Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence (AAAI-16)

Detection of Plan Deviation in Multi-Agent Systems

Bikramjit Banerjee
School of Computing
University of Southern Mississippi
Hattiesburg, MS 39402
tBikramjit.Banerjee@usm.edu

Steven Loscalzo
AFRL Information Directorate
26 Electronic Parkway
Rome, NY 13441
Steven.Loscalzo@us.af.mil

Daniel Lucas Thompson
School of Computing
University of Southern Mississippi
Hattiesburg, MS 39402
Daniel.L.Thompson@eagles.usm.edu

Abstract

Plan monitoring in a collaborative multi-agent system requires an agent to not only monitor the execution of its own plan, but also to detect possible deviations or failures in the plan execution of its teammates. In domains featuring partial observability and uncertainty in the agents' sensing and actuation, especially where communication among agents is sparse (as a part of a cost-minimized plan), plan monitoring can be a significant challenge. We design an Expectation Maximization (EM) based algorithm for detection of plan deviation of teammates in such a multi-agent system. However, a direct implementation of this algorithm is intractable, so we also design an alternative approach grounded on the agents' plans, for tractability. We establish its equivalence to the intractable version, and evaluate these techniques in some challenging tasks.

Introduction

The problem of computing multi-agent plans, when the domain features partial observability and uncertainty in the agents' sensing and actuation, is computationally very challenging. This problem of planning in multi-agent systems has received significant attention in the past. However, the related problem of *multi-agent plan monitoring* has hardly garnered similar attention during the same time. Plan monitoring allows an agent to monitor the execution of its own plan, and detect possible failures or deviations that might entail the need to repair its plan, or replan. By contrast, a multi-agent system is particularly fragile to plan deviations, because one agent's deviation may affect other agents, and the debilitating effects may snowball quickly leading to catastrophic failures. Therefore in a collaborative multi-agent setting, an agent needs to detect any deviation or failure on part of its teammates. This activity is an important part of multi-agent plan monitoring. In a partially observable setting where agents cannot directly observe the actions of its teammates, it may be challenging for an agent to perform such monitoring. It must rely on its local observations and information to deduce the probability of deviation of its teammates from their part of the joint plans.

The problem of diagnosing multi-agent plan execution has been approached from various perspectives. Nearly two decades ago, Tennenholtz (1997) formulated the concept of *stable joint plans*, and proceeded to explore the computability and computational procedures for such plans. This was a mechanism design perspective of the planning problem with an additional constraint that plans must be such that any deviation by one agent is detectable by the other agents. This interesting initiative at the intersection of multi-agent planning and plan monitoring does not appear to have been pursued further in either literature. Later, Kalech and Kaminka (2003) introduced the idea of *social diagnosis*, using an abstract representation for multi-agent plans based on *behaviors*, to explain why agents selected conflicting behaviors. This approach adds communication among agents as a diagnostic tool, in addition to those required by the agents' plans. In view of the significant added communication complexity, Kalech and Kaminka also extended their approach for scalability (Kalech and Kaminka 2005). In contrast with social diagnosis, many authors have used more explicit representation of multi-agent plans in terms of agents' actions and resources. For instance, de Jonge, Roos and Witteveen (2009) introduced the notion of *plan diagnosis*—a subset of actions which when assumed to be executed anomalously make the plan execution consistent with the protagonist's observation. When the same observation sequence may be consistent with multiple plan diagnoses, the authors identify preferred diagnoses based on their predictive power. The diagnostic inference procedure is, however, largely centralized.

Through a series of papers, Micalizio and Torasso (2008; 2009; 2014), developed a relational framework with explicit representation of multi-agent plans and extended action models to include both nominal and faulty evolutions. A distinct advantage of this framework is that it is fully distributed and does not rely on synchronized action executions by the agents. However, it does impose cooperative communications among agents to share beliefs and mutually facilitate the diagnostic process. Similar to de Jonge, Roos and Witteveen (2009), Micalizio and Torasso also distinguish *primary failures* (original failures) from *secondary failures* (caused by primary failures), with an eye toward plan repair which typically targets primary failures.

In this paper, we derive a procedure for the detection of deviation in a teammate's plan in partially observable multi-agent systems, from first principles via *expectation maximization*. However, a direct implementation of this approach would be intractable, so we design an alternative (and tractable) implementation that is grounded on the agents' plans, called *controller based EM* (CBEM). Experimental evaluation in two domains—one existing and another new—not only establishes the relative tractability of CBEM, but also the validity of its inferences.

Our work is distinct from the existing literature in several ways. Firstly, we use explicit representation of multi-agent plans that is popular in the multi-agent planning community, rather than an abstract or explicit representation that is dictated by the needs of plan diagnosis. In other words, we wish to apply diagnosis to the execution of plans generated by off-the-shelf planners. Consequently, a second distinction is that we do not assume additional communication or collaboration among agents to facilitate diagnosis. Thirdly, we do not assume the availability of explicit and extended action models that may be tedious to encode. Fourthly, our approach is fully decentralized, allowing each agent to independently draw inferences about other agents' deviations. On the downside, we assume that agents' actions are indeed synchronous, although not necessarily simultaneous, or even of equal duration. We also treat diagnosis as an end rather than a means for (eventual) plan repair, and consequently, we do not distinguish primary failures from secondary ones.

