function of the number of active defenders and the targets for a joint state s (see Section 2.2 for details). The test results were averaged over 15 randomly generated metro-based graph networks and the rewards were generated randomly in the range of [0,100]. We run the scenarios with a probability delay of .2 and a maximum of 5 agents (varies from 10% to 25% across scenarios) with an ability to leave the system, defined by probability vector Δ. The defender agents are homogeneous (due to same reward and transition function) but differ from each other in their starting states (generated randomly for every agent) and their capability to leave the system.

Solution Quality: We compare different approaches with respect to average team utility in Figure 1(a) as the number of defenders $|\mathcal{A}g|$ is increased. Specifically, we consider a metro network with targets $\tau = 40$, horizon $H = 20$ and effectiveness parameter $\epsilon = 0.7$. Similarly, in Figure 1(b), we vary the targets, τ for a fixed number of agents $|\mathcal{A}g| = 20$, horizon $H = 10$ and effectiveness parameter $\epsilon = 0.7$. The key observations are summarized as following:
(1) The average team utility increases with increasing defenders for a fixed number of targets and planning horizon due to the submodular reward structure. Similarly, the team utility increases with increasing targets due to increased number of choices for obtaining better rewards.
(2) Dec-ILA provides low team utility solutions since the remaining agents continue with existing policies even after agents leave. This impacts the scope of improvement in rewards and is a cause of serious concern in security domain since it allows easy access to an adversary to plan an attack in unprotected areas.
(3) Offline-Greedy provides similar or better solutions than Dec-ILA. We observe that with fewer agents and targets, and smaller planning horizon, Offline-Greedy performs almost similar to Dec-O-Rev as agent exits are given due importance during the offline policy design but the performance degrades quickly with increasing count of agents and the planning horizon. In the worst case, the solution quality was seen to be even lower than Dec-ILA.
(4) Offline-Online provides a steady performance, almost at par with the upper bound benchmarks (Dec-O-Rev and Dec-OPT) even with increasing problem sizes. Due to the random selection of agents at different observation times and the presence of submodular reward function, in the best case, Offline-Online could provide better team utility than Dec-O-Rev (uses lazy Greedy).
(5) Dec-OPT provides a good upper bound but is not always guaranteed to provide better utility compared to Dec-O-Rev due to the dominance of rewards in earlier timesteps explained in the section 4.1. However, on average, Dec-OPT provides slightly better solution quality compared to Dec-O-Rev after having the knowledge of samples before-hand.

Solution Runtime: With respect to runtime, we compare only online runtime since the offline runtimes do not matter. Due to decentralized planning of agents, individual agent planning time varies from 100 ms to 5000 ms from the smallest problem instance (20 targets and 10 timesteps) to the largest instance (40 targets and 20 timesteps). For Dec-O-Rev, due to the use of lazy greedy approach at every timestep of revamp, the revamp time varies from 15 seconds to 1700 seconds depending on the number of defenders and the problem size per defender. Further, there can be multiple revamps for one planning scenario making Dec-O-Rev infeasible for providing new policies quickly. However, our Offline-Online approach uses a proactive offline planning which reduces the online execution time to milliseconds, even in the worst case (although it requires offline training time). Similarly, the online runtime is minimal for Dec-ILA, Dec-OPT and Offline-Greedy.

Online Bound Comparison: For the online bound comparison, we use a consistent reward structure for every randomly generated metro network. For every metro network, we generate various scenarios of agents availabilities for different number of defenders and varying effectiveness of defenders. We compute the online bound for every scenario using Equation 6 and average the online

Grid-Size	Sensors,Targets Global States	ϵ		
		.3	.5	.7
5×5	5, 1, 10	58.3	64.5	71.5
5×5	5, 2, 6*6	58.6	65	71.5
10×5	6, 3, 14*10*10	57.4	61.2	64.2
10×5	6, 4, 5*5*5*5	57.3	61.6	63.7
10×5	6, 5, 6*5*5*5*5	55.7	61.3	65.3
10×5	10, 3, 14*10*10	58.5	63.5	69
10×5	10, 4, 5*5*5*5	55.5	58.5	61.7
10×5	10, 5, 6*5*5*5*5	55.9	61.5	67.7
10×10	10, 4, 6*5*5*5	58.7	64.5	71.2
10×10	15, 4, 6*5*5*5	58.5	63.8	72.2
10×10	20, 4, 6*5*5*5	58.9	65.5	72.7
10×10	10, 5, 5*5*5*5*5	55.5	61.2	67.3

