Distributed Information Gathering Policies under Temporal Logic Constraints

Kevin Leahy, Austin Jones, Mac Schwager, and Calin Belta

***Abstract*— We present an algorithm for synthesizing distributed control policies for networks of mobile robots such that they gather the maximum amount of information about some *a priori* unknown feature of the environment, e.g. hydration levels of crops or a lost person adrift at sea. Natural motion and communication constraints such as "Avoid obstacles and periodically communicate with all other agents", are formulated as temporal logic formulae, a richer set of constraints than has been previously considered for this application. Mission constraints are distributed automatically among sub-groups of the agents. Each sub-group independently executes a receding horizon planner that locally optimizes information gathering and is guaranteed to satisfy the assigned mission specification. This approach allows the agents to disperse beyond inter-agent communication ranges while ensuring global team constraints are met. We evaluate our novel paradigm via simulation.**

I. Introduction

We consider a group of robots that are tasked with conducting a mission while gathering information about a large environment. For example, our method could be employed by a group of robots autonomously tending to an agricultural field. The robots must irrigate and monitor crops (the "mission"), while simultaneously searching for and reacting to pest infestation (the "information gathering"). In order to complete this mission, the agents have to satisfy certain motion constraints such as "Always avoid obstacles" and "Visit a centralized station to upload gathered data." Constraints on the mission might require agents to cooperate or perform tasks in a certain order, e.g. harvest grain before depositing grain at a silo. Additionally, the agents face the dual constraints of spreading out to explore the environment while also communicating effectively with each other to share gathered information and ensure cooperative tasks are fulfilled. The motion and communication constraints can naturally be described by a temporal logic (TL) formula. TL constraints combine Boolean and temporal operators to capture rich and complex specifications such as "Visit regions A and B in any order before visiting region C while always avoiding region D".

We use tools from distributed formal methods to distribute the TL formula among sub-teams of the agents such that if each sub-team satisfies its individual formula, the global constraints are satisfied. Once a sub-team has been assigned an individual mission, it executes a computationally efficient receding horizon planner that locally maximizes the amount of information gained and is guaranteed to satisfy the individual mission. An implementation of our procedure is applied to a surveillance case study. Results from Monte Carlo simulations demonstrate that our approach outperforms a random walk constrained to satisfy the given specification.

Maximally gathering information in a distributed manner is an example of a decentralized partially observable markov decision process (DEC-POMDP), whose optimal solution has been proven to be NEXP-complete (and therefore infeasible to calculate) in the worse case [1]. Common methods used in this domain include one-step-look-ahead [2], receding horizon [3], [4], and off-line planning [5], [6]. Recent work has also included sampling trajectories [7] and methods using rapidly-exploring random trees and graphs [8]. Our work incorporates TL constraints into the path planning problem. These constraints permit richer, more realistic constraints on the motion of the agents than have previously been addressed for multi-agent systems. The receding horizon algorithm used for local information gathering is based on the single agent method proposed in [9], which maximizes information gathered by a single agent subject to TL constraints.

In [10], the authors provided a method for distributing a global task given as a regular expression to tasks for individual robots using methods from concurrency theory [11] and distributed formal methods [12], [13]. In that work, agents are given a pre-computed path to follow through the environment, and may be required to wait to communicate with other agents before completing their task. The authors provide a broad framework in which sub-teams of agents can act independently to ensure that global, cooperative behaviors are produced. In this paper, we extend this framework to include more typical—and more restrictive—communication constraints based on agents' distance from each other in the environment. Our method also allows the agents to act according to reactive control policies rather than follow pre-specified paths, giving the agents a greater degree of flexibility in conducting their mission. Further, we show how to distribute specifications among sub-teams of agents rather than among individual agents.

II. Preliminaries

For a set Σ, we denote the cardinality and power set as $|\Sigma|$ and 2^{Σ}, respectively. Σ^* denotes the set of all finite words that can be constructed from Σ. For two sets, A and B, $A \times B$ indicates their Cartesian product, and $A^n = A \times \ldots \times A$. For a collection of sets $\{\Sigma_i\}_{i \in I}$ where I is an index set, we use

K. Leahy is with the Department of Mechanical Engineering; A. Jones is with the Division of Systems Engineering; and M. Schwager and C. Belta are with the Division of Systems Engineering and the Department of Mechanical Engineering at Boston University, Boston, MA 02215. Email: {kjleahy,austinmj,schwager,cbelta}@bu.edu

This work was partially supported by ONR MURI N00014-09-1051, ONR N00014-12-1-1000, NSF NRI-1426907, and NSF CNS-1035588.

