Model Estimation Within Planning and Learning

Alborz Geramifard[†*], Joshua D. Redding[†], Joshua Joseph[*], Nicholas Roy[*], and Jonathan P. How[†]

***Abstract*— Risk and reward are fundamental concepts in the cooperative control of unmanned systems. In this research, we focus on developing a constructive relationship between cooperative planning and learning algorithms to mitigate the learning risk, while boosting system (planner & learner) asymptotic performance and guaranteeing the safety of agent behavior. Our framework is an instance of the intelligent cooperative control architecture (iCCA) where the learner incrementally improves on the output of a baseline planner through interaction and constrained exploration. We extend previous work by extracting the embedded parameterized transition model from within the cooperative planner and making it adaptable and accessible to all iCCA modules. We empirically demonstrate the advantage of using an adaptive model over a static model and pure learning approaches in an example GridWorld problem and a UAV mission planning scenario with 200 million possibilities. Finally we discuss two extensions to our approach to handle cases where the true model can not be captured exactly through the presumed functional form.**

I. Introduction

The concept of *risk* is common among humans, robots and software agents alike. Risk models are routinely combined with relevant observations to analyze potential actions for unnecessary risk or other unintended consequences. Risk mitigation is a particularly interesting topic in the context of the intelligent cooperative control of teams of autonomous mobile robots [1]. In such a multi-agent setting, cooperative planning algorithms rely on knowledge of underlying transition models to provide guarantees on resulting agent performance. In many situations however, these models are based on simple abstractions of the system and are lacking in representational power. Using simplified models may aid computational tractability and enable quick analysis, but at the possible cost of implicitly introducing significant risk elements into cooperative plans [2,3].

Aimed at mitigating this risk, we adopt the intelligent cooperative control architecture (iCCA) as a framework for tightly coupling cooperative planning and learning algorithms [4]. Fig. 1 shows the template iCCA framework which is comprised of a cooperative planner, a learner, and a performance analysis module. The performance analysis module is implemented as a risk-analyzer where actions suggested by the learner can be overridden by the baseline cooperative planner if they are deemed unsafe. This synergistic planner-learner relationship yields a "safe" policy in the eyes of the planner, upon which the learner can improve.

Our research has focused on developing a constructive relationship between cooperative planning and learning algorithms to reduce agent risk while boosting system (planner and learner) performance. The paper extends previous work [5] by extracting the parameterized transition model from within the cooperative planner and making it accessible to all iCCA modules. This extension enables model adaptation which facilitates better risk estimation and allows the use of multiple planning schemes. In addition, we consider two cases where the assumed functional form of the model is 1) exact and 2) inaccurate.

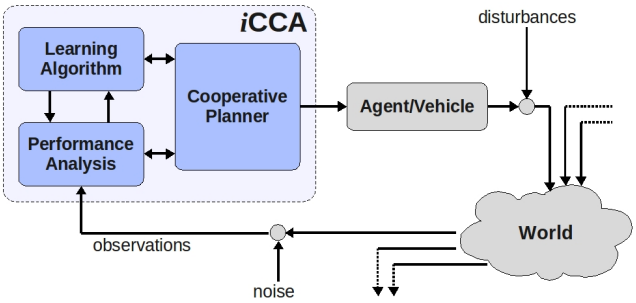

Fig. 1. The intelligent cooperative control architecture (iCCA) is a customizable template for tightly integrating planning and learning algorithms.

When the functional form is exact, we update the assumed model using an adaptive parameter estimation scheme and demonstrate that the performance of the resulting system increases. Furthermore, we explore two methods for handling the case when the assumed functional form can not represent the true model (*e.g.,* a system with state dependent noise represented through a uniform noise model). First, we enable the learner to be the sole decision maker in areas with high confidence, in which it has experienced many interactions. This extension eliminates the need for the baseline planner altogether asymptotically. Second, we use past data to estimate the reward of the learner's policy. While the policy used to obtain past data may differ from the current policy of the agent, we can still exploit the Markov assumption to piece together trajectories the learner would have experienced given the past history [6]. By using an inaccurate but simple model, we can further reduce the amount of experience required to accurately estimate the learner's reward using the method of control variates [7,8].

