Constrained-Action POMDPs for Multi-Agent Intelligent Knowledge Distribution

Michael Fowler, Pratap Tokekar, T. Charles Clancy, and Ryan K. Williams

Abstract— **This paper addresses a fundamental question of multi-agent knowledge distribution: what information should be sent to whom and when, with the limited resources available to each agent? Intelligent Knowledge Distribution is a framework that answers these questions. Communication requirements for multi-agent systems can be rather high when an accurate picture of the environment and the state of other agents must be maintained. To reduce the impact of multi-agent coordination on systems, including communications, this paper introduces the concept of action-based constraints on partially observable Markov decision processes, rewards based upon the value of information driven by Kullback-Leibler Divergence, and probabilistic constraint satisfaction through discrete optimization and Markov chain Monte Carlo analysis. Intelligent Knowledge Distribution is driven by determining the *information content* an agent believes another agent will obtain by receiving certain information, along with the importance or relevance of that information to the system objective. To perform constraint analysis on an infinite-horizon policy, policies are represented as a Finite State Controller allowing Markov chain Monte Carlo analysis to determine a probabilistic level of guarantee that the constraints will be satisfied. The analysis of performance for an example mission presented in this paper shows the constrained controllers, during the highest constraint seen in simulations, can be constructed to meet minimal constraint guarantees (80%) while impacting the optimal value less than 50%, where the unconstrained optimal controller only satisfied the constraint 10% of the time.**

I. Introduction

Coordination and collaboration between multiple autonomous systems in the field often creates a significant burden on communication systems in order to achieve global objectives. To provide resiliency to systems, decentralized or distributed approaches have been implemented that allow for an agent to act independently with little to full information of the other agents' states and actions. The key concepts and contributions of our approach is to provide a framework for *Intelligent Knowledge Distribution* which decides on what information is transmitted to whom and when, while minimizing the impact this coordination between agents will have on the limited resources available to each agent, such as power or bandwidth.

In disaster response or military operations, remote sensing by autonomous systems provides a stream of data back to critical personnel. Disaster recovery and other sensor based applications rely on high bandwidth links that are stretched to their limits under unpredictable channel and link qualities [1] in support of infrared, electrooptical, video surveillance, chemical sensors, radar, electronic sensing, etc. At the same time, the communications necessary for multiple agents to coordinate in support of a mission needs to stay within a constrained allowance so as to not impact overall information flow in the system. Along with common bandwidth constraints, unmanned aerial vehicles (UAV) for example have the additional restriction of minimizing the utilization of energy to maximize their flight time in support of field operations.

We assume each agent is executing its own decision making in a decentralized or distributed multi-agent network as in decentralized POMDPs (Dec-POMDP) [2]–[4], where the agent has local state observations of itself but local indirect observations of other agents and the environment. The key is to determine what information another agent will need at what point in time to improve the efficiency of the tasking and behavior without assuming unconstrained resource availability, e.g., communication are instantaneous and free [5].

In this paper, the Constrained-Action POMDP (CA-POMDP) for *Intelligent Knowledge Distribution* (IKD) is validated using a model of monitoring of a ground asset during a disaster response to ensure they safely avoid hazards and dangerous situations they might not be aware of from their perspective on the ground. The constraints are soft and the resulting infinite-horizon policy is evaluated as the probability it will not violate the constraint during a period of time. Soft constraints are often relevant in communication decisions over the enforcement of hard constraints for multiple reasons: Infinite-horizon POMDPs are often represented as a finite state controller [6] and evaluating hard constraints on cyclic graphs is intractable; resource availabilities are not instantaneous but long-term as with total battery power to maintain flight times needed for safety monitoring; and, available bandwidth in the network then is being allocated to collaboration and network protocols like ZeroMQ permit queueing techniques to account for myopic link saturation.

The remainder of this paper is organized as follows. Secion II provides a perspective of the concept in relation to work being done in the field. In Section III, we go over the fundamental models at the core of the system that our architecture is based upon. This is all put together in a new approach to provide Constrained-Action POMDPs in Section IV. Section V describes how a CA-POMDP was formulated and tested. The paper wraps up with future and continuing research in Section VI.

M. Fowler, P. Tokekar, T. C. Clancy, and R. K. Williams are with the Department of Electrical and Computer Engineering, Virginia Polytechnic Institute and State University, Blacksburg, VA USA, E-mail: {*mifowler, tokekar, tcc, rywilli1*}*@vt.edu*

II. Related Work

Capitan's paper [5] is a key comparison in analyzing the performance of our paper in the scenario of target tracking, but the authors made the assumption that point-to-point communications are free and instantaneous between nodes and used a collaboration policy of decentralized auctions with a type of data fusion similar to consensus. In this paper, we are particularly removing the assumption of free and instantaneous communication to provide multi-agent communication to stay within the bounds of resource availability.

Constrained MDPs have been used historically to solve two major drawbacks of standard discrete MDPs: Multiple objectives and limited resources [7]–[14]. Applying constraints to Markov Decision Processes (MDPs) has been well established in restricting utilization of states to prevent collisions [9], and has been expanded to hierarchical cases to reduce the complexity of the linear programming formulation [10], [11]. The primary objective of the Constrained MDP (CMDP) is to find a policy that's within a restricted state, cost, and/or reward structure of an unconstrained MDP. The CMDP extends the MDP model by including the matrix of decision variables, $Q_t \in \Re^{n,p}$, and upper bounds on the state and transition probabilities, $\mathbf{d}$. Finding solutions to constrained MDPs is reliant on solving linear programming (LP) formulations, either as a pure LP problem [10] or incorporated into dynamic programming (DP) [9].