Multi-agent Plans

We first describe the multi-agent planning domain—the substrate for computing and executing plans for a multi-agent team. Then we shall describe the plan representation of the agents. Together, they will serve as the basis for the mechanism of detection of plan deviation.

Multi-agent Planning

We consider the traditional multi-agent planning domain—featuring uncertainty and partial observability—given by a tuple $\langle n, S, A, P, R, \Omega, O, \beta_0\rangle$ where

- n is the number of agents.
- S is a finite set of environment states that are not directly observable.
- $A = \times_i A_i$ is the (product) set of joint actions, where A_i is the set of individual actions that agent i can execute.
- $P(s'|s, \boldsymbol{a})$ gives the probability of transition to state $s' \in S$ when joint action $\boldsymbol{a} \in A$ is executed in state $s \in S$.
- $R : S \times A \times S \to \mathbb{R}$, where $R(s, \boldsymbol{a}, s')$ gives the immediate reward the agents receive upon executing action $\boldsymbol{a} \in A$ in state $s \in S$, and transitioning to state $s' \in S$.
- $\Omega = \times_i \Omega_i$ is the (product) set of joint observations, where Ω_i is the finite set of individual observations that agent i can receive from the environment.
- $O(\boldsymbol{\omega}|s', \boldsymbol{a})$ gives the probability of the agents jointly observing $\boldsymbol{\omega} \in \Omega$ if the current state is $s' \in S$ and the previous joint action was $\boldsymbol{a} \in A$.
- β_0 is the initial state belief.

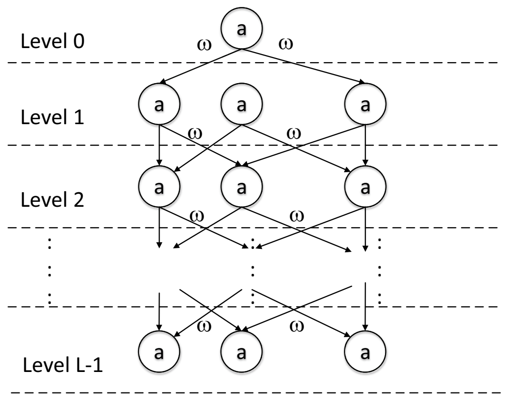

Figure 1: A layered controller/policy. "a" represents action, while "ω" represents observation.

The reward function R, transition model P, and observation model O are defined over joint actions and/or observations, which forces the agents to coordinate. A problem *horizon*, notated as T, is often specified in a planning problem to indicate the number of interactions that the agents are going to have in the shared environment. The goal of planning is to find a policy for each agent (i.e., a *joint policy*) that maximizes the total expected reward over T steps, given that the agents cannot communicate their observations and actions to each other. Note that if communication is possible but comes at a cost, then the communicative actions can be included in A and their costs can be included in the R function to make communication a part of the optimization problem.

We assume that the planning problem has been solved and the agent policies computed. We also assume that agents are not only given their own policies for execution, but also the policies of their teammates to enable deviation detection.

Plan/Policy/Controller Representation

We consider multi-agent policies that map the observation history of an agent to its (next) action. Thus the policy of agent i is $\pi_i : \mathcal{O}_i \mapsto A_i$. For finite horizon problems with horizon T, the policy may map any observation history of any length $t < T$, $o_i^t \in \mathcal{O}_i$. In practice, however, an agent's policy may be represented as a deterministic finite state controller (FSC) in problems with large, indefinite, or infinite horizon. In such cases, each observation history uniquely maps to a controller (FSC) state/node which is associated with an action. Therefore, FSC policies also map observation histories to actions, and our definition of π_i applies to such policies as well. We do not consider stochastic FSCs in this paper as they themselves typically require optimization over stochastic transitions, while deterministic FSCs are often much simpler to hand-code for a given task.

In this paper, we further limit focus to a class of deterministic FSCs, called *layered controllers*. In such a controller, states/nodes are organized in a set of L layers or levels, with a fixed number of nodes per layer. The node transition function (based on an agent's observations) is constrained to only map nodes in layer ℓ to nodes in layer $\ell + 1$,

$\ell = 0, \ldots, L-2$. Figure 1 shows an example of a layered FSC. Such controllers are quite common in the multi-agent planning literature (Seuken and Zilberstein 2007; Pajarinen and Peltonen 2011). We assume that the problem horizon T does not exceed L.

Detection of Deviation

We first establish some notations and conventions used in this paper. Let θ_i^t represent the action-observation history of length t of agent i, i.e., $\theta_i^t = \langle a_i^0, \omega_i^1, a_i^1, \omega_i^2, \ldots, a_i^{t-1}, \omega_i^t \rangle$. Let o_i^t represent the observation (only) history of length t, i.e., $o_i^t = \langle \omega_i^1, \omega_i^2, \ldots, \omega_i^t \rangle$. Also let $\phi : \Theta \mapsto \mathcal{O}$ be a function that extracts the observation-only history from $\theta_i^t \in \Theta_i$ and return $o_i^t \in \mathcal{O}_i$ for any agent i.