When the assumed functional form of the model is exact, the proposed approach of integrating an adaptive model into the planning and learning framework improved the sample complexity in a GridWorld navigation problem by a factor of two. In addition, empirical results in an example UAV mission planning domain led to more than 20% performance improvement on average compared to the best performing algorithm.

The paper proceeds as follows: Section II provides background information and Section III highlights the problem of interest by defining a pedagogical scenario where planning

† Aerospace Controls Laboratory, MIT Cambridge, MA 02139 USA
∗ Robust Robotics Group, MIT Cambridge, MA 02139 USA
{agf,jredding,jmjoseph,nickroy,jhow}@mit.edu

and learning algorithms are used to mitigate stochastic risk. Section IV outlines the proposed technical approach for learning to mitigate risk. Section V empirically verifies the effectiveness of our new approach. Section VI highlights two extensions to our approach when the true model can not be captured exactly in the parametric form assumed for the model. Section VII concludes the paper.

II. Background

A. Markov Decision Processes (MDPs)

MDPs provide a general formulation for sequential planning under uncertainty [9]. MDPs are a natural framework for solving multi-agent planning problems as their versatility allows modeling of stochastic system dynamics as well as inter-dependencies between agents. An MDP is defined by tuple $(\mathcal{S}, \mathcal{A}, \mathcal{P}^a_{ss'}, \mathcal{R}^a_{ss'}, \gamma)$, where $\mathcal{S}$ is the set of states, $\mathcal{A}$ is the set of possible actions. Taking action a from state s has $\mathcal{P}^a_{ss'}$ probability of ending up in state s' and receiving reward $\mathcal{R}^a_{ss'}$. Finally $\gamma \in [0, 1]$ is the discount factor used to prioritize early rewards against future rewards.[1] A trajectory of experience is defined by sequence $s_0, a_0, r_0, s_1, a_1, r_1, \cdots$, where the agent starts at state s_0, takes action a_0, receives reward r_0, transits to state s_1, and so on. A policy π is defined as a function from $\mathcal{S} \times \mathcal{A}$ to the probability space $[0, 1]$, where $\pi(s, a)$ corresponds to the probability of taking action a from state s. The value of each state-action pair under policy π, $Q^\pi(s, a)$, is defined as the expected sum of discounted rewards when the agent takes action a from state s and follows policy π thereafter:

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

The optimal policy π^* maximizes the above expectation for all state-action pairs: $\pi^* = \text{argmax}_a \, Q^{\pi^*}(s, a)$.

B. Reinforcement Learning in MDPs

The underlying goal of the reinforcement learning (RL) algorithms herein is to improve performance of the cooperative planner over time using observed rewards by exploring new agent behaviors that may lead to more favorable outcomes. In this paper, we focus on one of the simplest RL techniques, known as SARSA (state action reward state action) [10].

SARSA is a popular approach among MDP solvers that finds an approximation to $Q^\pi(s, a)$ (policy evaluation) and updates the policy with respect to the resulting values (policy improvement). Temporal Difference learning (TD) [11] 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:

$$Q^\pi(s_t, a_t) = Q^\pi(s_t, a_t) + \alpha \delta_t(Q), \tag{1}$$

where α is the learning rate. SARSA is essentially TD learning for control, where the policy is directly derived from the Q values as:

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

where ϵ is the probability of taking a random action. This policy is also known as the ϵ-greedy policy.[2]

