Intelligent Cooperative Control Architecture: A Framework for Performance Improvement Using Safe Learning

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

Intelligent Cooperative Control Architecture: A framework for performance improvement using safe learning

Alborz Geramifard† · **Joshua Redding**⋆ · **Jonathan P. How**†

Received: date / Accepted: date

Abstract Planning for multi-agent systems such as task assignment for teams of limited-fuel unmanned aerial vehicles (UAVs) is challenging due to uncertainties in the assumed models and the very large size of the planning space. Researchers have developed fast cooperative planners based on simple models (*e.g.,* linear and deterministic dynamics), yet inaccuracies in assumed models will impact the resulting performance. Learning techniques are capable of adapting the model and providing better policies asymptotically compared to cooperative planners, yet they often violate the safety conditions of the system due to their exploratory nature. Moreover they frequently require an impractically large number of interactions to perform well. This paper introduces the intelligent Cooperative Control Architecture (iCCA) as a framework for combining cooperative planners and reinforcement learning techniques. iCCA improves the policy of the cooperative planner, while reduces the risk and sample complexity of the learner. Empirical results in gridworld and task assignment for fuel-limited UAV domains with problem sizes up to 9 billion state-action pairs verify the advantage of iCCA over pure learning and planning strategies.

Keywords MDPs · Cooperative Planning · Reinforcement Learning · Risk Management

1 Introduction

Applications involving multiple autonomous robots typically require that participating agents remain capable of performing their advertised range of tasks

† Laboratory for Information and Decision Systems
Massachusetts Institute of Technology
E-mail: {agf,jhow}@mit.edu

⋆ VTOL Embedded Systems
Lockheed Martin Procerus Technologies
E-mail: josh.redding@lmco.com

in the face of noise, unmodeled dynamics and uncertainties. Many cooperative control algorithms have been designed to address these and other, related issues such as humans-in-the-loop, imperfect situational awareness, sparse communication networks, and complex environments [Beard et al., 2002, Casal, 2002, Choi et al., 2009, Ketan Savla, 2008, Olfati-Saber et al., 2007, Ren et al., 2007, Saligrama and Castañón, 2006, Wang et al., 2007, Xu and Ozguner, 2003]. While these methods had success in a variety of simulations and some focused experiments, there remains room to improve overall performance in real-world applications. For example, cooperative control algorithms are often based on simple, abstract models of the underlying system. This may aid computational tractability and enable quick analysis, but at the cost of ignoring real-world complexities such as noisy dynamics and stochastic outcomes [Cassandras and Li, 2002, Singh et al., 2009, Wang et al., 2005]. Section 2 further provides specific examples regarding the assumptions made in the literature to obtain fast planners.

Researchers have long understood the negative impact of these modeling errors on decision-making algorithms [Ben-Tal et al., 2006, Bertsimas et al., 2011, Zhu and Fukushima, 2009], but the simple and robust extensions of cooperative control algorithms to account for such errors are often overly conservative and generally do not utilize observations or past experiences to refine poorly known models [Alighanbari et al., 2006, Bertuccelli, 2008]. Despite these issues however, cooperative control algorithms provide a baseline capability for achieving challenging multi-agent mission objectives. In this context, the following research question arises: *How can current cooperative control algorithms be extended to result in improved plans?*

To address this question, we integrate learning with cooperative control algorithms through the intelligent Cooperative Control Architecture (iCCA), so that the planner can improve its solution over time based on its experiences. The general iCCA template shown in Figure 1 consists of customizable modules for implementing strategies against modeling errors and uncertainties by integrating cooperative control algorithms with learning techniques and a feedback measure of system performance. It is well known that most learning algorithms are more effective when given some prior knowledge to guide the search and/or steer exploration. In the iCCA framework, the cooperative planner offers this capability, ensuring that mission objectives are achieved even as learning proceeds. In return, the learning algorithm enhances the performance of the planner by offering adaptability to time-varying parameters.

Learning approaches for solving large-scale control problems can handle unknown models, but they do not have a mechanism to avoid risky behaviors. For example, in the context of an unmanned aerial vehicle (UAV) mission planning scenario, the learner might send UAVs with low fuel to remote locations solely for the purpose of learning about the consequences of such behaviors. To a human operator who has domain knowledge, such actions may not be acceptable because losing a UAV is costly and the risk of losing a UAV is high under this plan. On the contrary, cooperative planners provide safe plans based on prior knowledge, yet their performance may be suboptimal in ac-

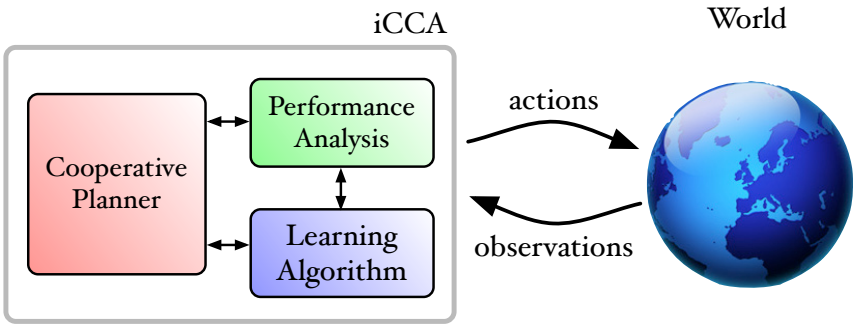


Fig. 1 **i**ntelligent **C**ooperative **C**ontrol **A**rchitecture, a template framework for the integration of cooperative control algorithms and machine learning techniques [Redding et al., 2010b].

tuality. [Alighanbari, 2004, Beard et al., 2002, Casal, 2002, Choi et al., 2009, Olfati-Saber et al., 2007, Ryan et al., 2004, Saligrama and Castañón, 2006, Wang et al., 2007, Xu and Ozguner, 2003].

This paper introduces a novel approach that takes advantage of cooperative planners and domain knowledge to mitigate the risk of learning, reduce the overall sample complexity, and boost the performance of cooperative planners. Empirical evaluations in gridworld domains and task assignment missions for multiple UAVs with problem sizes up to 9 billion state-action pairs demonstrate the advantage of our technique. Our approach is shown to provide better solutions compared to pure planning methods, and safer plans compared to pure learning techniques with lower sample complexity.

The structure of the paper is as follows. Section 2 provides the literature review on cooperative planners and safety in RL. Section 3 illustrates the goal of this paper through a pedagogical example. Section 4 covers the mathematical framework and two existing learning techniques used in the paper. Section 5 explains the iCCA and our instantiation of iCCA to integrate cooperative controllers with reinforcement learning (RL) techniques. Section 6 focuses on a simplified problem where **I**) the applied RL method has an explicit policy formulations and **II**) the model of the system is assumed to be known and fixed. Section 7 relaxes the first assumption by extending the work to support RL methods with implicit policy forms. Section 8 relaxes the second assumption by allowing the system model to be partially known and adapt over time. Section 9 concludes the paper by highlighting the contributions. Parts of this article were published as separate papers [Geramifard et al., 2011a,b, Redding et al., 2010a,b].

2 Related Work

In the controls community, many researchers have developed algorithms for task assignment among teams of vehicles under the name of cooperative plan-

ning [Alighanbari, 2004, Alighanbari et al., 2003, Beard et al., 2002, Berman et al., 2009, Cassandras and Li, 2002, Castanon and Wohletz, 2009, Choi et al., 2009, Ryan et al., 2004, Saligrama and Castañón, 2006, Wang et al., 2007, 2005]. As the name suggests, these methods use an existing world model for planning vehicles task assignment. While, in theory, such planning problems can be solved using dynamic programming (DP) [Bellman, 2003], the computation time required for solving realistic problems using DP is not practically feasible. Consequently, cooperative planners are often focused on problems with specific properties such as convexity, submodularity, *etc.* that render the problem more tractable. Furthermore, they investigate approximation techniques for solving planning problems.

[Cassandras and Li, 2002] adopted a receding horizon approach for planning in obstacle free 2D spaces where vehicles with constant speeds move according to deterministic dynamics. [Singh et al., 2009] proposed a non-myopic approach and derived theoretical guarantees on the performance of their algorithm. Their main results are built upon the submodularity (a diminishing returns property) assumption which is a limiting factor. [Alighanbari et al., 2003] formulated the task allocation problem as a mixed-integer linear program. They scaled their solution to large domains including 6 vehicles by approximating the decomposition of task assignment among vehicles. The main drawback of the work is the assumption of deterministic dynamics for vehicle movements. Wang et al. [2005] introduced a new method for maintaining the formation among vehicles in domains with dynamic obstacles. Their approach assumes a quadratic cost function and a Gaussian noise model. In our UAV mission planning scenarios both these assumptions are violated: the penalty function is step-shaped and the noise involved in the reward function has a multinomial distribution. Castanon and Wohletz [2009] focused on stochastic resource allocation problem. While their method scales well to domains with 20 resources and 20 tasks, it cannot be applied to cases where multiple assignments of simultaneous resources are required for the task completion. Berman et al. [2009] tackled the problem of task allocation for swarms of robots in a distributed fashion using stochastic policies. Their approach addresses the problem of maintaining a predefined distribution of robots on a set of locations, yet does not support time varying distributions. Choi et al. [2009] used a distributed auction-based approach for task assignment among agents. They bounded the sub-optimality of their solution, yet extending their work to stochastic models is still an open problem. In summary, cooperative planners often result in sub-optimal solutions due to the model inaccuracy or unsatisfied assumptions. Furthermore, these methods do not incorporate learning to use perceived trajectories in order to improve future performance.