$\prod_{i\in I}\Sigma_i$ to denote the Cartesian product of all the sets in the collection. We denote the empty string with ϵ.

A. Discrete Models

A *deterministic transition system* (TS) is a tuple $TS = \left(Q, q^0, Act, Trans, AP, \models\right)$, where Q is a set of states, $q^0 \in Q$ is the initial state, Act is a set of actions, $Trans \subseteq Q \times Act \times Q$ is a deterministic transition relation, AP is a set of atomic propositions, and $\models \subseteq Q \times 2^{AP}$ is a satisfaction relation. We define $AP_q = \{\, p \in AP |\, (q,p) \in \models\}$ as the set of atomic propositions satisfied at state q. A finite run of a TS is a sequence of states $q^0 q^1 \ldots \in Q^*$ such that $\exists a^i \in Act$ such that $\left(q^i, a^i, q^{i+1}\right) \in Trans\ \forall i = 0, 1, \ldots$. An output trace of a run is a word $w = w^0 w^1 \ldots$ where $w^i = AP_{q^i}$.

A *discrete time Markov Chain* (MC) is a tuple $MC = \left(S, s^0, P\right)$, with a set of states S, an initial state s^0, and a probabilistic transition relation $P : S \times S \to [0,1]$ such that the probability of transitioning from state s to s' is $P\left(s, s'\right)$.

A *discrete time Markov Decision Process* (MDP) is a tuple $MDP = \left(S, s^0, P, Act\right)$, where S and s^0 are defined as for an MC, Act is a set of actions, and $P : S \times Act \times S \to [0,1]$ is a probabilistic transition relation with the probability of transitioning from state s to state s' under action a given by $P\left(s, a, s'\right)$. The set of actions a available at state s is $Act(s) \subseteq Act$ such that $\exists s' \in S$ with $P(s,a,s') > 0$. A sequence of states $s^0 s^1 \ldots s^l$ with $P\left(s^i, a, s^{i+1}\right) > 0$ for $a \in Act\left(s^i\right)\ \forall i = 0, \ldots, l-1$ is called a sample path.

B. Automata

In this work, we consider missions that can be specified using *syntactically co-safe Linear Temporal Logic* (scLTL) [14]. Given a set of atomic propositions AP, an scLTL formula is defined inductively as:

$$\phi = p | \neg p | \phi_1 \vee \phi_2 | \phi_1 \wedge \phi_2 | \phi_1 \mathcal{U} \phi_2 | \bigcirc \phi_1 | \Diamond \phi_1\,, \qquad (1)$$

where $p \in AP$, and ϕ_1 and ϕ_2 are scLTL formulae, $\neg$, $\vee$, and $\wedge$ are Boolean negation, conjunction and disjunction, respectively, and $\mathcal{U}$, $\bigcirc$, and $\Diamond$ are the temporal operators until, next, and eventually. The satisfaction of an scLTL formula can be checked in finite time, and all linear temporal logic (LTL) formulae that can be checked in finite time can be expressed as an scLTL formula. We denote the set of all words that satisfy ϕ as the language of ϕ, denoted $L(\phi)$ [14].

A *finite state automaton* (FSA) is a tuple $A = \left(X, \Pi, x^0, F, \rightarrow_A\right)$, where X is a set of states, Π is an input alphabet, $x^0 \in X$ is an initial state, $F \subseteq X$ is a set of final (accepting) states, and $\rightarrow_A \subseteq X \times \Pi \times X$ is a deterministic transition relation. A accepts a word $w \in \Pi^*$ if the last symbol in w is in the accepting set F. The set of all words accepted by an automaton A is called the *language* of the automaton and is denoted by $L(A)$. Given an automaton A, we use $\neg A$ to denote the automaton such that $L(\neg A) = \Sigma^* \setminus L(A)$. $\neg A$ can be constructed from A by replacing all accepting states with non-accepting states and all non-accepting states with accepting states. Given an scLTL formula ϕ, there exist off-the-shelf tools such as scheck [15] which can construct an automaton A_ϕ with input language 2^{AP} such that $L(\phi) = L(A_\phi)$.