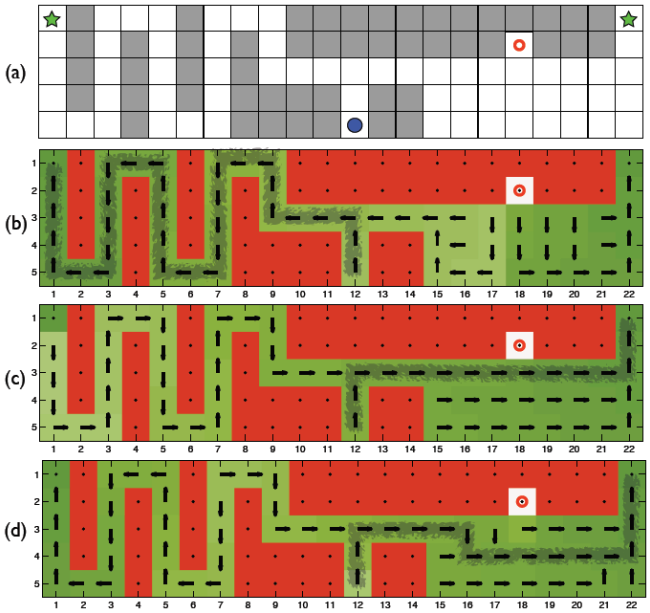

Fig. 2. The GridWorld domain is shown in (a), where the task is to navigate from the bottom middle (●) to one of the top corners (⋆). The the danger region (○) is an off-limit area where the agent should avoid. The corresponding policy and value function, are depicted with respect to (b) a conservative 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.

III. GridWorld Domain: A Pedagogical Example

Consider the GridWorld domain shown in Fig. 2-a, in which the task is to navigate from the bottom-middle (●) to one of the top corner gridcells (⋆), while avoiding the danger zone (○), 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 Fig. 2-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

[1] can set $\gamma = 1$ for episodic tasks, where trajectories are finite length.

[2] Ties are broken randomly, if more than one action maximizes $Q(s, a)$.

of blocked gridcells are shown as red.

Fig. 2-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 Fig. 2-d shows the optimal solution. Notice how the nominal path under this policy avoids getting close to the danger zone. Model-free learning techniques such as SARSA can find the optimal policy of the noisy environment through interaction, but require a great deal of training examples. More critically, they may deliberately move the agent towards dangerous regions in order to gain information about those areas. Previously, we demonstrated that when a planner (*e.g.,* methods to generate policies in Fig. 2-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 [4,5]. In our past work, we considered the case where the model used for planning and risk analysis were static. In this paper, we expand our framework by representing the model as a separate entity which can be adapted through the learning process. The focus here is on the case where the parametric form of the approximated model includes the true underlying model (T) (*e.g.,* assuming an unknown uniform noise parameter for the GridWorld domain). In Section VI, we discuss drawbacks of our approach when the approximation model class is unable to exactly represent T and introduce two potential extensions.

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 paper shows that by representing the model as a separate entity that can be adapted through the learning process, we enable the ability to intelligently switch-out the underlying policy, which is refined by the learning process.

IV. Technical Approach

This section first discusses the new architecture used in this research, which is shown in Fig. 3. Note the addition of the "Models" module in the iCCA framework as implemented when compared to the template architecture of Fig. 1. This module enables the agent's transition model to be adapted in light of actual transitions experienced. An estimated model, $\hat{T}$, is output and is used to sample successive states when simulating trajectories. As mentioned above, this model is assumed to be of the exact functional form (*e.g.,* a single uncertain parameter). Additionally, Fig. 1 shows a dashed boxed outlining the learner and the risk-analysis modules, which are formulated together within an MDP to enable the use of reinforcement learning algorithms in the learning module.