For notational convenience, we consider two-agent systems, agent i and agent j. Agent i is the protagonist, who wishes to compute the probability of plan deviation by other agents (but itself doesn't deviate from its plan). All other agents in the system are bundled together into a single (virtual) agent j. Then agent i's problem is to infer what history θ_j^T may have been observed by agent j, given that it has observed θ_i^T itself, after T steps of plan execution by both agents.

This problem can be posed as a *filtering* problem, i.e., the problem of (hidden) state estimation where $\langle s^t, a_j^t, \omega_j^t \rangle$ is considered the hidden state, and $\langle a_i^t, \omega_i^t \rangle$ is the observation at step t. The *forward backward algorithm* (Rabiner 1989) is a common approach; however this algorithm gives state estimations that are temporally optimized at individual steps, rather than optimized together. We are interested in estimating θ_j^T where a_j^t and $a_j^{t'}$ $(t, t' < T)$ are correlated. Consequently, the more appropriate approach for this problem is *sequence estimation*, which returns an optimized sequence of hidden states, given the observation sequence.

A popular algorithm for sequence estimation is the *Viterbi algorithm* (Viterbi 1967). This algorithm assumes the knowledge of a transition model guiding the evolution of the hidden state. While such models are available for s^t and ω_j^t, the only model that agent i has about a_j^t is the policy π_j^t (which is known to agent i). However, our precise interest is in the scenario where agent j *may deviate* from π_j^t. To solve this problem, we introduce an additional parameter δ_t, which represents the probability that agent j will execute the action dictated by its policy π_j at step t. We further assume that the probability mass of $(1-\delta_t)$, corresponding to deviation from π_j, is *uniformly* distributed over all actions that violate π_j at step t, i.e., probability $\frac{1-\delta_t}{|A_j|-1}$ for each such action.

There is yet another roadblock to exploiting the existing approaches for sequence estimation, however. Since we know nothing about δ_t, we would like to assume some reasonable prior, and be able to compute the posterior. Sequence estimation would output an optimized sequence of $\langle s^t, a_j^t, \omega_j^t \rangle$, but would not give a straightforward way to update δ_t. Consequently, we turn to *Expectation Maximization* for that purpose, as described later in the next section.

Our Approach

The likelihood of joint action-observation history $\boldsymbol{\theta}^t = \langle \theta_i^t, \theta_j^t \rangle$, with the system state being s^t at step t of policy execution, is given by the recurrence relation

$$\mathbf{Pr}(s^t, \boldsymbol{\theta}^t) = \sum_{s^{t-1} \in S} \mathbf{Pr}(s^{t-1}, \boldsymbol{\theta}^{t-1}) P(s^t, \boldsymbol{\omega}^t | s^{t-1}, \boldsymbol{a}^{t-1}) \cdot \mathbf{Pr}(\boldsymbol{a}^{t-1} | \boldsymbol{\theta}^{t-1}) \quad (1)$$

where $P(s^t, \boldsymbol{\omega}^t | s^{t-1}, \boldsymbol{a}^{t-1})$ is the shorthand notation for the product $P(s^t | s^{t-1}, \boldsymbol{a}^{t-1}) \cdot O(\boldsymbol{\omega}^t | s^t, \boldsymbol{a}^{t-1})$. Further, $\boldsymbol{a}^{t-1} = \langle a_i^{t-1}, a_j^{t-1} \rangle$ and $\boldsymbol{\omega}^t = \langle \omega_i^t, \omega_j^t \rangle$ are such that $\boldsymbol{\theta}^t = concat(\boldsymbol{\theta}^{t-1}, \boldsymbol{a}^{t-1}, \boldsymbol{\omega}^t)$. The base condition is $\mathbf{Pr}(s^0, \boldsymbol{\theta}^0) = \beta_0(s^0)$, the initial belief. The action likelihood $\mathbf{Pr}(\boldsymbol{a}^{t-1} | \boldsymbol{\theta}^{t-1}) = \prod_i \mathbf{Pr}(a_i^{t-1} | \theta_i^{t-1})$, where the protagonist (agent i) is assumed to execute his policy without deviation:

$$\mathbf{Pr}(a_i^{t-1} | \theta_i^{t-1}) = \begin{cases} 1 & \text{if } \pi_i(\phi(\theta_i^{t-1})) = a_i^{t-1} \\ 0 & \text{otherwise.} \end{cases}$$

Furthermore, based on the parametrization using δ_t that we introduced in the previous section, we also have

$$\mathbf{Pr}(a_j^{t-1} | \theta_j^{t-1}) = \begin{cases} \delta_{t-1} & \text{if } \pi_j(\phi(\theta_j^{t-1})) = a_j^{t-1} \\ \frac{1-\delta_{t-1}}{|A_j|-1} & \text{otherwise} \end{cases}$$

From the joint probability in equation 1, we get

$$\mathbf{Pr}(\boldsymbol{\theta}^t) = \sum_{s^t} \mathbf{Pr}(s^t, \boldsymbol{\theta}^t).$$

