An Intelligent Cooperative Control Architecture

Joshua Redding, Alborz Geramifard, Aditya Undurti, Han-Lim Choi and Jonathan P. How

Abstract— **This paper presents an extension of existing cooperative control algorithms that have been developed for multi-UAV applications to utilize real-time observations and/or performance metric(s) in conjunction with learning methods to generate a more intelligent planner response. We approach this issue from a cooperative control perspective and embed elements of feedback control and active learning, resulting in an new *intelligent Cooperative Control Architecture* (iCCA). We describe this architecture, discuss some of the issues that must be addressed, and present illustrative examples of cooperative control problems where iCCA can be applied effectively.**

I. Introduction

Most applications of heterogeneous teams of UAVs require participating agents to remain capable of performing their advertised range of tasks 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. While many of these approaches have been successfully demonstrated 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 intelligently evasive targets, adversarial actions, possibly incomplete data and delayed or lossy communications.

Additionally, although the negative impacts with modeling errors are relatively well understood, simple and robust extensions of cooperative control algorithms to account for such errors are frequently overly conservative and generally do not utilize observations or past experiences to refine poorly known models [1], [2]. 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 more adaptable planning approaches?*

To address this question while improving long-term performance in real-world applications, we propose a tighter integration of cooperative control algorithms with recent learning techniques. Many learning algorithms are well suited for online adaptation in that they explicitly use available data to refine existing models, leading to policies that fully exploit new knowledge as it is acquired [3], [4]. Such learning algorithms, combined with a cooperative planner, would be better able to generate plans that are not overly conservative. However, learning algorithms are also prone to limitations, including the following:

- They may require significant amounts of data to converge to a useful solution.
- Insufficient coverage of the training data can lead to "overfitting" and/or poor generalization.
- There are no guarantees on the robustness of the closed *learner-in-the-loop* system (robustness in learning algorithms typically refers to the learning process itself).
- Exploration is often explicit (e.g., by assigning optimistic values to unknown areas) which, in the context of cooperative control, can lead to catastrophic mistakes.
- Scenarios where agents do not share complete knowledge of the world may cause the learning algorithm to converge to local minima or to fail to converge at all.

In this work, we propose that by combining learning with an underlying cooperative control algorithm in a general, synergistic, solution paradigm, some of these limitations can be addressed. Firstly, the cooperative planner can generate information-rich feedback by exploiting the large number of agents available for learning, addressing problems raised by insufficient data. Second, learning algorithms are more effective when given some prior knowledge to guide the search and steer exploration away from catastrophic decisions. A cooperative planner can offer 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. It is proposed that this combination of cooperative control and learning will result in more successful executions of real-world missions.

Figure 1 shows a solution framework called an *intelligent Cooperative Control Architecture*, (*iCCA*), that was designed to provide 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. The remainder of this paper describes each of the *iCCA* modules and provides a sampling of example *iCCA* applications. Specifically, Section II discusses the cooperative control algorithm requirements, Section III describes the observations and performance metric(s) and Section IV outlines the requirements

J. Redding, Ph.D. Candidate, Aerospace Controls Lab, MIT
A. Geramifard, Ph.D. Candidate, Aerospace Controls Lab, MIT
A. Undurti, Ph.D. Candidate, Aerospace Controls Lab, MIT
H.-L. Choi, Postdoctoral Associate, Aerospace Controls Lab, MIT
J. P. How, Professor of Aeronautics and Astronautics, MIT
{jredding,agf,aundurti,hanlimc,jhow}@mit.edu

