Distributed Learning for Planning Under Uncertainty Problems with Heterogeneous Teams

Citation: Ure, N. Kemal, Girish Chowdhary, Yu Fan Chen, Jonathan P. How, and John Vian. "Distributed Learning for Planning Under Uncertainty Problems with Heterogeneous Teams." J Intell Robot Syst 74, no. 1–2 (November 19, 2013): 529–544.

As Published: http://dx.doi.org/10.1007/s10846-013-9980-x

Publisher: Springer Netherlands

Persistent URL: http://hdl.handle.net/1721.1/103618

Version: Author's final manuscript: final author's manuscript post peer review, without publisher's formatting or copy editing

Journal of Intelligent and Robotic Systems

Distributed Learning for Planning Under Uncertainty Problems with Heterogeneous Teams

--Manuscript Draft--

Manuscript Number:	JINT-D-13-00458
Full Title:	Distributed Learning for Planning Under Uncertainty Problems with Heterogeneous Teams
Article Type:	Full/Regular paper
Keywords:	Distributed Learning; Planning Under Uncertainty; Unmanned Aerial Systems.
Corresponding Author:	Nazim Kemal Ure Massachusetts Institute of Technology Cambridge, Massachusetts UNITED STATES
Corresponding Author Secondary Information:	
Corresponding Author's Institution:	Massachusetts Institute of Technology
Corresponding Author's Secondary Institution:	
First Author:	Nazim Kemal Ure
First Author Secondary Information:	
Order of Authors:	Nazim Kemal Ure
	Girish Chowdhary, Assistant Professor
	Yu Fan Chen
	Jonathan P. How, Professor
	John Vian
Order of Authors Secondary Information:	
Abstract:	This paper considers the problem of multiagent sequential decision making under uncertainty and incomplete knowledge of the state transition model. A distributed learning framework, where each agent learns an individual model and shares the results with the team, is proposed. The challenges associated with this approach include choosing the model representation for each agent and how to effectively share these representations under limited communication. A decentralized extension of the model learning scheme based on the Incremental Feature Dependency Discovery (Dec-iFDD) is presented to address the distributed learning problem. The representation selection problem is solved by leveraging iFDD's property of adjusting the model complexity based on the observed data. The model sharing problem is addressed by having each agent rank the features of their representation based on the model reduction error and broadcast the most relevant features to their teammates. The algorithm is tested on the multiagent block building and the persistent search and track missions. The results show that the proposed distributed learning scheme is particularly useful in heterogeneous learning setting, where each agent learns significantly different models. We show through large-scale planning under uncertainty simulations and flight experiments with state-dependent actuator and fuel-burn-rate uncertainty that our planning approach can outperform planners that do not account for heterogeneity between agents.

Cover Letter

Dear Editor,

We are excited to submit our paper "Distributed Learning for Planning Under Uncertainty Problems with Heterogeneous Teams" to the Journal of Intelligent and Robotic Systems UAS Special Volume. We would like to emphasize that this paper contains the following differences and enhancements from our ICUAS'13 submission:

- The abstract and the introduction are completely revised and new references on distributed learning are added.

- Figure 2, which shows the distributed learning framework is re-drawn to display interactions among agents.

- The Dec-iFDD Algorithm is explained more thoroughly with additional figures and equations. (Section 3.1, 3.2 and 3.3)

- New results are added to the simulation section for a multiagent block building problem. (Section 4.2)

- New flight results are added to the flight results section for the 10 agent problem. (Section 5.1)

Please do let us know if you need any further information from us. We are looking forward to making progress in having this paper appear in the JINT UAS Special Issue.

Best Regards

Noname manuscript No.
(will be inserted by the editor)

Distributed Learning for Planning Under Uncertainty Problems with Heterogeneous Teams

Scaling Up the Multiagent Planning with Distributed Learning and Approximate Representations

N. Kemal Ure · Girish Chowdhary · Yu Fan Chen · Jonathan P. How · John Vian

Received: date / Accepted: date

Abstract This paper considers the problem of multiagent sequential decision making under uncertainty and incomplete knowledge of the state transition model. A distributed learning framework, where each agent learns an individual model and shares the results with the team, is proposed. The challenges associated with this approach include choosing the model representation for each agent and how to effectively share these representations under limited communication. A decentralized extension of the model learning scheme based on the Incremental Feature Dependency Discovery (Dec-iFDD) is presented to address the distributed learning problem. The representation selection problem is solved by leveraging iFDD's property of adjusting the model complexity based on the observed data. The model sharing problem is addressed by having each agent rank the features of their representation based on the model reduction error and broadcast the most relevant features to their teammates. The algorithm is tested on the multiagent block building and the persistent search and track missions. The results show that the proposed distributed learning scheme is particularly useful in heterogeneous learning setting, where each agent learns significantly different models. We show through large-scale planning under uncertainty simulations and flight experiments with state-dependent actuator and fuel-burn- rate uncertainty that our planning approach can outperform planners that do not account for heterogeneity between agents.

N. Kemal Ure
Massachusetts Institue of Technology E-mail: ure@mit.edu

Girish Chowdhary
Oklahoma State University E-mail: girish.chowdhary@okstate.edu

Yu Fan Chen
Massachusetts Institue of Technology E-mail: chenyuf2@mit.edu

Jonathan P. How
Massachusetts Institue of Technology E-mail: jhow@mit.edu

John Vian
Boeing Research and Technology E-mail: john.vian@boeing.com

Keywords Distributed Learning · Planning Under Uncertainty · Unmanned Aerial Systems

1 Introduction

Multiagent planning problems[12] are ubiquitous in engineering, with applications ranging from manufacturing[13] to surveillance with Unmanned Aerial Vehicles (UAVs)[26]. A common theme among such missions is the uncertainty that stems from stochastic vehicle dynamics and external disturbances[24]. The planners developed to address such problems typically rely on models of the mission dynamics[17], but these models may not always be available and/or the available models might be inaccurate, which can lead to poor performance[1]. A common solution is to incorporate a model learning algorithm within the planning framework to address this problem[3] , but this coupled planning/learning problem is particularly challenging for the multi-agent systems[19] due to exponential increase in the size of the planning space with number of agents. This paper presents a decentralized learning algorithm that decomposes the learning problem to be per agent, and enables the agents to share the relevant models/features while accounting for communication constraints. The main objective of the paper is to demonstrate that the developed learning algorithm can be integrated within a planning framework to enable online solution of large-scale multiagent planning problems with uncertain dynamics.

Markov Decision Processes (MDPs) provide a natural framework for solving stochastic sequential decision making problems[18]. Many researchers have developed MDP formulations of robotic planning missions and used variety of tools such as dynamic programming to generate optimal solutions[11]. However, these exact methods usually do not scale well with the number of agents for the multiagent problems. Approximate Dynamic Programming (ADP) methods[1] aim to address the problem of solving such large-scale MDPs by employing approximate representations of the value or policy functions, thereby trading-off the optimality of the resulting plan with the computational efficiency. ADP methods have been demonstrated to be a powerful set of tools for solving large-scale and/or continuous planning problems[4]. In particular, the authors have applied ADP techniques to solve UAV persistent search and track missions, and demonstrated the applicability of the developed methods with flight results[26,28,30].

The planning techniques based on solution of MDPs operate on the assumption that the transition model of the MDP is available to the designer. To aid this problem, researchers have integrated machine learning techniques and planning algorithms to update the transition model of the MDP based on the observations that are gathered by the agents through interactions with the mission environment. In particular, the Dyna algorithm[22] and approximate versions[23,32] have been successful in planning with learned models for MDPs with unknown transition dynamics. One of the disadvantages of such methods is the requirement that the features (also referred to as bases) of the approximate representation are fixed. Selection of features is a difficult problems in itself, and recognized as one of the most important challenges in ADP and Reinforcement Learning[1,14]. In the previous work, the authors have used the Incremental Dependency Discovery (iFDD)[7] function approximation method to learn the transition models of MDPs[27], which incrementally expands the set of features used in the approximation architecture.

The main advantage of the technique is allowing the complexity of the approximation architecture to be dynamically adjusted based on the observed interactions with the environment, therefore relaxing the strict requirement on hand-coding a fixed set of features.