Safe exploration has been of special interest to practitioners applying RL techniques to real world domains involving expensive resources. Pessimistic approaches find optimal policies with respect to the worse possible outcome of selected actions [Heger, 1994]. Consequently, resulting policies are often overly constrained, yielding undesirable behavior in practice. Bounding the probability of failure, risk-sensitive RL methods [Geibel and Wysotzki, 2005,

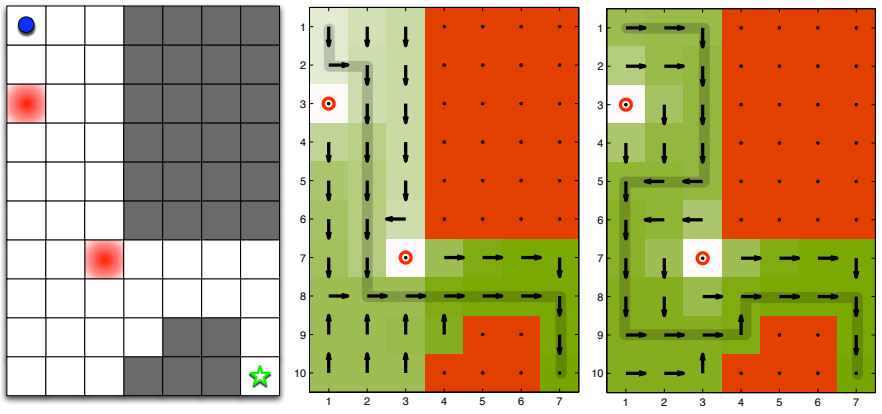
Mihatsch and Neuneier, 2002] provide a less conservative approach by maximizing the performance subject to an acceptable risk level. Because these methods do not guarantee the performance before the learning phase, they are not suitable for online settings. Abbeel and Ng [2005] investigated exploration within the context of batch apprenticeship learning. They conjectured that running least-squares policy iteration [Lagoudakis and Parr, 2003] over teacher generated trajectories yields safe policies for practical domains. While in practice they reported safe policies, their approach does not have mathematical guarantees. For deterministic MDPs, Hans et al. [2008] extended the risk-sensitive RL approach by identifying states as super critical, critical, and safe. While they demonstrated promising empirical results, their method cannot be applied to stochastic domains. Knox and Stone [2010] discussed various ways that an initial policy can bias the learning process, yet their approach requires access to the value function of the given policy which may be hard to calculate. Robust MDPs is another framework to capture the uncertainties about the model and tune the policy to be robust within the space of possible models [Bagnell et al., 2001, Nilim and Ghaoui, 2005]. These methods do not take advantage of the online data and their resulting policies can be overly conservative in practice. Recent work has extended robust MDPs to an adaptive framework by incorporating observed interactions to reduce the model uncertainty [Bertuccelli et al., 2012]. While similar in some aspects to the approach given here, it is restricted to parameter estimation within the model learning, and that work also does not explicitly capture the notion of risk that is a key element of the iCCA learning. Within the controls community, safe exploration is pursued under robust RL in which the stability of the system is of main interest, but current state of the art methods do not extend to general MDPs as they consider systems with linear transition models [Anderson et al., 2007].

3 A Pedagogical Example: GridWorld-1

This section provides a pedagogical example, demonstrating how inaccurate models can lead into suboptimal policies. In Section 6, we show how the optimal policy can be obtained by using learning techniques.

Consider a grid world scenario shown in Figure 2(a), in which the task is to navigate a UAV from the top-left corner ($\bullet$) to the bottom-right corner ($\star$). Red areas highlight the danger zones where the UAV will be eliminated upon entrance. At each step the UAV can take any action from the set $\{\uparrow, \downarrow, \leftarrow, \rightarrow\}$. However, due to wind disturbances, there is 30% chance that the UAV is pushed into any unoccupied neighboring cell while executing the selected action. The reward for reaching the goal region and off-limit regions are $+1$ and -1 respectively, while every other move results in -0.001 reward.

Figure 2(b) illustrates the policy (shown as arrows) calculated by a planner using dynamic programming [Sutton and Barto, 1998] that is unaware of the



(a) GridWorld-1 (b) Plan with the noiseless model (c) Plan with the true model

Fig. 2 GridWorld-1 (a), the corresponding policy calculated with a planner assuming deterministic movement model and its true value function (b) and the optimal policy with the perfect model and its value function (c). The task is to navigate from the top left corner highlighted as • to the right bottom corner identified as ⋆. Red regions are off-limit areas which the UAV should avoid. The dynamics model has 30% noise of moving the UAV to a random free neighboring grid cell. Gray cells are not traversable.

wind, together with the nominal path highlighted as a gray tube. As expected, the path suggested by the planner follows the shortest path that avoids directly passing through off-limit areas. The color of each cell represents the true value of each state (*i.e.,* including the wind) under the planner's policy. Green indicates positive, white indicates zero, and red indicates negative values. We set the value for blocked areas to $-\infty$, hence the intense red color. The optimal policy and its corresponding value function and nominal path are shown in Figure 2(c). Notice how the optimal policy avoids the risk of getting close to off-limit areas by making wider turns. While the new nominal path is longer, it mitigates the risk better. In fact, the new policy raises the mission success rate from 29% to 80%, while boosting the value of the initial state by a factor of approximately three. Model-free learning techniques such as SARSA can find the optimal policy through mere interaction, although they require a plethora training examples. More importantly, they might deliberately move the UAV towards off-limit regions just to gain information about those areas. However, when integrated with the planner, the learner can rule out intentionally poor decisions. Furthermore, the planner's policy can be used as a starting point for the learner to bootstrap on, reducing the amount of data the learner requires to master the task.

This paper explains how planner solutions based on approximated models (*i.e.,* Figure 2-b) can be improved using learning techniques, while at the same time, the risk in the learning process is reduced.

4 Background

4.1 Markov Decision Process (MDP)

An MDP [Sutton and Barto, 1998] is a tuple defined by $(\mathcal{S}, \mathcal{A}, \mathcal{P}^a_{ss'}, \mathcal{R}^a_{ss'}, \gamma)$ where $\mathcal{S}$ is a set of states, $\mathcal{A}$ is a set of actions, $\mathcal{P}^a_{ss'}$ is the probability of getting to state s' by taking action a in state s, $\mathcal{R}^a_{ss'}$ is the corresponding reward, and $\gamma \in [0, 1]$ is a discount factor that balances current and future rewards . A *trajectory* is a sequence $s_0, a_0, r_0, s_1, a_1, r_1, s_2, \ldots$, where the action $a_t \in \mathcal{A}$ is chosen according to a *policy* $\pi : \mathcal{S} \times \mathcal{A} \to [0, 1]$ mapping each state-action pair to a probability. The agent selects the action in each specified state using its policy. Every consequent state is generated by the environment according to the transition model (*i.e.,* $i \geq 1, s_{i+1} \sim \mathcal{P}^{a_i}_{s_i}$). Given a policy π, the state-action value function, $Q^\pi(s, a)$, is the expected sum of the discounted rewards for an agent starting at state s, taking action a, and then following policy π thereafter:

$$Q^\pi(s, a) = E_\pi \left[\sum_{t=0}^{\infty} \gamma^t r_t \middle| s_0 = s, a_0 = a \right]. \tag{1}$$

In finite state spaces, $Q^\pi(s, a)$ can be stored in a table. The goal of solving an MDP is to find the optimal policy which maximizes the expected cumulative discounted rewards in all states. In particular, the optimal policy π^* is defined as:

$$\forall s, \pi^*(s) = \underset{a \in \mathcal{A}}{\operatorname{argmax}} \, Q^{\pi^*}(s, a). \tag{2}$$

4.2 Reinforcement Learning in MDPs

The underlying goal of the two reinforcement learning (RL) algorithms presented here is to improve performance of the cooperative planning system over time using observed rewards by exploring new agent behaviors that may lead to more favorable outcomes. The details of how these algorithms accomplish this goal are discussed in the following sections.

4.2.1 SARSA

A popular approach among MDP solvers is to find an approximation to $Q^\pi(s, a)$ (policy evaluation) and update the policy with respect to the resulting values (policy improvement). Temporal Difference learning (TD) [Sutton, 1988] is a traditional policy evaluation method in which the current $Q(s, a)$ is adjusted based on the difference between the current estimate of Q and a better approximation formed by the actual observed reward and the estimated value of the following state. Given $(s_t, a_t, r_t, s_{t+1}, a_{t+1})$ and the current value estimates, the temporal difference (TD) error, δ_t, is calculated as:

$$\delta_t(Q) = r_t + \gamma Q^\pi(s_{t+1}, a_{t+1}) - Q^\pi(s_t, a_t).$$

The one-step TD algorithm, also known as TD(0), updates the value estimates using:

$$Q^{\pi}(s_t, a_t) = Q^{\pi}(s_t, a_t) + \alpha \delta_t(Q), \tag{3}$$

where α is the learning rate. SARSA (state action reward state action) [Rummery and Niranjan, 1994] is basic TD for which the policy is directly derived from the Q values as:

$$\pi^{SARSA}(s, a) = \begin{cases} 1 - \epsilon & a = \text{argmax}_a \, Q(s, a) \\ \frac{\epsilon}{|\mathcal{A}|} & \text{Otherwise} \end{cases},$$

in which ties are broken randomly, if more than one action maximizes $Q(s, a)$. This policy is also known as the ϵ-greedy policy.