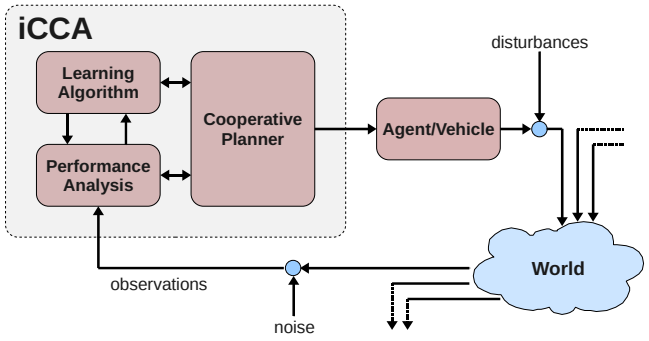

Fig. 1. An **i**ntelligent **C**ooperative **C**ontrol **A**rchitecture designed to mitigate the effects of modeling errors and uncertainties by integrating cooperative control algorithms with learning techniques and a feedback measure of system performance.

and assumptions associated with the learning element of *iCCA*. Following these descriptions, Section V provides a few examples of the application of *iCCA*.

II. Cooperative Control

In this section, we outline the cooperative control algorithm element of *iCCA*. For the purposes of this research, *cooperative control* refers to a large class of planning and control problems aimed at solving the multi-agent path planning or task allocation problem. Solving these problems is typically done in a manner which optimizes some performance criterion or *objective function*. In order to do so, the cooperative planner must maintain an internal model of which demands require which resources, and what the reward is for assigning a particular resource to a demand. These models may be probabilistic (as in the case of an MDP) or deterministic (as might be the case in a MILP-based allocation technique). As the cooperative control algorithm is the primary source of plan generation within *iCCA*, the performance and robustness properties of the integrated system rely on the accuracy of its models as well as the uncertainty representations used in the cooperative control optimization. Therefore, the planner in *iCCA* provides access to these internal models so that the the learning element can assist in their refinement.

Output from the performance analysis element is also available to the planner for use in its internal optimization. In general, the cooperative control algorithm can act directly upon the performance observed. For example, measured versus expected performance can produce what is often referred to as *temporal-difference errors* [5], which can drive a multitude of objective functions, including those found in many cooperative control algorithms. In effect, we connect the cooperative planner with both a learning method and a performance analysis method in an attempt to generate cooperative control solutions that are both robust and adaptable to errors and uncertainties without being unnecessarily conservative.

In Section V, we implement several very different approaches to the cooperative control problem and wrap *iCCA* around each. First, we use a decentralized auction-based algorithms called consensus-based bundle algorithm (CBBA) [6] as the cooperative planner. Second, we revisit previous work with multi-agent Markov decision processes (MDPs) and uncertain model parameters [4] and generalize it to fit within *iCCA*. Finally, we again use a CBBA planner, but with an actor-critic reinforcement learner.

III. Observations and Performance

The use of feedback within a planner is of course not new. In fact, there are very few cooperative control planners which do not employ some form of measured feedback. The focus of the performance analysis block within *iCCA* is to extract relevant information from the observation stream and formulate a meaningful metric that can be used in the planner itself, and/or as input to the learner.

One of the main reasons for cooperation in a cooperative control mission is to minimize the *objective function*, or some cost/reward metric. Very often this involves time, risk, fuel, or some other similar physically-meaningful quantity. The purpose of the performance analysis element in *iCCA* is to glean useful information buried in the noisy observations, then fuse, filter or otherwise transform it and present it to the learner and cooperative planner as a meaningful quantity for use in improving subsequent plans. Essentially, the purpose of the performance analysis element of *iCCA* is to assist in improving agent behavior by diligently studying its own experiences [7].

In Section V, we implement several methods of performance analysis based on observed data. In the first example, we construct temporal-difference errors based on expected and observed cost and use these errors to drive the learning of uncertain parameters. Second, we record state-dependent observations to construct the parameters used by the learner. Finally, we implement a method for analyzing risk as a form of performance and couple this with the learner.