Constrained POMDPs, which provide constraints for partially observable environments, have also been explored using mixed integer linear programming (MILP) for evaluating policies by introducing a discrete variable $d^i \in \{0, 1\}$ [13], [14], a standard trick with conditional constraints [14]. Both [13] and [14] solve the POMDP using value iteration, though [13] also describes a point-based value iteration (PBVI) approach to provide an upper bound to a heuristic search. [14] also introduces a policy iteration approach based upon a deterministic finite-state controller and policy improvement from [12].

One of the drawbacks of applying constraints to the state space in LP or MILP or as penalties in the value function is that the resulting policy may be too conservative or risky with any value of MC [13]. [13] compensates for this tendency through an online algorithm that maintains two values: $V_R(a, s)$ for the rewards; and, $V_C(s)$ for the constraint penalty. A finite horizon is computed off-line with a "constraint-penalty-to-go" every dynamic programming iteration using value iteration as defined in [14] for solving the MILP, but it doesn't provide a policy iteration solution which is necessary in IKD for constraint evaluation.

These approaches to constrained MDPs and POMDPs apply constraints to the state-space of the model and projection into the value space, but our formulation requires action-based constraints as we are limiting the utilization of resources that an action consumes. The resulting POMDP infinite-horizon policy is represented as a finite state controller, which requires analyzing how a cyclic graph will utilize the resources it consumes. To the best of authors' knowledge, it is not possible to represent resource utilization of actions of an entire cyclic controller in the state or value space as utilized in CMDP and CPOMDP.

III. Background

A. Partially Observable Markov Decision Processes

Markov decision processes (MDP) are a decision theoretic approach to determining the best actions over a time horizon based upon the current state of the system (s), the probability of transition to another state ($T(s'|s)$), and the rewards associated with a state and action ($R(s)$) [6], [15]. It is often used to determine optimal actions where the rewards are delayed or realized at some future point in time and the agent needs to determine a set of actions, or policy (π), that achieves this goal. If the state of the agent is not fully observable then the system becomes a Partially Observable MDP (POMDP) where belief states, $b \in \mathbf{B}$, are used to represent the probability distribution of being in any particular state based upon observations [16]. These belief states are tied to an observation model, O, that maps the probability of being in a state based upon what was observed. A POMDP can be described formally by the vector

$$< \mathbf{S}, \mathbf{O}, \mathbf{B}, \mathbf{A}, \mathbf{T}, \mathbf{R}, \mathbf{\Pi} >$$

where $\mathbf{S}$ is the set of states an agent can be in, $\mathbf{A}$ is the set of actions the agent can take, $\mathbf{T}$ is the set of transition probabilities ($T(s'|s, a)$) between states based on an action, $\mathbf{R}$ is the set of rewards ($R(s, a)$) for taking an action in a state, and $\mathbf{\Pi}$ is the set of policies, π, that are feasible consisting of a set of vectors of actions, $\{\pi_0, \pi_1, \ldots, \pi_t\}$. The objective is to select a policy, π, that maximizes the expected utility over time: $\pi^*(s) = \arg\max_\pi U^\pi(s)$ The transition probabilities are similar to those in the discrete MDP but expanded as

$$\begin{aligned}\tau(b'|b, a) &= P(b'|b, a) \\ &= \sum_o P(b'|b, a, o) \sum_{s'} O(o|s') \sum_s T(s'|s, a) b(s) \quad (1)\end{aligned}$$

where $P(b'|b, a, o)$ in the above equation is the Kronecker delta function, δ, to update the belief [16] to account for the belief states and observation model. The immediate reward is simply a summation of the product of the reward for each state by the belief that the agent is in that state, $R(b, a) = \sum_s R(s, a) b(s)$.

B. Markov Chain Monte Carlo

A Markov Chain Monte Carlo (MCMC) is a particle-based approximate inference sampling technique that uses a sequence of samples to iteratively approach a desired posterior distribution [17]. When samples are initially likely to be from the prior distribution, the sampling of a sequence of samples will successively approach the posterior, whereas likelihood based approaches are unlikely to account for this fact and place undue weighting on earlier samples. Another gain in utilizing MCMCs to determine posterior distributions is the versatility in the type of distribution that can be inferred without being bound to approximating distributions as Gaussian.

The Metropolis-Hastings algorithm is a useful construct for working with proposal distributions rather then ensuring we are always sampling from the next state. The algorithm uses a random walk approach that either accepts or denies a proposal rather than trying to track importance weights. The acceptance ratio reduces to

$$\mathcal{A} = \frac{P(x|\mu)P(\mu)}{P(x|\mu_0)P(\mu_0)} \tag{2}$$

of the proposed posterior distribution, $P(x|\mu)P(\mu)$, over the current posterior distribution, $P(x|\mu_0)P(\mu_0)$,. The most difficult part of calculating the posterior with the Bayes formula, the evidence $P(x)$, is common to both the proposed and current posterior and therefore conveniently cancels out. If the random number is lower than the acceptance ratio, then the proposed posterior is accepted. In cases where the proposal distribution is larger than the current distribution, then the acceptance ratio will be greater than one and therefore will always be accepted. On the contrary when the proposal is less, then there is a random chance that it will accept the proposed allowing the chain to be reversible and ensure it seeks the posterior.

