Planning for Actively Synchronized Multi-Robot Systems

Patrick Zhong[1], Federico Rossi[2] and Dylan A. Shell[1]

[1] Texas A&M University, College Station TX 77840, USA.
{patrickzhong|dshell}@tamu.edu
[2]Jet Propulsion Laboratory, California Institute of Technology, Pasadena, CA 91109, USA. federico.rossi@jpl.nasa.gov

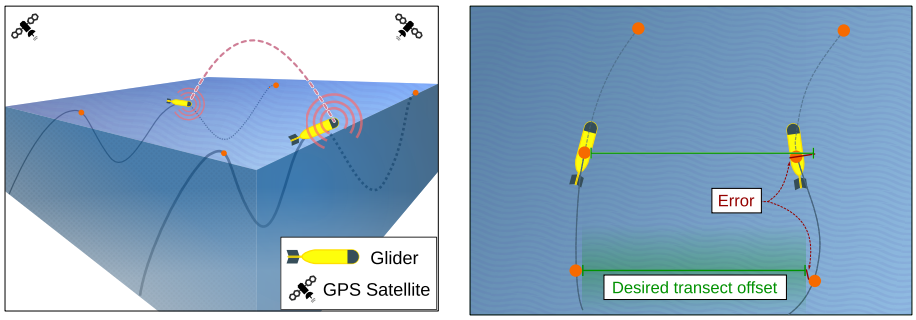
Abstract. We tackle planning under uncertainty when multiple robots must proactively plan perception and communication acts, and decide whether the cost needed to obtain a state estimate is justified by the benefit of the information obtained. The approach is suitable when observations are costly but, when they do occur, are of high quality and recover the system's joint state, either alone or along with communication. Such cases allow one to sidestep the construction of the full joint belief space, a well known source of intractability in planning. Formulating the problem as a class of Markov decision processes to be solved over joint states and structured to allow decentralized execution, we give a suitable Bellman recurrence using macro-actions. We solve for policies for the individual robots, providing a simulation case study and reporting on a physical robot implementation. Based on our experience with hardware, we examine some non-idealities identified in practice, proffering suitable enhancements to the basic model.

1 Introduction

A fundamental challenge in dealing with a typical distributed system is that the state of the overall system is not known to all the constituents at all points in time. In multi-robot systems, this challenge is particularly vexing owing to uncertainty, born of imperfect actuation and sensing, and intermittent or unreliable communication. Treating the whole multi-robot system jointly, as within a centralized model, usually imposes constraints that are either utterly unrealistic, or impracticably onerous to meet. Yet, fully general solutions that model local beliefs about the global state are very costly. Oftentimes prohibitively so.

This paper studies a class of multi-robot systems where information is obtained actively: the robots in the system plan and, through fully distributed execution, realize a *jointly endogenous* observation process. Observations are either direct simultaneous measurements of joint system state, or are sufficient when combined via communication. The system can be considered 'synchronized' when the joint state is known by all; at such junctures the system must also decide, as a function of the observed state, when the next observation should take place. And because such sensing and communication can be costly, the process of optimization may result in sparse observations.

Acknowledgement: Research supported by ONR under Award #N00014-22-1-2476. Part of this research was carried out at the Jet Propulsion Laboratory, California Institute of Technology, under a contract with the National Aeronautics and Space Administration (80NM0018D0004).



(a) A pair of gliders making measurements along two parallel transects of ocean.

(b) Control uncertainty leads to error, visible in this a bird's eye view.

Fig. 1: Ocean sampling example. A pair of underwater gliders are tasked with making near-simultaneous measurements with a pre-defined spatial offset within a region of interest, employing the characteristic sinking/rising pattern of locomotion. The devices can communicate and obtain reliable position estimates only once surfaced (depicted in orange). Observations are costly and, as varying conditions affect the reliability of their dead-reckoning, they determine when to check-in next based on their relative error.

Concrete Example 1 (Ocean sampling)**.** Consider the situation depicted in Figure 1 wherein a pair of underwater gliders sample a transect of ocean by tracing parallel tracks, with a pre-defined offset distance between them. The figure shows how these devices move: they dive and float by adjusting their buoyancy, and use hydrofoils to turn vertical forces into translational movement. Individual gliders only have access to precise pose measurements in the global reference frame at the surface, and challenges imposed by underwater attenuation mean that communication is all but infeasible except at the surface. The gliders' predominant mode of operation involves being sensing- and communication-deprived, punctuated by intermittent moments of high-quality state sensing and communication. Once surfaced, they determine how much their separation has deviated from the desired offset, what actions will help correct this error, and —more subtly— how deep to dive before resurfacing again. Given the energy required to use the GPS and communication hardware, longer sequences with deeper dives between re-surfacing represent an energy saving. On the other hand, sparse pose estimates mean greater between-track error, potentially tainting the gathered data. ◊

Note how using the full state, the system decides *when* next to sense state, balancing the cost of reducing uncertainty against the value that information provides. While there is uncertainty in the motion dynamics and the observations will likely be sparse, the sensors provide high-quality estimates of state and, at specific junctures, communication is available to reconstruct joint state. The contribution of this paper is the identification and specialized study of such multi-robot planning problems as a specific class of Markov decision processes (MDPs): this includes the formalization via two orthogonal species of composition (macro–joint actions are formed that aggregate across agents and across time), as well as experimental examination of solutions, and insights derived therefrom.