Next, we introduce the notion of *consistent histories*. Let Θ_j be the set of all possible action-observation histories of any length for agent j. Given the policy of agent j, π_j, we define the set of histories consistent at step t with π_j, as a subset $\Theta_j^{\pi,t} \subset \Theta_j$ such that any $\theta_j \in \Theta_j^{\pi,t}$ must satisfy one of the two following conditions:

- $|\theta_j| < 2t$, i.e., there are fewer than t steps of actions embedded in θ_j, or
- $|\theta_j| \geq 2t$, but $\pi_j(\phi(\theta_j^{t-1})) = a_j^{t-1}$ (where θ_j^{t-1} is the $2(t-1)$-length prefix of θ_j) and the concatenation of θ_j^{t-1} and a_j^{t-1} is also a prefix of θ_j. In other words, the action embedded in θ_j following the prefix θ_j^{t-1} agrees with π_j's prescribed action after observing $\phi(\theta_j^{t-1})$.

In words, histories that are consistent at step t with a policy, are either too short, or are long enough to suggest that there was no deviation at step t from that policy. Now we are ready to formulate a procedure for the refinement of δ_t, using the EM (meta) algorithm.

Expectation Maximization

When $\mathbf{Pr}(a_i^{t-1} | \theta_i^{t-1})$ and $\mathbf{Pr}(a_j^{t-1} | \theta_j^{t-1})$ are substituted into equation 1, we can rewrite $\mathbf{Pr}(\boldsymbol{\theta}^T)$ as

$$\mathbf{Pr}(\boldsymbol{\theta}^T) = c_{\theta_j^T} \cdot \left(\prod_{k \in N} \delta_k \right) \cdot \left(\prod_{k \in D} \frac{1-\delta_k}{|A|_j - 1} \right) \quad (2)$$

where $N \subseteq \{0, \ldots, T-1\}$ is the set of steps where θ_j^T follows the policy π_j (i.e., $\theta_j^T \in \Theta_j^{\pi,k}, \forall k \in N$), and D is the set of steps where θ_j^T deviates from π_j (i.e., $\theta_j^T \notin \Theta_j^{\pi,k}, \forall k \in D$). Note $|N| + |D| = T$. The coefficient $c_{\theta_j^T}$ is the collection over other terms from equation 1, and is a function of θ_j^T for a given θ_i^T, but is independent of the parameters δ_k.

When agent i observes θ_i^T, the hidden data is θ_j^T, and the (initial) parameter vector is $\boldsymbol{\delta}^0 = \langle \delta_0^0, \ldots, \delta_{T-1}^0 \rangle$. We now define the two key steps of the EM algorithm as follows:

Expectation/Estimation step: In this step, the standard procedure is to calculate

$$E[\ln \mathbf{Pr}(\boldsymbol{\theta}^T | \boldsymbol{\delta}^{h+1}) | \boldsymbol{\delta}^h, \theta_i^T].$$

The above expectation is taken relative to the distribution over the hidden data θ_j^T induced by the h-th parameter hypothesis $\boldsymbol{\delta}^h$, while $\mathbf{Pr}(\boldsymbol{\theta}^T)$ is expressed in terms of the next hypothesis $\boldsymbol{\delta}^{h+1}$. By equation 2, the above expression reduces to

$$E[\ln c_{\theta_j^T}] + \sum_{k \in N, \theta_j^T \in \Theta_j^{\pi,k}} \ln(\delta_k^{h+1}) \mathbf{Pr}(\theta_j^T | \theta_i^T, \boldsymbol{\delta}^h) + \sum_{k \in D, \theta_j^T \notin \Theta_j^{\pi,k}} \ln\left(\frac{1 - \delta_k^{h+1}}{|A_j| - 1}\right) \mathbf{Pr}(\theta_j^T | \theta_i^T, \boldsymbol{\delta}^h)$$

Maximization step: In this step we maximize the above expectation with respect to $\boldsymbol{\delta}^{h+1}$, which yields

$$\delta_k^{h+1} = \sum_{\theta_j^T \in \Theta_j^{\pi,k}} \mathbf{Pr}(\theta_j^T | \theta_i^T, \boldsymbol{\delta}^h), \ for\ k \in [0, T-1]. \quad (4)$$

Repetition of the above two steps is guaranteed to converge to the local maximum likelihood hypothesis $\boldsymbol{\delta}^\infty(\theta_i^T) = \langle \delta_0^\infty(\theta_i^T), \ldots, \delta_{T-1}^\infty(\theta_i^T) \rangle$, *for any given* θ_i^T. In other words, given an observation θ_i^T, the EM procedure gives the optimized (for θ_i^T) likelihood that agent j has deviated in any of the steps $k = 0, \ldots, T-1$, by calculating $1 - \delta_k^\infty(\theta_i^T)$.

The above procedure utilizes $\mathbf{Pr}(\boldsymbol{\theta}^T)$, and ultimately equation 1. Unfortunately, this has a complexity of $O(|A_j|^T |\Omega_j|^T |S|^{T+1})$ for each iteration of EM, since every possible history of agent j—whether consistent with π_j, or deviated from π_j—must be taken into account. In the next section, we develop an alternative approach to calculate equation 4 more tractably.

