Model Estimation Within Planning and Learning

Alborz Geramifard[†] AGF@MIT.EDU
Joshua D. Redding[†] JREDDING@MIT.EDU
Joshua Joseph[*] JMJOSEPH@MIT.EDU
Jonathan P. How[†] JHOW@MIT.EDU
†Aerospace Controls Laboratory, MIT Cambridge, MA 02139 USA
∗Robust Robotics Group, MIT Cambridge, MA 02139 USA

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 a grid world problem. Finally we discuss two extensions to our approach to handle cases where the true model can not be captured through the presumed functional form.

1. Introduction

The concept of *risk* is common among humans, robots and software agents alike. Amongst the latter two, 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 (Kim, 2003; Weibel & Hansman, 2005). 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.

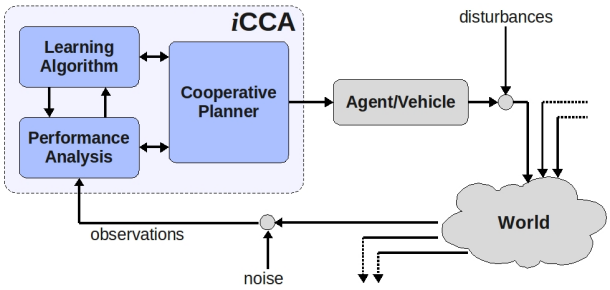

Figure 1. The intelligent cooperative control architecture (*iCCA*) is a customizable template for tightly integrating planning and learning algorithms.

Aimed at mitigating this risk, we adopt the intelligent cooperative control architecture (*iCCA*) as a framework for tightly coupling cooperative planning and learning algorithms (Redding et al., 2010). 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.

This research focused on developing a constructive relationship between cooperative planning and learning algorithms to reduce agent risk while boosting system (planner

Preliminary work in *Planning and Acting with Uncertain Models Workshop, ICML,* Bellevue, WA, USA, 2011.

+ learner) performance. We extend previous work (Geramifard et al., 2011) by extracting the parameterized transition model from within the cooperative planner and making it accessible to all *iCCA* modules. In addition, we consider the cases where the assumed functional form of the model is both correct and incorrect.

When the functional form is correct, 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 (Bowling et al., 2008). Additionally, we use the incorrect functional form of the model to further reduce the amount of experience required to accurately estimate the learner's reward using the method of control variates (Zinkevich et al., 2006; White, 2009).

The proposed approach of integrating an adaptive model into the planning & learning framework is shown to yield significant improvements in sample complexity in the grid world domain, when the assumed functional form of the model is correct. When incorrect, we highlight the drawbacks of our approach and discuss two extensions which can mitigate such drawbacks. The paper proceeds as follows: Section 2 provides background information and Section 3 highlights the problem of interest by defining a pedagogical scenario where planning and learning algorithms are used to mitigate stochastic risk. Section 4 outlines the proposed technical approach for learning to mitigate risk. Section 6 highlights two extensions to our approach when the true model can not be captured in the parametric form assumed for the model. Section 7 concludes the paper.

2. Background

2.1. Markov Decision Processes

Markov decision processes (MDPs) provide a general formulation for sequential planning under uncertainty (Howard, 1960; Puterman, 1994; Littman et al., 1995; Kaelbling et al., 1998; Russell & Norvig, 2003). 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 follow 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: $\pi^* = \text{argmax}_a \, Q^{\pi^*}(s, a)$.

2.2. Reinforcement Learning in MDPs

The underlying goal of the two reinforcement learning 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.

2.2.1. 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:

$$Q^\pi(s_t, a_t) = Q^\pi(s_t, a_t) + \alpha \delta_t(Q), \quad (1)$$

[1] γ can be set to 1 only for episodic tasks, where the length of trajectories are fixed.

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

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

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

3. The 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 grids (⋆), 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.