As we explain, the multi-robot case requires careful consideration in selecting reference frames, action representations and maximum strides, and in assessing the impact of observation non-idealities (asymmetry and delay).

2 Related work

Observations that are sparse can be treated via a partially-observed Markov decision process (POMDP), and when dealing with multiple agents there exists work on decentralized POMDPs [1]. These approaches, reviewed recently in [2], seek approximate search algorithms to handle the enormous space of beliefs arising from partial observability and the combinatorial nature of the joint-action space in DEC-POMDPs. The generality of the POMDP formulation overlooks structure ripe for exploitation: our formulation has no explicit representation of beliefs as, at the synchronization points, beliefs project back to the state space. Further, in contrast, we can compute exact solutions under our model.

Also within the literature, decentralized MDPs utilize local observations from distributed agents, such as in [3]. In that model, as also in our case, when observations arrive their union provides the joint system state. Unlike that model, we are considering sparse observations and must decide when observations ought next to occur. Worth further mention, a set of approaches consider decentralized reinforcement learning; one recent example having agents communicate intent appears in [4]. Those authors employ sampling to learn the transition model, whereas our approach is model based.

The work of [5] optimizes communication policies, selecting observations to share between agents in an online fashion. Also, decentralized planning via tree search has been effectively used to improve multi-robot active perception [6].

Although it never considers multiple agents, a different method of limiting sensor use that allows for dynamic observations is the use of MDPs for self-triggered control [7] wherein a subpolicy determines when sensor inputs will be needed next. Their approach, however, relies on the reduction of the action space to time-homogeneous macro-actions which constantly repeat a single action. Recent work of [8] suggests this may be expanded to a limited set of time-inhomogeneous macro-actions, but they are to be prescribed *a priori*.

3 Problem treatment

Our development will use two forms of composition: sequential and parallel. We introduce the former to model sparse observations subject to costs. Then, in Section 3.4, parallel composition leads to Definition 2 wherein multi-agency is brought into the mix. Owing to the concurrent use of both species of composition, the former employs subscript notation, the latter, superscripts.

3.1 Preliminaries

We assume the reader is familiar with a standard (centralized) Markov decision process; a comprehensive reference is [9]. In the classic discrete time model, at each stage, some observation discloses the state of the underlying Markov process and the decision maker chooses an action and the process evolves. This repeats with the decision maker minimizing some cost; we treat the usual infinite horizon case with cumulative discounted costs.

3.2 Scheduling and macro-action composition

Consider a decision maker who pays an additional cost to receive each observation. When that cost is substantial, the decision maker might choose to forgo the information obtained and instead take some actions blindly. For instance, the observation cost needs only to be paid half as often if actions are selected in pairs. The balance to strike is between saving the observation cost versus losing performance in employing coarser action selection (losing the ability to condition the choice of the second action on the first's outcome). Longer action sequences might be preferable to pairs: an apt choice depends on circumstance—an agent near a cliff likely wants few steps between observations, even if they're costly. The schedule of times between observations is *dynamic* in that it is state-dependent.

The following decision problem treats arbitrary length action sequences.

Definition 1. *A* dynamically-scheduled macro-action MDP *(*DS-MA-MDP*) is a 6-tuple* $(S, A, \vec{T}, \vec{C}^{\gamma}_{\text{ex}}, C_{\text{obs}}, \gamma)$ *where*

- S *is the set of states;*
- A *is the set of elementary actions, from which* $\vec{A} \coloneqq A \cup A^2 \cup A^3 \cup \cdots$ *defines the macro-action sequences, where* A^m *is a sequence of* m *elements of* A*;*
- $\vec{T} : S \times \vec{A} \times S \to [0,1]$ *gives the transition dynamics, or macro-action transition model, describing the stochastic state transitions of the system, assumed to be Markovian in the states, where:*
$$\Pr(s_{t+|\vec{a}|} = s' \mid s_t = s, \vec{a}_t = \vec{a}) = \vec{T}(s', \vec{a}, s),\ t \in \{0, 1, 2, \dots\};$$
- $\vec{C}^{\gamma}_{\text{ex}} : S \times \vec{A} \to \mathbb{R}$ *is the macro execution cost function given so that* $\vec{C}^{\gamma}_{\text{ex}}(s, \vec{a})$ *prescribes the cost incurred for taking macro-action* $\vec{a} \in \vec{A}$ *in state* s*;*
- $C_{\text{obs}} \in \mathbb{R}$ *is the cost incurred for making an observation;*
- $\gamma \in (0, 1)$ *is the discounting factor.*

The intuition is that though the elementary actions A are provided, the decision maker selects sequences of elements of A, i.e., open-loop macro-actions in $\vec{A}$. For simplicity, we will treat state spaces where S is finite; also, as will be described below, macro-action sequences will be computed up to some bounded length. We use the operations, introduced in [10], of transition and cost composition to form $\vec{T}$ and $\vec{C}^{\gamma}_{\text{ex}}$ from an MDP with elementary actions A.