Although the iFDD model learning algorithm was shown to be an effective method for solving a variety of planning problems[26,30], the algorithm's convergence rate slows down considerably for large-scale multiagent planning scenarios. One possible solution to this challenge is to decompose the learning to be per agent rather than operating in the joint state space of all agents, by using methods from distributed learning[6] and transfer learning[15]. Such methods accelerate the speed of the learning process by sharing model parameters. However these techniques have their own set of challenges, such as how to share learned model parameters under limited communication. In particular, in the *heterogeneous* learning setting[16], where the agents are learning different models, sharing model information may actually harm the overall system performance. Such heterogeneous team settings are common in UAV missions, where the dynamics or the operational space of each agent is different from each other.

Our previous work introduced the Dec-iFDD algorithm[28], which decomposes the problem to per agent to relax the computational complexity, and allows agents to share features with each other to improve their corresponding models. Results in that paper showed that the Dec-iFDD algorithm is more scalable compared to its centralized variant and the feature sharing helps to improve the convergence speed considerably. The main contribution of this paper is the presentation of the Dec-iFDD algorithm in a more general setting rather than limit it's applicability to UAV health management problems, as well as additional simulations and flight test results to show the applicability of the algorithm across different problems. These additional results confirm that the Dec-iFDD is applicable to general multiagent learning problems. In addition, results verify the earlier results on how the decentralized learning can be more beneficial than centralized learning when the the team is heterogeneous.

The paper is organized as follows, Section 2 provides the definition of the learning/planning problem, Section 3 introduces the Dec-iFDD algorithm. The next section presents simulation results, for the multiagent block assembly problem and the UAV persistent search and track (PST) problem. Finally, Section 5 provides the flight results for a 10 mixed real/virtual agent implementation of the PST mission[1].

2 Problem Definition

The main problem we address in the paper is planning and distributed learning with MDPs with uncertain transition models in cooperative multiagent setting. Note that there has been significant amount of research in the game theory community for competitive scenarios[9,31].

We assume that the uncertainty in the model can be represented by a parameter p that can be factorized per agent, that is $p = [p^1, \ldots, p^n]$ where n is the number of agents. Depending on the context of the problem, p^i can be a scalar, a probability

[1] Parts of this work was published before on[28,29]

distribution or a mapping. The distributed learning problem involves development of the estimation laws for p^i, as well as the coordination plan for distribution of the learned models across the team.

We make more formal definitions of the problem and our approach in Section 3. Note that after the model is updated, the corresponding MDP must be solved by a planning algorithm to generate a policy. This paper does not focus on the planning algorithms, however the authors have developed several multiagent planning algorithms in the past for such problems [20,26,30].

2.1 Motivating Problem: UAV Persistent Search and Track Missions (PST):

In this section we present the PST mission as a motivating example for a multiagent planning under uncertainty problem and how the decentralized learning can be used to improve the planning performance. The goal of the mission is to find an optimal policy to persistently perform surveillance on a group of targets with limited endurance UAVs in the presence of uncertainty and both communication and health constraints [20], see Fig. 1.

The number of UAVs is denoted by $n_{\texttt{UAV}}$, and each UAV's individual state consists of its location, fuel, actuator status, and sensor status. The full state space is the combination of states for all UAVs. There are three available actions for each UAV: $\{Advance, Retreat, Loiter\}$. The objective of the mission is to fly to the surveillance region and perform find/track the targets, while ensuring that a communication link with the base is maintained. The base is assumed to be out of line-of-sight, therefore, a UAV needs to act as a communication relay to maintain the link. If a UAV runs out of fuel (remaining charge in the battery), it is assumed to be unable to continue the mission. It is difficult to estimate the time for the battery to discharge due to uncertainties in the environment such as external disturbances, and uncertainties in the mission such as the aggressiveness of the targets to be pursued. Therefore, a probabilistic model is used to capture the fuel burned while performing the mission: each UAV starts with 10 units of fuel and is assumed to burn one unit for all actions with probability $p_{\texttt{nom}}$ and 2 units with probability $1 - p_{\texttt{nom}}$. This model allows designer to choose $p_{\texttt{nom}}$ to reflect the uncertainty in the fuel burning dynamics for the given mission. Each UAV carries an actuator and a sensor with probabilistically modeled failure rates. The sensor and actuator of each UAV can fail on each step with probabilities p_{act} and $p_{\texttt{sns}}$ correspondingly. A UAV with a failed sensor is assumed to be unable to perform surveillance, and UAV with a failed actuator unable to perform surveillance or communication. When a UAV returns to the base, it is refueled and all failures are assumed repaired. We refer to the wellness of the sensor and actuator as the health of the UAV.

The previous works on PST missions [2,26,30] showed that having an accurate health model of each agent is crucial to generate an efficient plan. In particular such health models can be *state-correlated*, meaning that the uncertain parameter for each agent p^i is a mapping from the state space $\mathcal{S}$ to $[0, 1]$. For instance, the nominal fuel burning probability may depend on the location of the UAV. Assuming that this map is constant (state independent uncertainty) or ignoring to update the parameter might negatively affect mission performance [27].

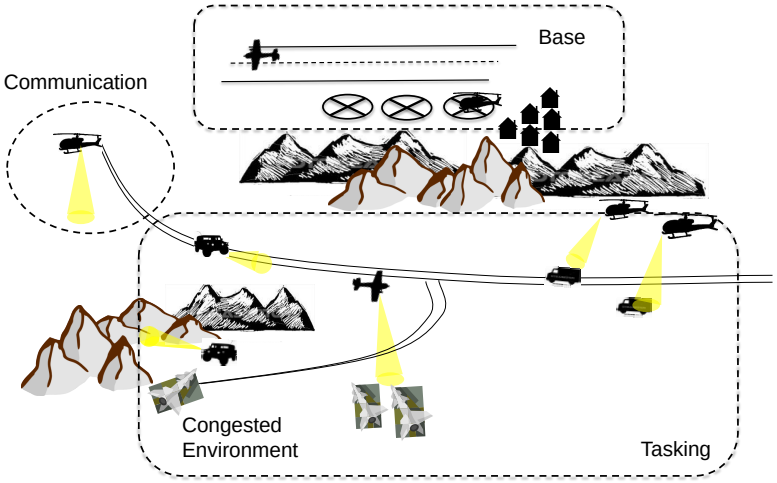


Fig. 1: Persistent Search and Track mission. Goal is to maintain persistent surveillance of two targets, while maintaining a data link with the base by having a capable UAV with loiter in the communication area.

There are two facets of the problem of learning such health models. First, when the state space is large, representing $p^i(s)$ as a look-up table becomes infeasible and an appropriate set of features should be selected for approximation. Although these health models can be learned by each agent independently, due to fact that agents operate in the same mission environment, sharing of relevant features might boost the convergence of learning significantly. The second challenge is the development of a feature sorting and sharing method under limited communication. In the next subsection we give an overview of how these challenges were addressed.

2.2 An Overview of Our Approach

To address the problem of planning with learned models in large-scale heterogeneous teams, an integrated planning-learning approach was employed, displayed in Figure 2. The basic idea of the framework is enabling each agent to have its own decentralized learning algorithm (The Dec-iFDD algorithm developed in Section 3) and communicate these learned models across each other and to a planning algorithm. The planning algorithm can be also decentralized depending on the mission formulation and requirement. This framework was originally developed to solve UAV Persistent Search and Track Mission with uncertain health dynamics [28], in this work we show that it is generalizable to a larger class of planning under certainty problems.

The framework uses the linear function approximation represent the state-correlated uncertainties,

$$p(s) \approx \bar{p}(s) = \theta^T \phi(s), \tag{1}$$

where $\phi(s)$ is the feature set and θ is the weight vector. The challenge of finding a good ϕ is handled by the iFDD algorithm's ability to expand ϕ based on the observed transitions [7], hence the problem of determining model complexity is solved

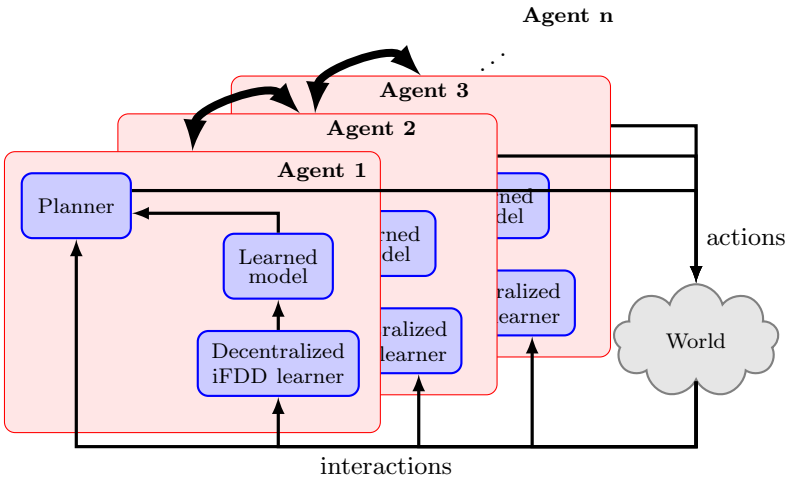


Fig. 2: The integrated planning and learning architecture used in this paper employs decentralized collaborative online learning (see Section 3) to learn the models of mission dynamics. These learned models are fed to an planning algorithm. Arrows in the figure indicate that the agents communicate with each other their representations of the uncertainty.

internally. The Dec-iFDD algorithm sorts the features ϕ^i for each agent based on their corresponding weights θ^i and allows each agent to only share their most heavily weighted features. The maximum number of shared features can be tuned based on the communication constraints. This sharing methodology addresses the problem of model sharing with communication constraints and improves the convergence speed of the planning performance. The next section presents the formal presentation of the algorithm.