IV. Learning

Learning has many forms. We aim to be minimally restrictive in defining the learning component of *iCCA*. However, contributions of the learner include helping the planner handle uncertainty in its internal models, and perhaps suggesting potential exploratory actions to the planner that will expedite the learning process itself. This "exploration" is a key concept in learning and brings significant challenges [8]. One of these challenges is how to *bound* or *bias* exploration such that the learner will explore the parts of the world that are likely to lead to a better model and better performance, while ensuring that it remain safely within its operational envelope and clear of undesirable states [9]–[11]. To facilitate this, the baseline cooperative control solution within *iCCA* can be used to guide the learning, acting as a constraint to prevent the learner from catastrophic errors during exploration, or perhaps as a prior distribution over the policy space.

Learning can leverage the multi-agent setting by collecting observations from the team and using the information from sensor data and observed or inferred mission successes (and

failures) as feedback signals to identify possible improvements, such as tuning the weights of an objective function. However, a trademark of any learning algorithm is that negative information is extremely useful, albeit extremely costly in the cooperative control setting. Active learning algorithms can explicitly balance the cost of information gathering against the expected value of information gathered. In section V, we give examples of active and passive maximum likelihood learners in the context of a multi-agent Markov Decision Processes and a decentralized market-based planner. Finally, we give an example of an actor-critic reinforcement learner [12] biased and guided in its exploration by a decentralized market-based planner.

V. Example Applications

A. Consensus-Based Cooperative Task Allocation

In this example, we consider a multi-agent task-allocation scenario and implement an approximate, decentralized, market-based planning algorithm called consensus-based bundle algorithm (CBBA) [6]. CBBA is a decentralized auction protocol that alternates between two phases: In the first phase, each vehicle generates a single ordered bundle of tasks by sequentially selecting the task giving the largest marginal score. The second phase resolves inconsistent or conflicting assignments through local communication between neighboring agents.

CBBA allows for various design objectives, agent models, and constraints by defining appropriate scoring functions. If the resulting scoring scheme satisfies a certain property called *diminishing marginal gain* (DMG), a provably good feasible solution is guaranteed. While the score functions primarily used in Ref [6] was time-discounted reward, Ponda (et al) [13] has extended the algorithm to appropriately handle that have finite time windows of validity, heterogeneity in the agent capabilities, and vehicle fuel costs while preserving the robust convergence properties. Here, we take this extended CBBA algorithm as the cooperative planner.

1) Problem Statement: Given a list of N_t tasks and N_a agents, the goal of the task allocation is to find a conflict-free matching of tasks to agents that maximizes some global reward. An assignment is said to be free of conflicts if each task is assigned to no more than one agent. Each agent can be assigned a maximum of L_t tasks, and the maximum overall number of assignments is given by $N_{\max} \triangleq \min\{N_t, N_a L_t\}$. The global objective function is assumed to be a sum of agents' reward values, each of which is the sum of all time-discounted values of task accomplished, less associated fuel costs. Details of the CBBA problem formulation for multi-agent task assignment can be found in Ref [13].

2) Implementation under iCCA: Figure 2 shows how this example was wrapped within the framework of *iCCA*. As seen, the CBBA algorithm serves as the cooperative planner which uses internal models of vehicle dynamics to rank bids placed by participating agents. These models, like many used in cooperative control algorithms, are good approximations and serve their purpose well. However, refining these model parameters online can increase overall performance without

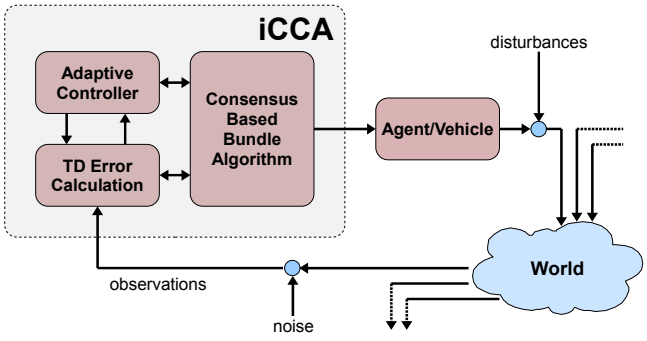

Fig. 2. *iCCA* formulation with a CBBA planner and a simple learner driven by temporal-difference errors