Table 1: Online Bound Comparison for Sensor Domain

bounds over all test samples and all randomly generated graphs. This leads to the inference that the online guarantees are significantly better than the a priori guarantees (of 50% from optimal), with the best case of atleast 90% from optimal for different values of effectiveness parameter. Figure 2 compares the online (or posterior) quality guarantees obtained by Dec-O-Rev for different values of agents ($\mathcal{A}g$), targets (τ) and effectiveness parameter (ϵ). It shows that the online guarantees improve with increasing agents and decreasing targets over varying effectiveness, with highest guarantee being reported for 10 targets and 40 agents. Further, with increasing effectiveness of agents, the optimal bound increases with highest quality guarantees (up to 99%) observed for $\epsilon = 0.9$. To avoid clutter, we do not plot the quality guarantees provided by policies generated using Offline-Online in the same graph. However, Offline-Online fared slightly lower than Dec-O-Rev in terms of guarantees and provided a guarantee that was 0.7% lower than Dec-O-Rev in the best case, while in the worst case, it was 2 % lower than Dec-O-Rev.

5.2 Sensor Network Domain

We use the similar settings as Kumar *et al.* [2017] for this domain. The environment is modelled as a grid and a submodular reward function with n-ary interactions (any number of sensors can track a target) is used where the reward of tracking a target is dependent on the number of sensors tracking it. The sensors are randomly placed at junctions of cells on the grid and can track four target cells surrounding the sensor. However, due to wear and tear or due to unforeseen conditions, some sensors may get spoilt and the neighbouring sensors must track the targets of damaged sensors to maximize the reward. Therefore, reconfiguration of sensors after one or more sensors are spoilt is important. The targets move stochastically (according to some fixed distribution) in the grid and follow a path of fixed length for movement. The product of path lengths of all available targets defines the total number of global states for the sensor domain.

Online Bound Comparison: Table 1 shows the online guarantees obtained for offline-online by varying the grid-size, number of sensors and their effectiveness, number of targets and the number of global states. We vary the effectiveness parameter from 0.3 to 0.7 and observe that the online bounds vary from 55% to 73% for Offline-Online, while the guarantees provided by Dec-O-Rev were 4% and 1.8% better than Offline-Online in the worst case and best case, respectively. An important observation is that with increasing targets, the number of global states increases exponentially, leading to memory issues. We note that 5 targets for a 10×10 grid with every target having a path-length of 5 was very difficult instance to solve with 5^5 or 3125 global states. However, increasing the number of sensors with a fixed number of targets was comparatively easier to solve since every sensor agent problem was solved independent of other agents due to the decentralized settings. With respect to runtime, the time taken by any sensor agent for individual planning varies from few milliseconds to 10 seconds with increasing number of targets and the global states. Due to lazy greedy evaluations for Dec-O-Rev, every revamp may take time ranging from less than a minute to 40 minutes depending on the complexity of the problem being solved. This makes the usage of Dec-O-Rev infeasible in online settings. Further, there can be multiple revamps for every scenario to worsen the situation. Similar to the security domain, the performance of Offline-Greedy is very similar to Dec-ILA while Dec-OPT provides results similar to Dec-O-Rev. More interestingly, our Offline-Online approach continues to perform gracefully with increasing number of sensors, targets and the grid size, while taking minimal time (in milliseconds) for solving the largest problem. Finally, we conclude from the experiments that Offline-Online is the best choice considering the trade-off of running time and compromise in solution quality.

6 CONCLUSION

In this work, we focussed on cooperative decentralized stochastic planning for non-dedicated agent teams. We provided a general model for decentralized non dedicated agent teams. Our offline greedy based approach provided good results in small instances while our Offline-Online approach provided the best results even in large instances in an effective manner. Finally, our extensive experiments on benchmark problems demonstrate that our Offline-Online approach provides the best solutions that are on par with benchmarks that provide an upper bound on the performance while taking negligible online runtime making it effective even for taking decisions at every step.