Let's 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 of blocked grids 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 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 just 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 intentionally poor decisions, 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 (Redding et al., 2010; Geramifard et al., 2011). 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 ($\hat{T}$) includes the true underlying model (T) (*e.g.,* assuming an unknown uniform noise parameter for the gridworld domain). In Section 6, we discuss drawbacks of our approach when $\hat{T}$ is unable to exactly represent T and introduce two potential extensions.

Adding a parametric model to the planning/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 which 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.

4. Technical Approach

In this section, we introduce the architecture in which this research was carried out. First, notice the addition of the "Models" module in the *iCCA* framework as implemented when compared to the template architecture of Figure 1. This module allows us to adapt the agent's transition model 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 correct functional form (*e.g.,* a single uncertain parameter). Additionally, Figure 1 shows a dashed boxed outlining the learner and the risk-analysis modules, which are formulated together within a Markov decision process (MDP) to enable the use of reinforcement learning algorithms in the learning module.

The Sarsa (Rummery & Niranjan, 1994) algorithms is implemented as the system learner which uses past experiences to guide exploration and then suggests behaviors that are likely to lead to more favorable outcomes than those suggested by 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 iCCA blocks in further detail.

4.1. 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. This means that it seeks

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

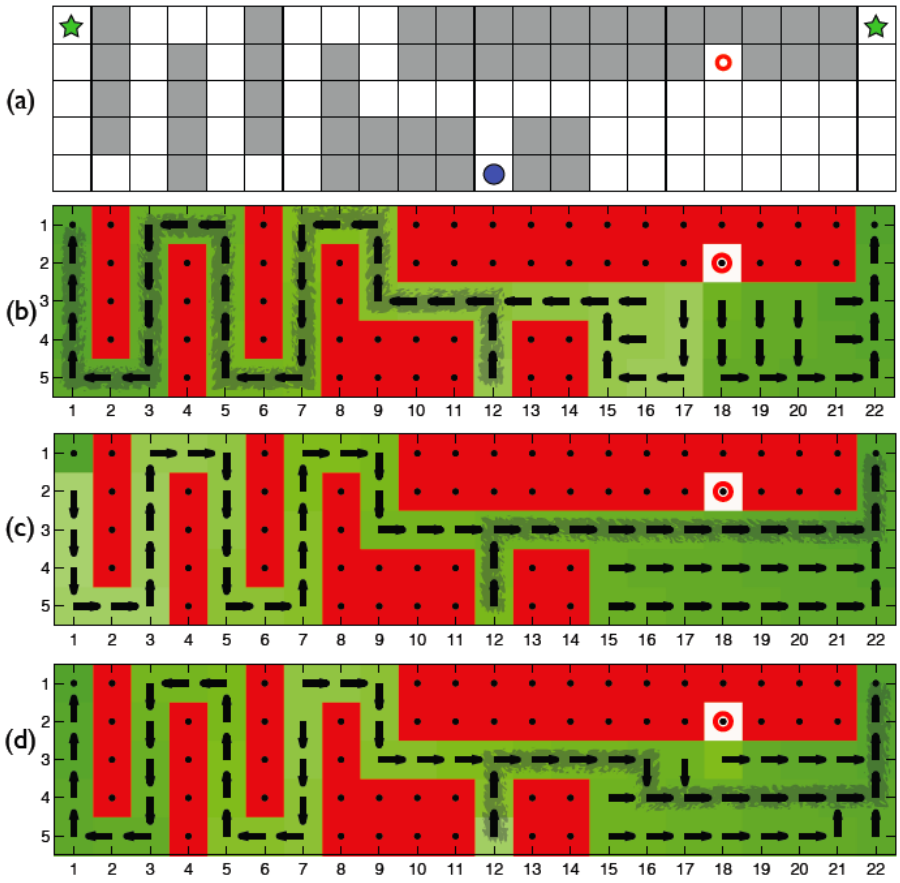

Figure 2. The gridworld domain is shown in (a), where the task is to navigate from the bottom middle (•) to one of the top corners (⋆). The the danger region (○) is an off-limit area where the agent should avoid. The corresponding policy and value function, are depicted with respect to (b) a conservative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

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 (Murphey & Pardalos, 2002; Bertsekas & Tsitsiklis, 1996). Regardless of how it is formulated (e.g. MILP, MDP, CBBA), the cooperative planner, or cooperative control algorithm, is the source for baseline plan generation within *iCCA*. We assume that this module can provide safe solutions to the problem in reasonable amount of time.