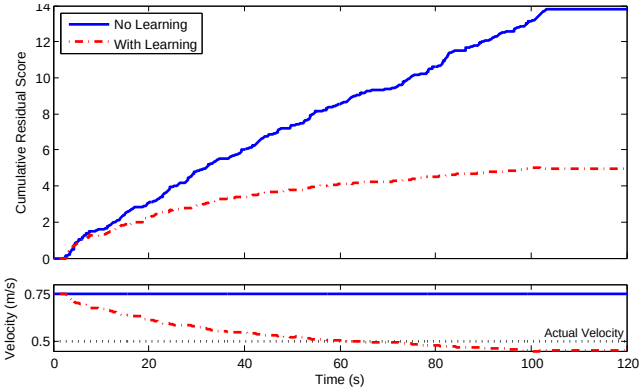

Fig. 3. Cumulative residual score (expected - actual) over the duration of the mission. When coupled with a learner, the planner more accurately anticipates performance.

sacrificing robustness. As an example, we selected the mobile agents' expected cruise velocity as an adaptive parameter since it is highly sensitive to variations in external conditions, such as congestion and collision avoidance routines. Also, this parameter is used in the CBBA scoring function and is available to the learning algorithm for online refinement. We implemented a simple adaptive controller as the learning algorithm which receives input from the performance analysis element in the form of temporal difference errors.

The performance analysis element, labeled "TD Error Calculation", observes and collects the scores achieved while servicing the tasks won during the bidding process. It then compares these actual with expected scores received from queries directly to the CBBA planner. A temporal-difference (TD) error of the form $\delta = E[\underline{x}] - \underline{x}$ captures the discrepancy between the two scores and is used to drive a direct adaptive controller which learns the true cost of fuel. In the simulated mission scenario, a time-discounted reward is received as a result of accomplishing a 5s task within the allowable time window of 15s, while costs are accrued as a result of travel.

Figure 3 shows the residual mission score as a function of time. As expected, the performance of the planning scheme increases when coupled with a learning algorithm to reduce the uncertainty around nominal cruise velocity. This enables agents to place more accurate bids and collect actual scores that are closer to the expected.

B. Multi-agent Persistent Surveillance

In this example, we formulate a multi-agent Markov Decision Process (MDP) while considering a persistent surveillance mission scenario [14]. MDPs are a natural framework for solving multi-agent planning problems as their versatility allows modeling of stochastic system dynamics as well as interdependencies between agents [15], [16].

1) Problem Statement: We formulate the persistent surveillance problem as in Ref [14], where a group of N_a UAVs are each equipped with some type(s) of sensors and are initially located at a base location. The base is separated by some (possibly large) distance from the surveillance location and the objective of the problem is to maintain a specified number N_r of requested UAVs over the surveillance location at all times while minimizing fuel costs. This represents a practical scenario that can show well the benefits of agent cooperation.

The uncertainty in this case is a simple fuel consumption model based on the probability of a vehicle burning fuel at the nominal rate, p_f. That is, with probability p_f, vehicle i will burn fuel at the known nominal rate and with probability $1 - p_f$, vehicle i will burn fuel at twice the known nominal rate during time step $j, \forall(i, j)$. When p_f is known exactly, a policy can be constructed to optimally hedge against running out of fuel while maximizing surveillance time and minimizing fuel consumption. Otherwise, policies constructed under overly conservative (p_f too high) or naive (p_f too low) estimates of p_f will respectively result in vehicles more frequently running out of fuel, or a higher frequency of vehicle phasing (which translates to unnecessarily high fuel consumption). Given the qualitative statement of the persistent surveillance problem, an MDP is formulated as detailed in Ref [4].