References

Pritee Agrawal and Pradeep Varakantham. Proactive and reactive coordination of non-dedicated agent teams operating in uncertain environments. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, pages 28–34, 2017.

Pritee Agrawal, Pradeep Varakantham, and William Yeoh. Scalable greedy algorithms for task/resource constrained multi-agent stochastic planning. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, 2016.

Matthew Brown, Sandhya Saisubramanian, Pradeep Varakantham, and Milind Tambe. STREETS: game-theoretic traffic patrolling with exploration and exploitation. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*, pages 2966–2971, 2014.

Archie Chapman and Pradeep Varakantham. Marginal contribution stochastic games for dynamic resource allocation. In *Proceedings of the International Conference on Principles and Practice of Multi-Agent Systems (PRIMA)*, pages 333–340, 2014.

Marshall L Fisher, George L Nemhauser, and Laurence A Wolsey. An analysis of approximations for maximizing submodular set functions- ii. In *Polyhedral combinatorics*, pages 73–87. Springer, 1978.

Daniel Golovin and Andreas Krause. Adaptive submodularity: Theory and applications in active learning and stochastic optimization. *Journal of Artificial Intelligence Research*, 42:427–486, 2011.

Akshat Kumar and Shlomo Zilberstein. Event-detecting multi-agent MDPs: Complexity and constant-factor approximation. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, pages 201–207, 2009.

Akshat Kumar and Shlomo Zilberstein. Message-passing algorithms for large structured decentralized POMDPs. In *Proceedings of the Tenth International Conference on Autonomous Agents and Multiagent Systems*, pages 1087–1088, Taipei, Taiwan, 2011.

Rajiv Ranjan Kumar, Pradeep Varakantham, and Akshat Kumar. Decentralized planning in stochastic environments with submodular rewards. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*, pages 3021–3028, 2017.

Francisco S Melo and Manuela Veloso. Decentralized mdps with sparse interactions. *Artificial Intelligence*, 175(11):1757–1789, 2011.

Michel Minoux. Accelerated greedy algorithms for maximizing submodular set functions. In *Optimization Techniques*, pages 234–243. Springer, 1978.

Ranjit Nair, Pradeep Varakantham, Milind Tambe, and Makoto Yokoo. Networked distributed pomdps: A synthesis of distributed constraint optimization and pomdps. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*, volume 5, pages 133–139, 2005.

Yash Satsangi, Shimon Whiteson, Frans A Oliehoek, et al. Exploiting submodular value functions for faster dynamic sensor selection. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*, pages 3356–3363, 2015.

Eric Anyung Shieh, Albert Xin Jiang, Amulya Yadav, Pradeep Varakantham, and Milind Tambe. Unleashing dec-mdps in security games: Enabling effective defender teamwork. In *Proceedings of the European Conference on Artificial Intelligence (ECAI)*, pages 819–824, 2014.

Pradeep Varakantham, Jun-Young Kwak, Matthew Taylor, Janusz Marecki, Paul Scerri, and Milind Tambe. Exploiting coordination locales in distributed POMDPs via social model shaping. In *Proceedings of the International Conference on Planning and Scheduling (ICAPS)*, pages 313–320, 2009.

Pradeep Varakantham, Hoong Chuin Lau, and Zhi Yuan. Scalable randomized patrolling for securing rapid transit networks. In *Proceedings of the Conference on Innovative Applications of Artificial Intelligence Conference (IAAI)*, 2013.

Pradeep Varakantham, Yossiri Adulyasak, and Patrick Jaillet. Decentralized stochastic planning with anonymity in interactions. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*, pages 2505–2512, 2014.

Prasanna Velagapudi, Pradeep Varakantham, Paul Scerri, and Katia Sycara. Distributed model shaping for scaling to decentralized POMDPs with hundreds of agents. In *Proceedings of the International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, pages 955–962, 2011.

Zhengyu Yin and Milind Tambe. Continuous time planning for multiagent teams with temporal constraints. In *IJCAI Proceedings-International Joint Conference on Artificial Intelligence*, volume 22, page 465, 2011.