IV. Constrained-Action POMDP

The objective of *Intelligent Knowledge Distribution* is to determine what information needs to be shared with whom at an appropriate time. Therefore, actions are simply to transmit the information learned, $\iota \in I$, to agent $n \in N$ designated as a^n_ι. As the number of agents grows in the system and naive communication grows drastically $\frac{1}{2}|A|(|A|-1)$, it is necessary to reduce the complexity of the model by determining an appropriate coordination graph [18]–[20]. In a homogeneous system, the coordination graph can be formulated geographically as those operating in the area of the current agent that can assist or needs to know information. On the contrary in heterogeneous systems, the coordination graph can focus more on collaborative capabilities and interoperability.

The goal is to adapt an infinite-horizon finite state controller to probabilistically guarantee not to violate the soft constraints within a desired period of time. Policies are represented as a finite-state controller (FSC), since an FSC is a cyclic graph that represents an infinite-horizon POMDP well [21]. An infinite-horizon policy allows the controller to run continuously without needing to restart back at the initial horizon point of finite horizon. It also can accelerate the convergence of an appropriate FSC, since an infinite horizon policy in value iteration is not guaranteed to converge [6].

The state of the system is the discrete variable that another agent has relevant and timely information that an agent has obtained, which is not directly observable. Therefore the belief state $b(s)$ is a probability distribution that indicates the confidence that an agent has that a collaborating agent $n_{-i} \in \mathcal{N}_i$ is “up to date.”

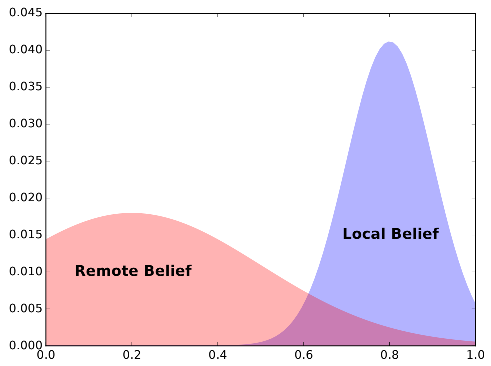


Fig. 1: Comparison between the value of information of the local agent’s belief state (blue) to the belief state of one of the collaborative agents (red)

A. Value of Information

The rewards for the POMDP are determined by calculating the value of the information based upon the known belief of information to the agent in comparison to what the agent believes the information will improve the belief state of a neighboring agent. The relative entropy metric provides the framework for determining the information theoretic information gain between the expected entropy of another agent and the actual entropy of the local agent. Relative entropy [22], [23], using Kullback-Liebler divergence

$$D_{KL}(P||Q) = \sum_x p(x) \log \frac{p(x)}{q(x)} \geq 0, \tag{3}$$

is based on the distribution of the agent’s beliefs, which can easily be calculated from the agent’s existing belief state and its estimated belief state of the another agent’s belief state.

In Figure 1, two distributions are shown with the belief of the local agent and its estimated belief of another neighbor agent in the coordination graph. To determine the reward for communicating what the local agent knows, it needs to understand what information the other agent would gain from receiving an update. Kullback-Leibler Divergence, D_{KL}, provides the ability to determine the information gain obtained from the local agent sharing its local beliefs, b'_0, in comparison to what it believes the other agent contains, b'_n.

$$V_\iota = \sum_{s \in S} b'_0(s) \log \frac{b'_0(s)}{b'_n(s)} \tag{4}$$

The KL Divergence is dependent on $|N|$ probability distributions for each local agent, which requires a belief update mechanism for both the local agents belief in the information it has and in predicting the belief states of the other agents. As the local agent receives new information about an item, it can update its own belief states depending on the noise and accuracy of the sensor input. At the same time, it updates the belief state of the other agents (5) [16] as it performs a

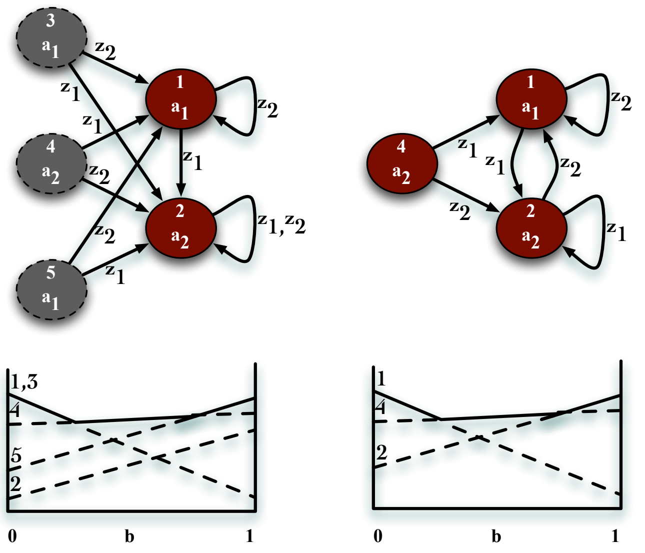


Fig. 2: The new vectors 3, 4, and 5 from the dynamic programming update are connected to the original controller δ. 3 and 1 are matches for action, values, and transitions and therefore 1 is kept in δ'. 5 point-wise dominates 2 and therefor 2 inherits the parameters of 5. [24]