4.2.2 Actor-Critic

Actor-critic methods parameterize the policy and store it as a separate entity named *actor*. In this paper, the actor is a class of policies represented as the Gibbs softmax distribution:

$$\pi^{AC}(s, a) = \frac{e^{\rho(s,a)/\tau}}{\sum_b e^{\rho(s,b)/\tau}},$$

in which $\rho(s, a) \in \mathcal{R}$ is the preference of taking action a in state s, and $\tau \in [0, \infty)$ is a knob allowing for shifts between greedy and random action selection. Since we use a tabular representation, the actor update amounts to:

$$\rho(s, a) \leftarrow \rho(s, a) + \alpha Q(s, a)$$

following the incremental natural Actor-Critic framework [Bhatnagar et al., 2007]. The value of each state-action pair ($Q(s, a)$) is held by the *critic* and is calculated/updated in an identical manner to SARSA, mentioned in Eqn. (3).

5 Intelligent Cooperative Control Architectur

Figure 1 depicts the general template of intelligent cooperative control architecture [Redding et al., 2010b]. The left rectangle with the gray boundary is the control box and consists of three elements:

- **Cooperative Planner**: Given a problem model, this module provides *safe* solutions with cheap computational complexity, often gained by simplifying the model. Cooperative planners are usually domain-dependent.
- **Learning Algorithm**: This component implements learning by looking at the past experiences of interactions. While in general any machine learning algorithm can be used, RL methods are used in this paper.

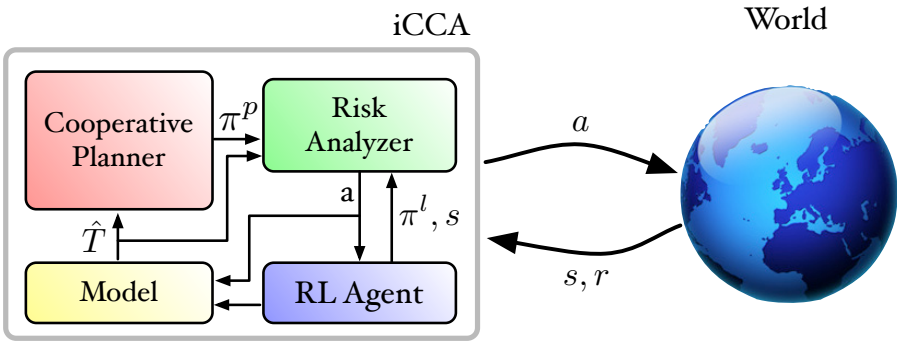


Fig. 3 The iCCA framework instantiation for safe integration of RL algorithms with cooperative planners with the ability to adjust the model online.

– **Performance Analysis**: This module regulates the interaction between the learner and the cooperative planner. The duties of this module can vary based on its instantiation. In this paper, its purpose is to evaluate the risk involved in executing the actions suggested by the learner.

The rest of the figure resembles the conventional interaction between an agent and an environment, in which, on every step the decision made by the control system is sent to the world and executed (*e.g.,* move a UAV to a certain location). During the execution, the command might get distorted (*e.g.,* the UAV moves forward, but a wind gust impedes it). The outcome of each command execution affects the world, and observations are returned to the control system. The observation may also get distorted through noise, but this paper does not consider partial observability. Note that Figure 1 depicts a general framework, and depending on the instantiation of the template, numerous algorithms can be derived.

5.1 Instantiating iCCA for Cooperative Learning

This sections explains how RL methods can be combined with cooperative planners through the iCCA instantiation in order to 1) mitigate the risk involved in the learning process, 2) improve the sample complexity of the learning methods, and 3) improve the performance of the cooperative planners. The high level idea is to use the solution of the cooperative planner fed with an approximate model to bias the policy of the RL agent in order to explore solutions close to the behavior of the cooperative planner. Furthermore, actions that are deemed not *safe* (*i.e.,* risky) are switched with the cooperative planner solution.

Figure 3 depicts our instantiation of the iCCA framework for merging RL methods with cooperative planners. An RL agent realizes the learning algorithm module, while the risk analyzer instantiates the performance analysis

box. Also note that observations are replaced with s, r, due to the full observability assumption. The underlying problem is formulated as an MDP with the true model $T = (\mathcal{P}, \mathcal{R})$. An approximate model of the MDP, $\hat{T} = (\hat{\mathcal{P}}, \hat{\mathcal{R}})$ is assumed to be available shown in the yellow box and is shared both by the cooperative planner and the risk analyzer.

5.1.1 Cooperative Planner

For small MDPs, given the approximated model, the Value Iteration algorithm [Sutton and Barto, 1998] is used to generate the planner's policy, (*i.e.,* π^p). For large UAV mission planning scenarios, however, running value iteration is not feasible. Therefore the consensus based bundle algorithm (CBBA) [Choi et al., 2009, Redding et al., 2010a] provides the solution. CBBA is a fast algorithm for task assignment among UAVs that uses a deterministic model of the system. All stochastic elements in $\hat{T}$ are replaced with the events with maximum expected values before being used in CBBA. For example if there is an 80% chance to move from one state to another state, achieving a reward of 100, the transition is assumed to be successful all the time with the corresponding reward of 80. This paper uses CBBA as a black box that takes an approximate model of the system and quickly provides a safe plan with respect to the assumed model that obtains good cumulative rewards. Other cooperative planners could easily be used to replace CBBA provided that they return safe policies quickly.

5.2 RL Agent

For the learning module, any RL method can be used, although since our work focuses on fast interaction between the control box and the world, using computationally expensive methods such as LSPI [Lagoudakis and Parr, 2003] is not advised. In this paper, we investigate the use of SARSA and Actor-Critic methods that were covered in Section 4.

5.3 Risk Analyzer

When using learning techniques, providing safety plays a major role in the applicability of the method in practical environments, where human lives and expensive equipments are involved. The purpose of the risk analyzer is to regulate the learner's actions by filtering those that are deemed too risky based on the assumed model. In general the model can involve stochasticity, hence the notion of risk should be defined based on probabilities accordingly. Algorithm 1 explains the risk analysis process. In particular, it is assumed that there exists a function $\texttt{constrained} : \mathcal{S} \rightarrow \{0, 1\}$, which indicates if being in a particular state is allowed or not.[1] Risk is defined as the probability of visiting

[1] Extending the `constrained` function to include actions as well as states is straight forward, yet for simplicity excluded.

Algorithm 1: safe (check the safety of the action suggested by the learner)

Input: s, a
Output: `isSafe`

```
risk ← 0
for i ← 1 to M do
    t ← 1
    s_t ~ T̂(s, a)
    while not constrained(s_t) and not isTerminal(s_t) and t < H
    do
        s_{t+1} ~ T̂(s_t, π^p(s_t))
        t ← t + 1
    risk ← risk + (1/i)(constrained(s_t) − risk);    /* Update Mean */
isSafe ← (risk < ε)
```

any of the constrained states. The core idea is to use Monte-Carlo simulation to estimate the risk level associated with the given state-action pair if planner's policy is applied thereafter by simulating $\mathcal{M}$ trajectories from the current state s. The first action is the learner's suggested action a, and the rest of actions come from the planner's policy, π^p. The approximated model, $\hat{T}$, is utilized to sample successive states. Each trajectory is bounded to a fixed horizon $\mathcal{H}$. The risk of taking action a from state s is estimated by the probability of a simulated trajectory reaching a risky state within horizon $\mathcal{H}$. If this risk is below a user-defined threshold, ε, the action is deemed to be safe. Note that the planner's policy is assumed to be safe with respect to the model in all states. Hence the estimated risk corresponds solely to the first action of each trajectory advised by the learner.

5.4 Model

This box captures the model estimate of the MDP, $\hat{T} = (\hat{\mathcal{P}}, \hat{\mathcal{R}})$. This estimate can be adjusted as more observations is received. In this paper, we first focus on fixed models. In Section 8, we relax this assumption by allowing the model to evolve over time, realizing a more accurate estimation.

5.5 Cooperative Learning: The High Level Control Loop

All sections so far described the mechanics inside each individual box. Consequently, a high level process, named *Cooperative Learning*, is required to tailor the functionality of all these modules together. We introduce three cooperative learning loops in the next three sections. The first cooperative learner assumes that **I**) the model estimate of the system remains fixed, and **II**) the RL agent has an explicit form of the policy (*e.g.,* Actor-Critic). The second cooperative

Algorithm 2: Cooperative Learning-1

Input: s, r
Output: a

1 $a^p \sim \pi^p(s)$ `/* CooperativePlanner */`
2 $a^l \sim \pi^l(s)$ `/* Learner */`
3 $a \leftarrow a^l$
4 **if not** $safe(s, a)$ **then**
5 $\quad a \leftarrow \pi^p$
6 $\quad \rho(s, a) \leftarrow \rho(s, a) - \lambda$
7 `ActorCritic.learn`(s, r, a)
8 **return** a

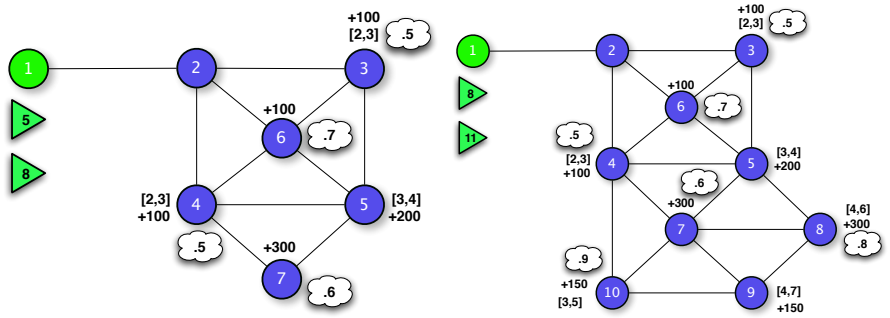
learner relaxes assumption **I**, while the third cooperative learner relaxes both assumptions **I** and **II**.