Controller Based EM

We have shown in the last section that grounding equation 1 on histories led to a complexity of $\boldsymbol{\delta}$ update that is exponential in the size of a history. In this section, we attempt to ground equation 1 on the controller rather than histories. It is not immediately clear whether the resulting procedure for $\boldsymbol{\delta}$ might be exponential in the size of the controller. We shall answer this question in the negative, and show that this change allows us to compute equation 4 in time *polynomial in the size of the controller*.

Let Q_i^ℓ be the set of nodes in agent i's layered FSM controller at level $\ell = 0, \ldots, L-1$, and $\lambda : \mathcal{O}_i \to Q_i$ be a function that maps its observation history to a node in Q_i. This node is unique because the controller only allows deterministic transitions. We represent the action that agent i must execute when it reaches node $q_i \in Q_i$ as $a(q_i)$.

Under the assumption that agent i has observed θ_i^T, we define $\rho(k, s, \boldsymbol{q}^\ell)$ as the probability that the agents reach joint node $\boldsymbol{q}^\ell = \langle q_i^\ell, q_j^\ell \rangle$ at level ℓ in their controllers and state s, without agent j deviating at level k, where $\ell = 0, \ldots, T$. Note that ℓ may be $\geq k$ or $< k$. We compute ρ recursively as shown in equation 3, where $\boldsymbol{q}^\ell = \langle q_i^\ell, q_j^\ell \rangle$ with $q_i^\ell = \lambda(\phi(\theta_i^\ell))$, and likewise for q_j^ℓ. Similarly, $\boldsymbol{\omega}^{\ell+1} = \langle \omega_i^{\ell+1}, \omega_j^{\ell+1} \rangle$, where the sum over $\omega_j^{\ell+1}$ in equation 3 is taken over all observations in $\Omega_j^{\ell+1} \subseteq \Omega_j$ that enable transition from q_j^ℓ to $q_j^{\ell+1}$. Furthermore, $\boldsymbol{a}^\ell = \langle a(q_i^\ell), a(q_j^\ell) \rangle$, but $\boldsymbol{\alpha}^\ell = \langle a(q_i^\ell), \alpha_j \rangle$ where $\alpha_j \neq a(q_j^\ell)$.

The function ρ is related to equation 1 in a simple manner as given in the following Lemma (proof omitted for space limitation).

Lemma 1 *For any* $\boldsymbol{\theta}^t = \langle \theta_i^t, \theta_j^t \rangle$ *such that* $\theta_j^t \in \Theta_j^{\pi,k}$:

$$\boldsymbol{Pr}(s^t, \boldsymbol{\theta}^t) = \rho(k, s^t, \boldsymbol{q}^t)$$

where $\boldsymbol{q}^t = \langle q_i^t, q_j^t \rangle$ *such that* $q_i^t = \lambda(\phi(\theta_i^t))$, $q_j^t = \lambda(\phi(\theta_j^t))$.

Based on Lemma 1, it is fairly straightforward to show the main result of this section, given as the following Theorem.

Theorem 1 *After observing* θ_i^T, *agent i can compute* δ_k^{h+1} *as*

$$\delta_k^{h+1} = \frac{\sum_{q_j^T, s} \rho^h(k, s, \boldsymbol{q}^T)}{\sum_{q_j^T, s} \rho^h(T, s, \boldsymbol{q}^T)}$$

for $k = 0, \ldots, T-1$.

Proof (sketch): Using Lemma 1, $\sum_{q_j^T, s} \rho^h(k, s, \boldsymbol{q}^T) = \sum_{\lambda(\phi(\theta_j^T)) | \theta_j^T \in \Theta_j^{\pi,k}} \mathbf{Pr}(\boldsymbol{\theta}^T)$, which simplifies to $\sum_{\theta_j^T \in \Theta_j^{\pi,k}} \mathbf{Pr}(\boldsymbol{\theta}^T)$. Also note that the last (joint) action in $\boldsymbol{\theta}^T$ is $\boldsymbol{a}^{T-1}$, so that $\boldsymbol{a}^T$ is beyond the horizon. Therefore, $\rho^h(T, s, \boldsymbol{q}^T)$ does not impose non-deviation at any of the steps $k = 0, \ldots, T-1$, and thus equates to $\sum_{\theta_j^T \in \Theta_j} \mathbf{Pr}(\boldsymbol{\theta}^T)$. Hence the right hand side in the above theorem is indeed $\sum_{\theta_j^T \in \Theta_j^{\pi,k}} \mathbf{Pr}(\theta_j^T | \theta_i^T, \boldsymbol{\delta}^h)$.