2) Implementation under iCCA: Wrapping the above problem statement within *iCCA*, the multi-agent MDP fits as the cooperative planner, while the performance analysis block consists of a state-dependent observation counter that tracks discretely observed fuel burn events. Two learning algorithms were implemented in this example scenario: an active and a passive learner. Both of which are based on a maximum likelihood algorithm whose parameters are updated using the number of observed fuel burn events, α_i, as counted by the performance analysis element, as depicted in Figure 4

First, the passive learner simply uses these α_i inputs to calculate $\hat{p}_f$ and the corresponding variance. Second, the active learner which, after calculating $\hat{p}_f$ and the corresponding variance, searches the possible actions and "suggests" to the planner that it take the action leading to the largest reduction in variance around p_f.

For the case of the active learner, we embed a ML learner into the MDP formulation such that the resulting policy will bias exploration toward those state transitions that will result in the largest reduction in the expected variance of the ML estimate $\hat{p}_f$. The resulting cost function is then formed as

$$g'(\mathbf{x}, \mathbf{u}) = g(\mathbf{x}, \mathbf{u}) + C\sigma^2(\widehat{p}_f)(\mathbf{x})$$

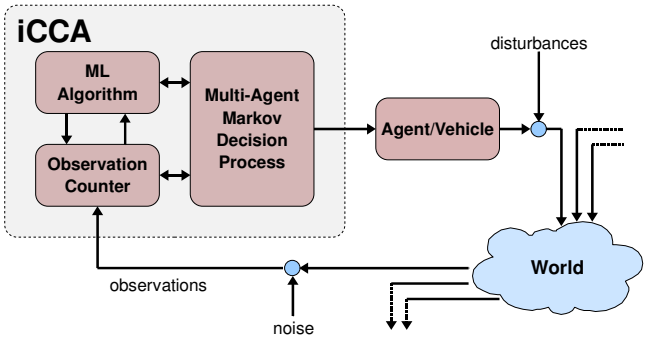

Fig. 4. *iCCA* formulation with a multi-agent MDP planner and a ML learner driven by β-distribution observation counts.

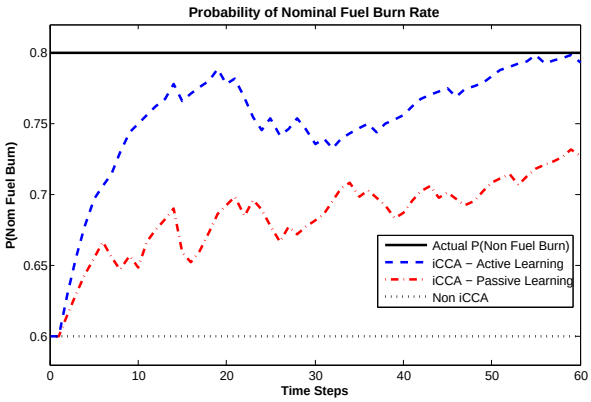

Fig. 5. Comparison of the fuel burn model parameter learned over time under the MDP formulation

where C represents a scalar gain that acts as a knob we can turn to weight exploration, and $\sigma^2(\widehat{p}_f)(\mathbf{x})$ denotes the variance of the model estimate in state $\mathbf{x}$. The variance of the Beta distribution is expressed as

$$\sigma^2(\widehat{p}_f)(\mathbf{x}) = \frac{\alpha_1(\mathbf{x})\alpha_2(\mathbf{x})}{(\alpha_1(\mathbf{x}) + \alpha_2(\mathbf{x}))^2(\alpha_1(\mathbf{x}) + \alpha_2(\mathbf{x}) + 1)}$$

where $\alpha_1(\mathbf{x})$ and $\alpha_2(\mathbf{x})$ denote the counts of nominal and off-nominal fuel flow transitions observed in state $\mathbf{x}$ respectively, by the performance module labeled "Observation Counter" in Figure 4.