Construction 1 (Transition composition [10])**.** Given elementary actions A and their associated transitions $T : S \times A \times S \to [0,1]$, the macro-action transition model is the function $\vec{T} : S \times \vec{A} \times S \to [0,1]$ defined as

$$\vec{T}\left(s', \vec{a}, s\right) \;=\; \vec{T}\left(s', (a_1, a_2, \dots, a_{|\vec{a}|}), s\right) \;=\; \sum_{\substack{(s_1, s_2, \dots, s_{|\vec{a}|}) \in S^{|\vec{a}|} \\ \text{where } s_0 = s \text{ and } s_{|\vec{a}|} = s'}} \prod_{i=1}^{|\vec{a}|} T(s_{i+1}, a_i, s_i).$$

Essentially, this repeatedly convolves the elementary transition dynamics, as determined by $\vec{a}$. The same idea applies to the cost as well.

Construction 2 (Cost composition [10])**.** For discount $\gamma \in (0, 1)$ and elementary cost function $C_{\text{ex}} : S \times A \to \mathbb{R}$, the corresponding macro execution cost is $\vec{C}^{\gamma}_{\text{ex}} : S \times \vec{A} \to \mathbb{R}$ defined as

$$\vec{C}^{\gamma}_{\text{ex}}(s, \vec{a}) = \vec{C}^{\gamma}_{\text{ex}}\left(s, (a_1, \dots, a_{|\vec{a}|})\right) = \sum_{k=1}^{|\vec{a}|} \gamma^{(k-1)} \sum_{\substack{(s_1, \dots, s_{|\vec{a}|}) \in S^{|\vec{a}|} \\ \text{where } s_1 = s}} C_{\text{ex}}(s_k, a_k) \prod_{i=1}^{|\vec{a}|} T(s_{i+1}, a_i, s_i).$$

Costs incurred later owing to sequences of actions are diminished via γ.

3.3 Solutions to dynamically-scheduled macro-action MDPs

The optimal state-action value function satisfies this Bellman recurrence:

$$Q^*(s, \vec{a}) = \vec{C}^{\gamma}_{\text{ex}}(s, \vec{a}) + C_{\text{obs}} + \vec{\gamma}(s, \vec{a}) \sum_{s' \in S} \vec{T}(s', \vec{a}, s) \min_{\vec{a}' \in \vec{A}} Q^*(s', \vec{a}'), \tag{1}$$

where $\vec{\gamma}(s, \vec{a}) = \gamma^{|\vec{a}|}$. The optimal policy is obtained via

$$\vec{\pi}^*(s) = \arg\min_{\vec{a} \in \vec{A}} Q^*(s, \vec{a}). \tag{2}$$

Observation cost C_{obs} shows up once per macro-action (at the beginning of each macro-action), and the future expected value is discounted by the time until the next observation; above $\vec{\gamma}(\vec{a})$ expresses this using the length of $\vec{a}$. When $C_{\text{obs}} \leq 0$, actions with $|\vec{a}| > 1$ offer no advantage over elementary ones.

Finally, the reader may argue that since $\vec{A}$ is not finite, the $(\arg)\min_{\vec{a} \in \vec{A}}$ above, technically, may not exist and ought to be an $(\arg)\inf_{\vec{a} \in \vec{A}}$. This will not concern us as, in computing solutions, we will bound the length of macro-actions via a parameter m. Such limits may arise naturally from the system's constraints: the gliders in Example 1 will have a maximum depth they can tolerate.

3.4 Synchronized multi-robot systems

Multiple robot systems comprise the core of the paper: for a system of n robots, with each $i \in \{1, \ldots, n\}$ having states S_i and actions A_i, any specific sets $\mathcal{S} \subseteq \left\{ \langle s^{(1)} | s^{(2)} | \cdots | s^{(n)} \rangle : s^{(i)} \in S_i \right\}$ and $\mathcal{A} \subseteq \left\{ \langle a^{(1)} | a^{(2)} | \cdots | a^{(n)} \rangle : a^{(i)} \in A_i \right\}$ will be termed *joint states* and *joint actions*, respectively. The key definition now follows.

Definition 2. *An* actively synchronized multi-robot MDP *for a system of n robots is a* DS-MA-MDP $(\mathcal{S}, \mathcal{A}, \vec{T}, \vec{C}^{\gamma}_{\text{ex}}, C_{\text{obs}}, \gamma)$ *defined over joint states and joint actions, i.e., it has states $\langle s^{(1)} | s^{(2)} | \cdots | s^{(n)} \rangle \in \mathcal{S}$ and actions $\langle a^{(1)} | a^{(2)} | \cdots | a^{(n)} \rangle \in \mathcal{A}$.*

The transitions $\vec{T}$ are for macro–joint actions built via Construction 1 on single time-step joint actions, i.e., the elements comprising $\mathcal{A}$. The transition function for those elements will typically model interactions, but if the robots have no couplings (e.g., they are fully independent/share no resources) it may be formed by directly composing n individual actions. Observe that joint actions may be a subset of the possibly combinatorially large space of available choices.