3 Decentralized Learning with Dec-iFDD

3.1 Background

3.1.1 Markov Decision Processes

The problem of sequential decision making under uncertainty is formulated as an MDP, which is defined as the tuple

$$\mathcal{M} = \langle \mathcal{S}, \mathcal{A}, \mathcal{P}^a_{ss'}, \mathcal{R}^a_{ss'}, \gamma \rangle \,,$$

where $\mathcal{S}$ is the discrete state space, $\mathcal{A}$ is the discrete set of actions, $\mathcal{P}^a_{ss'}$ is the state transition model that denotes the probability of transitioning to state s' when action a is applied at state s. $\mathcal{R}^a_{ss'}$ is a known reward model representing the reward for executing action a in state s and transitioning to state s'. $\gamma \in [0, 1]$ is the discount factor used to balance relative weights of current and future rewards.

Only MDPs with discrete and finite state space are considered. A *policy* is a mapping $\pi : \mathcal{S} \rightarrow \mathcal{A}$ from state to action space. Together with the initial state s_0, a policy forms a trajectory $z = \{s_0, a_0, r_0, s_1, a_1, r_1, \cdots\}$. For each time-step t, $a_t = \pi(s_t)$, and both r_t and s_{t+1} are sampled from the reward and transition models correspondingly. The *value* of each state-action pair under policy π is defined as:

$$Q^{\pi}(s,a) = E_{\pi}\left[\sum_{t=0}^{\infty} \gamma^t r_t \middle| s_0 = s, a_0 = a\right], \tag{2}$$

which is the expected sum of discounted rewards obtained starting from state s, taking action a and following the policy π thereafter. The optimal policy π^* is given by the solution of the following optimization problem:

$$\pi^*(s) = \arg\max_a Q^{\pi^*}(s,a). \tag{3}$$

3.1.2 Learning State-Dependent Uncertainties

Let $p : \mathcal{S} \rightarrow [0,1]$ denote the state dependent uncertainty function that needs to be learned to model the uncertainty in the underlying MDP. For instance, in the context of the PST mission $p(s)$ can be the probability of burning nominal amount of fuel in state s. Hence, in order to capture environment dynamics, this correlation should be estimated. The uncertain parameter p, can be treated as function that maps the state $s \in \mathcal{S}$ to a probability $p(s) \in [0,1]$,

$$p(s) = P(\langle s, a, s' \rangle \in E | s), \tag{4}$$

where E is the event associated with the uncertain parameter. Events are sets of state transitions that define the physical meaning behind the uncertain parameter, such as sensor failure, fuel consumption and communication loss. A straightforward approach to solve the parameter/model estimation problem would be treating each $p(s), s \in \mathcal{S}$ differently, hence estimating $|\mathcal{S}|$ parameters concurrently. It is evident that this approach easily becomes inefficient and even intractable when $|\mathcal{S}|$ is large. This issue can be alleviated by introducing a linear function approximation:

$$p(s) \approx \hat{p}^l(s) = \phi^{\top}(s)\theta^l, \tag{5}$$

where $\hat{p}^l(s)$ is the approximate representation at l^{th} step and $\phi(s)$ is the vector of features. Each component ϕ_j is a binary feature characterized by a mapping from state to a binary value; $\phi_j(s) : s \rightarrow \{0,1\}, j = 1, ..., m$, where m is the total number of features and $\theta^l \in \mathbb{R}^m$ is the weight vector at step l. A feature ϕ_j is called *active* at state s if $\phi_j(s) = 1$. Set of active features at state s is given by $\mathbb{A}(s) = \{j | \phi_j(s) = 1\}$. The weight θ of the representation can be updated as follows. At each step, new estimates are formed by updating θ by looping through each observed state transition at that step. Let z be the observed trajectory of state transitions at so, $z = \{(s_l, a_l, s_l^{'}), l = 1, 2, ..., N_{exec}\}$, where N_{exec} is the length of the trajectory. The steps l are referred to as learning steps. At each learning step, θ^l is updated to θ^{l+1} by processing the l^{th} element of observed trajectory z. Update is performed using gradient descent on the squared estimation error $\frac{1}{2}[p(s^l) - \hat{p}(s^l)]^2$. This results in an update law of the form,

$$\theta^{l+1} = \theta^l + \alpha^l \Delta^l(s^l)\phi(s^l), \tag{6}$$

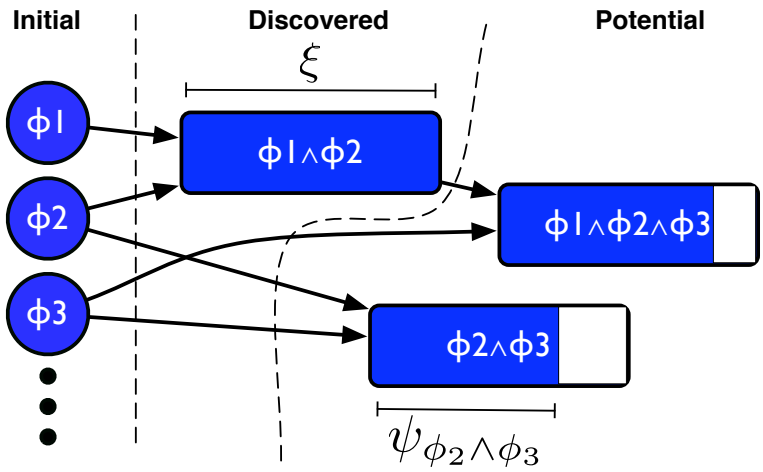


Fig. 3: The discovery step of the iFDD algorithm, initial features are circles, conjunctive features are rectangles. The relevance $\Delta(s)$ of a potential feature is the filled part of the rectangle. Potential features are discovered if their relevance $\Delta(s)$ reaches the discovery threshold ζ.[7]

where $\alpha^l \in [0,1]$ appropriate step-size parameter and $\Delta^l(s) = \zeta(s) - \hat{p}(s) = \zeta(s) - \phi(s)^T\theta^l$. Here $\zeta(s)$ is an indicator function for the event induced by p, that is $\zeta(s) = 1$ if $(s,a,s^{'}) \in E$ and $\zeta(s) = 0$ otherwise. Full derivation of the update law in Eq. 6 can be found in[27]. Updates of this form are usually referred to as Stochastic Gradient Descent (SGD)[10].

3.2 Adaptive Feature Selection with iFDD

It is easy to see that selection of the feature vector in Eq. 5, is a key step in the design process of the learning algorithm. Quality of the resulting approximation depends strongly on the representation capability of these features[4]. Hence it is desirable to add an additional loop to the learning process, which adaptively adjusts the set of features based on the performance of the learning algorithm.

The iFDD method is an adaptive function approximation method for estimating value functions encountered in reinforcement learning problem. Results showed that, when coupled with a value function update rule that is stable under function approximation (such as SARSA[22]), iFDD outperforms the existing methods both in sparsity and planning performance. Ure[27] extended the use of iFDD to represent transition probabilities of MDPs. The basic idea of iFDD is to expand the representation by adding conjunctions of the initial features based on an error metric, thus reducing the error in parts of the state space where the feedback error persists (see Fig. 3). The general outline of the algorithm is as follows: given a set of initial binary features, when performing the update for state $s \in z$, a conjunction set $\phi^c(s) = \{\phi_j(s) \wedge \phi_k(s) | j,k \in \mathbb{A}(s)\}$ is formed. These features are referred to as candidate features. If the sum of sampled estimation error $\Delta(s)$ over active

candidate features exceeds some pre-determined threshold ξ, these conjunctions are added to set of features. Interested reader is directed to[27] for further insight on the implementation of the algorithm and analysis of its convergence properties.