NoOp action without any observation to improve accuracy (e.g, $\mathcal{N}(\mu, \sigma)$ for a Kalman Filter).

$$b'(s') = \frac{p(z|s') \sum_s p(s'|a,s)b(s)}{\sum_{s,s''} p(z|s'')p(s''|a,s)b(s)} \tag{5}$$

B. POMDP Policy Iteration

The policy iteration algorithm of Hansen in [24] is the basis for improving the finite state controller which uses simple transformations that search within the policy space until epsilon convergence [24]. The function starts by initializing the policy for a finite state controller δ and then evaluates the value of a policy through a system of linear equations. The finite state controller (FSC), δ, is a directed cyclical graph where the nodes in the graph represent actions, $a \in \mathcal{A}$, and the edge transitions are driven by observations, $o \in O$. A dynamic programming update is used to generate a new set of candidate machine states that the policy improvement loop uses to compare the vectors of V and V' and modifies the FSC accordingly. Any change to the FSC for a link $l(i,z)$ from a machine state i due to an observation z triggers a recalculation of the policy. The algorithm terminates when the Bellman residual between the two value functions V and V' is within epsilon convergence.

Evaluating the policy of a finite state controller requires solving a system of linear equations [6]

$$v^i(s) = r(s, a(i)) + \gamma \sum_{s'z} P(s'|s,a)P(z|s',a)V^{l(i,z)}(s'). \tag{6}$$

For each machine state in the controller and state in the model, the value of a particular machine state i is the reward for taking the action dictated by the machine state plus the discounted summation of the rewards of transitions to another state $l(i,z)$, based upon receiving the observation, z. The dynamic programming update uses

$$\Theta_n^{a_i,o_i}(b,s) = \frac{r(s,a_i)}{|\Omega|} + \gamma \sum_{s' \in S} p(s,a_i,s')O(s',o_i)\Theta_{n-1}^{a_i,o_i}(b^{a_i,o_i}, s) \tag{7}$$

$$\Theta_n^{a_i}(b) = \sum_{o \in \Omega} \Theta_n^{a_i,o}(b) \tag{8}$$

to derive a new set of vectors for consideration from each vector (machine state) in the current policy, δ, machine states 3–5 are derived from 1 and 2 in Figure 2. For each vector in the existing policy, the algorithm determines the resulting value vector from each action proceeding after the current action. The piece $\frac{r(s,a_i)}{|\Omega|}$ averages the reward across observations so that the final summation of $\Theta_n^{a_i}(b)$ across all observations, $o \in \Omega$, is correct.

This new set of vectors is run through a linear programming formulation (9) to prune the set of vectors to just the dominant vectors, which are represented by the new states 3, 4, and 5 in Figure 2. It introduces Δ to objective function with the value coefficients for the value factors set to zero. If it finds a value of $\Delta > 0$ or the formulation fails to find a solution, then that vector is dominated by another and is removed from the set [6].

$$\begin{aligned} &Objective: \quad \max \Delta \\ &Subject\ to: \quad x(\theta - \bar{\theta}) \geq \Delta, \quad \forall \bar{\theta} \in \mathbf{\Theta}, \theta \neq \bar{\theta} \end{aligned} \tag{9}$$

The algorithm makes two checks between the new controller V' and the previous controller V. The first determines if there are any existing vectors in V that have the same action and observation edge transitions, $l(a,z)$, as a vector in V' and places the previous vector into the new controller δ' was done with 1 and 3 in Figure 2. If there is no match, then it determines if there is any vector in V that is point-wise dominated by a vector in V' and places the previous vector into the new controller but with the actions and transitions of the new vector in V' as 5 pointwise dominated 2 in the graph. The final step in improving the controller requires pruning the new controller δ', which removes vectors $i \in \delta$ from the controller δ' that have no correlating vector in V' if they are not reachable from a machine state in V'.

C. Constraint Improvement

Constraint improvement of the existing ϵ-optimal policy is solved by using a combinatorial discrete optimization technique. The approach is to use a search and bound algorithm using the constraint evaluation of the optimal FSC as an initial heuristic and bound for pruning. Since we know *a priori* the branching function, an ordered set can be precomputed using the action's resource utilization and the reward. The first ordering

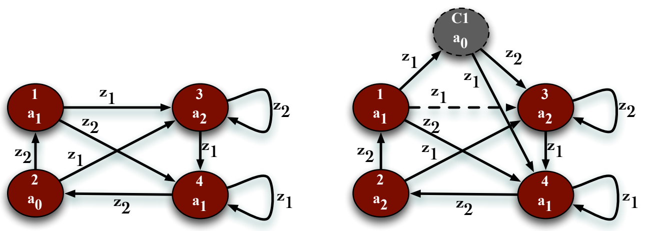


Fig. 3: The constraint $C1$ has been injected into the controller to reduce the resource utilization of machine state 3. The edge determination algorithm has selected the $1 \rightarrow 3$ to be reconnected as $1 \rightarrow C1$ and $C1$ inherits the outgoing edges of 3. Notice the loop for observation z_2 at 3 becomes a transition of $C1 \rightarrow 3$

$$V^i \prec Y^j \prec \cdots \prec V^{|K|}; i, j, \ldots, |K| \in \delta \tag{10}$$