The *synchronous product* of a set of automata $A_i = \left(X_i, \Pi_i, x_i^0, F_i, \rightarrow_{A_i}\right)$ for i in index set I is the automaton $A_p = \|_{i\in I} A_i = \left(X_p, \Pi_p, x_p, F_p, \rightarrow_{A_p}\right)$ where $X_p = \prod_{i\in I} X_i$, $\Pi_p = \bigcup_{i\in I}\Pi_i$, $x_p = (x_i^0)_{i\in I}$, and $F_p = \prod_{i\in I} F_i$. The transition relation $\rightarrow_{A_p} \subseteq X_p \times \Pi_p \times X_p$ is defined such that $((q_i)_{i\in I}, \pi, (q_i')_{i\in I}) \in \rightarrow_{A_p} \Leftrightarrow \forall j \in I$ such that $\pi \in \Pi_j, (q_i, \pi, q_i') \in \rightarrow_{A_i}$ and $\forall k$ such that $\pi \notin \Pi_k$, $q_k = q_k'$.

A *product automaton* between a transition system $TS = \left(Q, q^0, Act, Trans, AP, \models\right)$ and an FSA $A_\phi = \left(X, 2^{AP}, x^0, F, \rightarrow_A\right)$ is an FSA $\mathcal{P} = TS \times A_\phi = \left(X_\mathcal{P}, 2^{AP}, \chi_0, Act, F_\mathcal{P}, \rightarrow_\mathcal{P}\right)$. $X_\mathcal{P} \subseteq Q \times X$ is the state space of $\mathcal{P}$, $\chi^0 = (q^0, \sigma_0)$ is the initial state, and $F_\mathcal{P} \subseteq Q \times F$ is the set of accepting states. The transition relation is defined as $\rightarrow_\mathcal{P} = \{(q,x), p, (q',x') \,|\, (q,p,q') \in Trans, (x, AP_q, x') \in \rightarrow_A\}$. The state of $\mathcal{P}$ at time k is (q^k, x^k), which we denote as χ^k for brevity. If $\chi^{0:\ell}$ is an accepting run on $\mathcal{P}$, then the associated run $q^{0:\ell}$ satisfies ϕ.

For a set Σ, we call the set of subsets $\{\Sigma_i \subseteq \Sigma, i \in I\}$, a *distribution*[1] Δ of Σ if $\cup_{i\in I}\Sigma_i = \Sigma$, where I is an index set. For a word $\omega \in \Sigma^*$ and a subset $\Sigma_i \subseteq \Sigma$, the *projection* of ω onto Σ_i, written $\omega \restriction_{\Sigma_i}$, is obtained by removing all symbols in ω that are not in Σ_i. For a language $L \subseteq \Sigma^*$ and a subset $\Sigma_i \subseteq \Sigma$, the projection of L onto Σ_i, $L \restriction_{\Sigma_i}$ is the set $\{\omega \restriction_{\Sigma_i} | \omega \in L\}$.

Given a distribution $\{\Sigma_i\}_{i\in I}$ of Σ and $\omega, \omega' \in \Sigma^*$, ω' is *trace-equivalent* to ω ($\omega' \sim \omega$), iff $\omega \restriction_{\Sigma_i} = \omega' \restriction_{\Sigma_i}$ for $i \in I$. The *trace-equivalence class* of ω for the distribution is $[\omega] = \{\omega' \in \Sigma^* | \omega' \restriction_{\Sigma_i} = \omega \restriction_{\Sigma_i} \quad \forall i \in I\}$. A *trace-closed language* over the distribution is a language L such that $[\omega] \subseteq L$, $\forall \omega \in L$.

III. Problem Formulation

A. Motion and Service Model

We consider a team of agents with heterogeneous motion capabilities operating in a shared environment. The environment is modeled as a graph $\mathcal{E} = (V, \rightarrow_\mathcal{E})$, with a set of states V and a set of edges $\rightarrow_\mathcal{E} \subseteq V \times V$. Such a discrete graph may be constructed as the quotient graph of a partitioned continuous environment. We define a labeling function, $\mathcal{L} : V \to 2^{AP}$, which maps regions in the environment to a set of atomic propositions which may be satisfied at the regions.