6 Using Fixed Models with Explicit Policy Forms

This section introduces the first cooperative planner that incorporates fixed models of the environment, while assumes that the RL agent has an explicit policy form (assumptions **I** and **II**). Algorithm 2 shows the pseudo-code for the main loop of the iCCA control box. First the safe action of the planner, a^p, and the learner action a^l are generated using the corresponding policies π^p and π^l (lines 1,2). The safety of the learning agent is then tested using the *safe* function (line 4). If the action is safe, it will be executed on the next step, otherwise the action is replaced with the planner's action, a^p, which is assumed to be safe (line 5). What this process dictates, however, is that state-action pairs explicitly forbidden by the *safe* function will not be intentionally visited. Therefore, if the *safe* function is built on a poor model, it can hinder the learning process in parts of the state space for which the safety is miscalculated due to the wrong model. To reduce the probability of the learner suggesting the same action, the preference corresponding to the unsafe action, $\rho(s, a)$, is reduced by λ (line 6). The λ parameter is picked by the domain expert to discourage suggesting unsafe actions by the learner. Furthermore, in order to bias the policy of the actor initially, the preferences of state-action pairs sampled from π^p are increased by the user-defined value λ. This initialization encourages the agent to select actions similar to the cooperative planner in the beginning of the learning process.

6.1 Empirical Evaluation

This section compares the performance of iCCA with respect to pure learning and pure cooperative planning approaches in both GridWorld-1 and more



(a) UAV 7-2 Scenario (b) UAV 10-2 Scenario

Fig. 4 The mission scenarios of interest: a team of two UAVs plan to maximize their cumulative reward along the mission by cooperating to visit targets. Target nodes are shown as circles with rewards noted as positive values and the probability of receiving the reward shown in the accompanying cloud. Note that some target nodes have no value. Constraints on the allowable visit time of a target are shown in square brackets.

complicated UAV mission planning scenarios. Figures 4 depicts the mission scenarios of interest where a team of two fuel-limited UAVs cooperate to maximize their total reward by visiting valuable target nodes in the network. The base is highlighted as node 1 (green circle), targets are shown as blue circles and agents are shown as triangles. The total amount of fuel for each agent is highlighted by the number inside each triangle. For those targets with an associated reward, it is given as a positive number nearby. The constraints on the allowable times when the target can be visited are given in square brackets and the probability of receiving the known reward when the target is visited is given in the white cloud near the node. If two agents visit a node at the same time, the probability of visiting the node would increase accordingly. Each reward can be obtained only once and traversing each edge takes one fuel cell and one time step. UAVs may loiter at any of the nodes indefinitely if, for some reason, they believe loitering to be the "optimal" action. The fuel burn for a loitering action is also one unit, except for any UAV at the base, where it is assumed to be stationary and its fuel level is therefore not depleted. The mission horizon was set to 8 time steps for UAV 7-2 scenario and 11 for the UAV 10-2 scenario.

6.1.1 UAV Mission Planning: MDP formulation

The state space was formulated as $[l_1, f_1, \ldots, l_n, f_n, v_1, \ldots, v_m, t]^T$, where l_i and f_i were integer values highlighting the location and the remaining fuel respectively for UAV i ($i \in 1 \ldots n$). v_j was a single bit signaling if node j had been visited before, where ($j \in 1 \ldots m$), and t was the current time step. There were n UAVs and m nodes participating in the scenario. The action space was $[l_1^+, \ldots, l_n^+]$ where l_i^+ was the node to which the agent was traveling.

The transition function ($\mathcal{P}^a_{ss'}$) was deterministic for the UAV position, fuel consumption, and time variables of the state space, while it was stochastic for the visited list of targets. The detailed derivation of the complete transition function should be trivial following the corresponding graph in Figure 4. That is, transitions were allowed between nodes for which there was an edge on the graph. The reward on each time step was stochastic and calculated as the sum of rewards from visiting new desired targets minus the total burnt fuel cells on the last move. Notice that a UAV received the target reward only if it landed on an unvisited node and lucky enough to obtain the reward. In that case, the corresponding visibility bit turned on, and the agent received the reward. The crash penalty or mission failure was equal to the negative sum of rewards at all nodes for both scenarios in order to prioritize safety over visiting targets. The mission failed if any UAV ran out of fuel or was not at the base by the end of the mission horizon.

6.1.2 Experimental Results

Both for GridWorld-1 and UAV 7-2 scenario, the optimal solutions were obtained using dynamic programming on the true model and used as the baseline for the optimality. Unfortunately, calculating an optimal solution was not feasible for the UAV 10-2 case, with about 9 billion state action pairs. This computation for UAV 7-2 scenario took about a day to calculate all expected values over more than 100 million state-action pairs. Thus, this approach cannot be easily scaled for larger sizes of the problem. For GridWorld-1, the `constrained` function was +1 for all off-limit (red) region and 0 otherwise. As for the UAV mission planning scenarios, the fixed model was the expected stochastic model of the world. The `constrained` function returned +1 only if any of the UAVs ran out of fuel or was not at the base by the end of horizon.[2] Since the assumed model in both domains were static one Monte-Carlo simulation is enough to calculate the risk $\in \{0, 1\}$. For baseline planners, the Value Iteration and CBBA methods were used for GridWorld-1 and the UAV mission planning scenarios correspondingly. Note that Value Iteration did not have access to the true model, while CBBA could not use the exact stochastic model due to its deterministic assumption of the dynamics. The quality of CBBA was probed on each domain by executing its policy online for 10,000 episodes. For Value Iteration the expected value of the initial state was simply fetched from the table. For each learning algorithm (*i.e.,* Actor-Critic and iCCA) the best learning rate was found empirically where the learning rate was calculated by:

$$\alpha_t = \alpha_0 \frac{N_0 + 1}{N_0 + \text{Episode } \#^{1.1}}.$$

[2] Since the transition model for this case is deterministic, for faster computation, we first calculated and stored all-pairs shortest paths using Floyd-Warshall algorithm [Cormen et al., 2001]. On each step, an action was assumed safe if after executing the action, the UAV has enough fuel to return to the base using the shortest path values. This process returns identical answers compared to Algorithm 1.

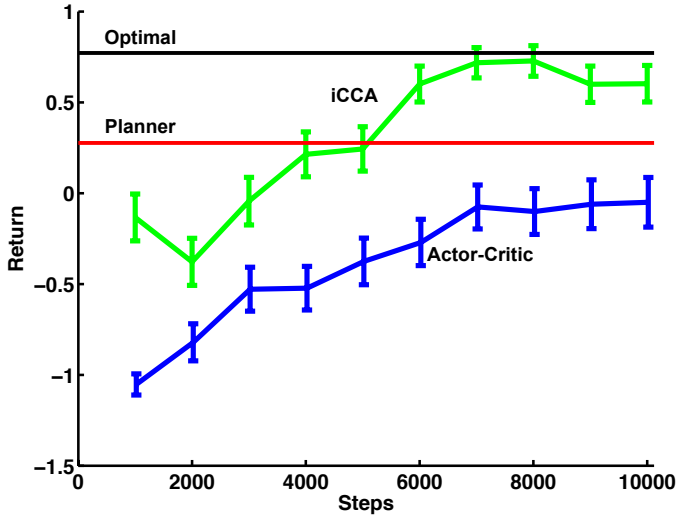
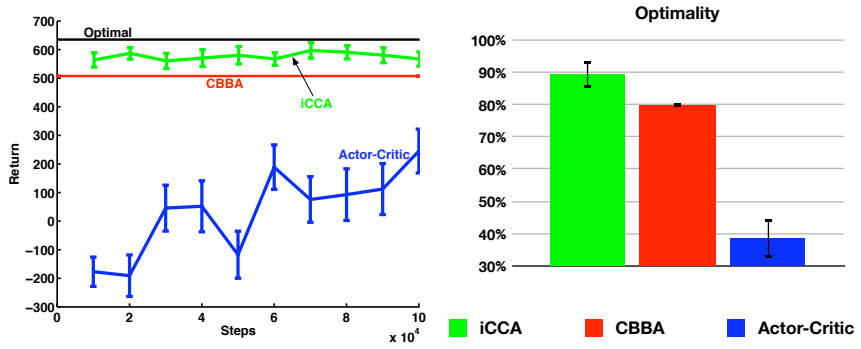


Fig. 5 A comparison of the collective rewards received in GridWorld-1 using cooperative planning alone (red), pure learning (blue), and when both are coupled via the iCCA framework (green). The optimal performance (black) was calculated via dynamic programming.

The best α_0 and N_0 were selected through experimental search of the sets of $\alpha_0 \in \{0.01, 0.1, 1\}$ and $N_0 \in \{100, 1000, 10^6\}$ for each algorithm and scenario. λ was set 100 and τ was set to 1 for the actor. The number of interactions for each simulation was limited to 10^4 and 10^5 steps for GridWorld-1 and UAV mission planning scenarios respectively. This led to a cap of 40 minutes of computation for each simulation on an Intel Xeon 2.40 GHz with 4 GB of RAM and Linux Debian 5.0. The performance of learning algorithms was extracted by running the greedy policy with respect to the existing preferences of the actor. For iCCA, unsafe moves again were replaced by the cooperative planner's solution. All learning method results were averaged over 60 runs except for the UAV 10-2 scenario for which it was averaged over 30 runs. Error bars represent 95% confidence intervals.

GridWorld-1 Figure 5 compares the performance of Actor-Critic, iCCA, the baseline planner (Fig 2-b), and the expected optimal solution (Fig 2-c) in GridWorld-1. The X-axis shows the number of steps the agent executed an action, while the Y-axis highlights the cumulative rewards of each method after each 1,000 steps. Notice how iCCA outperformed pure learning Actor-Critic. In particular iCCA outperformed the planner (red) after 6,000 steps by navigating farther from the danger zones. Actor-Critic, on the other hand, could not outperform the planner by the end of 10,000 steps. Note that the black line represents the expected return under the optimal policy, yet simulated trajectories may obtain higher return. This explains why the standard error around the iCCA's performance slightly exceeds the optimal line.