4.2. 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 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. As 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 details 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 (Geramifard et al., 2011). In this research, we pull this embedded 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

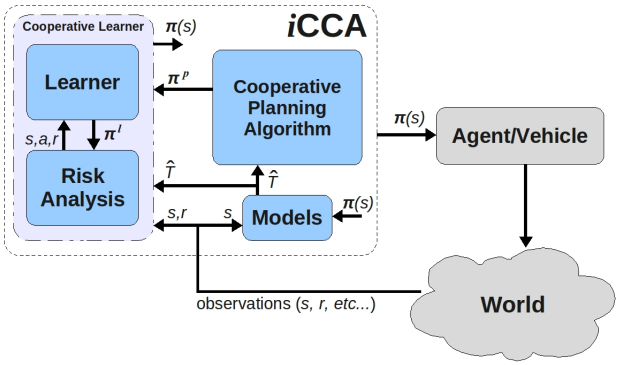

Figure 3. The intelligent cooperative control architecture as implemented. The conventional reinforcement learning method (*e.g.,* Sarsa, Natural Actor Critic) sits in the learner box while the performance analysis block is implemented as a risk analysis tool. Together, the learner and the risk-analysis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

Algorithm 1: safe

Input: $s, a, \hat{T}$
Output: $isSafe$
1 $risk \leftarrow 0$
2 **for** $i \leftarrow 1$ **to** M **do**
3 $\quad t \leftarrow 1$
4 $\quad s_t \sim \hat{T}(s, a)$
5 $\quad$ **while not** $constrained(s_t)$ **and not** $isTerminal(s_t)$ **and** $t < H$ **do**
6 $\quad\quad s_{t+1} \sim \hat{T}(s_t, \pi^p(s_t))$
7 $\quad\quad t \leftarrow t + 1$
8 $\quad risk \leftarrow risk + \frac{1}{i}(constrained(s_t) - risk)$
9 $isSafe \leftarrow (risk < \psi)$

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.

It is important to note that, though not explicitly required, the cooperative planner should take advantage of the updated transition model by replanning. 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 actor component (Redding et al., 2010). Our idea was motivated by the concept of the R_{max} algorithm (Brafman & Tennenholtz, 2001). 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 s/he takes actions advised by the parent. While learning from such actions, after a while, the child feels comfortable about taking a self-motivated actions as s/he has been through the same situation many times. Seeking permission from the parent, the child could take the action if the parent thinks the 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 cyan section highlights our previous cooperative method (Geramifard et al., 2011), 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 shift 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 Algorithm 2 is that state-action pairs explicitly forbidden by 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 in order to let the model 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 (line 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

Algorithm 2: Cooperative Learning

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

$a \leftarrow \pi^p(s)$
$\pi^l \leftarrow learner.\pi$
$knownness \leftarrow \min\{1, \frac{count(s,a)}{N}\}$
if $rand() < knownness$ **then**
  $a^l \sim \pi^l(s,a)$
  if *safe*$(s, a^l, \hat{T})$ **then**
    $a \leftarrow a^l$
else
  $count(s,a) \leftarrow count(s,a) + 1$
Take action a and observe r, s'
$learner.update(s, a, r, s')$
$\hat{T} \leftarrow NewModelEstimation(s, a, s')$
if $||\hat{T}^p - \hat{T}|| > \xi$ **then**
  $\hat{T}^p \leftarrow \hat{T}$
  $\pi^p \leftarrow Planner.replan()$
  if π^p *is changed* **then**
    reset all *counts* to zero

[ACC 2011]

Adaptive Model

times. Otherwise, most actions suggested by the learner will be deemed too risky by mistake, as they are followed by the planner's policy.