3.3 The Dec-iFDD Algorithm

In many realistic scenarios the model of the uncertainty is not only state dependent, but is also agent dependent. Let n represent the number of agents and let $p^i(s)$ represent the state dependent uncertainty functions that needs to be learned to model the uncertainty in the MDP of the i^{th} agent. A naive generalization of SGD-iFDD to the decentralized setting can be obtained by running a separate SGD-iFDD for each UAV. In this setting, each UAV updates its own representation based on the individual observations. This approach does not leverage the ability of the agents to collaborate. In particular, even though the environment may affect each agent in a different way the underlying features used to represent the environment should be common, as the agents all operate in the same environment. The main idea behind the Decentralized iFDD (Dec-iFDD) algorithm presented in this subsection is increasing the efficiency in learning by allowing agents to share the iFDD discovered features with each other under communication constraints. The pseudocode of the Dec-iFDD algorithm is provided in Algorithm 1.

The line by line description of the Algorithm 1 is as follows, the algorithm takes number of agents n, the set of binary features for each agent ϕ_{init} and the cap on shared features ϕ_{cap} as the input. At each step, agents interact with the environment to receive observations (line 10), and then each agent applies the standard iFDD algorithm independent of each other to update it's current set of features ϕ^i as well as the corresponding weights θ^i (line 11). When the current step is a sharing step, each agent sorts it's feature based on the weights (line 5), and then the first ϕ_{cap}/n number of features with the largest weights are broadcast in the network (line 6 and 7). Then all the agents update their feature set with the broadcasted features and then set the weights for their new features (line 8).

There are two striking properties of the algorithm. First, note that the algorithm only allows agents to share features and not the weights each other. This is especially important for the heterogeneous teams, where the agents may share common features due to operating in the same environment, but each feature is weighted differently due to heterogeneity of the team. Secondly, the algorithm takes the communication limits into consideration with capping the total number of shared features in the network by the parameter ϕ_{cap}, and forces agents to only share the more heavily weighed features in their representations at each sharing step. The implications of these two properties are shown in the following simulations.

4 Simulation Results

4.1 The Simulation Setting and The Compared Approaches

For each multi-agent MDP, we consider the learning of a single state-correlated uncertainty $p^i(s)$ per agent, where $i = 1, .., n$ indexes the agents. For all missions

Algorithm 1: Dec-iFDD Model Learning Algorithm

Input: Number of Agents n, Set Initial Binary Features ϕ_{init}, Shared Feature Cap ϕ_{cap}
Output: Estimated model $\hat{p}^i$ for each agent $i = 1, ..., n$

Initialize $\phi^i = \phi_{init}$ Initialize $\theta^i = 0$
foreach *Step* **do**
  foreach *Agent* **do**
    if *Step* = *Sharing Step* **then**
      $\phi'^i \leftarrow$ Sort Features$(\phi^i, \phi_{cap}, \theta^i)$
      Broadcast(ϕ'^i)
      $\phi^+ \leftarrow$ Listen()
      $\phi^i \leftarrow \phi^i \cup \phi^+$, $\theta^i \leftarrow$ Update Theta(ϕ^i)
    else
      $s, a, s' \leftarrow$ Observe Transition
      $\phi^i, \theta^i \leftarrow$ Expand Representation(s, a, s')

it is assumed that a nominal model $p^{\text{nom}}(s)$ and the underlying $p^i(s)$ for each agent was generated by randomly perturbing this nominal model. A team is called *homogeneous*, when the underlying $p^i(s)$ for each agent is within 5% limits of the nominal model. A team is called *heterogeneous*, when the underlying $p^i(s)$ for each agent is within 20% limits of the nominal model.

For both heterogeneous and homogeneous teams, we compare 4 different learning approaches. *The centralized learner with homogeneity assumption* assumes that $p^i(s)$ are the same for all agents, and therefore concatenates observations from all agents and processes these by the iFDD algorithm to generate a single model. Then all agents plan using this single model. Alternative to the centralized learner, the Dec-iFDD algorithm (Algorithm 1) with 3 different feature sharing caps (FSC), $\{0, 10, 100\}$, was compared. Note that FSC= 0 corresponds to the case where each agent learns completely independently and shares nothing with each other.

Two domains are inspected, the multi-agent block building problem, which is the multi-agent extension of a classical AI problem and the PST mission, which was used as the motivating problem at the Section 2. The main objective of these simulation results is to show that, if the team is heterogeneous, the centralized learning can actually hurt the performance and it is outperformed by distributed learning approach.

4.2 Multiagent Block Building Problem

This problem formulation is motivated by classical blocksworld problem that appears as a benchmark in many different AI applications [21]. Problem consists of picking up the blocks on a table and arranging them into a specific configuration, usually a tower. The stochastic dynamics are introduced to the system though p_{fall}, which denotes the state-independent probability that a block may fall from the top of the tower back to the table. This formulation of the problem have been solved by many different planning algorithms in the past [7]. The authors have formulated an alternate version of the problem [27] with state-dependent p_{fall}. In this formulation, probability of the block falling down increases as the tower gets

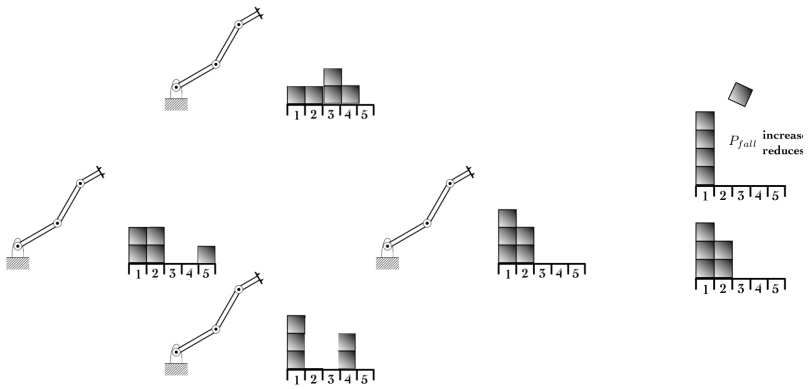


Fig. 4: The multiagent blocksworld problem with 4 agents. The baseline probability model correlates the configuration and the probability of blocks falling

higher and decreases with the presence of adjacent blocks. The objective of the planning/learning problem is to learn the structure of $p_{\text{fall}}(s)$ and build the optimal block configuration.

This paper presents a decoupled multi-agent version of the blocksword problem with state dependent uncertainty. As shown in Fig. 4, the problem consists of coordinating $n = 4$ robotic arms for building the blocks. The dynamic of each block building problem is completely decoupled from each other, thus the planning can be performed independently. For this mission learning is executed at every 300 steps for each robot, and the MDP that corresponds to the learned models were solved by using the value iteration algorithm [1]. All results are averaged over 30 runs.

The nominal model is given as follows. Let n_{slot} be the number of blocks in the destination slot, and n_{adj} be the maximum number of blocks in adjacent towers. Define $\bar{p}_{\text{fall}} = 0.1 \times (n_{slot} - n_{adj})^2 + 0.1$. Then p_{fall} is calculated as:

$$p_{\text{fall}}^{\text{nom}} = \begin{cases} 0 & \bar{p}_{\text{fall}} \leqslant 0 \text{ or } n_{slot} = 0 \\ \bar{p}_{\text{fall}} & 0 < \bar{p}_{\text{fall}} < 1 \\ 1 & \bar{p}_{\text{fall}} \geqslant 1 \end{cases}.$$

The results for the homogeneous team is given in Fig. 5. Results show that the centralized algorithm required $\approx 30\%$ less number of steps to converge compared to the Dec-iFDD algorithm with the largest FSC. It can be concluded that for this homogeneous scenario, using the same weights θ and features ϕ for all the agents resulted in a good approximation.

The results for the heterogeneous team are given in Fig. 6. The results show that centralized learning algorithm is unable to converge in this case because the algorithm generates a single model by processing all the observations, but the underlying samples are generated by substantially different models. The Dec-iFDD algorithm on the other hand, learns individual models for each agent and hence the agents can adapt their plans to their specific models. In addition, as the FSC increases, agents are able to share more relevant features with each other and the convergence rate of the overall performance increases significantly.