Exemplification via Example 1 If the pair of robots are denoted ℓ and r, with former in state $s^{(\ell)}$ and the latter in $s^{(r)}$, then the system's state is $\langle s^{(\ell)} | s^{(r)} \rangle$; joint actions will also, of course, involve two slots, the first for robot ℓ, the second for robot r. Suppose that state is known to be $\langle s^{(\ell)} | s^{(r)} \rangle$ because the pair generated a joint observation at time t_k, meaning that both gliders surfaced, collected GPS data, and communicated their positions. Then, if they possess a policy $\vec{\pi}^*$, the gliders (each) look up a joint macro-action:

$$\vec{\pi}^*\big(\langle s^{(\ell)} | s^{(r)} \rangle\big) = \vec{a} = \big(\langle a^{(\ell)}_1 | a^{(r)}_1 \rangle, \langle a^{(\ell)}_2 | a^{(r)}_2 \rangle, \ldots, \langle a^{(\ell)}_m | a^{(r)}_m \rangle\big), \text{where } m = |\vec{a}|.$$

For actions to execute individually, they cleave the action sequence in two: glider ℓ performs actions $(a^{(\ell)}_1, a^{(\ell)}_2, \ldots, a^{(\ell)}_m)$, while r does $(a^{(r)}_1, a^{(r)}_2, \ldots, a^{(r)}_m)$. Thence, at time t_{k+m}, they will surface again, and the process repeats.

An implied schedule Continuing the example, the gliders execute their individual actions without reference to the other robot for the times $t \in (t_k, t_{k+m})$. Each glider first uses GPS to obtain its location (sensing a sort of "half" state), followed by communication to ensure that the joint state will be known by all. The next check-in/synchronization time depends on the pair's configuration (i.e., the joint state). It is encoded in the policy itself through the macro-action's length—no separate communication policy or explicit schedule is required.

Centralized planning, but with decentralized execution Definition 2 does not specify how joint states are obtained nor how macro–joint actions are transformed into individual robot actions. For situations other than Example 1, decentralized execution may not depend on any communication at all; in our physical robots, local sensing provides the requisite information. Moreover, though solving (1) at planning time considers joint actions, each individual robot actually only needs just enough to provide its slot of the macro–joint action sequence from the joint state, assuming that the robots all use the same $\vec{\pi}^*$.

The reasoning above applies *mutatias mutandus* to when $n \geq 3$. Potential concerns about mismatched measurements of joint states or delays will be examined in Section 5.5.

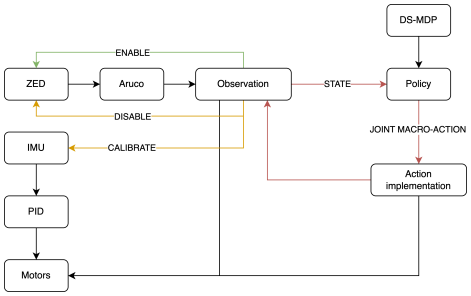
4 Demonstration case study

Next, we describe a formation-keeping problem that permitted investigation of the theory presented above in operation on the hardware we have available. Formation-keeping connects with and bears similarity to Example 1, sharing commonalities such as the need for agents to mutually decide when to make the next check-in; agents each also bear the responsibility of providing "half" of the state information at those observation points. An interesting difference is that they perform a (sometimes fallible) action to make such observations, which, owing to mismatching measurements, occasionally turn out to be imperfect.

4.1 Formation-keeping whilst driving problem

A pair of mobile robots are tasked with moving, indefinitely, in some general direction while keeping a predefined displacement between them. The robots navigate independently, making observations of their relative poses only sporadically during check-ins. We model the problem by treating the joint state to be a description of the pose of one robot relative to the other, having each of the robots' actions comprise fine-grained adjustment over a coarse baseline movement in the direction of motion. For this model, the state consists of two parameters: along-track error and cross-track error. A robot's along-track distance to its partner is the separation along the axis of motion, and its cross-track distance is the separation along the perpendicular axis. For the formation moving east, positive along-track is east and positive cross-track is north. If the proper formation is that of the two robots being side by side, along-track separation would be zero with some desired cross-track separation when in formation. The cross-track and

(a) Turtlebots in formation.



(b) Turtlebot system architecture.

Fig. 2: Turtlebots at NASA's Jet Propulsion Laboratory.

along-track *error* therefore measure how far off the current separations are from the desired formation separations.

Unlike a naïve MDP formulation with x/y coordinates for each robot, there are a number of advantages of this state description. Firstly, there is a reduction in the number of dimensions. Also, the state space need only accommodate the maximum separation of the robots, not the overall distance traveled (recall that they have a maintenance goal rather than an achievement one). The measurements do not require any global metrical reference frame, but merely measure distance along- and cross-track errors. Finally, each robot carries out its portion of the joint action independently of the other, but the joint actions have direct interpretations *in toto*: the pair get closer, separate, or maintain distance along each axis. Because some combinations of individual actions lead to nonsensical or redundant joint actions, those are omitted from the set of joint actions.

4.2 Hardware implementation

We assembled two Turtlebot3 Burger robots [11] each with a single-board computer, an IMU, and a ZED Mini camera used to detect Aruco markers [12] (see Figure 2). The system has no global localization and each robot is only able to determine the relative spacing in the formation by directly sensing the other robot. The Turtlebots drive in arcs and cannot see each other when driving as the Aruco markers and cameras are positioned on the front of each robot. To obtain a check-in when the state is needed, the Turtlebots perform an expensive panning maneuver to seek each other out: the robots rotate toward and sweep the area where they believe their partner would be. With known marker size, camera parameters, and robot heading, each robot can compute the location and orientation of the opposing Aruco marker and therefore derive both the cross- and along-track distances to its partner. After checking in, the robots return to their initial heading and drive according to the policy—reducing cross-track separation requires the robots to drive in arcs towards each other, while reducing