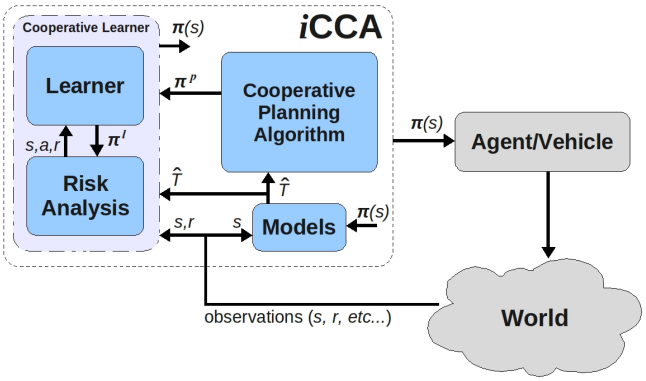

Fig. 3. The intelligent cooperative control architecture as implemented. The conventional reinforcement learning method (*e.g.,* SARSA) sits in the learner box while the performance analysis block is implemented as a risk analysis tool. Together, the learner and the risk-analysis modules are formulated within a Markov decision process (MDP).

SARSA is implemented as the system learner, which uses past experiences to guide exploration and then suggests behaviors that have potential to boost the performance of the baseline planner. The performance analysis block is implemented as a risk analysis tool where actions suggested by the learner can be rejected if they are deemed too risky. The following sections describe the iCCA blocks in further detail.

A. Cooperative Planner

At its fundamental level, the cooperative planning algorithm used in iCCA yields a solution to the multi-agent path planning, task assignment or resource allocation problem, depending on the domain that seeks to optimize an underlying, user-defined *objective function*. Many existing cooperative control algorithms use observed performance to calculate *temporal-difference errors* which drive the objective function in the desired direction [3,12]. Regardless of how it is formulated (*e.g.,* MILP or MDP), the cooperative planner, or cooperative control algorithm, is the source for baseline plan generation within iCCA. We assume that given a model ($\hat{T}$), this module can provide a safe solution (π^p) in a reasonable amount of time.

B. Learning and Risk-Analysis

As discussed earlier, learning algorithms may encourage the agent to explore dangerous situations (*e.g.,* flying close to the danger zone) in the hope of improving the long-term performance. While some degree of exploration is necessary, unbounded exploration can lead to undesirable scenarios such as crashing or losing a UAV. To avoid such undesirable outcomes, we implemented the iCCA performance analysis module as a risk analysis element where candidate actions are evaluated for safety against an adaptive estimated transition model $\hat{T}$. Actions deemed too "risky" are replaced with the safe action suggested by the cooperative planner. The risk-analysis and learning modules are coupled within an MDP formulation, as shown by the dashed box in Fig. 3. We now discuss the detail of the learning and risk-analysis algorithms.

Previous research employed a risk analysis scheme that used the planner's transition model, which can be stochastic, to mitigate risk [5]. In this research, we pull this embedded

Algorithm 1: safe

Input: $s, a, \hat{T}$
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)
isSafe ← (risk < ψ)
```

model from within the planner and allow it to be updated online. This allows both the planner and the risk-analysis module to benefit from model updates. Algorithm 1 explains the risk analysis process where we assume the existence of the function $constrained$: $\mathcal{S} \rightarrow \{0, 1\}$, which indicates if being in a particular state is allowed or not. We define risk as the probability of visiting any of the constrained states. The core idea is to use Monte-Carlo sampling to estimate the risk level associated with the given state-action pair if the planner's policy is applied thereafter. This is done by simulating 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 policy, π^p. The adaptive approximate model, $\hat{T}$, is utilized to sample successive states. Each trajectory is bounded to a fixed horizon H and the risk of taking action a from state s is estimated by the probability of a simulated trajectory reaching a risky (*e.g.,* constrained) state within horizon H. If this risk is below a given threshold, ψ, the action is deemed to be safe.

Note that the cooperative planner has to take advantage of the updated transition model and replan adaptively. This ensures that the risk analysis module is not overriding actions deemed risky by an updated model with actions deemed "safe" by an outdated model. Such behavior would result in convergence of the cooperative learning algorithm to the baseline planner policy and the system would not benefit from the iCCA framework.

For learning schemes that do not represent the policy as a separate entity, such as SARSA, integration within iCCA framework is not immediately obvious. Previously, we presented an approach for integrating learning approaches without an explicit component for the policy [4]. Our idea was motivated by the concept of the R_{max} algorithm [13]. We illustrate our approach through the parent-child analogy, where the planner takes the role of the parent and the learner takes the role of the child. In the beginning, the child does not know much about the world, hence, for the most part they take actions advised by the parent. While learning from such actions, after a while, the child feels comfortable about taking a self-motivated actions as they have been through the same situation many times. Seeking permission from the parent, the child could take the action if the parent thinks the

Algorithm 2: Cooperative Learning

Input: $N, \pi^p, s, learner$
Output: a