is determined by sorting in ascending order the rewards of the vectors in the existing controller δ, which is easily calculated since we know the belief region that the action dominates from the policy improvement algorithm. This determines the first branching function that will drive the selection of which existing machine state will have a constraint state injected as a companion to increase constraint satisfaction. The second ordering

$$U^i \succ U^j \succ \cdots \succ U^{|K|}; i, j, \ldots, |A| \in A \tag{11}$$

is computed by the resource utilization of the actions $a \in A$ in descending order, which is assumed to be inversely proportional to the rewards, where K is the vectors or machine states in the controller δ, all ordered items are unique, and A is the set of actions available with their utilization U. The vector's rewards for that introduced constraint action are calculated using the optimal controller's belief region that it is attempting to replace from the previous branching function. This allows the search and bound to prune branches that will not successfully achieve a constraint satisfaction greater than current bound on the search.

Discrete optimization will terminate when all the branches have been pruned and there is no more improvement, but this is not the desired behavior operationally. In CA-POMDP, the constraint improvement terminates when either the discrete optimization terminates indicating failure to find a feasible solution or the constraint satisfaction meets or exceeds the desired behavior. Due to branching function's ordered sets, we will have a controller that meets the desired behavior while minimizing the impact on the optimal value.

During the search and bound, the constraint state being considered is injected into the controller and the edges leading to this machine state are selected as either maintaining their transition to the existing machine state or to the new machine state. This edge determination is selected by Monte Carlo with a probability of acceptance driven by simulated annealing

$$P(b, C, T) = k_1 e^{-(e-C)/T} \tag{12}$$

with the "temperature" defined by the proximity of current constraint satisfaction bound as it approaches the desired behavior, where b is the current bound of the search and bound algorithm, C is the desired constraint behavior, T is pre-calculated to provide the asymptotic curve shape, and k_1 to scale the maximum value to be around $p(0) = 0.75$. Outgoing edges are inherited by the constraint state from the existing machine state. The values for the temperature T and k_1 are specifically chosen to ensure that at the extremes there is still a probability of selecting or not selecting a new edge to the constraint space.

Figure 3 provides an example of the injection process where the search has determined to insert constraint state $C1$ as a replacement for machine state 3 because it is the machine state that has the best impact on constraint satisfaction with the minimal impact to optimal value. The edge determination has randomly selected to redirect the edge between 1 and 3 to $C1$ and $C1$ inherits the outgoing edges from machine state 3. It's important to note that the loop at machine state 3 for observation z_2 does not create a loop in $C1$ for z_2 but is directed to the original edge $l^3(a_2, z_2) = 3 \rightarrow l^{C1}(a_0, z_2) = 3$.

D. Constraint Satisfaction Evaluation

Constraint satisfaction is determined by random sampling of the controller and determining the appropriate resource utilization through a Markov chain Monte Carlo (MCMC). Using the length of time or epochs that the constraints are being applied, the finite state controller is sampled by Monte Carlo following the edges around the controller recording the resource utilization and variation of the associated actions per machine state. This is repeated from multiple initial belief states to gather a sequence of samples for MCMC mixing for a minimum number of steps, which is evaluated through a variety of heuristics [17]. To determine how long we need to randomly select proposals from the successor states until the chain has approached the posterior, the algorithm needs to know when it has reached an ϵ-mixing time determined by variational distance [17].

$$\mathbf{D}(P^{(T)}; \pi) = \max_{\alpha \in S} |P^{(T)}(\alpha) - \pi(\alpha)| \leq \epsilon \tag{13}$$

Utilizing ϵ-mixing time, the system knows when it can terminate the Metropolis-Hastings proposal sampling, but it is also necessary to determine when to stop collecting samples. This is achieved by leveraging a generalization of the central limit theorem to Markov chains and terminate on the autocovariance (14) between samples from the chain [17]. The samples of the controller are terminated when the autocovariance of the lagged MCMC distributions have converged

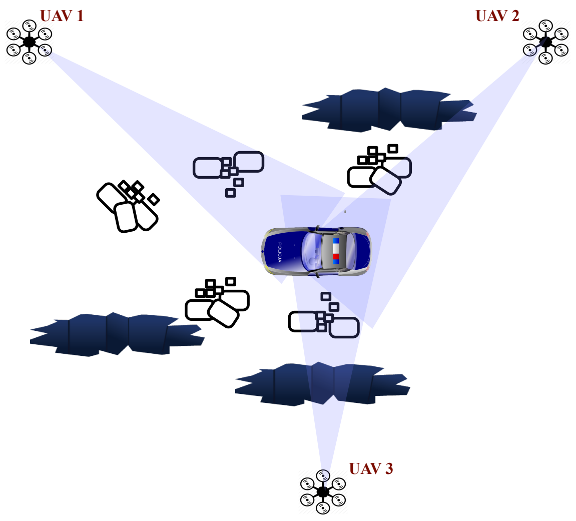


Fig. 4: The conceptual diagram of the situation being modeled with an autonomous emergency vehicle traversing a disaster site with blocked views of potential hazards (e.g. sinkholes)

$$Cov\left[f(\mathbf{X}[m]); f(\mathbf{X}[m+l)\right] \approx \frac{1}{M-l}\sum_{m=1}^{M-l} f(\mathbf{X}[m] - \hat{\mathbf{E}}_D(f))(f(\mathbf{X}[m+l] - \hat{\mathbf{E}}_D(f)) \tag{14}$$

where $\mathbf{M}$ is the set of samples collected and $\hat{\mathbf{E}}_D(f)$ is an unbiased estimator if the chain has mixed. Once the MCMC has terminated, the resource utilization posterior distribution is used to determine probability that a controller is within the cumulative distribution function (CDF), $\int_{\infty}^{S} p(x)dx$, which is our probability of satisfying the constraint S.