We call the procedure where Theorem 1 is used for each EM iteration, CBEM (Controller Based EM). Theorem 1 provides an alternative to equation 4, and is now $O(N^2 T^2 |S|^2 |A_j| |\Omega_j|)$, where N is the number of nodes per layer of agent j's controller. Thus this approach has a complexity polynomial in the size of the controller (which is $O(NL|\Omega_j|)$), and is much more scalable than equation 4. However, it is important to note $|A_j|$ itself can be exponential in n (number of agents) when j is a virtual agent representing the product space of all agents other than i. Scalability in terms of n usually requires that the transition (P) and observation (O) distributions be factored using some prior

$$\boxed{\rho^h(k,s',\boldsymbol{q}^{\ell+1}) = \begin{cases} \sum_{s,q_j^\ell,\omega_j^{\ell+1}} \rho^h(k,s,\boldsymbol{q}^\ell)\left[P(\boldsymbol{\omega}^{\ell+1},s'|s,\boldsymbol{a}^\ell)\delta_\ell^h + \sum_{\boldsymbol{\alpha}^\ell} P(\boldsymbol{\omega}^{\ell+1},s'|s,\boldsymbol{\alpha}^\ell)\frac{1-\delta_\ell^h}{|A_j|-1}\right] & \text{if } k \neq \ell \\ \sum_{s,q_j^\ell,\omega_j^{\ell+1}} \rho^h(k,s,\boldsymbol{q}^\ell)P(\boldsymbol{\omega}^{\ell+1},s'|s,\boldsymbol{a}^\ell)\delta_\ell^h & \text{otherwise} \end{cases}} \quad (3)$$

knowledge about the dependence relations among agents, which we do not pursue in this paper.

Evaluation

We evaluate CBEM in two domains: *multi-agent tiger* (Nair et al. 2003), and *drone search*—a new domain that we introduce here. First, we use the small domain of multi-agent tiger to verify the speed gain of CBEM over EM. Then we describe and use the more complex domain of drone search to analyze the output of CBEM in two handcrafted scenarios, and demonstrate its effectiveness.

Multi-agent Tiger

In multi-agent tiger there are two agents, each deciding whether to open one of two doors, one of which hides a tiger while the other conceals treasure. An agent can listen for the tiger's growl instead of opening a door, and this gives it a noisy clue about the tiger's location. In this domain, $|S| = 2$, $|\Omega_{i/j}| = 2$, $|A_{i/j}| = 3$. More details of the planning domain parameters can be found in (Nair et al. 2003). We ran both EM and CBEM using known optimal policies for this domain, for horizons $T = 3, \ldots, 7$. For the choice of θ_i^T, we used every possible θ_i^T. The runtimes of EM and CBEM (in secs), averaged over the choices of θ_i^T (i.e., per history of agent i), are reported in Figure 4 (left). This verifies the relative scalability of CBEM, as claimed in the previous section.

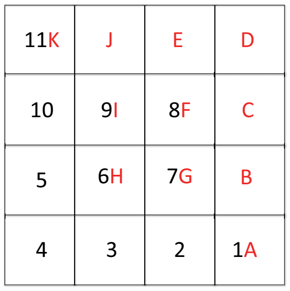

Figure 2: The Drone Search domain. Black numbers represent the successive locations of Drone 1, and red letters that of Drone 2.

Drone Search

We introduce the main evaluation domain of this paper, *Drone Search*, situated in a 4×4 grid, shown in Figure 2. Two agents/drones take off from a base at the lower right corner (1A) and follow different path plans (shown in black numbers for Drone 1 and red letters for Drone 2) to end in a refueling station at the upper left corner. Along their paths,

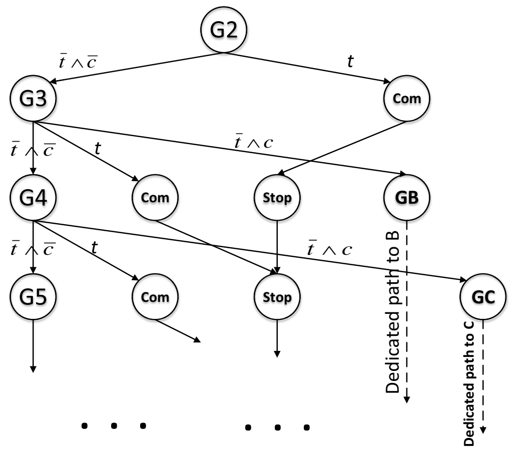

Figure 3: A part of the layered controller used for Drone 1. Here "Gx" represents the action "Go to x". "Com" represents the action of communicating to the other drone that the target has been found. Drone 2 has a similar controller.

the drones essentially perform a distributed search for a single, stationary target, which can be located in any of the 15 cells (excluding the start cell). The search paths of the drones cover exclusive territory for the most part, although their search territories overlap at 1A, 6H, 7G, 9I, 8F and 11K. Furthermore, they occupy the diagonal cells (1A, 7G, 9I and 11K) *simultaneously*. We assume that the drones cannot observe each other even when they co-occupy the same cell. If a drone co-occupies the target cell then it is assumed that the drone detects the target (observation t), and then it must send a signal to the other drone (communication "Com", which can be perceived by the other drone at the next step as observation c) and stop. If a drone receives a communication from the other drone, but has not itself seen the target, then it infers the location of the other drone where it must have seen the target, and proceeds toward that inferred location. Thus, ideally, both drones end up in the location of the target, for further joint surveillance. To keep it simple, we assume that the planning domain functions O, P are deterministic. Figure 3 shows a part of the manually designed layered controller used for Drone 1. This domain has $|S| = 256$ joint drone locations. Each drone can move in one of the 4 cardinal directions, or send a communication, or stop, for a total of $|A_{i/j}| = 6$ actions. The number of observations is $|\Omega_{i/j}| = 4$, corresponding to the combinations $\bar{t} \wedge \bar{c}$, $\bar{t} \wedge c$, $t \wedge \bar{c}$, and $t \wedge c$. Furthermore, $T = 10$, which amounts to $(256 \cdot 6 \cdot 4)^{10}$ operations for raw EM. This is well beyond the capability of any personal computer existing today.

