An Intelligent Cooperative Control Architecture

Josh Redding, 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 decentralized 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 or performing their advertised range of tasks in the face of noise, unmodeled dynamics (both internal and external) 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 a partially observable, dynamic, stochastic, and/or hostile environment [1] [2] [3]. While many of these approaches have been successfully demonstrated in a variety of simulations and some focused experiments, there remains ample room for improving 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, delayed and/or lossy communications, and non-Gaussian noise models.

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 [4]–[6]. Furthermore, it is often difficult, if not impossible, to tune design parameters *a priori* when it is unclear what objective functions and constraints should be included to achieve the desired mission performance. Despite these issues however, cooperative control algorithms provide a baseline capability for achieving challenging multi-agent mission objectives. In this context, relevant research questions include:

J. Redding, 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,aundurti,hanlimc,jhow}@mit.edu

> *How can current cooperative control algorithms be extended to result in more adaptable planning approaches?*

and

> *How can such planning algorithms be generalized to develop richer policy classes that provide a more intelligent planner response?*

To address this, and improve long-term performance in real-world applications, we propose a tighter integration of cooperative control algorithms with recent learning techniques [7]–[9]. While machine learning is certainly not a panacea for cooperative control, many learning algorithms are well suited for on-line adaptation in that they explicitly use available data to refine existing models, leading to policies that fully exploit new knowledge as it is acquired [10], [11]. Such learning algorithms could address a key issue with current cooperative control algorithms. In general however, learning algorithms are prone to limitations that are relevant in this context. Among these are 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 as robustness in learning algorithms typically refers to the robustness of 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.

However, learning and/or refining the underlying cooperative control models in real-time would lead to a reduction in the associated uncertainties and could greatly improve overall multi-agent performance. Therefore, we propose an architecture that utilizes both cooperative control and machine learning algorithms for that which each was intended and creates a general, synergistic solution paradigm, which we call an *intelligent Cooperative Control Architecture* (*iCCA*). In general, the *iCCA* combines a cooperative planner, a learner, some metric of performance-to-date and on-line observations, as shown in Figure 1. In context, learning algorithms can be effective given some prior knowledge to guide the search and/or exploration away from catastrophic decisions. A cooperative planner can offer this knowledge by providing the baseline capability for achieving mission

objectives. In addition, the cooperative planner can generate information-rich feedback by exploiting the large number of agents available for learning. In return, the learning algorithm enhances the performance of the planner by adapting it to time-varying parameters. The synergistic combination of these two approaches will help bridge the gap to successful execution in real-world missions.

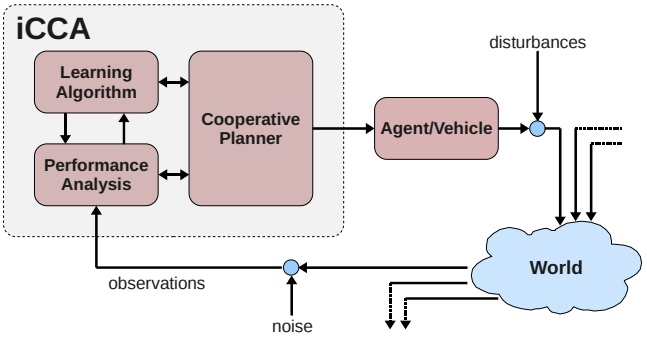

Fig. 1. An **i**ntelligent **C**ooperative **C**ontrol **A**rchitecture

In short, *iCCA* is a framework for 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 and assumptions associated with the learning element of *iCCA*. Following these descriptions, Section V provides a few baseline 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, and indeed in general, "cooperative control" refers to a large class of planning and control problems aimed at solving the multi-agent path planning and resource allocation problems. In general, an algorithm that solves a resource allocation problem is one that matches demands with resources, i.e. assigns resources to the demands in a manner that optimizes some performance criterion that is commonly called the *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). That the planning algorithm does in fact, solve this problem is a requirement for the cooperative planner within *iCCA*. As the cooperative control algorithm is the primary source of plan generation, even 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 should provide access to these internal models if it is desired that the learning element assist in refining these models online.