A team of m agents is indexed by the set I. The motion of a single robot i in $\mathcal{E}$ is modeled by a TS $Robot_i = \left(Q_i, q_i^0, Act_i, Trans_i, \Sigma_i, \models_i\right)$, where $Q_i \subseteq V$ are the set of states that $Robot_i$ can occupy, $q_i^0 \in Q_i$ is the initial state, Act_i is the set of actions the robots can take, $\Sigma_i \subseteq AP \cup \{\epsilon\}$are the robot's service capabilities, and $\models_i \subseteq Q_i \times \Sigma_i$ captures how atomic propositions may be satisfied by agent i at the states, where $(q, \epsilon) \in \models_i$ for all $q \in Q_i$ and $(q, \sigma) \in \models_i, \sigma \in \Sigma_i$, if and only if $\sigma \in \mathcal{L}\left(q\right)$. The model uses

[1]The use of the term *distribution* should not be confused with the notion of a probability distribution. We use this term to remain consistent with the related literature.

a discrete clock k which is initialized to zero and increments by 1 every time $Robot_i$ takes an action. The state of $Robot_i$ at time k is q_i^k, and the action executed at time k is a_i^k. We write $\boldsymbol{a}^k$ to represent the vector of actions taken by the team at time k. Similarly, the state of the team at time k is written as $\boldsymbol{q}^k$. Two or more agents may occupy the same state, and the clocks of each agent in the team are assumed to initialize and evolve in time synchronously.

Each run $r_i = q_i^0 q_i^1 \ldots$ of $Robot_i$ generates a corresponding output word $\omega_i = \omega_i^0 \omega_i^1 \ldots$. Each symbol in ω_i comes from the alphabet of $Robot_i$ such that $\left(q_i^l, \omega_i^k\right) \in \models_i$. For the team as a whole, the output word $\omega_{team} = \omega_{team}^0 \omega_{team}^1 \cdots$ is generated such that $\omega_{team}^k = \bigcup_{i=1}^m \omega_i^k$ is the union of all propositions serviced at time k.

A distribution Δ captures the service capabilities of a team of agents. The capabilities of agent i are given by $\Sigma_i \in \Delta$. For a request $\sigma \in \Sigma_i$, agent i is said to "own" the request, and that agent is the only agent that can service that request. If more than one agent owns the request, that is, if σ appears in more than one set Σ_i, it must be serviced by all of the agents that own it simultaneously in order to be satisfied.

Example 1. Three agents must perform a surveillance mission in the environment pictured in Fig. 2. This environment is modeled as a graph with 64 nodes, which are inherited by the transition systems $\{Robot_i\}_{i=1,2,3}$. Agents must survey regions of interest labeled $\pi_i,\ i = 1, \ldots, 4$ while avoiding obstacles and tracking a target whose position is a priori unknown. Each agent begins in a region labeled π_H and has motion primitives $\{N, S, E, W\}$, corresponding to each of the four directions on the grid. Obstacles in the environment are labeled π_O. The distribution Δ of agent capabilities is $\Sigma_1 = \{\pi_1, \pi_4\}$, $\Sigma_2 = \{\pi_2, \pi_3\}$, and $\Sigma_3 = \{\pi_2, \pi_3\}$.

B. Communication Model

We assume that communication among agents is based on proximity to other agents. A parameter, $CommDist$, captures the maximum distance over which two agents may communicate directly. For robots communicating wirelessly, the threshold represents the maximum distance over which wireless communication has a high probability of success. For two agents i and l, $Q_{Comm} \subseteq Q_i \times Q_l$ is the set of states where communication is possible. That is, $Q_{Comm} = \{(q_1, q_2) \,|d(q_1, q_2) \leq CommDist\}$, where $d(q_1, q_2)$ is the distance between q_1 and q_2 on the graph of the environment $\mathcal{E}$. The agents therefore form a mobile ad hoc network, and we assume the use of a protocol for efficient communication over such a network, given changing topology [16].

C. Sensing Model

The team of robots is tasked with estimating a feature (or target) in the environment that evolves stochastically over timethat is modeled as a Markov Chain $Targ = \left(S, s^0, P\right)$ which evolves synchronously with $Robot_i$. The state of $Targ$ at time k is denoted as s^k. The initial state of $Targ$, s^0, is a priori unknown. When $Robot_i$ moves to state q_i^k at time k it measures s^k using noisy sensors, resulting in measurement $y_i^k \in R_Y$. The vector of the measurements of all agents at time k is written $\boldsymbol{y}^k$. Each measurement is a realization of a discrete random variable, Y_i^k. The distribution of Y_i^k depends on the true underlying state of the feature s^k, the position of the robot taking measurement q_i^k, and the statistics of the sensor. We capture this with measurement likelihood function

$$h\left(y, s, q\right) = Pr[\text{ measurement is } y | Targ \text{ in state } s, Robot \text{ in state } q]. \tag{2}$$