Figure 5 compares the rate at which the model parameter p_f is learned using a ML estimator that uses online observations of vehicle fuel consumption. In the passive learning case, the planner chooses actions based on an internal objective function, without being "nudged" by the learner. These actions lead to observations, which in turn reduced the variance around p_f, albeit not as quickly as the active learner, as seen in Figure 5. For the case without *iCCA*, no learning is achieved and the planner assumes p_f is known and therefore acts sub-optimally: running a higher risk of crashing, or wasting fuel.

C. Actor-Critic Policy Learning

In this example, we integrate a variant of the CBBA planner presented in Section V-A with an actor-critic reinforcement learner in the context of a stochastic multi-agent

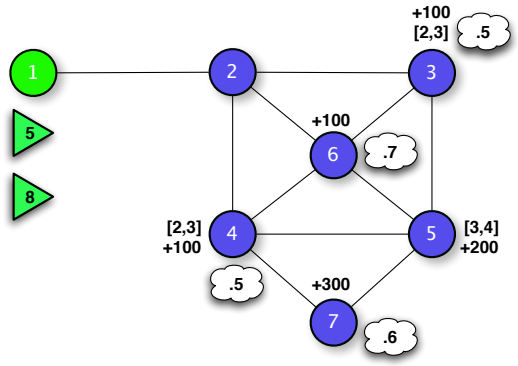

Fig. 6. A team of two UAVs (triangles) maximize their reward by cooperating to visit targets. Target nodes (circles) have a probability of receiving a reward (positive values with probability in nearby cloud). Note that some target nodes have no value. Allowable visit times shown in brackets.

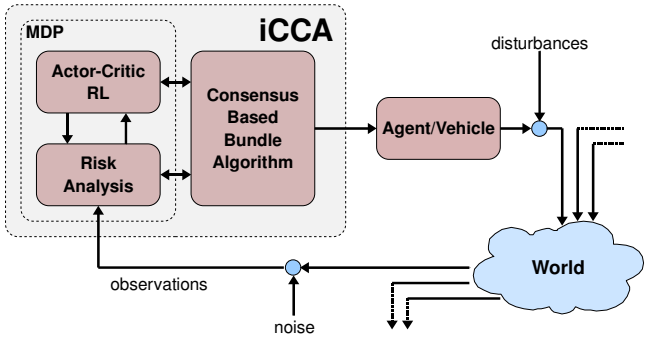

Fig. 7. *iCCA* framework as implemented. CBBA planner with the risk analysis and actor-critic learner formulated under an MDP.

task allocation scenario. We discuss a method for learning and adapting cooperative control strategies in stochastic domains.

1) Problem Statement: We consider a small team of agents that start by following the "safe" plan calculated by the baseline planner and incrementally adapting it to maximize rewards by cooperating to visit target nodes in the network.

Figure 6 depicts the problem scenario where the base is highlighted as node 1 (green circle), targets are shown as blue circles and agents 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 a positive number nearby. The constraints on 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 nearest the node.[1] Each reward can be obtained only once and all edges take one fuel cell and one time step. We also allow UAVs to loiter on any nodes for the next time step. The fuel burn for loitering action is also one except for the UAVs staying in the base, where they assumed to be stationary and sustain no fuel cost. The mission horizon was set to 8 time steps.

[1]If two agents visit a node at the same time, the probability of visiting the node would increase accordingly.

2) Implementation under iCCA: We implemented the consensus-based bundle algorithm under the same formulation as outlined in Section V-A with additional fuel constraints. Specifically, each agent was given a limited amount of fuel and the combined path length of its task bundle was constrained to be feasible with respect to fuel consumption while allowing the agent enough reserve to return to base upon completion.

For the learning algorithm, we implemented an actor-critic reinforcement learner [12] which uses information regarding performance to explore and suggest new behaviors that would likely lead to more favorable outcomes than the current behavior would produce. In actor-critic learning, the actor handles the policy, where in our experiments actions are selected based on Gibbs softmax method:

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

in which $P(s,a)$ is the preference of taking action a in state s, and $\tau \in (0,\infty]$ is the temperature parameter acting as knob shifting from greedy towards random action selection. Since we use a tabular representation the actor update amounts to $P(s,a) \leftarrow P(s,a) + \alpha Q(s,a)$, where α is the learning rate.