V. Model & Results

The model used for testing the Constrained-Action POMDP for Intelligent Knowledge Distribution is based on the monitoring of ground assets during a disaster response to ensure they safely avoid hazards and dangerous situations they might not be aware of from their perspective on the ground. The agents are station keeping unmanned aerial vehicles (UAVs) situated at critical observation locations around the disaster site monitoring the ground assets communicating across a first responders mobile ad-hoc network (MANET) as shown in Figure 4. In this initial study to validate the concept, the agents are only tracking a single asset and need to determine what sensor information needs to be shared between them to ensure they understand the risk to the asset and provide a warning.

The model used for intelligent knowledge distribution is a combinatorial problem between the neighboring nodes, the sensor data, the relevance of the information, and the current level of collaboration, which is assumed to be Markovian. Since an agent can only communicate with a single node at a time, it needs to determine the risk it believes an asset is under (relevance) and how confident it is in the data it has received so far from other nodes (collaboration) to ascertain the appropriate information to send to whom to provide proper and timely warning to the ground asset, while not consuming the limited resources of bandwidth and battery available to the MANET and UAV respectively.

A. *Collaborative Actions, States, & Rewards*

Due to size, weight, and power (SWAP) restrictions of the UAVs, they are assumed to have heterogeneous sensor support with overlapping sensor support. Each UAV has been assigned to carry two sensors on their platform from a total possibility of three: (1) RF Geolocation, (2) Optical Tracking, and (3) Laser Range Finding. Therefore each node has the option of sending the result of a sensor to any one of the nodes that it's connected to in the coordination graph. For a UAV, connected to two other drones with RF geolocation and optical tracking capabilities, there is five total actions to choose during any time epoch:

1) Do not communicate (Silence)
2) Send RF Geolocation results to Node A
3) Send Optical Tracking results to Node A
4) Send RF Geolocation results to Node B
5) Send Optical Tracking results to Node B

The state information is assumed to be a combinatorial set of the relevance of the data, the risk to the ground asset in this case, and the state of collaboration, the expected value of information based upon previous communications and the quality of the data.

The rewards were also combinatorial with an assumption that they are independent and therefore can be expressed as the sum of individual rewards for relevance and collaboration with each agent per

$$R(a, s) = R(a, s_{rel}) + \sum_{n \in N} R(a, s_{coll_n}). \tag{15}$$

The collaboration is pre-calculated based upon an analysis of the range of distribution possibilities and their KL-Divergence, discussed in Section IV-A, that has been quantized. The reward for relevance comes from a parallel process that is determining the probability the ground asset will encounter the hazard based upon its current behavior and motion.

B. *Transitions & Observations*

Transitions and observation probabilities from one state to another or for an observation given a state based upon an action is also assumed for computational simplicity in validating the approach to be independent, as in rewards, and therefore are the product of the independent transitions.

$$T(a, s_{rel}, s \in S_{coll}, s' \in S_{coll}) = T(a, s_{rel}, s'_{rel}) \cdot \prod_{n \in N} T_n(a, s_{coll_n}, s'_{coll_n}) \quad (16)$$

$$O(a, s_{rel}, s \in S_{coll}, o \in O) = O(a, s_{rel}, o_{rel}) \cdot \prod_{n \in N} O_n(a, s_{coll_n}, o_{coll_n}) \quad (17)$$

The transition probabilities for relevance are driven by the relationship between the monitoring UAVs and the ground asset. The ground asset is assumed to be reachable by the UAVs through the MANET or a multi-hop network so that it can be influenced in finding a more conservative route. The timeliness and usefulness of the data being transmitted and received is key to calculating the collaboration transition probabilities. Collaboration observation probabilities are determined by the uncertainties in the sensor measurements while the relevance observations are driven by the quality of the sensor to work with the sensors they have on board.

C. Constraints on Communication

The constraints being analyzed by the constraint evaluation algorithm is the utilization of bandwidth (bytes per second) and power (Watt second) over a second, which is a sampling length of 100 epochs as the UAV makes a collaboration decision every $10ms$. The constraints are considered soft constraints because their is more available bandwidth in the network then is being allocated to collaboration and network protocols like ZeroMQ permit queueing techniques to account for saturated links. The power utilization is considered an average power consumption needed for the communication system to use to ensure the UAV can maintain the flight time needed to maintain safety monitoring for the ground asset.

Resource utilization is dependent on the type of information they are sending and it is assumed that sending optical data requires more data and therefore higher power requirements to achieve the SNR needed for the higher bandwidth. After communicating optical tracking results, the data for RF geolocation and laser range finder take sequentially less data and power.

D. Constraint Satisfaction and Optimal Value

The model was run through the Constrained-Action POMDP for multiple values of the resource constraints or limitations for predetermined resource utilizations per action. Figure 5 shows the analysis of the data from results of the constraint satisfaction and the value of the finite state controller. For all cases, the CA-POMDP was able to achieve over 90% constraint satisfaction for the final FSC. In the worse-case scenario the optimal controller was only able to achieve 10% constraint satisfaction and with the addition of constraint states was able to improve to 90% satisfaction with a constrained FSC within 54% of the optimal value. As the optimal unconstrained FSC naturally approaches the constraint satisfaction, then the required constraint states to make it compliant drastically decrease its impact on the value function.