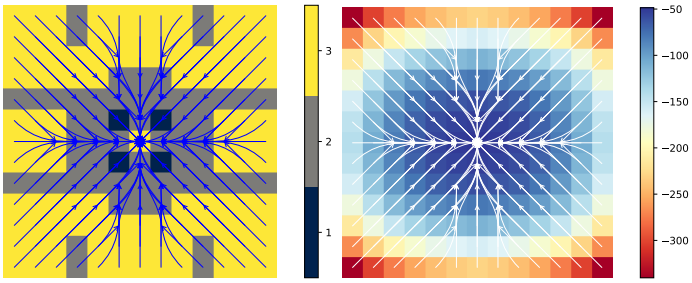


Fig. 3: The policy for the formation-keeping problem with $m = 3$ (blue arrows), color-coded with length of policy macro-actions on the left, and state values on the right.

along-track separation requires the lagging robot to drive at a higher velocity than the lead. Both robots are equipped with identical policies, from which they extract and execute their respective individual macro-actions.

For our setup, the check-in mechanism aided in the practical realization of the system. Disabling camera and Aruco processing except during observations yields far smoother motion as more CPU is given to our PID controller. While the IMU provides a fairly accurate heading, it drifts over time; it is used only during the check-in, and re-calibrated when the robot is stationary while the camera is releasing its resources. Thus, as our robots cannot drive well while scanning for markers, and cannot re-calibrate the gyroscope while moving—the check-in serves simultaneously for observation and for calibration.

5 Results

In the context of the formation-keeping problem, the following discusses a specific generated policy as well as the effects of observation cost and transition uncertainty on other policies.

5.1 Computing policies

The MDP was solved using linear programming (LP), as a faster equivalent to value iteration. The majority of computation time was in transition composition, as well as building LP constraints. We use m to denote the bound of macro-action length. With $m = 3$, the MDP took 21 s to build, the constraints took 27 s, and solving the LP took 6.6 s, for a total of 57 s. Solving the LP was thus only 11 % of the runtime.

Figure 3 displays the main policy we will examine in the following discussion of our experiments; it was computed using $m = 3$. Colors indicate for each state the length of the macro-action dictated by the policy, allowing for visually identifying the number of steps until the subsequent check-in at a glance.

Figure 4 shows the policy for the larger bound of $m = 4$, where the MDP, constraints, and LP took 6 min, 8.8 min, and 4.3 min, respectively, for a total of 20.7 min (of which solving was 20 %). The length-4 strides (orange) are only ever used around the boundary of the world, and indeed the execution of the $m = 3$ policy never travelled to any of the regions which would have a length-4 stride in

the $m = 4$ policy—excepting the start state, which had the robots intentionally far apart to stress the experiment. As a result, there was insufficient benefit in raising m to 4 to justify its extra computation time, so we remained with $m = 3$.

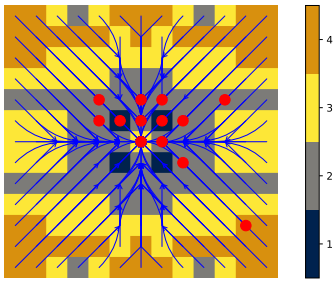


Fig. 4: Policy with $m = 4$, overlaid with states visited during the $m = 3$ execution (red dots). The bottom-right highlighted state was the initial starting position.

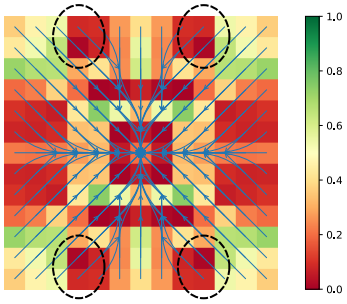


Fig. 5: Value differences to runner-up actions of different lengths are shown for the $m = 3$ policy, and notable artifacts are circled.

5.2 Hardware execution

Figure 6 shows two check-ins of the robots during execution. The onboard camera footage shows the Aruco marker of the other robot being recognized as the two robots look at each other during the check-in. In the first check-in, there is positive along-track error (upper robot is too far forward) and negative cross-track error (upper robot is too close), which translates to the red state in the policy diagram. The policy dictates for the robots to drive apart (upper goes further up, lower goes further down) and for the lower robot to drive at a higher velocity. This results in the new positions at the next check-in, one which is almost in formation. The robots believe they are in formation, as evidenced by the red state in the policy being in center, a result of the rounding error from discretizing the continuous world into grid cells. The action from that information state is then for both robots to stay the course and move in sync.

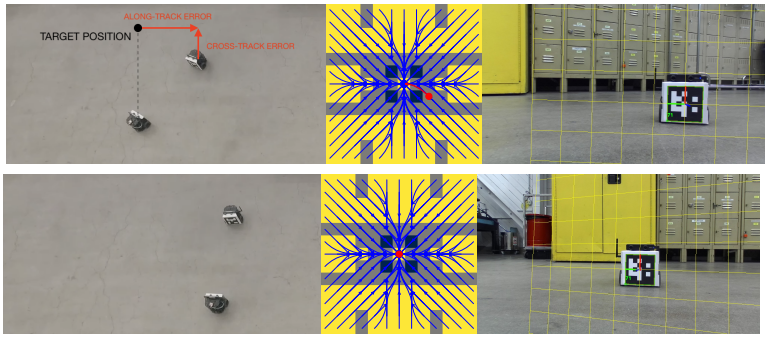


Fig. 6: Snapshots of two check-ins, with an overhead view, current policy state, and camera footage from upper robot. Video at https://youtu.be/Mv4Kgc9-1XY. (An inconsistency in the transition dynamics leads to a visual difference in the policy shown *vs* Figure 3—still, both policies agree for every state visited, all actions executed.)