As for the critic, we employ the temporal-difference learning algorithm [17] to update the associated value function estimate. We initialized the actor's policy by bootstrapping the initial preference of the actions generated by CBBA.

The performance analysis block is implemented as a "Risk Analysis" tool where actions suggested by the learner can be overridden by the baseline cooperative planner if they are deemed too "risky". This synergistic relationship yields a "safe" policy in the eyes of the planner, upon which the learner can only improve. A trademark of learning algorithms in general, is that negative information is extremely useful in terms of the value of information it provides. This is unacceptable in a multi-agent environment. We therefore introduce the notion of a "virtual reward" - a large negative value delivered to the learner for suggesting risky actions. When delivered, the learner associates this negative reward with the previously suggested action, dissuading the learner from suggesting it again.

The formulation shown in Figure 7 allows for the key concept of *biased & guided* exploration such that the learner can explore the parts of the world that are likely to lead to better system performance while ensuring that the agent remain safely within its operational envelope and away from states that are known to be undesirable, such as running out of fuel. The *bias* is introduced as the initial policy is seeded using the initial plan generated by the baseline CBBA planner. In order to facilitate the *bound*, the risk analysis module inspects all actions suggested by the actor and replaces the "risky" ones with the action specified by CBBA, thus guiding the learning away from catastrophic errors. In essence, the baseline cooperative control solution provides a form of "prior" over the learner's policy space while also acting as a backup policy in the case of an emergency.

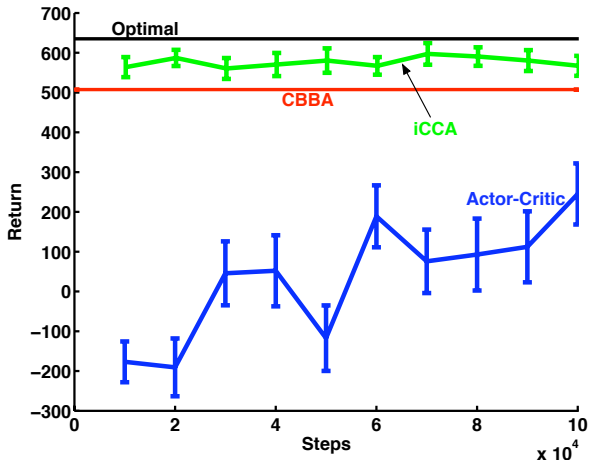

Fig. 8. A comparison of the collective rewards received when strictly following plans generated by CBBA alone, actor-critic reinforcement learning inside/outside of the iCCA environment. All are compared against the optimal performance as calculated via dynamic programming.

We first solved the problem using backward dynamic programing in order to use this solution as a benchmark for comparison (this took about a day and cannot be easily scaled for larger sizes of the problem). We then ran CBBA on the expected deterministic problem (as converted from the stochastic problem), and ran it for 10,000 episodes. For all experiments, we set the preference of the advised CBBA state-action pairs to 100. τ was set to 1 for the actor. Figure 8 depicts the performance of iCCA and Actor-Critic averaged over 60 runs. The Y-axis shows the cumulative reward, while the X-axis represents the number of interactions. Each point on the graph is the result of running the greedy policy with respect to the existing preferences of the actor. For iCCA, risky moves again were replaced by the CBBA baseline solution. Error bars represent the standard error with $\%90$ confidence interval. In order to show the relative performance of these methods with offline techniques, the optimal and CBBA solutions are highlighted as lines. It is clear that the actor-critic performs much better when wrapped into the iCCA framework and performs better than CBBA alone. The reason is that CBBA provides a good starting point for the actor-critic to explore the state space, while the risk analyzer filters risky actions of the actor which leads into catastrophic scenarios.