Output from the performance analysis element is also available to the planner for use in its internal optimization. In general, the cooperative control algorithm could act directly upon the performance observed. For example, measured versus expected performance can produce what is often referred to as "temporal-difference errors" [12], which can drive a multitude of objective functions, including those found in many cooperative control algorithms [13]. In short, 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 uncertainties without being unnecessarily conservative.

In Section V, we implement two very different approaches to the cooperative control problem and wrap *iCCA* around each. The first example is an auction-based algorithms called consensus-based bundle algorithm (CBBA) [14]. CBBA is a distributed task allocation framework developed to address complex missions for heterogeneous teams of autonomous agents in dynamic environments. Second, we revisit previous work with multi-agent Markov decision processes (MDPs) with uncertain model parameters [10] and generalize it to fit within *iCCA*.

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 some cost metric. Very often this involves time, risk, fuel, or some other similar physically-meaningful quantity. In this section, we outline the the performance analysis element of *iCCA*. In general, this module may massage raw observations into some useful form, glean useful information buried in the noisy observations and categorize it and use it to improve subsequent plans. In other words, the purposes of the performance analysis element of *iCCA* is to assist in improving agent behavior by diligently studying its own experiences [15].

In Section V, we implement two methods of performance analysis based on observed data. In the first example, we construct temporal-difference errors based on expected and observed costs. These errors then drive the learning of uncertain parameters. Second, we use the observations to construct the parameters of a maximum likelihood (ML) estimator for a fuel consumption parameter with a prior Beta distribution. Both applications fit within *iCCA* performance analysis module.

IV. Learning

Learning has many forms. We aim to be minimally restrictive in defining the learning component of *iCCA*.

However, contributions of the learner should include helping the planner handle uncertainty in its internal models, and perhaps even 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, including how to perform *bounded* exploration such that the learner can explore the parts of the world that may lead to a better model while ensuring that the agent remain safely within its operational envelope and free from harm. The 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.

Any of supervised, unsupervised or reinforcement learning methods could potentially fill this element of *iCCA*, depending on the application details. In supervised learning, an all-knowing "teacher" is always immediately on hand to give correct labels to training samples while a hypothesis function is constructed. Given such supervision, a "knows what it knows" (KWIK [7]) learner could fit nicely into this box.

Learning can also leverage the multi-agent setting by observing self and others 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. A canonical failure of any learning algorithm however, is that negative information is extremely useful, albeit extremely costly overall. 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 a maximum likelihood learning in the context of a Consensus-Based Bundle Algorithm and a multi-agent Markov Decision Process.

V. Example Applications

In this section, we implement *iCCA* in the context of several cooperative control scenarios and show how mission performance is improved over more traditional cooperative control strategies.

A. Consensus-Based Cooperative Task Allocation

In this example, we consider a multi-agent task-allocation scenario and implement an approximate planning algorithm called consensus-based bundle algorithm (CBBA) [14]. CBBA is a decentralized auction protocol that produces conflict-free assignments relatively robust to disparate situational awareness over the network. The task allocation problem CBBA solves is described as follows.

Problem Formulation

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 local reward values, while each local reward is determined as a function of the tasks assigned to each agent. Mathematically, this allocation problem is written as the following integer (possibly nonlinear) program with binary decision variables x_{ij} that indicate whether or not task j is assigned to agent i:

$$\begin{aligned} \max \quad & \sum_{i=1}^{N_a} \left(\sum_{j=1}^{N_t} c_{ij}(\tau_{ij}(\mathbf{p}_i(\mathbf{x}_i)))x_{ij} \right) \\ \text{subject to:} \quad & \sum_{j=1}^{N_t} x_{ij} \leq L_t, \ \forall i \in \mathcal{I} \\ & \sum_{i=1}^{N_a} x_{ij} \leq 1, \ \forall j \in \mathcal{J} \\ & x_{ij} \in \{0,1\}, \ \forall (i,j) \in \mathcal{I} \times \mathcal{J} \end{aligned} \tag{1}$$