5.3 Properties of the policies

An aspect readily apparent in the policy is the dependence on distance—the farther the robots deviate from the formation, the longer the actions to take. Intuitively, an agent far from the target state should move as far as it can, since the optimal action will not change even if its trajectory slips marginally, whereas when closer it is more prone to overshooting into a different policy zone. For instance, an agent starting in a corner needs to move diagonally—shorter steps cause it to pass through states themselves requiring the same diagonal motion, so longer actions forgo the cost of check-ins which, in this case, add little extra value. Also, at the center, the policy is to execute long actions again, taking advantage of being in formation to perform a long stride.

Note how not all policy macro-actions are to go straight, or diagonal. Some macro-actions are curved, being made up of different actions—for example, going diagonal then straight—and therefore are time-inhomogeneous (cf. [7,8]).

The non-radial nature and the vertical gray states (indicated via ellipses in Figure 5) can be explained as artifacts from the transition dynamics. In Figure 5 the difference in value between the optimal policy action and the next-best action of a different length is illustrated. When the difference is low (red), the policy is essentially indifferent to the length and selects it nearly arbitrarily, which leads to the jagged artifacts on the color map representing macro-action length.

5.4 Check-in cost and transition uncertainty

Further interesting relationships appear when one examines how differing parameters impact the policy, varying from low to high check-in costs and perfect to imperfect transitions, as visualized in Figure 7.

With no check-in cost in the top row, the agents observe each other at every step (the artifacts here are also from numerically close values). With absolutely perfect transitions in the left column, the agents have no need for frequent check-ins and opt for as few as possible. As one proceeds from the left to the right with higher entropy transitions, the yellow of longer actions is replaced by the grey and blue of shorter actions as more frequent check-ins are needed to correct for those imperfect actions. Conversely, going downwards with costlier observations, actions lengthen as the agents take more risk to mitigate the cost of check-ins.

5.5 Non-idealities

The model we have presented makes two key assumptions about the state observation process. Firstly, that the agents receive perfect observations of the state, an assumption central to ensuring coordination between the agents. Secondly, that all agents take the same, deterministic duration to perform their portion of an action and observe the system state. These assumptions can be violated in practice. Different agents may reach different conclusions as to the state and, hence, execute mismatched actions. Different agents may vary in the time to perform an action followed by a joint observation; it may depend on the state and action themselves (e.g., farther agents may require extra time to communicate).

In this section, we show how, under mild assumptions, the impact of the first effect can be quantified; and the second explicitly accounted for in policy design.

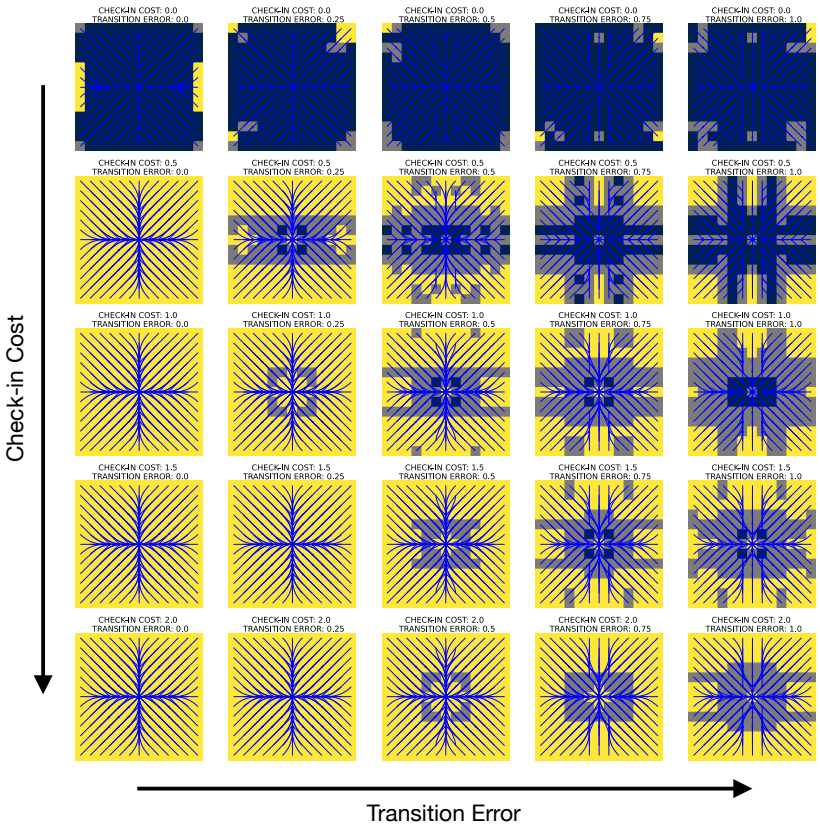


Fig. 7: Policies for various check-in costs and transition uncertainties. The element at position $(4, 4)$ with check-in cost 1.5 and transition error 0.75 corresponds to Figure 3.