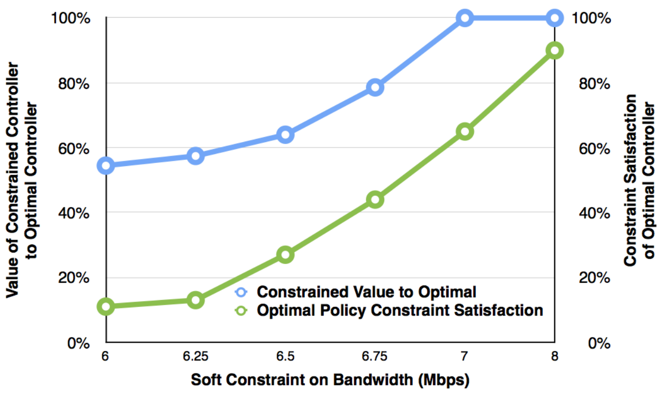


Fig. 5: The impact of the value function to the optimal value and constraint satisfaction between the optimal controller without any constraint requirements and the constrained controller versus bandwidth limitations for a system communication with two collaborative nodes with three different information types.

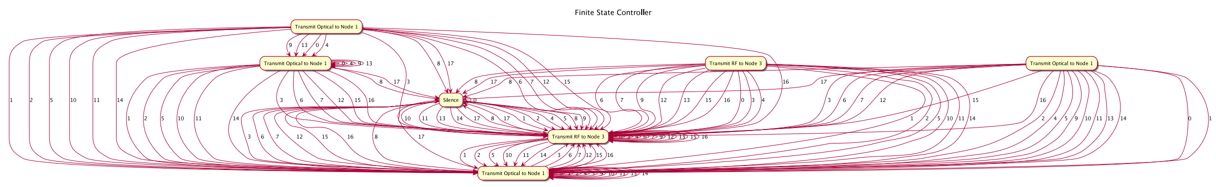
E. Constraint Impact on Automata

CA-POMDP introduced a constraint state that brings the constraint satisfaction above 80% while minimizing the impact to the value of the FSC. The farther away the optimal unconstrained FSC is from the desired behavior, the more likely the CA-POMDP will assign constraint states that utilize the least resources which often negatively impacts the multi-agent collaboration. As the unconstrained FSC approaches but does not meet the desired behavior, an action that utilizes more resources, though less than matched machine state, and has a greater value to the collaboration of the system is chosen. An example seen during this simulation is the inclusion of *Silence*, Figure 6, as the constraint state when the constraint satisfaction was naturally low and the inclusion of a *Transmit RF Geolocation* was selected since it requires less resources then the optical data.

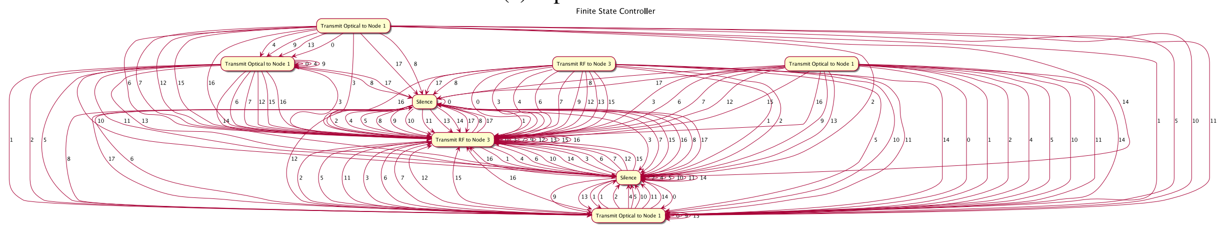
One interesting result seen in Figure 5 is the point where the constrained FSC value is the same as the optimal FSC despite the unconstrained FSC being below the 80% desired behavior. This was determined to be due to the fact that it introduced into the FSC a constraint state that was exactly the one it replaced but optimized the edge transitions to achieve the desired constraint satisfaction behavior.

VI. Conclusion

The approach provides a framework for IKD by constraining the actions of a POMDP to only transmit information to a neighbor when the value of that information warrants the communication to overcome the constraints of performing that action. Constrained-Action POMDPs provide a level of guarantee of constraint satisfaction to a desired behavior while still allowing short-term bursts of critical information. The methodology also provides a mechanism for evaluating the value of information through KL Divergence to determine


Finite State Controller
Transmit Optical to Node 1
Transmit Optical to Node 1
Transmit RF to Node 3
Transmit Optical to Node 1
Silence
Transmit RF to Node 3
Transmit Optical to Node 1

(a) Optimal Controller


Finite State Controller
Transmit Optical to Node 1
Transmit Optical to Node 1
Transmit RF to Node 3
Transmit Optical to Node 1
Silence
Transmit RF to Node 3
Silence
Transmit Optical to Node 1

(b) Constrained Controller

Fig. 6: Comparison of (a) an Optimal Controller with only 10% constraint satisfaction and (b) a Constrained Controller showing the addition of an action to bring the system within resource limits of the UAV

if a set of information to an agent that is part of a collaboration is warranted. It also introduces the concept of using Markov chain Monte Carlo analysis to evaluate a infinite-horizon policy represented as a finite state controller for its probability of satisfying constraints and combinatorial discrete optimization for achieving desired constraint behavior while maximizing the controller value.