where $x_{ij} = 1$ if agent i is assigned to task j, and $\mathbf{x}_i \in \{0,1\}^{N_t}$ is a vector whose j-th element is x_{ij}. The index sets are defined as $\mathcal{I} \triangleq \{1, \ldots, N_a\}$ and $\mathcal{J} \triangleq \{1, \ldots, N_t\}$. The vector $\mathbf{p}_i \in (\mathcal{J} \cup \{\emptyset\})^{L_t}$ represents an ordered sequence of tasks for agent i; its k-th element is $j \in \mathcal{J}$ if agent i conducts j at the k-th point along the path, and becomes $\emptyset$ (denoting an empty task) at the k-th point if agent i conducts less than k tasks. The summation term in brackets in the objective function represents the local reward for agent i. As indicated in the objective function, it is assumed that:

1) The score c_{ij} that agent i obtains by performing task j is defined as a function of the arrival time τ_{ij} at which the agent reaches the task (or possibly the expected arrival time in a probabilistic setting).
2) The arrival time τ_{ij} is *uniquely* defined as a function of the path $\mathbf{p}_i$ that agent i takes.
3) The path $\mathbf{p}_i$ is *uniquely* defined by the assignment vector of agent i, $\mathbf{x}_i$.

Algorithm Description

CBBA consists of iterations 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. In the local communication round, some agent i sends out to its neighboring agents two vectors of length N_t: the winning agents vector $\mathbf{z}_i \in \mathcal{I}_t^N$ and the winning bids vector $\mathbf{y}_i \in \mathbb{R}_+^{N_t}$. The j-th entries of the $\mathbf{z}_i$ and $\mathbf{y}_i$ indicate who agent i thinks is the best agent to take task j, and what is the score that agent gets from task j, respectively. The essence of CBBA is to enforce every agent to agree upon these two vectors, leading to agreement on some conflict-free assignment regardless of inconsistencies in situational awareness over the team.

There are several core features of CBBA identified in [14]. First, CBBA is a decentralized decision architecture. For a large team of autonomous agents, it would be too restrictive to assume the presence of a central planner (or server) with which every agent communicates. Instead, it

is more natural for each agent to share information via local communication with its neighbors. Second, CBBA is a polynomial-time algorithm. The worst-case complexity of the bundle construction is $\mathcal{O}(N_t L_t)$ and CBBA converges within $\max\{N_t, L_t N_a\}D$ iterations, where N_t denotes the number of tasks, L_t the maximum number of tasks an agent can win, N_a the number of agents and D is the network diameter, which is always less than N_a. Thus, the CBBA methodology scales well with the size of the network and/or the number of tasks (or equivalently, the length of the planning horizon). Third, various design objectives, agent models, and constraints can be incorporated 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 [14] was time-discounted reward, the authors have 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 [16]. This paper takes this extended CBBA algorithm as a cooperative planner.

iCCA Application

Figure 2 shows how this example was wrapped within the

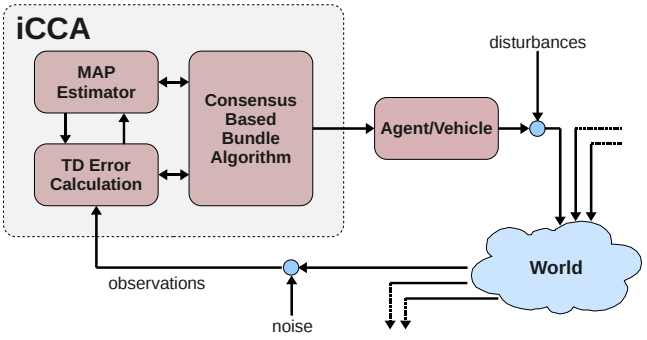

Fig. 2. *iCCA* formulation with a CBBA planner and a MAP learner driven by TD error observations.