Inconsistent state observations The assumption that, at check-in, all agents have access to a perfect observation of the joint system state can be violated in real systems with noisy sensors and unreliable communication. Agents then risk obtaining different macro–joint actions from the policy and executing mismatched individual actions, including ones with differing durations, which can misalign the agents' future check-in times.

Two additional assumptions can address this. First, that each agent has a "holding" action which waits for others to conclude their own actions, so all agents can measure the system state at the same time. If the system dynamics present no drift, a no-op action (with no cost or effect on the system) is suitable. Second, we assume that agents have a synchronization mechanism that indicates when all agents have concluded their actions followed by an observation. In practice, this is not overly restrictive: if agents communicate to assess the system state, an acknowledgement mechanism suffices; and, if sensor observations are used to measure the shared state, agents can use in-band signaling (e.g.,

a colorful LED visible to other agent's cameras). Importantly, the mechanism never requires an agent to communicate outside of its check-in times.

Under these assumptions, the effect of inconsistent observations on state value can be characterized, provided an observation model for the agents. Specifically, suppose a model $\Pr(\tilde{s}_1, \ldots, \tilde{s}_i, \ldots, \tilde{s}_n | s)$ characterizes the likelihood that agent i observes system state $\tilde{s}_i$ while the system is in state s. Note that the subscript i refers to agent i's observation of the *overall* system state—the concatenation of agents' individual states.

Upon observing state $\tilde{s}_i$, each agent i will select the macro–joint action given by the policy $\vec{\pi}^*(\tilde{s}_i)$ and execute its portion: $\big(\overset{(i)}{a}_1(\tilde{s}_i), \ldots, \overset{(i)}{a}_m(\tilde{s}_i)\big)$, where we make explicit the dependency of actions on the measured state $\tilde{s}_i$. The resulting macro–joint action executed in state s is then

$$\vec{\tilde{a}}(s) = \big(\langle \overset{(1)}{a}_1(\tilde{s}_1) | \cdots | \overset{(i)}{a}_1(\tilde{s}_i) | \cdots | \overset{(n)}{a}_1(\tilde{s}_n) \rangle, \ldots, \langle \overset{(1)}{a}_m(\tilde{s}_1) | \cdots | \overset{(i)}{a}_m(\tilde{s}_i) | \cdots | \overset{(n)}{a}_m(\tilde{s}_n) \rangle\big)$$

with probability $\Pr(\tilde{s}_1, \ldots, \tilde{s}_i, \ldots, \tilde{s}_n | s)$. Individual agents' actions may differ in length; in that case, agents who end early perform the holding action until everyone is ready to check in.

The resulting state values can be computed through policy evaluation. Figure 8 shows state values changing as uncertainty of the observation model increases. Each robot has likelihood o of observing one of the eight neighbors of the system state s; the robots' observations are independent, and the likelihood that both robots observe the correct state is thus $(1 - 8o)^2$.

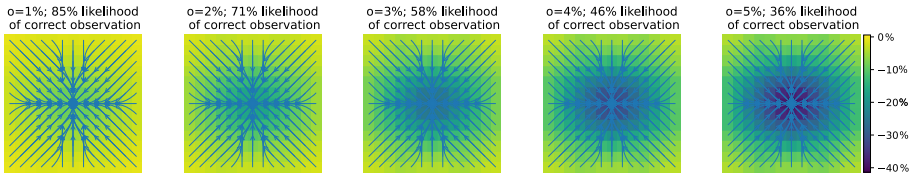


Fig. 8: Decrease in state value with increasing uncertainty in the observation model.

As observation uncertainty increases, state values decrease significantly, especially in the central region of the state where the policy is less spatially uniform. To further investigate this phenomenon, we computed a policy that is locally optimal for the $o = 5\%$ case (i.e., no single change to an action can improve the policy) through brute-force search, and compared the value of that policy with the value of the optimal policy $\vec{\pi}^*$ across multiple levels of uncertainty. The result is shown in Figure 9. The difference between the state value of the two policies is relatively small, generally under 5%, even in presence of large observation noise; this suggests that the reduction in state value shown in Figure 8 is due to the intrinsic complexity of the problem, and that the performance of the optimal policy $\vec{\pi}^*$ is remarkably robust to observation noise.

Imperfect observation time Equation (1) assumes that the next observation s' is available to all agents simultaneously. In practice, measuring the joint state takes a finite amount of time that can depend on the system state and can differ for each agent. The state-dependent delay does not affect action costs; however, it does affect the cost-to-go through the discount factor $\vec{\gamma}(s, \vec{a})$ in Equation (1).

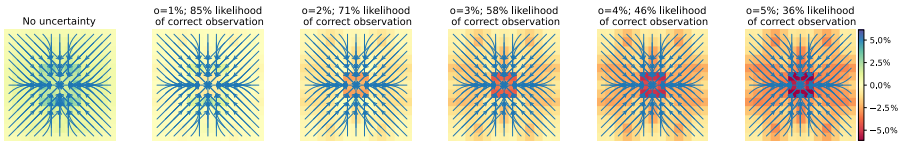


Fig. 9: Difference between the state value of the optimal policy and the state value achieved by a policy optimized for the $o = 5\%$ case.