5. Experimental Results

We compared the effectiveness of the adaptive model approach combined with iCCA framework (AM-iCCA) with respect to (i) our previous work with a static model (iCCA) and (ii) the pure learning approach. 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 the conservative policy (Fig. 2-c). We used 5 Monte-Carlo simulations to evaluate risk and rejected actions for which any of the trajectories entered the danger zone. The knowness parameter (N) was set to 10. 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 planner switched to the aggressive policy (Fig. 2-b) whenever the wind disturbance estimate reached below 25%. Each algorithm was tested for 100 trials. Error bars represent 95% confidence intervals.

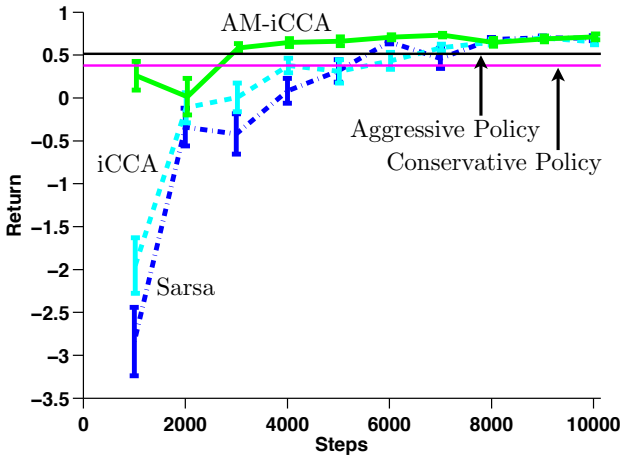

Figure 4. Empirical results of AM-iCCA, iCCA, and Sarsa algorithms in the grid world problem.

Fig. 4 compares the cumulative return obtained in the grid world 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. 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 in green in Figure 4) 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 switch, AM-iCCA outperformed both iCCA and Sarsa. 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.

6. Extensions

So far, we assumed that the true model can be represented accurately within functional form of the approximated model. In this section, we are going to 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 grid world domain, consider the case where the 20% noise is not applied to all states. Fig. 5 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

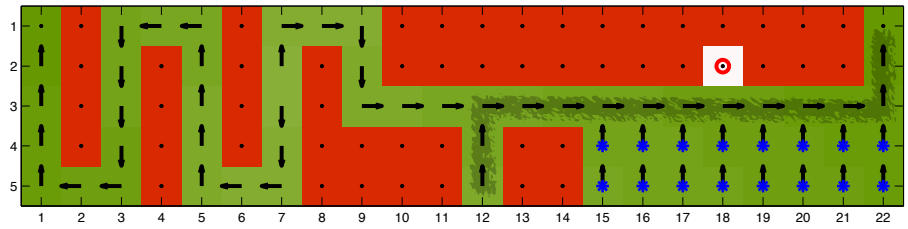

Figure 5. The solution to the grid world scenario where the noise is only applied in windy grid cells (∗).

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 incorrect 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 turns the risk analyzer mandatory only to a limited degree. Back to our parent/child analogy, the child may simply stop checking if the parent thinks an action is safe once s/he feels comfortable taking a self-motivated action. Thus, the learner would eventually circumvent the need for a planner altogether. More specifically, line 6 of Algorithm 2 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 would introduce another parameter to the framework, it guarantees that in the limit, the learner would be the final decision maker. Hence the resulting method will converge to the same solution of Sarsa asymptotically.