framework of *iCCA*. As seen, the CBBA algorithm serves as the cooperative planner which uses models of vehicle dynamics and fuel burn to score 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 serve to both make the planner more robust as well as increase overall performance. As an example, we selected the fuel burn cost as a model parameter worth adapting and refining due to its high sensitivity to variations in external conditions. As mentioned, this fuel burn parameter is used in the CBBA scoring function and is available to the learning algorithm for online refinement. We implemented a maximum likelihood estimator as the learning algorithm which receives input from the performance analysis element in the form of temporal difference errors.

The performance analysis element, labelled "TD error Calculation", calculates expected costs and collects actual

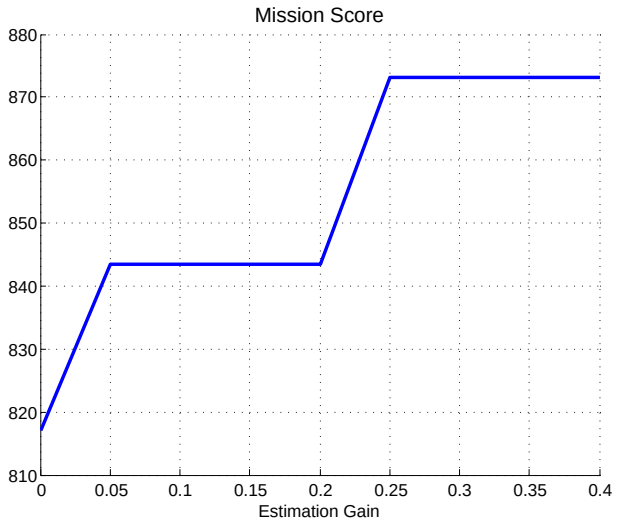

Fig. 3. Comparison of mission score as a function of estimator gain. Without the *iCCA* framework corresponds to gain = 0.

costs from the observation stream. A TD error of the form $\delta = E[\underline{x}] - \underline{x}$ captures any discrepancies between actual and expected costs of servicing the tasks won during the bidding process. Results of the multi-agent task allocation scenario both within the *iCCA* framework and outside are shown in Figure 3. In the mission scenario, rewards are received as a result of accomplishing tasks while costs are accrued as a result of traveling between tasks. Figure 3 shows the conglomerate mission score as a function of the ML estimator gain $k(\sigma^2)$. In the case under the *iCCA* framework, we see the mission score increases with the estimation gain.

B. Multi-agent Persistent Surveillance

In this example, we formulated a multi-agent Markov Decision Process (MDP) while considering a persistent surveillance mission scenario. Markov decision processes (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 [2], [15], [17]–[19].

In the persistent surveillance problem [20], there is a group of n UAVs, each equipped with some type(s) of sensors. The UAVs are initially located at a base location, which is separated by some (possibly large) distance from the surveillance location. The objective of the problem is to maintain a specified number r of requested UAVs over the surveillance location at all times. 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_{nom}. That is, with probability p_{nom}, vehicle i will burn fuel at the known nominal rate during time step $j, \forall(i,j)$ When p_{nom} 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_{nom} too high) or naive (p_{nom} too low) estimates of p_{nom} 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).