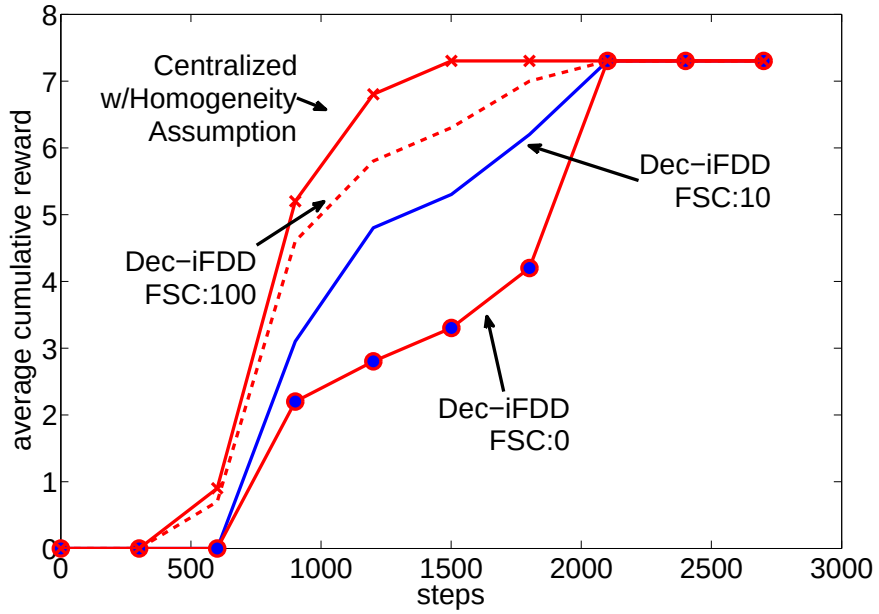


Fig. 5: Comparison of centralized approach versus decentralized approaches with different feature sharing caps (FSC) for the multiagent blocksworld scenario where model perturbation is 5% from the nominal model

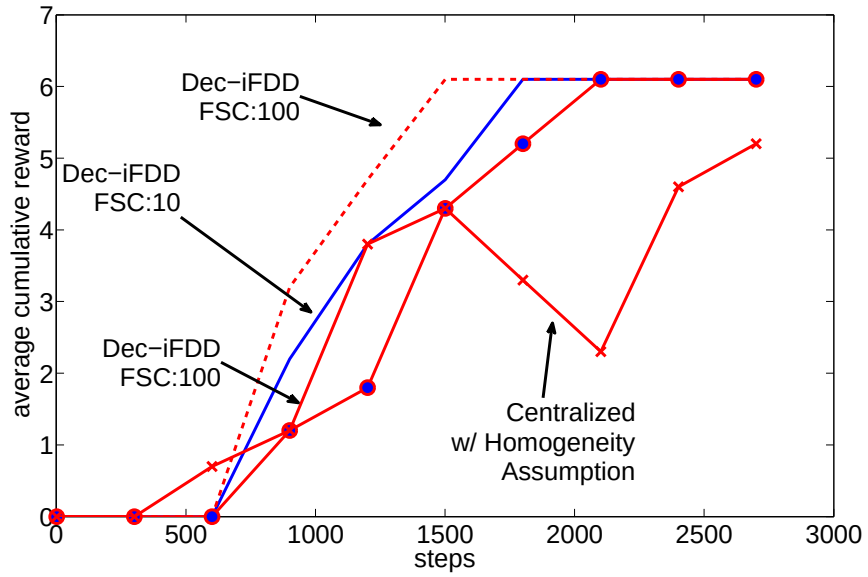


Fig. 6: Comparison of centralized approach versus decentralized approaches with different feature sharing caps (FSC) for the multiagent blocksworld scenario where model perturbation is 20% from the nominal model

Table 1: Baseline probability of actuator failure across different locations and fuel levels.

	Fuel Level	
Location	High Fuel Level	Low Fuel Level
Base	0.0	0.0
Communication	0.05	0.1
Surveillance	0.2	0.3

Table 2: Baseline probability of nominal fuel rate across different locations and health states.

	Health		
Location	Healthy	Sensor Fail	Actuator Fail
Base	0.0	0.0	0.0
Communication	0.95	0.92	0.86
Surveillance	0.95	0.92	0.86
Surveillance (Windy)	0.90	0.80	0.75

4.3 Persistent Search and Track with UAVs

For simulations, a large scale PST simulation with 10 collaborating UAVs was considered. It was assumed that the state-correlated actuator failure probability $p_a(s)^i$ and state-correlated nominal fuel burn rate for each agent i is unknown at the beginning of the mission, and must be learned through interactions with the environment. The baseline probability of actuator failure model that is used in the simulations is given at Table 1 and the baseline nominal fuel burning rate is given at Table 2. State-dependency of the fuel burning rate was motivated from the possibility of having non-uniform wind in the mission environment and thus each region induces a different fuel burning rate. Table 2 shows the probability of incurring nominal fuel burn in the respective states. Non-nominal fuel burn rate is set as twice the nominal fuel burn rate.

The Group Aggregate Dec-MMDP (GA-Dec-MMDP) algorithm[26] was used to replan after every model update. Similar to how the Dec-iFDD algorithm decomposes the learning problem to per agent, the Ga-Dec-MMDP planner decomposes the planning problem per agent to enable online planning in large-scale multiagent missions. The development, and additional simulation and flight results for the GA-Dec-MMDP algorithm are available[26,28].

Figure 7 compares the cumulative cost performance of several instances of the Dec-iFDD algorithm with different feature sharing caps for almost homogeneous UAV team. In this case, it can be seen that as the cap on features to be shared is increased, the performance of the algorithm approaches that of the centralized algorithm. This result reinforces the intuition that the decentralized algorithm can do no better than a centralized one when the agent models of uncertainty are homogeneous. Figure 8 on the other hand, shows the performance of the Dec-iFDD algorithm for the same set of feature caps but in a network of collaborating UAVs with more heterogeneity (20% perturbation from a nominal model). In this case, it can be seen that planning using the output of the decentralized algorithm results

in much less cumulative cost, because the planner is able to adjust the plan to suite the capability of each individual agent.

In order to demonstrate the proactive behavior of the resulting policy, two different mission metrics were averaged over 100 simulation runs and results are displayed on Table 3. It is seen that the proposed planning-learning approach leads to policies with less number of total failures in the expense of commanding vehicles to return to base more frequently. This behavior is due to ability of Dec-iFDD to learn a specific model for each agent, opposed to centralized planner.

Finally, it is possible to compare feature sharing to an alternative approach in which all observations are shared by estimating the average number of shared features, in terms of communication cost. Let κ be the size of a message that can be passed in the network, which corresponds to an single integer. Thus, each initial feature and a component of the state vector corresponds to messages of size κ. The average number of features shared in the simulations were computed to be $\approx 4.2\kappa$. Hence the total load on the network is $4.2\phi_{cap}\kappa$, since he maximum amount of features shared in the network is limited to ϕ_{cap} by Algorithm 1. In the considered scenario, each observed state transition corresponds to 84κ sized messages. Hence if n agents share observations instead of features, they would need to send $84\kappa n$ messages. For the simulation results presented above, number of agents $n = 10$, thus the load on network for passing observations instead of features is around 840κ. Hence, even with a feature sharing cap of 100, sharing features require less load on network in terms of communication cost, compared to sharing observations. In addition, it is seen that size of observations scale linearly with number of agents, while the number of shared features is fixed for a constant $\phi_{cap}f$. Hence the designer can tune ϕ_{cap} solely based on the limits of the communication network, while direct sharing of measurements is limited by the size of the observations and the number of agents in the scenario.

5 Flight Results

This section presents the results of an hour long indoor flight test on a large-scale PST mission with 10 UAV agents, 10 recharge stations[25,26], and several other ground agents. Of the 10 UAV agents, 9 are simulated (referred to as virtual agents) using physics based simulations, and 1 agent is physically present in the experiment area (referred to as real agent). The experiment is conducted at Aerospace Controls Lab (ACL) RAVEN testbed[8], which is equipped with a Vicon

Table 3: Evaluation of averaged mission metrics for centralized planner with homogeneity assumption and coupled GA-Dec-MMDP Dec-iFDD planner with FSC = 100.