This delay can be explicitly accounted for in planning. Assume that the time required for *all* agents to observe state s follows the distribution $\Pr(t_{\text{obs}}(s) = \tau)$. Then the discount in (1) can be rewritten as $\vec{\gamma}(s, \vec{a}) = \mathbb{E}_{\tau \sim (|\vec{a}| + \Pr(t_{\text{obs}}(s)))}(\gamma^{\tau})$, and the resulting Bellman recursion solved for a policy that accounts for delays.

Four cases of check-in delay are presented in Figure 10 for visual comparison to the no-delay case (leftmost). Observe that action length generally decreases in regions with larger expected delays. In the example, state values are generally negative; accordingly, as the discount increases, the expected cost-to-go becomes less negative, making additional observations comparatively more attractive.

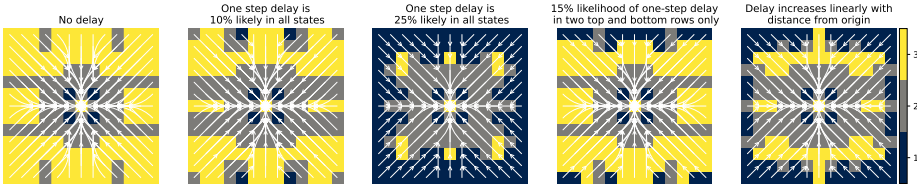


Fig. 10: Optimal policy and policy length with different expected check-in delays

Imperfect action execution time Finally, agents may take different durations to execute their part of a macro–joint action. In general, Constructions 1 and 2 are no longer directly applicable, since one agent's i^{th} action may act on the system at the same time as another agent's j^{th} action. When the system is sufficiently decoupled, Construction 1 still applies, and Construction 2's γ^{k-1} factor can be replaced by a term to model the expected discount. Though we omit the details, we remark that both examples considered in this paper (the gliders and Turtlebots) have a suitable treatment in this fashion.

6 Conclusion

This paper proposes an MDP formulation for multi-agent sequential decision-making problems where agents must decide not only what actions to perform, but also when to jointly observe the system state. This is, thus, a jointly endogenous observation process and, since observations are modeled as bearing costs, optimal solutions often end up employing observations rather sparingly. We find that, at least for small-scale problems, the approach is computationally tractable and demonstrate its practicality on hardware for a stylized formation-keeping scenario. Having conducted an appraisal of the properties of the policies that are produced, we see that they appear effective even in the presence of noisy and delayed state observations.

A number of directions for future research are of interest. First, the action set $\mathcal{A}$ is a subset of all possible macro–joint actions, so one might explore how to efficiently prune this set to reduce the computational requirements of the problem, extending our prior work [10] that addressed fixed check-in times. Second, we plan to propose techniques to make the policy more robust to imperfect state observations by penalizing highly inconsistent actions in neighboring (and therefore easily-confused) states. Thirdly, we plan to extend the formulation to capture selected *partial* state observations that can arise in multi-agent systems (e.g., improved knowledge of the agent's own state, or Boolean-valued observations reporting whether the system state lies inside a given set), while retaining computational tractability. Finally, it would be interesting to explore decentralized planning as well.

References

1. C. Amato, G. Konidaris, L. P. Kaelbling, and J. P. How, "Modeling and Planning with Macro-Actions in Decentralized POMDPs," *Journal of Artificial Intelligence Research*, vol. 64, pp. 817–859, 2019.
2. K. Zhang, Z. Yang, and T. Başar, *Multi-Agent Reinforcement Learning: A Selective Overview of Theories and Algorithms*. Springer, 2021, pp. 321–384.
3. D. S. Bernstein, R. Givan, N. Immerman, and S. Zilberstein, "The Complexity of Decentralized Control of Markov Decision Processes," *Mathematics of Operations Research*, vol. 27, no. 4, pp. 819–840, 2002.
4. T. Yang, Y. Cao, and G. Sartoretti, "Intent-based Deep Reinforcement Learning for Multi-agent Informative Path Planning," in *International Symposium on Multi-Robot and Multi-Agent Systems*, Dec. 2023, pp. 71–77.
5. R. J. Marcotte, X. Wang, D. Mehta, and E. Olson, "Optimizing multi-robot communication under bandwidth constraints," *Autonomous Robots*, vol. 44, no. 1, pp. 43–55, 2020.
6. G. Best, O. M. Cliff, T. Patten, R. R. Mettu, and R. Fitch, "Dec-MCTS: Decentralized planning for multi-robot active perception," *The International Journal of Robotics Research*, vol. 38, no. 2-3, pp. 316–337, 2019.
7. Y. Huang and Q. Zhu, "Self-Triggered Markov Decision Processes," in *IEEE Conference on Decision and Control (CDC)*, 2021, pp. 4507–4514.
8. C. Reisinger and J. Tam, "Markov decision processes with observation costs: framework and computation with a penalty scheme," 2023, arXiv:2201.07908.
9. M. L. Puterman, *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. John Wiley & Sons, 2014.
10. F. Rossi and D. A. Shell, "Planning under periodic observations: bounds and bounding-based solutions," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Oct. 2022.
11. ROBOTIS, "Turtlebot3 Specifications," https://emanual.robotis.com/docs/en/platform/turtlebot3/features/.
12. OpenCV, "Detection of ArUco Markers," https://docs.opencv.org/4.x/d5/dae/tutorial_aruco_detection.html.