1) MDP Formulation: Given the qualitative description of the persistent surveillance problem, an MDP can now be formulated. The MDP is specified by $(\mathcal{S}, \mathcal{A}, P, g)$, where $\mathcal{S}$ is the state space, $\mathcal{A}$ is the action space, $P_{\mathbf{xy}}(\mathbf{u})$ gives the transition probability from state $\mathbf{x}$ to state $\mathbf{y}$ under action $\mathbf{u}$, and $g(\mathbf{x}, \mathbf{u})$ gives the cost of taking action $\mathbf{u}$ in state $\mathbf{x}$. Future costs are discounted by a factor $0 < \alpha < 1$. A policy of the MDP is denoted by $\mu : \mathcal{S} \to \mathcal{A}$. Given the MDP specification, the problem is to minimize the so-called cost-to-go function J_μ over the set of admissible policies Π:

$$\min_{\mu \in \Pi} J_\mu(\mathbf{x}_0) = \min_{\mu \in \Pi} \mathbb{E}\left[\sum_{k=0}^{\infty} \alpha^k g(\mathbf{x}_k, \mu(\mathbf{x}_k))\right].$$

State Space $\mathcal{S}$

The state of each UAV is given by two scalar variables describing the vehicle's flight status and fuel remaining. The flight status y_i describes the UAV location,

$$y_i \in \{Y_b, Y_0, Y_1, \ldots, Y_s, Y_c\}$$

where Y_b is the base location, Y_s is the surveillance location, $\{Y_0, Y_1, \ldots, Y_{s-1}\}$ are transition states between the base and surveillance locations (capturing the fact that it takes finite time to fly between the two locations), and Y_c is a special state denoting that the vehicle has crashed.

Similarly, the fuel state f_i is described by a discrete set of possible fuel quantities,

$$f_i \in \{0, \Delta f, 2\Delta f, \ldots, F_{max} - \Delta f, F_{max}\}$$

where Δf is an appropriate discrete fuel quantity. The total system state vector $\mathbf{x}$ is thus given by the states y_i and f_i for each UAV, along with r, the number of requested vehicles:

$$\mathbf{x} = (y_1, y_2, \ldots, y_n; f_1, f_2, \ldots, f_n; r)^T$$

Control Space $\mathcal{A}$

The controls u_i available for the i^{th} UAV depend on the UAV's current flight status y_i.

- If $y_i \in \{Y_0, \ldots, Y_s - 1\}$, then the vehicle is in the transition area and may either move away from base or toward base: $u_i \in \{\text{“}+\text{”}, \text{“}-\text{”}\}$
- If $y_i = Y_c$, then the vehicle has crashed and no action for that vehicle can be taken: $u_i = \emptyset$
- If $y_i = Y_b$, then the vehicle is at base and may either take off or remain at base: $u_i \in \{$“take off”,“remain at base”$\}$
- If $y_i = Y_s$, then the vehicle is at the surveillance location and may loiter there or move toward base: $u_i \in \{$“loiter”,“$-$”$\}$

The full control vector $\mathbf{u}$ is thus given by the controls for each UAV:

$$\mathbf{u} = (u_1, \ldots, u_n)^T \tag{2}$$

State Transition Model P

The state transition model P captures the qualitative description of the dynamics given at the start of this section. The model can be partitioned into dynamics for each individual UAV.

The dynamics for the flight status y_i are described by the following rules:

- If $y_i \in \{Y_0, \ldots, Y_s - 1\}$, then the UAV moves one unit away from or toward base as specified by the action $u_i \in \{\text{“}+\text{”}, \text{“}-\text{”}\}$.
- If $y_i = Y_c$, then the vehicle has crashed and remains in the crashed state forever afterward.
- If $y_i = Y_b$, then the UAV remains at the base location if the action “remain at base” is selected. If the action “take off” is selected, it moves to state Y_0.
- If $y_i = Y_s$, then if the action “loiter” is selected, the UAV remains at the surveillance location. Otherwise, if the action “$-$” is selected, it moves one unit toward base.
- If at any time the UAV's fuel level f_i reaches zero, the UAV transitions to the crashed state ($y_i = Y_c$).

The dynamics for the fuel state f_i are described by the following rules:

- If $y_i = Y_b$, then f_i increases at the rate $\dot{F}_{refuel}$ (the vehicle refuels).
- If $y_i = Y_c$, then the fuel state remains the same (the vehicle is crashed).
- Otherwise, the vehicle is in a flying state and burns fuel at a stochastically modeled rate: f_i decreases by $\dot{F}_{burn}$ with probability p_{nom} and decreases by $2\dot{F}_{burn}$ with probability $(1 - p_{nom})$.