VI. Conclusions

In conclusion, we have shown how existing cooperative control algorithms can be extended to utilize real-time observations and performance metric(s) in conjunction with learning methods to generate a more intelligent planner response. We approached the issue from a cooperative control perspective and have embedded elements of feedback control and active learning, resulting in an intelligent Cooperative Control Architecture (iCCA). We described this architecture and presented illustrative examples of cooperative control problems where iCCA was applied.

VII. Acknowledgements

This research was supported in part by AFOSR (FA9550-08-1-0086) and Boeing Research & Technology. In addition, the authors would like to thank Sameera Ponda and MIT Professors Nick Roy and Emilio Frazzoli for their many insightful conversations and contributions on this subject.

References

[1] M. Alighanbari, L. F. Bertuccelli, and J. P. How, "A robust approach to the uav task assignment problem," in *IEEE CDC*, 2006.

[2] L. F. Bertuccelli, "Robust decision-making with model uncertainty in aerospace systems," Ph.D. dissertation, MIT, 2008.

[3] B. Bethke, L. F. Bertuccelli, and J. P. How, "Experimental Demonstration of Adaptive MDP-Based Planning with Model Uncertainty." in *AIAA Guidance Navigation and Control*, Honolulu, Hawaii, 2008.

[4] J. Redding, B. Bethke, L. Bertuccelli and J. How, "Active Learning in Persistent Surveillance UAV Missions," in *AIAA Infotech@Aerospace*, Seattle, WA, 2009.

[5] D. Bertsekas and J. Tsitsiklis, *Neuro-Dynamic Programming*. Belmont, MA: Athena Scientific, 1996.

[6] H.-L. Choi, L. Brunet, and J. P. How, "Consensus-based decentralized auctions for robust task allocation," *IEEE Trans. on Robotics*, vol. 25 (4), pp. 912 – 926, 2009.

[7] S. Russell and P. Norvig, "Artificial Intelligence, A Modern Approach," 2003.

[8] S. Thrun, "Efficient exploration in reinforcement learning," *Technical ReportCMU-CS-92-102, CarnegieMellonUniversity, SchoolofComputerScience*, 1992.

[9] A. Moore and C. Atkeson, "Prioritized sweeping: Reinforcement learning with less data and less time," *Machine Learning*, vol. 13, no. 1, pp. 103–130, 1993.

[10] M. Dorigo and L. Gambardella, "Ant colony system: A cooperative learning approach to the traveling salesman problem," *IEEE Transactions on evolutionary computation*, vol. 1, no. 1, pp. 53–66, 1997.

[11] G. Chalkiadakis and C. Boutilier, "Coordination in multiagent reinforcement learning: A bayesian approach," in *Proceedings of the second international joint conference on Autonomous agents and multiagent systems*. ACM, 2003, p. 716.

[12] S. Bhatnagar, R. S. Sutton, M. Ghavamzadeh, and M. Lee, "Incremental natural actor-critic algorithms." in *NIPS*, J. C. Platt, D. Koller, Y. Singer, and S. T. Roweis, Eds. MIT Press, 2007, pp. 105–112.

[13] S. Ponda, J. Redding, H.-L. Choi, B. Bethke, J. P. How, M. Vavrina, and J. L. Vian, "Decentralized planning for complex missions with dynamic communication constraints," in *American Control Conf*, 2010.

[14] B. Bethke, J. P. How, and J. Vian, "Group Health Management of UAV Teams With Applications to Persistent Surveillance," in *American Control Conference*, June 2008, pp. 3145–3150.

[15] M. L. Puterman, "Markov decision processes," 1994.

[16] M. L. Littman, T. L. Dean, and L. P. Kaelbling, "On the complexity of solving markov decision problems," in *11th Int'l Conf on Uncertainty in AI*, 1995, pp. 394–402.

[17] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*. MIT Press, 1998. [Online]. Available: citeseer.csail.mit.edu/sutton98reinforcement.html