```
1  a ← π^p(s)
2  π^l ← learner.π
3  knownness ← min{1, count(s,a)/N}
4  if rand() < knownness then
5      a^l ~ π^l(s, a)
6      if safe(s, a^l, T̂) then
7          a ← a^l
8  else
9      count(s, a) ← count(s, a) + 1
10 Take action a and observe r, s'
11 learner.update(s, a, r, s')
12 T̂ ← NewModelEstimation(s, a, s')
13 if ||T̂^p − T̂|| > ξ then
14     T̂^p ← T̂
15     π^p ← Planner.replan()
16     if π^p is changed then
17         reset all counts to zero
```

[ACC 2011]

Adaptive Model

action is safe. Otherwise the child should follow the action suggested by the parent.

Our approach for safe, cooperative learning is shown in Algorithm 2. The red section highlights our previous cooperative method [5], while the green region depicts the new version of the algorithm which includes model adaptation. 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 (line 3). The knownness parameter controls the transition speed from following the planner's policy to the learner's policy. Given the knownness of the state-action pair, the learner probabilistically decides to select an action from its own policy (line 4). If the action is deemed to be safe, it is executed. Otherwise, the planner's policy overrides the learner's choice (lines 5-7). If the planner's action is selected, the knownness count of the corresponding state-action pair is incremented (line 9). Finally the learner updates its parameter depending on the choice of the learning algorithm (line 11).

A drawback of the red part of Algorithm 2 (*i.e.,* our previous work) is that state-action pairs explicitly forbidden by the risk analyzer will not be visited. Hence, if the model is designed poorly, it can hinder the learning process in parts of the state space for which the risk is overestimated. Furthermore, the planner can take advantage of the adaptive model and revisits its policy. Hence we extended the previous algorithm (the red section) to enable the model to be adapted during the learning phase (line 12). Furthermore, if the change to the model used for planning crosses a predefined threshold (ξ), the planner revisit its policy and keeps record of the new model (lines 13-15). If the policy changes, the $counts$ of all state-action pairs are set to zero so that the

learner start watching the new policy from scratch (lines 16, 17). An important observation is that the planner's policy should be seen as 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.

V. Numerical Experiments

We compared the effectiveness of the adaptive model approach (Algorithm 2), which we refer to as AM-iCCA with respect to (i) our previous work with a static model —the red section of Algorithm 2— (iCCA), (ii) the pure learning approach, and (iii) pure fixed planners. All algorithms used SARSA for learning with the following form of learning rate:

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

For each algorithm, we used the best α_0 from $\{0.01, 0.1, 1\}$ and N_0 from $\{100, 1000, 10^6\}$. During exploration, we used an ϵ-greedy policy with $\epsilon = 0.1$. Value functions were represented using lookup tables. Both iCCA methods started with the noise estimate of 40% with the count weight of 100, and a conservative policy. We used 5 Monte-Carlo simulations to evaluate risk (*i.e.,* $M = 5$). Each algorithm was tested for 100 trials. Error bars represent 95% confidence intervals. We allowed 20% risk during the execution of iCCA (*i.e.,* $\psi = 0.2$). As for plan adaptation, each planner executed a conservative policy for noise estimates above 25% and an aggressive policy for noise estimates below 25%. This adaptation was followed without any lag (*i.e.,* $\xi = 0$). For the AM-iCCA, the noise parameter was estimated as:

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

The noise parameter converged to the real value in all domains.

A. The GridWorld Domain

For the iCCA algorithm, the planner followed the conservative policy (Fig. 2-b). As for AM-iCCA, the planner switched from the conservative to the aggressive policy (Fig. 2-c), whenever the noise estimate dropped below 25%. The knownness parameter (N) was set to 10.

Fig. 4 compares the cumulative return obtained in the GridWorld 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 (shown as green in Figure 4), the AM-iCCA approach 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 and SARSA, requiring one half the data compared to other

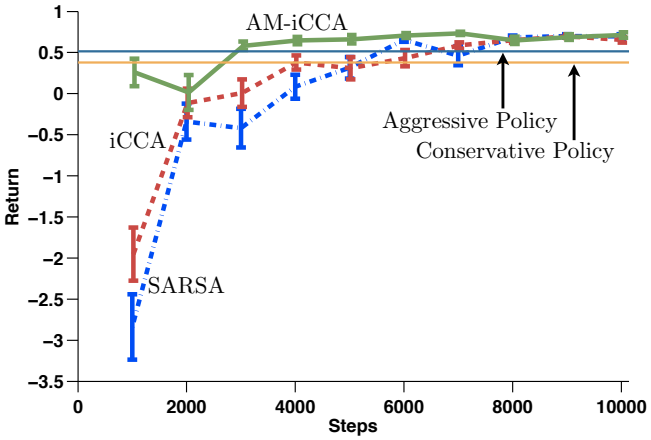

Fig. 4. Empirical results of AM-iCCA, iCCA, and SARSA algorithms in the GridWorld problem.

learning methods to reach the asymptotic performance.[3] Over time, however, all learning methods (*i.e.,* SARSA, iCCA, and AM-iCCA) reached the same level of performance that improved the performance of static policies, highlighting their sub-optimality.

B. The Multi-UAV Planning Scenario

Fig. 5-a depicts a mission planning scenario, where a team of two fuel-limited UAVs cooperate to maximize their total reward by visiting valuable target nodes in the network and return back to the base (green circle). Details of the domain is available in our previous work [4,5]. The probability of a UAV remaining in the same node upon trying to traverse an edge (*i.e.,* the true noise parameter) was set to 5%. The size of the possible state-action pairs exceeds 200 million.

As for the baseline cooperative planner, CBBA [3] 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 3 illustrates the conservative CBBA algorithm that adopts a pessimistic approach for planning. The input to the algorithm is the collection of UAVs (U) and the connectivity graph (G). First the current fuel of UAVs are saved and decremented by the dimeter of the connectivity graph (lines 1-2). This value is 3 for the mission planning scenario shown in Fig. 5-a. 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, 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 domain, the iCCA algorithm only took advantage of the conservative CBBA because the noise assumed to be 40%. In AM-iCCA, the planner switched between the conservative and the aggressive

[3]Compare AM-iCCA's performance after $4,000$ steps to other learning methods' performance after $8,000$ steps.

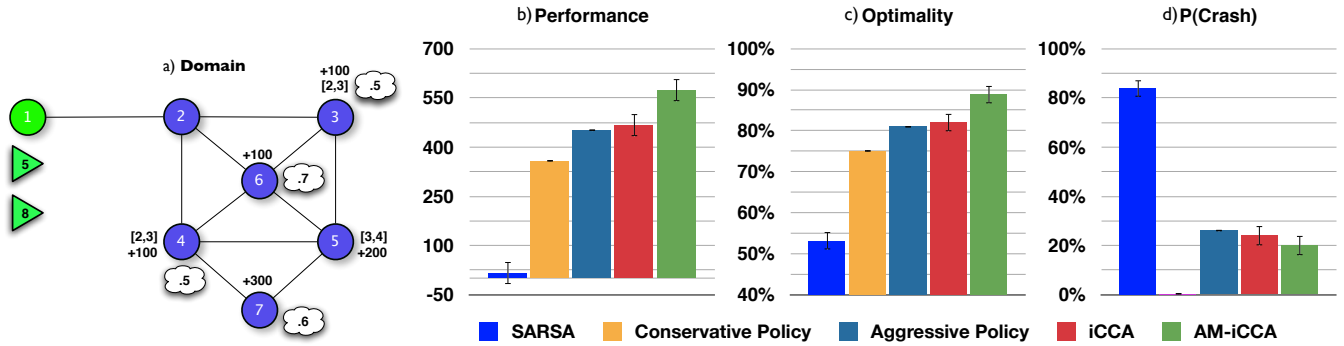

Fig. 5. (a) Mission scenarios of interest: A team of two UAVs plan to maximize their cumulative reward along the mission by cooperating to visit targets. Target nodes are shown as circles with rewards noted as positive values and the probability of receiving the reward shown in the accompanying cloud. Note that some target nodes have no value. Constraints on the allowable visit time of a target are shown in square brackets. (b,c,d) 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%.

Algorithm 3: Conservative CBBA

Input: U, G
Output: $Plan$