Cost Function g

The cost function $g(\mathbf{x}, \mathbf{u})$ penalizes three undesirable outcomes in the persistent surveillance mission. First, any gaps in surveillance coverage (i.e. times when fewer vehicles are on station in the surveillance area than were requested) are penalized with a high cost. Second, a small cost is associated with each unit of fuel used. This cost is meant to prevent the system from simply launching every UAV on hand; this approach would certainly result in good surveillance coverage but is undesirable from an efficiency standpoint. Finally, a high cost is associated with any vehicle crashes. The cost function can be expressed as

$$g(\mathbf{x}, \mathbf{u}) = C_{loc} \max\{0, (r - n_s(\mathbf{x}))\} + C_{cr} n_{cr}(\mathbf{x}) + C_f n_f(\mathbf{x})$$

where:

- $n_s(\mathbf{x})$: number of UAVs in surveillance area in state $\mathbf{x}$,
- $n_{cr}(\mathbf{x})$: number of crashed UAVs in state $\mathbf{x}$,
- $n_f(\mathbf{x})$: total number of fuel units burned in state $\mathbf{x}$,

and C_{loc}, C_{cr}, and C_f are the relative costs of loss of coverage events, crashes, and fuel usage, respectively.

2) iCCA Application: Wrapping the above MDP formulation within *iCCA*, the multi-agent MDP fits nicely as the cooperative planner, while the performance analysis block consists of a few simple counters that track discretely

observed fuel burn events. Two learning algorithms were implemented in this example scenario, both of which are based on a maximum likelihood estimator whose parameters are updated using the number of observed fuel burn events, α_i, as counted by the performance analysis element. First, the passive learner simply uses these α_i inputs to calculate $\hat{p}_{nom}$ and the corresponding variance. Second, the active learner which, after calculating $\hat{p}_{nom}$ 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_{nom}.

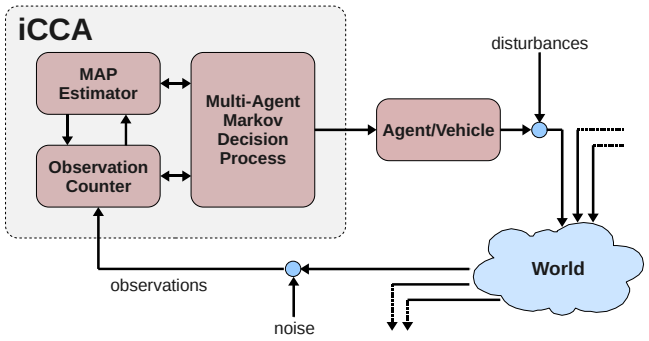

Fig. 4. *iCCA* formulation with a multi-agent MDP planner and a MAP learner driven by β distribution observation counts.

Specifically, for the case of the active learner, we embed a ML estimator 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 $\hat{p}_{nom}$. The resulting cost function is then formed as

$$g'(\mathbf{x}, \mathbf{u}) = g(\mathbf{x}, \mathbf{u}) + C_{\sigma^2}\sigma^2(\widehat{p}_{nom})(\mathbf{x})$$

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

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

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 labelled "Observation Counter".