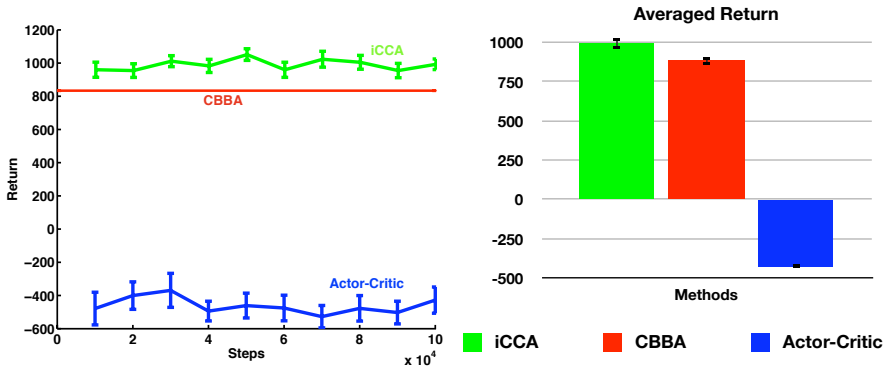


(a) Step based performance (b) Optimality after training

Fig. 6 (a) A comparison of the collective rewards received in UAV 7-2 using cooperative planning alone (red), pure learning (blue), and when both are coupled via the iCCA framework (green). The optimal performance (black) was calculated via dynamic programming. (b) The final performance of all methods scaled based on the expected performance of the worst and best policies.

UAV 7-2 Scenario Similarly, Figure 6(a) depicts the performance of Actor-Critic, iCCA, CBBA, and optimal policies in the UAV 7-2 scenario. The Y-axis shows the cumulative reward, while the X-axis represents the number of interactions. It is clear that the Actor-Critic performed much better inside the iCCA framework and performed better than CBBA alone. The reason is that CBBA provided a good starting point for the Actor-Critic to explore the state space, while the risk analyzer filtered risky actions of the actor leading to catastrophic situations. Figure 6(b) shows the performance of iCCA and Actor-Critic relative to the optimal policy after 10^5 steps of interaction with the domain and the averaged optimality of CBBA through 10,000 trials. Notice how the integrated algorithm could on average boost the best individual optimality performance (*i.e.,* CBBA's result) by 10%.

UAV 10-2 Scenario Figure 7(a) depicts the same set of results for the UAV 10-2 scenario. Since the size of the state-action pairs for this domain is about 9 billion, running dynamic programming to obtain the optimal policy was not feasible. For that reason the performance of the optimal policy is excluded. Since the state space was much larger than the UAV 7-2 scenario, Actor-Critic method had a hard time to find a sensible policy even after 10^5 steps. Online CBBA still could find a good policy to the approximated problem. When both CBBA and Actor-Critic were put together through the iCCA framework, the agent could achieve better performance even early on, after only 10^4 steps. Figure 7(b) shows the averaged performance of each method at the end of the learning phase. Notice that iCCA again could boost the performance of CBBA solution statistically significantly.



(a) Step based performance (b) Averaged return after training

Fig. 7 A comparison of the collective rewards received in UAV 10-2 scenario when strictly following plans generated by CBBA alone, Actor-Critic reinforcement learning outside of the iCCA environment, *i.e.,* without initialization and guidance from CBBA, and the result when these are coupled via the iCCA framework.

7 Relaxing the Explicit Policy Form Assumption

The initial policy of Actor-Critic type learners can be biased simply as they parameterize the policy explicitly. For learning schemes that do not represent the policy as a separate entity, such as SARSA, cooperative learning is not immediately obvious. This section presents a new cooperative learning algorithm for integrating learning approaches without an explicit actor component relaxing Assumption **I**. The idea is motivated by the concept of the $R_{\max}$ algorithm [Brafman and Tennenholtz, 2001]. The approach can be explained through the mentor-prodégé analogy, where the planner takes the role of the mentor and the learner takes the role of the prodégé. In the beginning, the prodégé does not know much about the world, hence, for the most part they take actions advised by the mentor. While learning from such actions, after a while, the prodégé feels comfortable about taking a self-motivated actions as they have been through the same situation many times. Seeking permission from the mentor, the prodégé could take the action if the mentor thinks the action is safe. Otherwise the prodégé should follow the action suggested by the mentor.

Algorithm 3 details the new cooperative learning process. On every step, the learner inspects the suggested action by the planner and estimates the "knownness" of the state-action pair by considering the number of times that state-action pair has been experienced following the planner's suggestion. The $\mathcal{K}$ parameter controls the transition speed from following the planner's policy to following the learner's policy. Given the knownness of the state-action pair, the learner probabilistically decides to select an action from its own policy. If the action is deemed to be safe, it is executed. Otherwise, the planner's policy

Algorithm 3: Cooperative Learning-2

Input: s, r
Output: a
1 $a \sim \pi^p(s)$ `/* CooperativePlanner */`
2 `knownness` $\leftarrow \min\{1, \frac{\texttt{count}(s,a)}{\mathcal{K}}\}$
3 **if** $rand() <$ `knownness` **then**
4 $\quad a' \sim \pi^l(s)$ `/* Learner */`
5 $\quad$ **if** $safe(s, a')$ **then**
6 $\quad\quad a \leftarrow a'$
7 **else**
8 $\quad$ `count`$(s, a) \leftarrow$ `count`$(s, a) + 1$
9 `learner.update`(s, r, a)
10 **return** a

overrides the learner's choice (lines 4-6). If the planner's action is selected, the knownness count of the corresponding state-action pair is incremented. Finally the learner is executed depending on the choice of the learning algorithm. Note that any RL algorithm, even the Actor-Critic family of methods, can be integrated with cooperative planners using Algorithm 3 since line 9 is the only learner-dependent line, defined in the general form.

7.1 Experimental Results

This section compares the empirical performance of SARSA combined with cooperative planners (CSARSA), with pure learning and pure planning methods in both the GridWorld-1 domain and the UAV 7-2 mission planning scenario. All cooperative planners and settings remained the same from the previous set of experiments in Section 6.1.2. The knownness parameter, $\mathcal{K}$, for CSARSA was empirically selected out of $\{10, 20, 50\}$. The exploration rate (ϵ) for SARSA and CSARSA was set to 0.1. All learning method results were averaged over 60 runs.

GridWorld-1 Figure 8 compares the performance of CSARSA, SARSA, the baseline planner (Fig 2-b), and the expected optimal solution (Fig 2-right) in the pedagogical GridWorld domain. The X-axis shows the number of steps the agent executed an action, while the Y-axis highlights the cumulative rewards of each method after each 1,000 steps. Notice how CSARSA outperformed pure learning approaches. Compared to Figure 5, SARSA based methods learned faster. This observation can be explained by two facts: 1) SARSA's policy is embedded in the $Q-$value function, whereas the actor requires another level of learning for the policy on the top of learning the $Q-$value function and 2) Algorithm 3 provides a better exploration mechanism (*i.e.,* $\text{R}_{\max}$ like) compared to Algorithm 2, where exploration is realized internally by the actor.

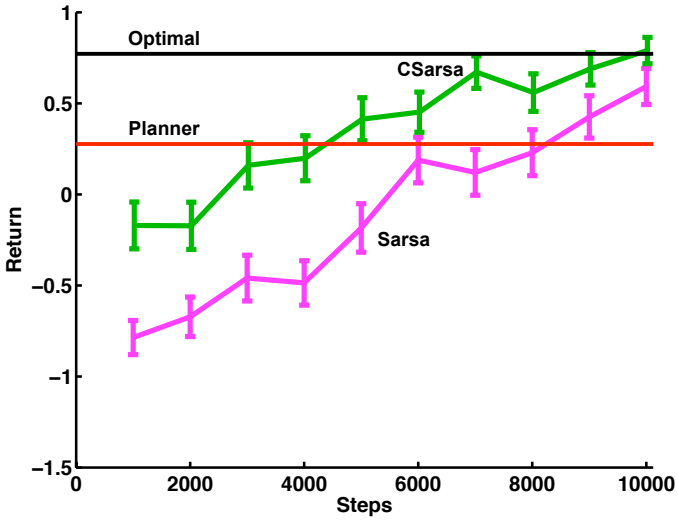


Fig. 8 In the pedagogical GridWorld domain, the performance of the optimal solution is given in black. The solution generated by the deterministic planner is shown in red. In addition, the performance of SARSA and CSARSA are shown. It is clear that CSARSA outperformed SARSA and eventually outperformed the baseline planner when given a sufficient number of interactions.

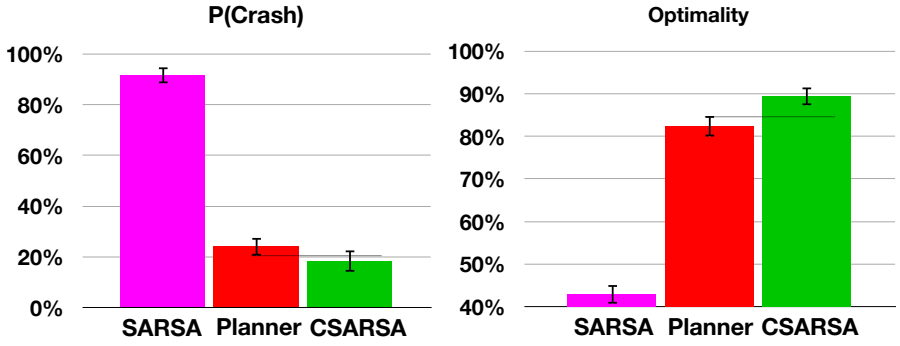


Fig. 9 Probability of crash (left) and optimality (right) of SARSA, CBBA (planner), and CSARSA algorithms at the end of the training session in the UAV 7-2 mission planning scenario. CSARSA improved the performance of both CBBA and SARSA. The optimality improvement of CSARSA was statistically significant.