Planner	# Failures	# Base Visits
Centralized Homogeneous	90.2	81.2
GA-Dec-MMDP with Dec-iFDD	55.1	125.4

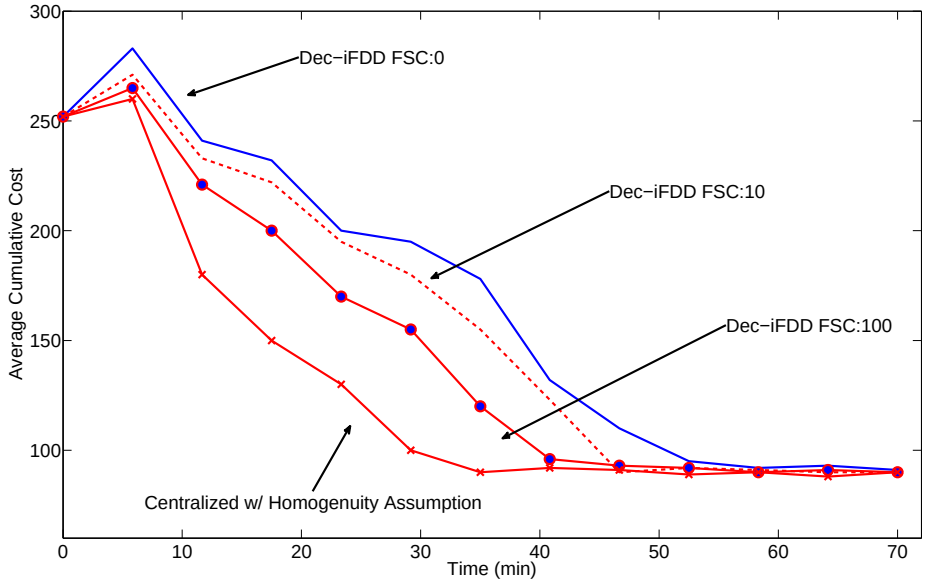


Fig. 7: Comparison of centralized approach versus decentralized approaches with different feature sharing caps (FSC) for the PST scenario where the model perturbation is 5% from the nominal model. Results are averaged over 100 runs.

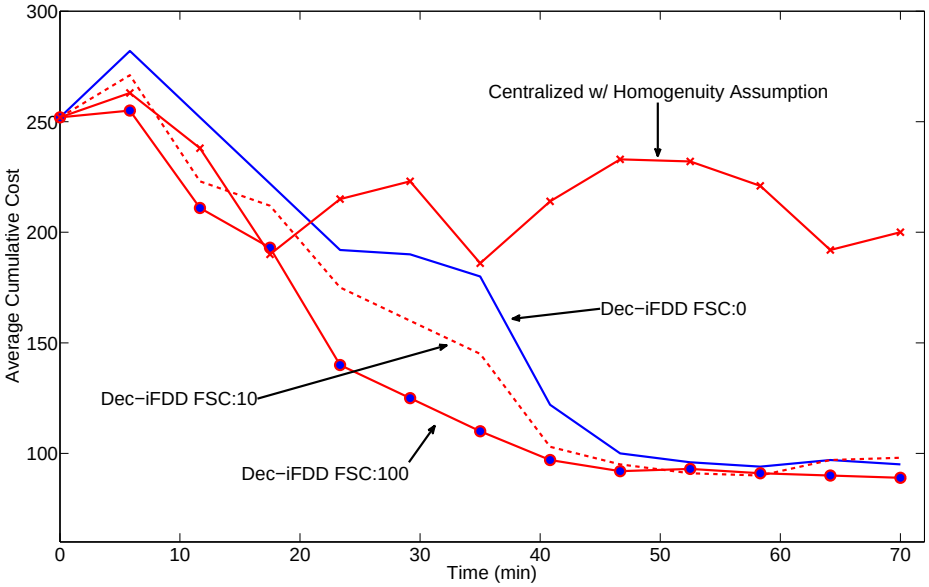


Fig. 8: Comparison of centralized approach versus decentralized approaches with different feature sharing caps (FSC) for the PST scenario where the model perturbation is 20% from the nominal model. Results are averaged over 100 runs.

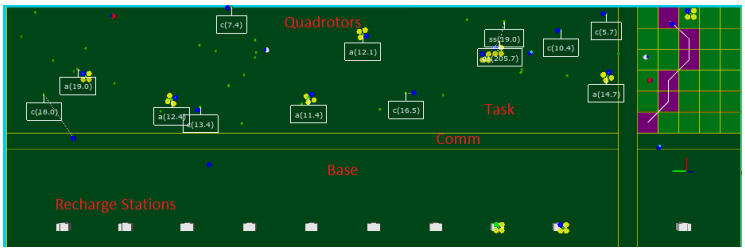


Fig. 9: Visual representation of the test environment showing Base, Communication and Tasking areas. The real quadrotor is constrained to operate in the RAVEN environment (right part of figure), and the rest of the team (9 quadrotors) operate in the simulated area to the left. The recharge station is shown at the bottom.

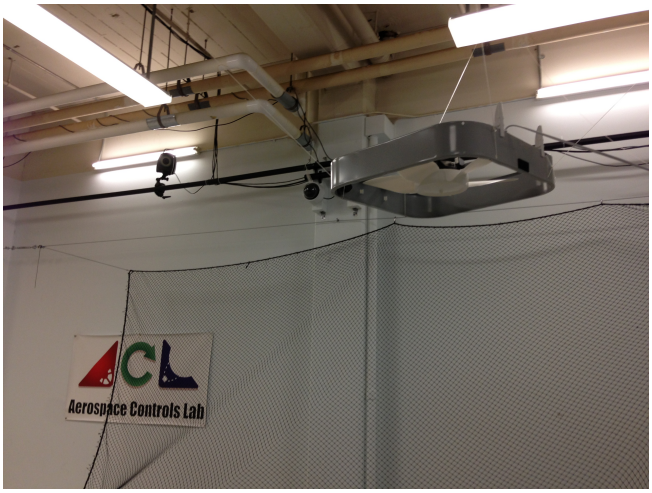


Fig. 10: A box fan is hang from the ceiling to simulate the effect of localized wind.

motion capture system. Custom designed quadrotor UAVs and an autonomous battery swap/recharge station [25,26] are used.

5.1 Experiment I: Single Fan and State-Independent Fuel Burn Rate Learning

In order to verify the applicability of the decentralized learning and planning approach developed in this paper, an additional scenario where each UAV has a different state-dependent fuel burning rate was considered in experimental setting. State-dependency of the fuel burning rate was motivated from the possibility of having non-uniform wind in the mission environment and thus each region induces a different fuel burning rate. Table 2 shows the probability of incurring nominal fuel burn in the respective states. Else, the vehicle experiences twice the nominal fuel burn.

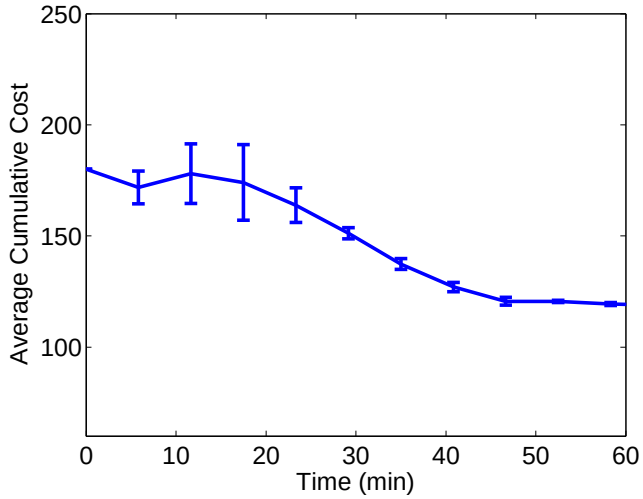


Fig. 11: Results from a large scale persistent mission experiment with 9 simulated quadrotors and one physical quadrotor. A stochastic model of agent fuel burning rate is learned using decentralized iFDD learning. It can be seen that as the fuel burning model is learned better the cumulative mission cost is reduced.

To implement the effect of localized wind, a 20-inch diameter box fan was hung from the ceiling in the RAVEN environment, as illustrated in Figure 10. On high power setting, the box fan can produce a downward wind flow of approximately 5 m/s over a 0.25 m^2 area. From combined on-board voltage and current reading, the system can reliably distinguish between windy and non-windy region. Note that the existence of a windy environment is not known to the planning-learning algorithm a priori, learning the existence of the wind and its impact on the fuel burning rate is the objective of the learning algorithm.