```
MaxFuel ← U.fuel
U.fuel ← U.fuel − G.diameter
ok ← False
while not ok or MaxFuel = U.fuel do
    Plan ← CBBA(U, G)
    ok ← True
    for u ∈ U, Plan[u] = ∅ do
        U.fuel[u] ← min(MaxFuel[u], U.fuel[u] + 1)
        ok ← False
return Plan
```

CBBA depending on the noise estimate. The best knowness parameter (N) was selected from $\{10, 20, 50\}$ for both iCCA and AM-iCCA.

Figures 5-b to 5-d show the results of learning methods (SARSA, iCCA, and AM-iCCA) together with two variations of CBBA (conservative and aggressive) applied to the UAV mission planning scenario. Fig. 5-b represents the solution quality of each learning method after 10^5 steps of interactions. The quality of planners 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. Fig. 5-c depicts the optimality of each solution, while Fig. 5-d 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 and, as expected, it led to 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 (Fig. 5-b). The risk involved in running AM-iCCA was also close to 20%, matching our selected ψ value.

These result highlights 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.

VI. Extensions

So far, we assumed that the true model can be represented accurately within functional form of the approximated model. In this section, we discuss the challenges involved in using our proposed methods when this condition does not hold and suggest two extensions to overcome such challenges. Returning to the GridWorld domain, consider the case where the 20% noise is not applied to all states. Fig. 6 depicts such a scenario 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 the risk analyzer has the final authority in selecting the actions from the learner and the planner, hence both of our extensions focus on revoking this authority. The first extension eliminates the need to analyze the risk after some time. Back to our parent/child analogy, the child may simply stop checking if the parent thinks an action is safe once they feel comfortable taking a self-motivated action. Thus, the learner will eventually circumvent the need for a planner altogether. More specifically, line 6 of Algorithm 2 is changed so that if the knowness

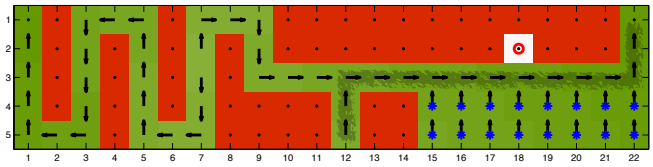

Fig. 6. The solution to the GridWorld scenario where the noise is only applied in windy grid cells ($*$).

of a particular state reaches a certain threshold, probing the safety of the action is not mandatory anymore (*i.e.,* new line 6: **if** $knownness > \eta$ **or** $safe(s, a^l, \hat{T})$). While this approach introduces another parameter to the framework, we conjecture that the resulting process converges to the optimal policy under certain conditions. This 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. Hence at some point the knownness of all states exceed any predefined threshold. This leads to 1) having SARSA suggest an action for every state, and 2) turning the risk analyzer off for all states. This means the whole iCCA framework is reduced 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 [14].

An additional approach to coping with the risk analyzer's inaccurate model is to estimate the reward of the learner's policy from previous experience. This can be achieved by standard on-policy importance sampling [9] but requires an impractical amount of data to accurately estimate the reward of the learner's policy. By taking the approach of [15], we hope to reduce the sample complexity of this estimate by using a combination of two methods. The first, control variates [7], allow us to use the risk analyzer's approximate model to reduce the variance of the estimate in states that have sparse data. The second, based on [6], leverages the Markov assumption to stitch together episodes of data from previous experience that the learner's policy would have taken. We conjecture that this approach increases the effective number of episodes on-policy importance sampling is performed with, leading to a more accurate estimate.