Figure 5 compares the rate at which the model parameter p_{nom} is learned using the ML estimator that was constructed using online observations of vehicle fuel consumption. In the active learning case, the planner was given additional bonuses in the objective function for visiting states that the learner believed would result in a large reduction of the variance surrounding the uncertain parameter, p_{nom}. In the passive learning case, the planner chose actions based on its own internal objective function, without being "nudged" by the learner. These actions naturally led to observations, which in turn reduced the variance around p_{nom}, only not as much. As seen, in Figure 5 both active and passive learning

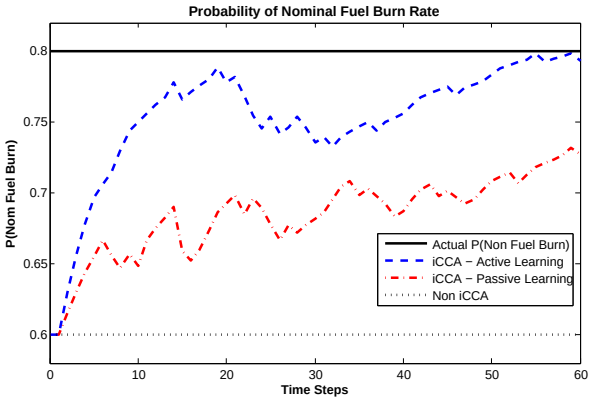

Fig. 5. Comparison of estimator results under MDP formulation

methods approach the true value for p_{nom} in time, however the active learner approaches much faster, as expected. For the case without *iCCA*, no learning is achieved and the planner assumes p_{nom} is known and therefore acts sub-optimally.

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 MIT Professors Nick Roy and Emilio Frazzoli for their many insightful conversations and contributions on this subject.

References

[1] E. F. Ketan Savla, Tom Temple, "Human-in-the-loop vehicle routing policies for dynamic environments," in *IEEE Conference on Decision and Control*, 2008.

[2] L. P. Kaelbling, M. L. Littman, and A. R. Cassandra, "Planning and acting in partially observable stochastic domains," *Artificial Intelligence*, vol. 101, pp. 99–134, 1998.

[3] A. B. H. Brendan McMahan, Geoffrey J. Gordon, "Planning in the presence of cost functions controlled by an adversary," in *Proceedings of the Twentieth International Conference on Machine Learning*, 2003.

[4] M. Alighanbari, L. F. Bertuccelli, and J. P. How, "A robust approach to the uav task assignment problem," in *IEEE Conf. on Decision and Control*, 2006.

[5] R. Rockafellar and S. Uryasev, "Optimization of conditional value at risk," *http://www.ise.ufl.edu/uryasev*, 1999.

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

[7] L. Li, M. Littman, and T. Walsh, "Knows what it knows: a framework for self-aware learning," in *Proceedings of the 25th international conference on Machine learning*. ACM New York, NY, USA, 2008, pp. 568–575.

[8] M. Bowling, "Convergence and no-regret in multiagent learning," in *Advances in neural information processing systems 17: proceedings of the 2004 conference*. The MIT Press, 2005, p. 209.

[9] A. Greenwald, A. Jafari, and C. Marks, "A general class of no-regret learning algorithms and game-theoretic equilibria," in *Learning theory and Kernel machines: 16th Annual Conference on Learning Theory and 7th Kernel Workshop, COLT/Kernel 2003, Washington, DC, USA, August 24-27, 2003: proceedings*. Springer Verlag, 2003, p. 2.

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

[11] M. Likhachev and A. Stentz, "PPCP: Efficient probabilistic planning with clear preferences in partially-known environments," in *PROCEEDINGS OF THE NATIONAL CONFERENCE ON ARTIFICIAL INTELLIGENCE*, vol. 21, no. 1. Menlo Park, CA; Cambridge, MA; London; AAAI Press; MIT Press; 1999, 2006, p. 860.

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

[13] R. Murphey and P. Pardalos, *Cooperative control and optimization*. Kluwer Academic Pub, 2002.

[14] 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.

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

[16] 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 *submitted to American Control Conference*, 2010.

[17] R. A. Howard, "Dynamic programming and markov processes," 1960.

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

[19] M. L. Littman, T. L. Dean, and L. P. Kaelbling, "On the complexity of solving markov decision problems," in *In Proc. of the Eleventh International Conference on Uncertainty in Artificial Intelligence*, 1995, pp. 394–402.

[20] 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.