Results of the experiment averaged over 4 different runs is displayed in Figure 11. As the learning algorithm learns a better model for the fuel burning rate, the performance of the policy produced by the planner for allocating UAVs between the base and surveillance locations improves. It can be seen that the planner computes a more efficient policy using the continually improving model. In particular, UAVs in windy areas are returned back to base more frequently to ensure that they do not run out of fuel and UAVs in non-windy areas are allowed to spend more time in the surveillance area to accomplish more tasks.

5.2 Experiment II: Two Fans and State-Dependent Fuel Burn Rate Learning

The test environment is depicted in Fig. 9. The mission environment is separated such that real and virtual agents do not cross each other's operational spaces. The actuator failure probabilities are set to the values used for the heterogeneous team model described in the Section 4.3, and the nominal fuel burn rate probabilities for all the virtual agents are set according to the heterogeneous team model described

in the Section 4.3. The nominal fuel burn probability of the real agent is influenced by the custom setup of vertical fans implemented in the experiment area (see Figs. 12 and 13), which is explained later in this section. The GA-Dec-MMDP planner[26] is used to coordinate the 10 agents across the base, communication and tasking areas, however the health models are not known to the planner, and must be estimated during the experiment. Objective of the experiment is to show that the Dec-iFDD algorithm succeeds in learning these models and the GA-Dec-MMDP planner can lower the mission cost progressively.

The tasking area for the virtual agents is populated with randomly generated tasks, which are represented by the colored dots in Fig. 9. Allocation of these tasks among virtual agents are handled by the CBBA task allocation algorithm[5].

The tasking environment for the real agent is shown in Figs. 12 and 13. As depicted in these figures, the real agent has two static target observation tasks (task 1 has a higher reward than task 2) and the agent must fly through a region with localized wind disturbances, created using two vertically aligned fans, in order to perform the tasks. However the locations and the magnitude of the localized wind disturbances are unknown to the agent a priori. The wind disturbances generated by the fans significantly impact the battery life of the quadrotor, as can be inferred from Fig. 14. The figure shows that flying under the wind (blue line) results in approximately 17% more current drawn from the battery compared to not flying under the wind (red line), thus it can be stated that flying under the wind corresponds to higher battery discharge rate, which results in shorter flight times before returning to base for battery swap. We would like to emphasize that impact of the wind on the battery life of the agent is not known to the planner in the beginning of the mission.

The tasking space of the real agent is separated into several grids, with each grid inducing a different unknown fuel burn rate based on whether the fan above the grid is active or not. Since the wind can appear and disappear during the mission, it is more appropriate to model their effect probabilistically rather than having a binary wind or no wind value for each grid. Hence, the probability of nominal fuel burn rate p_{fuel} is a function of the agent's grid location. Therefore, p_{fuel} can be treated as a state-dependent uncertainty (see Section 3) and is learned here using the iFDD algorithm. In the actual experiment, during the period where the real agent is in the tasking area, the agent collects observations of battery usage at each grid it had traveled to. When the agent returns to the base, the iFDD algorithm processes this batch of observations to generate an estimate of $p_{fuel}(grid)$. This process is referred to as one iteration of wind learning. After each wind learning iteration, a dynamic programming algorithm is used to re-plan a trajectory that minimizes the based on the current estimate of $p_{fuel}(grid)$. The actual policy that the agent implements is randomly chosen from the optimal policy obtained from the DP algorithm or from a random policy, with the probability of choosing random actions reducing over time . This is an ϵ-greedy approach (see[22]), with $\epsilon = 0.2$, designed to encourage exploration.

In the beginning of the mission, only the Fan 1 is on and the fan on the the top of Task 1 (Fan 2) is turned off. Fig. 15 displays a sequence of selected trajectories for the real agent during different stages of the learning process. At the first wind learning iteration, the planner routes the agent towards the task with the higher reward (Fig. 15a), during agents 3^{rd} visit to the tasking area The Fan 2 is manually turned on unknown to the agent. It can be seen that the learning

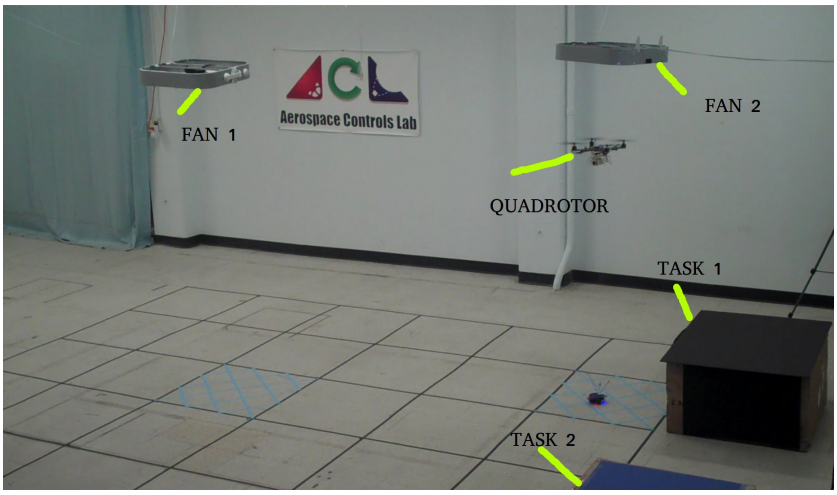


Fig. 12: View of the Task area of the real agent.

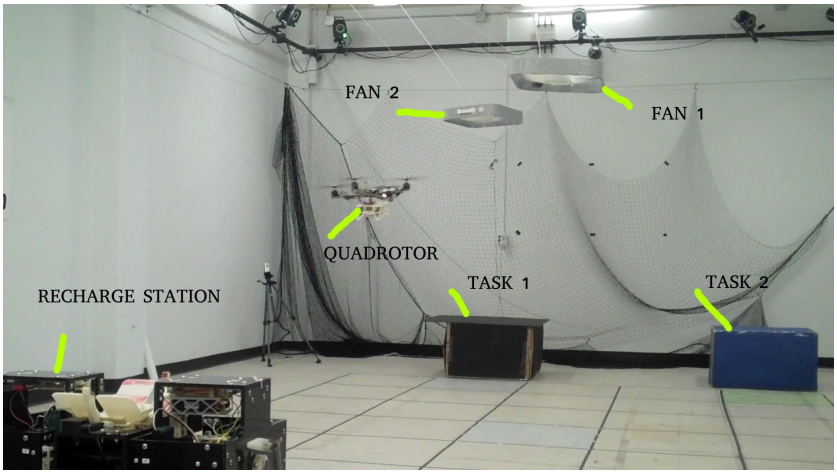


Fig. 13: View of the Task area of the real agent from Base.

algorithm identifies the non-zero probability of experiencing wind for this grid (Fig. 15b). After taking more observations of the environment, the planner discovers that doing Task 1 corresponds to higher probability of non-nominal fuel burn rates and starts to route the agent towards Task 2 (Fig. 15c). After 10 wind learning iterations, planner converges to a trajectory that has the least probability of burning non-nominal fuels (Fig. 15d). The overall planning performance of the whole team is presented in the average cumulative cost versus time plot on Fig. 16. These results are consistent with their simulation counterparts in Section 4.3, in the sense that the Dec-iFDD algorithm learns the actuator failure and fuel burn models across the team and the GA-Dec-MMDP planner is then able to provide

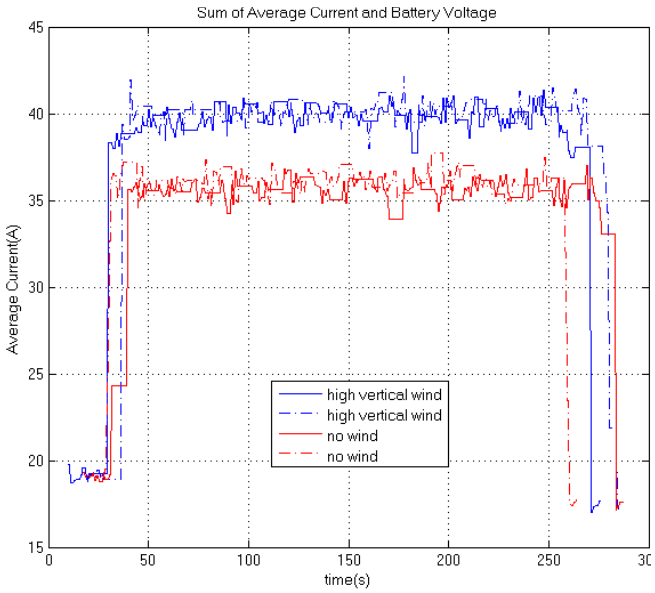
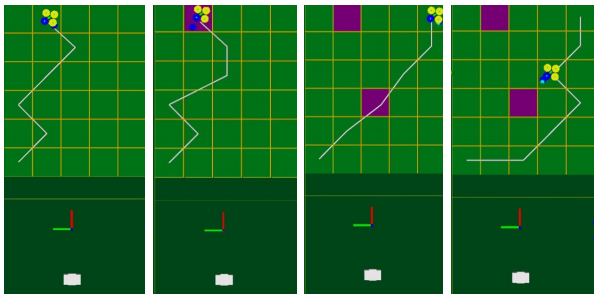


Fig. 14: Compares the current drawn from the battery while flying under the fan with current drawn while flying away from it.