We create the two following scenarios for the evaluation of CBEM in this domain. For each scenario, we use initial

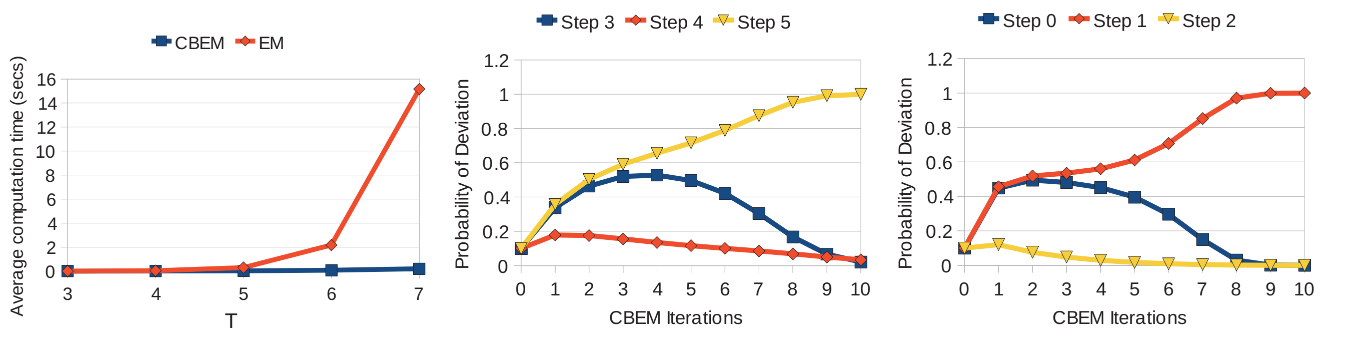

Figure 4: Left: Per history (θ_i^T) runtime of CBEM and EM on Multi-agent Tiger for horizons $T = 3, \dots, 7$. Middle: $(1 - \delta_k^h)$ plots for $k = 3, 4, 5$ and iterations $h = 0, \dots, 10$, in Drone Search Scenario 1. Right: $(1-\delta_k^h)$ plots for $k = 0, 1, 2$ and iterations $h = 0, \dots, 10$, in Drone Search Scenario 2.

$\delta_k^0 = 0.9$, $k = 0, \dots, 9$.

Scenario 1 (Communication failure) : In this scenario, we place the target in cell 8F. Although it is in overlapping search territory, Drone 2 should reach the target earlier (level 4 of its controller) than Drone 1. Suppose Drone 2 fails to communicate (at level 5 of its controller) to Drone 1 that it has seen the target at 8F, so Drone 1 does not receive a c observation. Eventually when it reaches 8F, it finds the target there, sends a "Com" signal, and stops. Thus Drone 1's observation sequence is $\bar{t} \wedge \bar{c}$ until after level 6. Figure 4 (middle) shows the plots of Drone 1's inference of $(1 - \delta_k^h)$ values for levels $k = 3, 4, 5$ against h (CBEM iterations). To reduce clutter, this plot does not show levels $k = 0, 1, 2, 6-9$, which have a similar pattern as level 4. The plot shows that Drone 1 does indeed believe that Drone 2 failed to execute "Com" at level 5. While alternative hypotheses about navigational deviation at other levels have some weight temporarily at the beginning, the probabilities of these hypotheses diminish within 10 iterations.

Scenario 2 (Navigation failure) : In this scenario, we place the target in cell 7G. We assume that Drone 2 encounters a navigational failure at level 1 and instead of travelling from B to C, goes west to 7G. There it sees the target, sends a "Com" to Drone 1 (at level 2), and stops. Drone 1 receives the communication, at which point it is located in cell 4. Drone 1 infers that Drone 2 must have seen the target in cell C, and hence proceeds toward cell C along the route $4 \rightarrow 5 \rightarrow 6H \rightarrow 7G \rightarrow B \rightarrow C$. However, when it reaches cell 7G, it sees the target too, earlier than expected. Figure 4 (right) shows the plots of Drone 1's inference of $(1 - \delta_k^h)$ values for levels $k = 0, 1, 2$. To reduce clutter, this plot does not show levels $k = 3 - 9$, which have a similar pattern as level 2. It is evident that Drone 1 believes Drone 2 failed to travel from cell B to C at level 1, rather than believing that Drone 2 sent a false "Com" signal at level 2.