For IKD in real-time scenarios, the agents available to collaborate may not be known *a priori*, therefore learning heterogeneous interactions and constructing coordination graph online is necessary. Along the same path, the communication links and their usage will not be known until the system is in the field and even then will change in a dynamic environment requiring that constraints also be learned online. The edge determination initially assumes a uniform distribution when randomly selecting edges to follow in the controller and improving the controller will require learning observation edge probability transitions to improve constraint satisfaction as its being utilized in the field.

References

[1] G. Tuna, B. Nefzi, and G. Conte, "Unmanned aerial vehicle-aided communications system for disaster recovery," *Journal of Network and Computer Applications*, vol. 41, pp. 27–36, 2014.

[2] S. Omidshafiei, A.-a. Agha-mohammadi, C. Amato, and J. P. How, "Decentralized control of partially observable Markov decision processes using belief space macro-actions," in *2015 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2015, pp. 5962–5969.

[3] F. A. Oliehoek and C. Amato, *Concise Introduction to Decentralized POMDPs*. Springer, 2016.

[4] S. Seuken and S. Zilberstein, "Improved memory-bounded dynamic programming for decentralized POMDPs," *CoRR*, vol. abs/1206.5295, 2012.

[5] J. Capitan, M. T. Spaan, L. Merino, and A. Ollero, "Decentralized multi-robot cooperation with auctioned POMDPs," *The International Journal of Robotics Research*, vol. 32, no. 6, pp. 650–671, 2013. [Online]. Available: http://ijr.sagepub.com/content/32/6/650.abstract

[6] O. Sigaud and O. Buffet, *Markov decision processes in artificial intelligence: MDPs, beyond MDPs and applications*. London: John Wiley & Sons, 2010.

[7] E. Altman, *Constrained Markov decision processes*. CRC Press, 1999, vol. 7.

[8] A. Beynier and A.-I. Mouaddib, "An iterative algorithm for solving constrained decentralized Markov decision processes," in *AAAI*, vol. 6, 2006, pp. 1089–1094.

[9] M. E. Chamie and B. Acikmese, "Finite-Horizon Markov decision processes with state constraints," *arXiv preprint arXiv:1507.01585*, Jul. 2015.

[10] S. Feyzabadi and S. Carpin, "HCMDP: A hierarchical solution to constrained Markov decision processes," in *2015 IEEE International Conference on Robotics and Automation (ICRA)*, May 2015, pp. 3971–3978.

[11] ——, "Risk-aware path planning using hirerachical constrained Markov decision processes," in *2014 IEEE International Conference on Automation Science and Engineering (CASE)*, Aug. 2014, pp. 297–303.

[12] E. A. Hansen, "Finite-memory control of partially observable systems," Ph.D. dissertation, University of Massachusets Amherst, 1998.

[13] A. Undurti and J. P. How, "An online algorithm for constrained POMDPs," in *Robotics and Automation (ICRA), 2010 IEEE International Conference on*. IEEE, 2010, pp. 3966–3973.

[14] J. D. Isom, S. P. Meyn, and R. D. Braatz, "Piecewise linear dynamic programming for constrained POMDPs." in *AAAI*, 2008, pp. 291–296.

[15] Q. Hu and W. Yue, *Markov decision processes with their applications*. Springer Science & Business Media, 2007, vol. 14.

[16] M. J. Kochenderfer, C. Amato, and H. J. D. Reynolds, *Decision making under uncertainty: theory and application*. MIT press, 2015.

[17] D. Koller and N. Friedman, *Probabilistic graphical models: principles and techniques*. MIT press, 2009.

[18] H. Ahmadzadeh and E. Masehian, "Fuzzy coordination graphs and their application in multi-robot coordination under uncertainty," in *Robotics and Mechatronics (ICRoM), 2014 Second RSI/ISM International Conference on*. IEEE, 2014, pp. 345–350.

[19] Y. Li, K. Gupta, and S. Payandeh, "Motion planning of multiple agents in virtual environments using coordination graphs," in *Proceedings of the 2005 IEEE International Conference on Robotics and Automation*. IEEE, 2005, pp. 378–383.

[20] N. Vlassis, R. Elhorst, and J. R. Kok, "Anytime algorithms for multiagent decision making using coordination graphs," in *Systems, Man and Cybernetics, 2004 IEEE International Conference on*, vol. 1. IEEE, 2004, pp. 953–957.

[21] D. S. Bernstein, E. A. Hansen, and S. Zilberstein, "Bounded policy iteration for decentralized POMDPs," in *Proceedings of the nineteenth international joint conference on artificial intelligence (IJCAI)*, 2005, pp. 52–57.

[22] W. R. Ashby and J. Pierce, "An introduction to cybernetics," *Physics Today*, vol. 10, no. 7, pp. 34–36, 1957.

[23] D. J. MacKay, *Information theory, inference and learning algorithms*. Cambridge university press, 2003.

[24] E. A. Hansen, "Solving POMDPs by searching in policy space," in *Proceedings of the Fourteenth conference on uncertainty in artificial intelligence*. Morgan Kaufmann Publishers Inc., 1998, pp. 211–219.