(a) (b) (c) (d)

Fig. 15: Selected trajectories of the real agent during the wind learning process. Purple grids, which are learned by the agent, correspond to grids with non-zero probability of experiencing wind. 15a: After 1 wind learning iteration, 15b: After 3 wind learning iterations ,15c: After 7 wind learning iterations, 15d: After 10 wind learning iterations

improved (lower) average cost. These flight experiment and simulation results verify the applicability of the GA-Dec-MMDP planner and the Dec-iFDD learner for large-scale UAV missions with unknown health models and heterogeneous team structure.

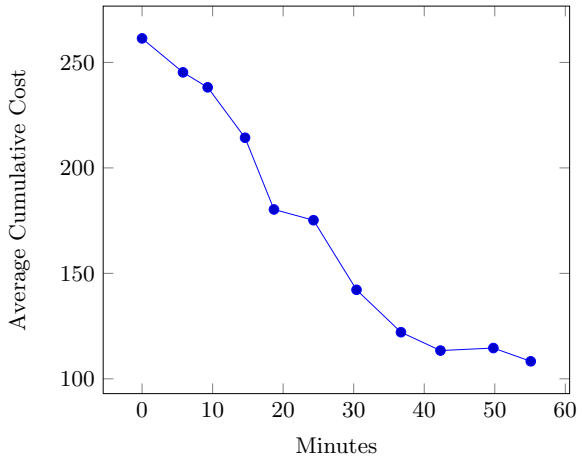


Fig. 16: Average cost of the mission as the learning and replanning progresses for the large-scale flight experiment

6 Conclusions

The paper presented an decentralized learning framework to address the problem of model learning and sharing for large-scale multiagent missions. The developed decentralized incremental feature dependency discovery (Dec-iFDD) algorithm enables each agent to rank the features of their model and share the most relevant features with their teammates under limited communication. The simulation and the flight tests results show that, when there is significant heterogeneity in agents, the integrated planning/learning framework that periodically re-plans using online estimated models of agent uncertainty outperforms the alternative centralized approach that assumes homogeneity in agents.

Acknowledgments

This research was generously supported by Boeing Research & Technology.

References

1. Bertsekas D (2005) Dynamic Programming and Optimal Control. Athena Scientific
2. Bethke B, Bertuccelli LF, How JP (2008) Experimental demonstration of adaptive MDP-based planning with model uncertainty. In: AIAA Guidance Navigation and Control, Honolulu, Hawaii
3. Busoniu L, Babuska R, De Schutter B (2008) A comprehensive survey of multiagent reinforcement learning. Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on 38(2):156–172

4. Busoniu L, Babuska R, Schutter BD, Ernst D (2010) Reinforcement Learning and Dynamic Programming Using Function Approximators. CRC Press
5. Choi HL, Brunet L, How JP (2009) Consensus-based decentralized auctions for robust task allocation. IEEE Transactions on Robotics 25(4):912–926, DOI 10.1109/TRO.2009.2022423, URL `http://dx.doi.org/10.1109/TRO.2009.2022423`
6. Djuric P, Wang Y (2012) Distributed bayesian learning in multiagent systems: Improving our understanding of its capabilities and limitations. Signal Processing Magazine, IEEE 29(2):65 –76, DOI 10.1109/MSP.2011.943495
7. Geramifard A, Doshi F, Redding J, Roy N, How J (2011) Online discovery of feature dependencies. In: Getoor L, Scheffer T (eds) International Conference on Machine Learning (ICML), ACM, pp 881–888
8. How JP, Bethke B, Frank A, Dale D, Vian J (2008) Real-time indoor autonomous vehicle test environment. IEEE Control Systems Magazine 28(2):51–64
9. Krishnamurthy V (2010) Quickest time herding and detection for optimal social learning. arXiv preprint arXiv:10034972
10. Kushner HJ, Yin GG (2003) Convergence of indirect adaptive asynchronous value iteration algorithms. Springer
11. LaValle S (2006) Planning Algorithms. Cambridge University Press
12. MacKenzie DC, Arkin R, Cameron JM (1997) Multiagent mission specification and execution. Autonomous Robots 4(1):29–52
13. Monostori L, Váncza J, Kumara SR (2006) Agent-based systems for manufacturing. CIRP Annals-Manufacturing Technology 55(2):697–720
14. Painter-Wakefield C, Parr R (2012) Greedy algorithms for sparse reinforcement learning. In: International Conference on Machine Learning (ICML), ACM, pp 968–975
15. Pan SJ, Yang Q (2010) A survey on transfer learning. Knowledge and Data Engineering, IEEE Transactions on 22(10):1345–1359
16. Panait L, Luke S (2005) Cooperative multi-agent learning: The state of the art. Autonomous Agents and Multi-Agent Systems 11(3):387–434
17. Powell W (2007) Approximate Dynamic Programming: Solving the curses of dimensionality. Wiley-Interscience, pp. 225–262
18. Puterman ML (1994) Markov decision processes
19. Redding JD (2012) Approximate multi-agent planning in dynamic and uncertain environments. PhD thesis, Massachusetts Institute of Technology, Department of Aeronautics and Astronautics, Cambridge MA
20. Redding JD, Toksoz T, Ure NK, Geramifard A, How JP, Vavrina M, Vian J (2011) Persistent distributed multi-agent missions with automated battery management. In: AIAA Guidance, Navigation, and Control Conference (GNC), (AIAA-2011-6480)
21. Russell S, Norvig P (2003) Artificial Intelligence: A Modern Approach, 2nd Edition. Prentice-Hall, Englewood Cliffs, NJ
22. Sutton R, Barto A (1998) Reinforcement Learning, an Introduction. MIT Press, Cambridge, MA
23. Sutton R, Szepesvári C, Geramifard A, Bowling M (2008) Dyna-style planning with linear function approximation and prioritized sweeping. In: Proceedings of the 25th International Conference on Machine Learning, Helsinki, Finland

24. Thrun S, Burgard W, Fox D (2005) Probabilistic Robotics (Intelligent Robotics and Autonomous Agents). The MIT Press
25. Toksoz T (2012) Design and Implementation of an Automated Battery Management Platform. Master's thesis, Massachusetts Institute of Technology
26. Ure NK, Chowdhary G, Redding J, Toksoz T, How J, Vavrina M, Vian J (2012) Experimental demonstration of efficient multi-agent learning and planning for persistent missions in uncertain environments. In: Conference on Guidance Navigation and Control, AIAA, Minneapolis, MN
27. Ure NK, Geramifard A, Chowdhary G, How JP (2012) Adaptive Planning for Markov Decision Processes with Uncertain Transition Models via Incremental Feature Dependency Discovery. In: European Conference on Machine Learning (ECML), URL `http://acl.mit.edu/papers/Ure12ECML.pdf`
28. Ure NK, Chowdhary G, Chen CM Yu Fan, How JP, Vian J (2013) Decentralized learning based planning multiagent missions in presence of actuator failures. In: International Conference on Unmanned Aircraft Systems, IEEE, Atlanta GA
29. Ure NK, Chowdhary G, Chen YF, How JP, Vian J (2013) Health-aware decentralized planning and learning for large-scale multiagent missions. In: Conference on Guidance Navigation and Control, AIAA, Washington DC
30. Ure NK, Chowdhary G, How JP, Vavarina M, Vian J (2013) Health aware planning under uncertainty for uav missions with heterogeneous teams. In: Proceedings of the European Control Conference, Zurich, Switzerland, (to appear)
31. Weibull JW (1997) Evolutionary game theory. MIT press
32. Yao H, Sutton RS, Bhatnagar S, Dongcui D, Szepesvári C (2009) Multi-step dynamic planning for policy evaluation and control. In: NIPS, pp 2187–2195

LAtex docs + style files + bib files + pictures
Click here to download Compressed File: Ure13JINT.zip