UAV 7-2 Scenario In order to test our approach under harder circumstances, 5% chance of edge traverse failure was added for each UAV, resulting in a fuel burn without no movement. This noise value was not included in the approximated model, hence the safety checking mechanism could not consider it. Figure 9 shows the results of the same battery of learning algorithms used in GridWorld-1 applied to the UAV mission planning scenario at the end of learning (*i.e.,* 10^5 steps of interaction). Akin to the previous section, CBBA

was used as the base line planner. The left plot exhibits the risk of executing the corresponding policy while the right plot depicts the optimality of each solution. At the end of learning, SARSA could barely avoid crashing scenarios (about 90%), thus yielding low performance with less than 50% optimality. This observation coincides with the previous experiments with this domain where the movement model was noise free (Figure 6), highlighting the importance of biasing the policy of learners in large domains and avoiding risky behaviors. On average, CSARSA reduced the probability of failure of SARSA and CBBA by 67% and 6% correspondingly, yet the latter improvement was not statistically significant. At the same time, CSARSA raised the optimality of CBBA by 7%. This improvement was statistically significant.

8 Relaxing the Fixed Model Assumption

So far the approximated model of the MDP, $\hat{T}$, was assumed to be static during the course of interaction with the world. In this section, the cooperative learner is extended so that the approximate model can also be adapted through the course of online learning, relaxing Assumptions **I, II**. In particular the parametric form of the model is assumed to be able to capture the true model, where parameters of the model are adjusted using adaptive parameter estimation. Empirical results demonstrate that the performance of the resulting system increases. Finally the drawbacks of having an underpowered model representation are covered and an alternative solution is suggested.

8.1 Pedagogical GridWorld-2

Consider the GridWorld-2 domain shown in Figure 10(a), in which the task is to navigate from the bottom-middle ($\bullet$) to one of the top corner grid cells ($\star$), while avoiding the danger zone ($\circ$), where the agent will be eliminated upon entrance. At each step the agent can take any action from the set $\{\uparrow, \downarrow, \leftarrow, \rightarrow\}$. However, due to wind disturbances unbeknownst to the agent, there is a 20% chance the agent will be transferred into a neighboring unoccupied grid cell upon executing each action. The reward for reaching either of the goal regions and the danger zone are $+1$ and -1, respectively, while every other action results in -0.01 reward.

First consider the conservative policy shown in Figure 10(b) designed for high values of wind noise. As expected, the nominal path, highlighted as a gray watermark, follows the long but safe path to the top left goal. The color of each grid represents the true value of each state under the policy. Green indicates positive, and white indicates zero. The value of blocked grid cells are shown as red. Figure 10(c) depicts a policy designed to reach the right goal corner from every location. This policy ignores the existence of the noise, hence the nominal path in this case gets close to the danger zone. Finally Figure 10(d) shows the optimal solution. Notice how the nominal path under this policy

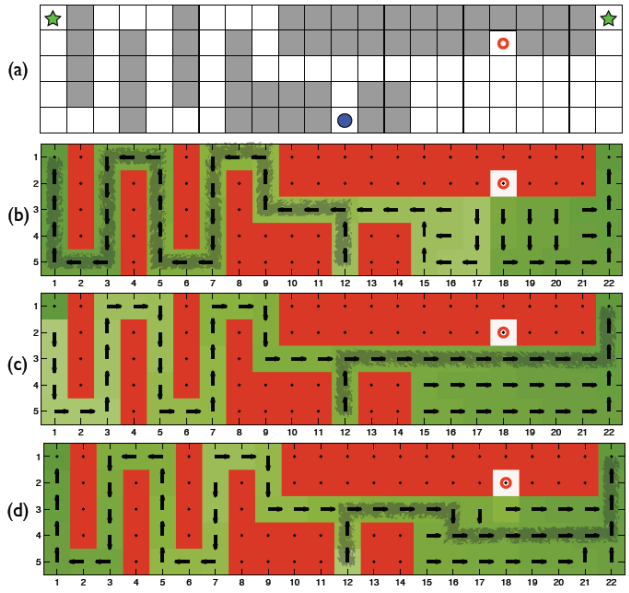


Fig. 10 The gridworld domain is shown in (a), where the task is to navigate from the bottom middle (•) to one of the top corners (⋆). The danger region (○) is an off-limit area where the agent should avoid. The corresponding policy and value function, are depicted with respect to (b) a conservative policy to reach the left corner in most states, (c) an aggressive policy which aims for the top right corner, and (d) the optimal policy.

avoids getting close to the danger zone. Section 6 demonstrated that when an approximate model (*e.g.,* the model used to generate policies in Figure 10-b,c) is integrated with a learner, it can rule out suggested actions by the learner that are poor in the eyes of the planner, resulting in safer exploration. Furthermore, the planner's policy can be used as a starting point for the learner to bootstrap on, potentially reducing the amount of data required by the learner to master the task. However, the model used for planning was static. This section introduces a new cooperative learner that allows the model box to be adapted through the learning process. The focus here is on the case where the parametric form of the approximated model ($\hat{T}$) includes the true underlying model (T) (*e.g.,* assuming an unknown uniform noise parameter for the gridworld domain). Section 8.3 discusses the drawbacks of this approach when $\hat{T}$ is unable to exactly represent T and introduces a potential alternative. Adding a parametric model to the planning and learning scheme is easily motivated by the case when the initial bootstrapped policy is wrong, or built from incorrect assumptions. In such a case, it is more effective to simply switch the underlying policy with a better one, rather than requiring a plethora of interactions to learn from and refine a poor initial policy. The remainder of this section introduces a cooperative learning process that is able to intelligently switch-out the underlying policy, refined by the learning process.

Algorithm 4: Cooperative Learning-3

Input: s, r
Output: a
1 $a \sim \pi^p(s)$
2 $\texttt{knownness} \leftarrow \min\{1, \frac{\texttt{count}(s,a)}{\mathcal{K}}\}$
3 **if** $rand() < \texttt{knownness}$ **then**
4 $\quad a' \sim \pi^l(s)$
5 $\quad$ **if** $safe(s, a')$ **then**
6 $\quad\quad a \leftarrow a'$
7 **else**
8 $\quad \texttt{count}(s, a) \leftarrow \texttt{count}(s, a) + 1$
9 $\texttt{learner.update}(s, r, a)$
10 $\texttt{model.update}(s, r, a)$ /* Update $\hat{T}$ */
11 **if** $||\hat{T}^p - \hat{T}|| > \Delta T$ **then**
12 $\quad \hat{T}^p \leftarrow \hat{T}$
13 $\quad \texttt{planner.update}()$ /* Update π^p */
14 $\quad$ **if** π^p *is changed* **then**
15 $\quad\quad$ reset all `counts` to zero
16 **return** a

Algorithm 4 depicts the new cooperative learning algorithm. Lines 1-7 are identical to Algorithm 3, while lines 8-13 highlight the new part of the algorithm which includes model adaptation. The risk mitigation process is the same as Algorithm 3. Line 10 updates the current estimate of the model, based on the observations, providing more accurate safety estimations compared to a fixed-model-based risk analyzer, as the Monte-Carlo simulations are generated using the most recent estimated model ($\hat{T}$). Furthermore, if the change to the model used for planning crosses a user-defined threshold (ΔT), the planner revisit its policy and keeps record of the new model (lines 10-12). If the policy changes, the *counts* of all state-action pairs are set to zero so that the learner start watching the new policy (mentor) from the scratch (line 13,14). An important observation is that the planner's policy should be seen safe through the eyes of the risk analyzer at all times. Otherwise, most actions suggested by the learner will be deemed too risky by mistake, as they are followed by the planner's policy. Hence, in this case, the output of the iCCA is reduced to the baseline planner's policy.

8.2 Experimental Results

This section probes the effectiveness of the adaptive modeling approach (*i.e.,* Algorithm 4), called AM-iCCA, compared to (i) the static model iCCA (*i.e.,* Algorithm 3) and (ii) the pure learning approach. The empirical settings omitted

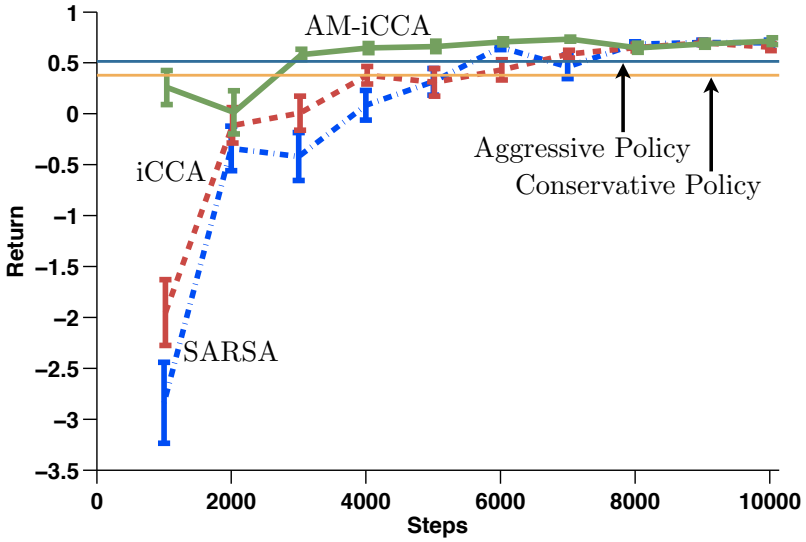


Fig. 11 Empirical results of AM-iCCA, iCCA, and SARSA algorithms in the GridWorld-2.

here were identical to those of Section 6.1.2. Five Monte-Carlo simulations were used to evaluate risk (*i.e.,* $\mathcal{M} = 5$). Each algorithm was tested for 100 trials. The risk tolerance (ε) was set to 20%. For the AM-iCCA, the noise parameter was estimated as:

$$noise = \frac{\#\text{unintended agents moves} + \text{initial weight}}{\#\text{total number of moves} + \text{initial weight}}.$$

Both iCCA methods started with the noise estimate of 40% with the count weight of 100.