Furthermore, an additional approach to dealing with an incorrect model is to use the previous experience of our agent to estimate the reward of the learner's policy. By accurately estimating the learner's policy from past data we can disregard the opinion of the risk analyzer, whose estimate of the learner's policy is based on an incorrect model. For a given action from the learner a and the planner's policy π^p we evaluate this joint policy's return by combining two approaches. First, we estimate the reward from the joint policy by calculating the reward our agent would receive from the training data when taking the joint policy, conditioned on the agent's current state. Unfortunately, this estimate of the joint policy's reward can suffer from high variance if little data has been seen. To reduce the variance of this estimate we use the method of control variates (Zinkevich et al., 2006; White, 2009). A control variate is a random variable y that is correlated with a random variable x and is used to reduce the variance of the estimate of x's expectation. Assuming we know y's expectation we can write

$$E[x] = E[x - \alpha(y - B)]$$

where $E[y] = B$. Using the incorrect model we can form a control variate that is correlated with the reward experienced by our agent and can therefore be used to reduce the variance of the joint policy's estimated reward.

7. Conclusions

In this paper, we 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 grid world domain verified the potential of the new approach in reducing the sample complexity. 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. For the future work, we are going to apply our algorithms to large UAV mission planning domains to verify their scalability.

References

Bertsekas, Dimitri P. and Tsitsiklis, John N. *Neuro-Dynamic Programming (Optimization and Neural Computation Series, 3)*. Athena Scientific, May 1996. ISBN 1886529108.

Bhatnagar, Shalabh, Sutton, Richard S., Ghavamzadeh, Mohammad, and Lee, Mark. Incremental natural actor-critic algorithms. In Platt, John C., Koller, Daphne, Singer, Yoram, and Roweis, Sam T. (eds.), *NIPS*, pp. 105–112. MIT Press, 2007.

Bowling, Michael, Johanson, Michael, Burch, Neil, and Szafron, Duane. Strategy evaluation in extensive games with importance sampling. *Proceedings of the 25th Annual International Conference on Machine Learning (ICML)*, 2008.

Brafman, Ronen I. and Tennenholtz, Moshe. R-MAX - a general polynomial time algorithm for near-optimal re-

inforcement learning. *Journal of Machine Learning*, 3: 213–231, 2001.

Geramifard, Alborz, Redding, Joshua, Roy, Nicholas, and How, Jonathan P. UAV Cooperative Control with Stochastic Risk Models. In *American Control Conference (ACC)*, June 2011.

Howard, R A. Dynamic programming and Markov processes, 1960.

Kaelbling, Leslie Pack, Littman, Michael L., and Cassandra, Anthony R. Planning and acting in partially observable stochastic domains. *Artificial Intelligence*, 101: 99–134, 1998.

Kim, Jongrae. Discrete approximations to continuous shortest-path: Application to minimum-risk path planning for groups of uavs. In *In Proc. of the 42nd Conf. on Decision and Control*, pp. 9–12, 2003.

Littman, Michael L., Dean, Thomas L., and Kaelbling, Leslie Pack. On the complexity of solving Markov decision problems. In *In Proc. of the Eleventh International Conference on Uncertainty in Artificial Intelligence*, pp. 394–402, 1995.

Murphey, R. and Pardalos, P.M. *Cooperative control and optimization*. Kluwer Academic Pub, 2002.

Puterman, M. *Markov Decision Processes: Discrete Stochastic Dynamic Programming.* Wiley, 1994.

Redding, J., Geramifard, A., Undurti, A., Choi, H.L., and How, J. An intelligent cooperative control architecture. In *American Control Conference (ACC)*, pp. 57–62, 2010.

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

Russell, S. and Norvig, P. *Artificial Intelligence: A Modern Approach, 2nd Edition*. Prentice-Hall, Englewood Cliffs, NJ, 2003.

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

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

White, Martha. A general framework for reducing variance in agent evaluation, 2009.

Zinkevich, Martin, Bowling, Michael, Bard, Nolan, Kan, Morgan, and Billings, Darse. Optimal unbiased estimators for evaluating agent performance. In *Proceedings of the 21st national conference on Artificial intelligence - Volume 1*, pp. 573–578. AAAI Press, 2006. ISBN 978-1-57735-281-5.