VII. Conclusions

This paper extended our previous iCCA framework by representing the model as a separate entity which can be shared by the planner and the risk analyzer. Furthermore, when the true functional form of the the transition model is known, we discussed how the new method can facilitate a safer exploration scheme through a more accurate risk analysis. Empirical results in a GridWorld domain and a UAV mission planning scenario verified the potential and scalability of the new approach in reducing the sample complexity and improving the asymptotic performance compared to our previous algorithm [5] and pure learning/planning techniques. Finally we argued through an example that model adaptation can hurt the asymptotic performance, if the true model can not be captured accurately. For this case, we provided two extensions to our method in order to mitigate the problem, which form the main thrust of our future work.

Acknowledgments

This work was sponsored by the AFOSR and USAF under grant FA9550-09-1-0522 and by NSERC. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of the Air Force Office of Scientific Research or the U.S. Government.

References

[1] R. Weibel and R. Hansman, "An integrated approach to evaluating risk mitigation measures for UAV operational concepts in the NAS," in *AIAA Infotech@Aerospace Conference*, 2005, pp. AIAA–2005–6957.

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

[3] H.-L. Choi, L. Brunet, and J. P. How, "Consensus-based decentralized auctions for robust task allocation," *IEEE Transactions on Robotics*, vol. 25, no. 4, pp. 912–926, August 2009. [Online]. Available: http://acl.mit.edu/papers/email.html

[4] 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 2010, (AIAA-2010-7586).

[5] A. Geramifard, J. Redding, N. Roy, and J. P. How, "UAV Cooperative Control with Stochastic Risk Models," in *American Control Conference (ACC)*, June 2011, pp. 3393 – 3398. [Online]. Available: http://people.csail.mit.edu/agf/Files/11ACC-iCCARisk.pdf

[6] M. Bowling, M. Johanson, N. Burch, and D. Szafron, "Strategy evaluation in extensive games with importance sampling," *Proceedings of the 25th Annual International Conference on Machine Learning (ICML)*, 2008.

[7] M. Zinkevich, M. Bowling, N. Bard, M. Kan, and D. Billings, "Optimal unbiased estimators for evaluating agent performance," in *Proceedings of the 21st national conference on Artificial intelligence - Volume 1*. AAAI Press, 2006, pp. 573–578.

[8] M. White, "A general framework for reducing variance in agent evaluation," 2009.

[9] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*. MIT Press, 1998.

[10] 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.

[11] R. S. Sutton, "Learning to predict by the methods of temporal differences," *Machine Learning*, vol. 3, pp. 9–44, 1988.

[12] D. P. Bertsekas and J. N. Tsitsiklis, *Neuro-Dynamic Programming (Optimization and Neural Computation Series, 3)*. Athena Scientific, May 1996.

[13] R. I. Brafman and M. Tennenholtz, "R-MAX - a general polynomial time algorithm for near-optimal reinforcement learning," *Journal of Machine Learning Research (JMLR)*, vol. 3, pp. 213–231, 2001.

[14] F. S. Melo, S. P. Meyn, and M. I. Ribeiro, "An analysis of reinforcement learning with function approximation," in *International Conference on Machine Learning (ICML)*, 2008, pp. 664–671.

[15] J. Joseph and N. Roy, "Reduced-order models for data-limited reinforcement learning," in *ICML 2011 Workshop on Planning and Acting with Uncertain Models*, 2011.