Pedagogical GridWorld-2 For the iCCA algorithm, the planner followed the conservative policy (Figure 10-b). As for AM-iCCA, the planner switched from the conservative to the aggressive policy (Figure 10-c), whenever the noise estimate dropped below 25%. The knownness parameter ($\mathcal{K}$) was set to 10. Figure 11 compares the cumulative return obtained in the GridWorld-2 domain for SARSA, iCCA, and AM-iCCA based on the number of interactions. The expected performance of both static policies are shown as horizontal lines, estimated by 10,000 simulated trajectories. The improvement of iCCA with a static model over the pure learning approach is statistically significant in the beginning, while the improvement is less significant as more interactions were obtained. Although initialized with the conservative policy, the adaptive model approach within iCCA (shown as green in Figure 11) quickly learned that the actual noise in the system was much less than the initial 40% estimate and switched to using (and refining) the aggressive policy. As a result of this early discovery and switching planner's policy, AM-iCCA outperformed both iCCA

Algorithm 5: Conservative CBBA

Input: UAVs
Output: Plan

```
1  MaxFuel ← UAVs.fuel
2  UAVs.fuel ← UAVs.fuel − 3
3  ok ← False
4  while not ok or MaxFuel = UAVs.fuel do
5  |  Plan ←CBBA(UAVs)
6  |  ok ← True
7  |  for u ∈ UAVs, Plan[u] = ∅ do
8  |  |  UAVs.fuel[u] ← min(MaxFuel[u], UAVs.fuel[u] + 1)
9  |  |  ok ← False
10 return Plan
```

and SARSA, requiring two times less data compared to other learning methods to reach the asymptotic performance. For example, compare AM-iCCA's performance after 4,000 steps to other learning methods' performance after 8,000 steps. Over time, however, all methods reached the same level of performance. On that note, it is important to see that all learning methods (SARSA, iCCA, AM-iCCA) improved on the baseline static policies, highlighting their sub-optimality.

UAV 7-2 Scenario The UAV 7-2 Scenario was implemented with 5% movement noise identical to Section 7.1. The baseline cooperative planner, CBBA, was implemented in two versions: aggressive and conservative. The aggressive version used all remaining fuel cells in one iteration to plan the best set of target assignments ignoring the possible noise in the movement. Algorithm 5 illustrates the conservative CBBA algorithm. The input to the algorithm is the collection of UAVs (U). First the current fuel of UAVs are saved and decremented by 3 (lines 1-2). Then on each iteration, CBBA is called with the reduced amount of fuel cells. Consequently, the plan will be more conservative compared to the case where all fuel cells are considered. If the resulting plan allows all UAVs to get back to the base safely, it will be returned as the solution. Otherwise, UAVs with no feasible plan (*i.e.,* $Plan[u] = \emptyset$) will have their fuels incremented by one, as long as the fuel does not exceed the original fuel value (line 8). Notice that aggressive CBBA is equivalent to calling CBBA method on line 5 with the original fuel levels. Akin to the GridWorld-2 domain, the iCCA algorithm only took advantage of the conservative CBBA because the noise assumed to be fixed at 40%. As for AM-iCCA, the planner switched from the conservative to the aggressive CBBA, whenever the noise estimate dropped below 25%. The best knownness parameter ($\mathcal{K}$) was selected from $\{10, 20, 50\}$ for both iCCA and AM-iCCA.

Figures 12 shows the results of learning methods (SARSA, iCCA, and AM-iCCA) together with two variations of CBBA (conservative and aggres-

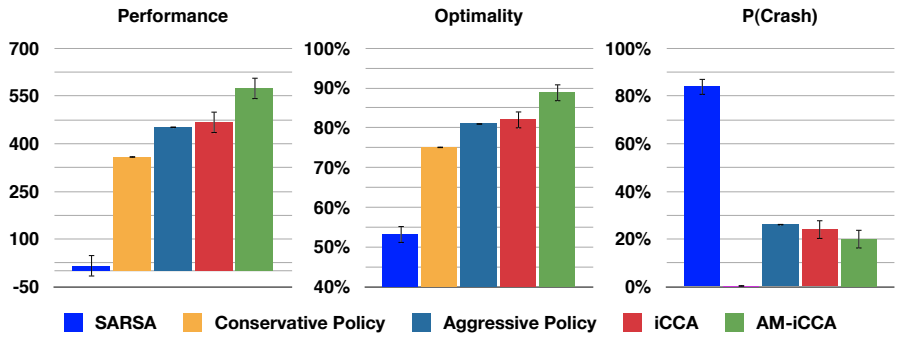


Fig. 12 Results of SARSA, CBBA-conservative, CBBA-Aggressive, iCCA and AM-iCCA algorithms at the end of the training session in the UAV mission planning scenario. AM-iCCA improved the best performance by 22% with respect to the allowed risk level of 20%.

sive) applied to the UAV mission planning scenario. Figure 12(a) represents the solution quality of each learning method after 10^5 steps of interactions. The quality of fixed CBBA methods were obtained through averaging over 10,000 simulated trajectories, where on each step of the simulation a new plan was derived to cope with the stochasticity of the environment. Figure 12(b) depicts the optimality of each solution, while Figure 12(c) exhibits the risk of executing the corresponding policy. First note that SARSA at the end of training yielded 50% optimal performance, together with more than 80% chance of crashing a UAV. Both CBBA variations outperformed SARSA. The aggressive CBBA achieved more than 80% optimality in cost of 25% crash probability, while conservative CBBA had 5% less performance, as expected, it realized a safe policy with rare chances of crashing. The iCCA algorithm improved the performance of the conservative CBBA planner again by introducing risk of crash around 20%. While on average it performed better than that aggressive policy, the difference was not statistically significant. Finally AM-iCCA outperformed all other methods statistically significantly, obtaining close to 90% optimality. AM-iCCA boosted the best performance of all other methods by 22% on average (Figure 12-a). The risk involved in running AM-iCCA was also close to 20%, matching the selected ε value.

These results highlight the importance of an adaptive model within the iCCA framework: 1) model adaptation provides a better simulator for evaluating the risk involved in taking learning actions, and 2) planners can adjust their behaviors according to the model, resulting in better policies serving as the stepping stones for the learning algorithms to build upon.

8.3 Extensions

So far, the true model was assumed to be representable within the functional form of the approximated model. But what are the consequences of using cooperative learning if this assumption does not hold? Returning to the GridWorld-

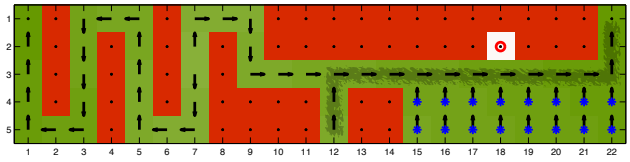


Fig. 13 The GridWorld-3 scenario: identical to GridWorld-2, yet the noise is only applied in windy grid cells (∗).

2 domain, consider the case where the 20% noise is not applied to all states. Figure 13 depicts such a scenario through GridWorld-3 where the noise is only applied to the grid cells marked with a ∗. While passing close to the danger zone is safe, when the agent assumes the uniform noise model by mistake, it generalizes the noisy movements to all states including the area close to the danger zone. This can cause the AM-iCCA to converge to a suboptimal policy, as the risk analyzer filters optimal actions suggested by the learner due to the inaccurate model assumption.

The root of this problem is that iCCA always filters the risky actions of the learner, even though the underlying assumed model might be incorrect. In order to allow the learning agent the freedom of trying potentially risky actions asymptotically, the safety check should be turned off for all states at some point. Back to the mentor/prodégé analogy, the prodégé may simply stop checking if the mentor thinks an action is safe once s/he feels comfortable taking a self-motivated action. Thus, the learner will eventually circumvent the need for a planner altogether. More specifically, line 4 of Algorithm 4 is changed, so that if the knownness of a particular state reaches a certain threshold, probing the safety of the action is not mandatory anymore. While this approach introduces another parameter to the framework, it is conjectured that the resulting process converges to the optimal policy under certain conditions. This conjecture is due to the fact that under an ergodic policy realized by the ϵ-greedy policy, all state-action pairs will be visited infinitely often. Thus at some point the knownness of all states should exceed any predefined threshold, which leads to 1) having SARSA suggest an action for every state, and 2) turning the risk filtering mechanism off for all states. As a result, the whole iCCA framework would eventually reduce to pure SARSA with an initial set of weights. Under certain conditions, it can be shown that the resulting method is convergent to the optimal policy with probability one [Melo et al., 2008]. Detailed analysis of these points is the topic of the ongoing research.

9 Contributions

This paper introduced cooperative learners formed by combining planners with RL algorithms through the intelligent Cooperative Control Architecture (iCCA). The first set of cooperative learners were built on top of RL methods with explicit policy parameterization and the underlying model was assumed

to be fixed. The second set of cooperative learners relaxed the assumption of an explicit parametric form for the RL methods, allowing the fusion of any online RL technique with cooperative planners. The third set of cooperative learners accommodated the notion of adaptive modeling within the framework. This paper also explained the drawbacks of adaptive modeling when the parametric form of the model cannot be captured in the exact model. We probed these three cooperative learners in pedagogical grid-world domains and UAV mission planning scenarios with state-action spaces up to 9 billion pairs. Empirical results demonstrated that cooperative learners boosted the performance of fixed planners and reduced the risk and sample complexity involved in the learning process.