Note that in each of the above scenarios there are potential alternative explanations. For instance, in Scenario 2 it is possible that Drone 2 sent a false "Com" signal at level 2. But the hypothesis that it performed a "Com" action in a navigation node of its controller has a lower probability ($\frac{1-\delta}{|A_j|-1}$) than doing a "Com" action in its "Com" node (δ), and this difference is reinforced by the fact that Drone 1 indeed received a c signal out of level 2. As a result, the hypothesis that there was a navigation failure before level 2 is preferred over the false "Com" hypothesis at level 2. Similarly in Scenario 1, the hypothesis that Drone 2 simply failed to send "Com" at level 5 is preferred over multiply fractured hypotheses of navigational failures at earlier levels. In general, (CB)EM prefers simple explanations involving brief deviations over more complex patterns of multiple step deviations.

In Scenario 2, there is an equally valid alternative explanation to navigation failure at level 2: rather than taking the path $1A \rightarrow B \rightarrow 7G$, Drone 2 might have taken path $1A \rightarrow 2 \rightarrow 7G$, thus deviating at level 0 rather than at level 1. Figure 4 (right) shows that both of these hypotheses had roughly equal probability (≈ 0.5) for the first two iterations of CBEM. However, one hypothesis had a slightly higher weight that acted as a perturbation to break symmetry, with the dynamics eventually selecting one decisively.

The controllers that we designed for the Drones (one of which is partly shown in Figure 3) do not meet Tennenholtz's criteria for *stable joint plans* (Tennenholtz 1997), although they come close. Some deviations of Drone 2 are, in fact, completely undetectable to Drone 1. For instance, if the target was at 8F, and Drone 2 stayed at C for two steps before moving west to 8F directly, thus skipping cells D and E, then Drone 1's observations would be no different than if Drone 2 had followed its path plan.

Conclusions and Future Work

We have developed an algorithm, and its tractable implementation, for the detection of plan deviation by teammates in a multi-agent system. Evaluation of our approach demonstrates a preference for simpler deviation hypotheses. One question that could spur future research is how to construct the priors, $\boldsymbol{\delta}^0$, or is it possible to perform this inference in a prior-free way? Some limitations of the approach include the inability to infer which other agent deviated, or what action it executed instead of its policy-action, and the assumption that $T < L$. In fact, most finite state controllers are designed for infinite horizon plan execution, therefore this last assumption deserves work in the immediate future.

Acknowledgments

We thank the anonymous reviewers for helpful comments and suggestions. This work was supported in part by an AFOSR Summer Faculty Fellowship awarded to Bikramjit Banerjee in 2014.

References

de Jonge, F.; Roos, N.; and Witteveen, C. 2009. Primary and secondary diagnosis of multiagent plan execution. *Journal of Autonomous Agent and Multiagent Systems* 18(2):267–294.

Kalech, M., and Kaminka, G. 2003. On the design of social diagnosis algorithms for multi-agent teams. In *Proc. International Joint Conference on Artificial Intelligence (IJCAI-03)*, 370–375.

Kalech, M., and Kaminka, G. 2005. Diagnosing a team of agents: Scaling up. In *Proc. of International Conference on Autonomous Agents and Multi-Agent Systems (AAMAS-05)*, 249–255.

Micalizio, R., and Torasso, P. 2008. Monitoring the execution of a multi-agent plan: Dealing with partial observability. In *Proceedings of the 2008 Conference on ECAI 2008: 18th European Conference on Artificial Intelligence*, 408–412. Amsterdam, The Netherlands, The Netherlands: IOS Press.

Micalizio, R., and Torasso, P. 2009. Agent cooperation for monitoring and diagnosing a map. In *Proc. of Multiagent System Technologies (MATES' 09)*, volume LNCS: 5774, 66–78.

Micalizio, R., and Torasso, P. 2014. Cooperative monitoring to diagnose multiagent plans. *Journal of Artificial Intelligence Research* 51:1–70.

Nair, R.; Tambe, M.; Yokoo, M.; Pynadath, D.; and Marsella, S. 2003. Taming decentralized POMDPs: Towards efficient policy computation for multiagent settings. In *Proceedings of the 18th International Joint Conference on Artificial Intelligence (IJCAI-03)*, 705–711.

Pajarinen, J., and Peltonen, J. 2011. Periodic Finite State Controllers for Efficient POMDP and DEC-POMDP Planning. In *Proceedings of the 25th Annual Conference on Neural Information Processing Systems (NIPS)*, 2636–2644.

Rabiner, L. R. 1989. A tutorial on hidden markov models and selected applications in speech recognition. In *Proceedings of the IEEE*, 257–286.

Seuken, S., and Zilberstein, S. 2007. Memory-bounded dynamic programming for Dec-POMDPs. In *Proceedings of the 20th International Joint Conference on Artificial Intelligence (IJCAI-07)*, 2009–2015.

Tennenholtz, M. 1997. On stable multi-agent behavior in face of uncertainty. In *Proc. UAI*, 445–452.

Viterbi, A. 1967. Error bounds for convolutional codes and an asymptotically optimum decoding algorithm. *IEEE Transactions on Information Theory* 13(2):260–269.