References

Pieter Abbeel and Andrew Y. Ng. Exploration and apprenticeship learning in reinforcement learning. In *International Conference on Machine Learning (ICML)*, pages 1–8, 2005.

M. Alighanbari. Task assignment algorithms for teams of UAVs in dynamic environments. Master's thesis, Department of Aeronautics and Astronautics, Massachusetts Institute of Technology, 2004.

M. Alighanbari, Y. Kuwata, and J. P. How. Coordination and control of multiple UAVs with timing constraints and loitering. In *American Control Conference (ACC)*, volume 6, pages 5311–5316 vol.6, 4-6 June 2003. ISBN 0743-1619. URL http://acl.mit.edu/papers/FP13_41.pdf.

M. Alighanbari, L. F. Bertuccelli, and J. P. How. A Robust Approach to the UAV Task Assignment Problem. In *IEEE Conference on Decision and Control (CDC)*, pages 5935–5940, 13–15 Dec. 2006. doi: 10.1109/CDC.2006.377012.

CW Anderson, PM Young, MR Buehnerand JN Knight, KA Bush, and DC Hittle. Robust reinforcement learning control using integral quadratic constraints for recurrent neural networks. *IEEE Trans. on Neural Networks*, 18(4):993–1002, 2007.

A. Bagnell, A. Ng, and J. Schneider. Solving uncertain Markov decision processes. *Advances in Neural Information Processing Systems (NIPS)*, 2001.

R. Beard, T. McLain, M. Goodrich, and E. Anderson. Coordinated target assignment and intercept for unmanned air vehicles. *IEEE Transactions on Robotics and Automation*, 18(6):911–922, 2002.

Richard Bellman. *Dynamic Programming*. Dover Publications, March 2003. ISBN 0486428095.

A. Ben-Tal, S. Boyd, and A. Nemirovski. Extending scope of robust optimization: Comprehensive robust counterparts of uncertain problems. *Mathematical Programming*, 107(1-2):63–89, 2006.

Spring Berman, Ádám Halász, M. Ani Hsieh, and Vijay Kumar. Optimized stochastic policies for task allocationin swarms of robots. *Trans. Rob.*, 25

(4):927–937, 2009. ISSN 1552-3098. doi: http://dx.doi.org/10.1109/TRO.2009.2024997.

D. Bertsimas, D.B. Brown, and C. Caramanis. Theory and applications of robust optimization. *SIAM Review*, 53(3):464–501, 2011.

Luca F. Bertuccelli. *Robust Decision-Making with Model Uncertainty in Aerospace Systems*. PhD thesis, Massachusetts Institute of Technology, Department of Aeronautics and Astronautics, Cambridge MA, September 2008. URL http://acl.mit.edu/papers/Bertuccelli_PhD.pdf.

Luca F. Bertuccelli, Alber Wu, and Jonathan P. How. Robust adaptive markov decision processes. *IEEE Control Systems Magazine*, 2012.

Shalabh Bhatnagar, Richard S. Sutton, Mohammad Ghavamzadeh, and Mark Lee. Incremental natural actor-critic algorithms. In John C. Platt, Daphne Koller, Yoram Singer, and Sam T. Roweis, editors, *Advances in Neural Information Processing Systems (NIPS)*, pages 105–112. MIT Press, 2007.

Ronen I. Brafman and Moshe Tennenholtz. R-MAX - a general polynomial time algorithm for near-optimal reinforcement learning. *Journal of Machine Learning Research (JMLR)*, 3:213–231, 2001.

A. Casal. *Reconfiguration Planning for Modular Self-Reconfigurable Robots*. PhD thesis, Stanford University, Stanford, CA, 2002.

C. Cassandras and W. Li. A receding horizon approach for solving some cooperative control problems. In *IEEE Conference on Decision and Control (CDC)*, pages 3760–3765, 2002.

D. A. Castanon and J. M. Wohletz. Model predictive control for stochastic resource allocation. *IEEE Transactions on Automatic Control*, 54(8):1739 – 1750, 2009.

H.-L. Choi, L. Brunet, and J. P. How. Consensus-based decentralized auctions for robust task allocation. *IEEE Transactions on Robotics*, 25(4):912–926, August 2009. ISSN 1552-3098. doi: 10.1109/TRO.2009.2022423.

Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. *Introduction to Algorithms*. The MIT Press, 2nd revised edition edition, September 2001. ISBN 0262531968.

Peter Geibel and Fritz Wysotzki. Risk-sensitive reinforcement learning applied to chance constrained control. *Journal of Artificial Intelligence and Research (JAIR)*, 24, 2005.

A. Geramifard, J. Redding, J. Joseph, and J. P. How. Model Estimation Within Planning and Learning. In *Workshop on Planning and Acting with Uncertain Models, ICML, Bellevue, WA, USA*, June 2011a.

A. Geramifard, J. Redding, N. Roy, and J. P. How. UAV Cooperative Control with Stochastic Risk Models. In *American Control Conference (ACC)*, pages 3393 – 3398, June 2011b. URL http://people.csail.mit.edu/agf/Files/11ACC-iCCARisk.pdf.

Alexander Hans, Daniel Schneegaß, Anton Maximilian Schäfer, and Steffen Udluft. Safe exploration for reinforcement learning. In *Proceedings of the 16th European Symposium on Artificial Neural Networks*, 2008.

Matthias Heger. Consideration of risk and reinforcement learning. In *International Conference on Machine Learning (ICML)*, pages 105–111, 1994.

Emilio Frazzoli Ketan Savla, Tom Temple. Human-in-the-loop vehicle routing policies for dynamic environments. In *IEEE Conference on Decision and Control*, 2008.

W. Bradley Knox and Peter Stone. Combining manual feedback with subsequent MDP reward signals for reinforcement learning. In *International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, May 2010.

Michail G. Lagoudakis and Ronald Parr. Least-squares policy iteration. *Journal of Machine Learning Research (JMLR)*, 4:1107–1149, 2003.

Francisco S. Melo, Sean P. Meyn, and M. Isabel Ribeiro. An analysis of reinforcement learning with function approximation. In *International Conference on Machine Learning (ICML)*, pages 664–671, 2008.

Oliver Mihatsch and Ralph Neuneier. Risk-sensitive reinforcement learning. *Journal of Machine Learning Research (JMLR)*, 49(2-3):267–290, 2002. ISSN 0885-6125. doi: 10.1023/A:1017940631555.

A. Nilim and L. El Ghaoui. Robust solutions to Markov decision problems with uncertain transition matrices. *Operations Research*, 53(5), 2005.

Reza Olfati-Saber, J. Alex Fax, and Richard M. Murray. Consensus and cooperation in networked multi-agent systems. *IEEE Proceedings*, 95(1):215–233, January 2007.

J. Redding, A. Geramifard, H.-L. Choi, and J. P. How. Actor-Critic Policy Learning in Cooperative Planning. In *AIAA Guidance, Navigation, and Control Conference (GNC)*, August 2010a. (AIAA-2010-7586).

J. Redding, A. Geramifard, A. Undurti, H. Choi, and J. How. An intelligent cooperative control architecture. In *American Control Conference (ACC)*, pages 57–62, Baltimore, MD, July 2010b. URL `http://people.csail.mit.edu/agf/Files/10ACC-iCCA.pdf`.

Wei Ren, R. W. Beard, and E. M. Atkins. Information consensus in multivehicle cooperative control. *IEEE Control Systems Magazine*, 27(2):71–82, April 2007. ISSN 0272-1708. doi: 10.1109/MCS.2007.338264.

G. A. Rummery and M. Niranjan. Online Q-learning using connectionist systems (tech. rep. no. cued/f-infeng/tr 166). *Cambridge University Engineering Department*, 1994.

Allison Ryan, Marco Zennaro, Adam Howell, Raja Sengupta, and J. Karl Hedrick. An overview of emerging results in cooperative UAV control. In *IEEE Conference on Decision and Control (CDC)*, pages 602–607, 2004.

V. Saligrama and D.A. Castañón. Reliable distributed estimation with intermittent communications. In *IEEE Conference on Decision and Control (CDC)*, pages 6763–6768, Dec. 2006. doi: 10.1109/CDC.2006.377646.

Amarjeet Singh, Andreas Krause, and William Kaiser. Nonmyopic adaptive informative path planning for multiple robots. In *International Joint Conference on Artificial Intelligence (IJCAI)*, 2009.

Richard S. Sutton. Learning to predict by the methods of temporal differences. *Machine Learning*, 3:9–44, 1988.

Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. MIT Press, 1998.

X. Wang, V. Yadav, and S. N. Balakrishnan. Cooperative UAV formation flying with obstacle/collision avoidance. *Control Systems Technology, IEEE Transactions on*, 15(4):672–679, 2007. doi: 10.1109/TCST.2007.899191.

Xiaohua Wang, Vivek Yadav, and S.N. Balakrishnan. Cooperative UAV formation flying with stochastic obstacle avoidance. *AIAA Guidance, Navigation, and Control Conference (GNC)*, 2005.

L. Xu and U. Ozguner. Battle management for unmanned aerial vehicles. In *IEEE Conference on Decision and Control (CDC)*, volume 4, pages 3585–3590. 2003. ISBN 0191-2216.

S. Zhu and M. Fukushima. Worst-case conditional value-at-risk with application to robust portfolio management. *Operations research*, 57(5):1155–1168, 2009. ISSN 0030-364X.