In this work, we assume that each robot has identical sensing capabilities, i.e. the measurement likelihood function for each agent is identical. Each robot maintains an individual estimate of s^k given its own measurements $b_i^k\left(s\right) = Pr[s^k = s | y_i^{1:k}, q_i^{0:k}]$. Each belief state b_i is initialized as an identical pmf b^0 which reflects any initial information about the state of s^0. For a sub-team of agents who are able to communicate, we denote the belief of the jth sub-team, which is identical among the agents in the team, as $\boldsymbol{b}_j^k$. As agents take measurements, they share those measurements with the team, and the team belief is updated as

$$\boldsymbol{b}_j^k\left(s\right) = \eta Pr\left(\boldsymbol{y}_j^k | s, \boldsymbol{q}_j^k\right) \sum_{s' \in S} P\left(s', s\right) \boldsymbol{b}_j^{k-1}\left(s'\right) \tag{3}$$

where η is the appropriate normalization factor, $\boldsymbol{y}_j^k$ is the collection of measurements taken by the sub-team, and $\boldsymbol{q}_j^k$ describes the positions of all of the agents in the sub-teams. We assume that the measurements of the agents are conditionally independent. The conditional distribution of the measurements $\boldsymbol{y}_j^k$ is

$$Pr\left(y_1^k, \ldots y_m^k \middle| \, s, q_1^k, \ldots q_m^k\right) = \prod_{i \in I} h\left(y_i^k, s, q_i^k\right) \ . \tag{4}$$

Each sub-team belief $\boldsymbol{b}_j^k$ evolves according to an MDP $Est_j = \left(B, b^0, P_{est}, Q_j\right)$. B is the set of all possible beliefs that can be outputs of the Bayes filter given initial belief b^0. P_{est} is a probabilistic transition relation such that if $\boldsymbol{b'}_j$ is the result of applying (3) after measuring $\boldsymbol{y}_j^k$ in states $\boldsymbol{q}_j^k$, then $P_{est}\left(\boldsymbol{b}_j, \boldsymbol{q}_j^k, \boldsymbol{b'}_j\right)$ is the total probability of observing $\boldsymbol{y}_j^k$, i.e.

$$P_{est}\left(\boldsymbol{b}_j, \boldsymbol{q}_j^k, \boldsymbol{b'}_j\right) = \textstyle\sum_{s_1, s_2 \in S} \prod_{i \in I} h\left(y_i^k, s, q_i^k\right) P\left(s_1, s_2\right) \boldsymbol{b}_j\left(s_1\right). \tag{5}$$

Example 2. In our example, each agent may detect the presence of the target, e.g. a vehicle of interest in an urban environment, in its neighborhood, $\mathcal{N}_i$. In other words, each agent can detect the target in its own location on the transition system or in adjacent states on the transition system. Detection is binary, with a 1 indicating that the target is detected in $\mathcal{N}_i$ and 0 otherwise.

D. Problem Definition

Here we formulate the problem of multi-agent information gathering under temporal logic constraints. We assume that the team of agents must satisfy its mission constraints ϕ before a deadline T. This deadline can be used to enforce energy constraints (limit the number of actions the robots take) or timeliness constraints (make sure information is shared in a timely manner). Our goal in this problem is to select the set of actions for the team $\boldsymbol{a}^{0:T-1}$ that minimizes

the uncertainty in the estimate of the state of $Targ$ while satisfying the mission constraints in time. In other words,

Problem III.1 (scLTL-constrained information gathering)**.** Given a team of m agents each with model, a feature *Targ*, an scLTL formula ϕ over *AP*, and a deadline T, solve

$$\min_{\boldsymbol{a}^{0:T-1}} E_{\boldsymbol{Y}^{0:T}}\left[H\left(\boldsymbol{b}^T|b^0,\boldsymbol{Y}^{0:T},\boldsymbol{q}^{0:T}\right)\right] \quad \text{subject to } \phi \text{ is satisfied,} \tag{6}$$

where $H(\cdot)$ is Shannon entropy.

Example 3. In our example, the 3 agents are required to satisfy the following mission specification

$$\phi = \Diamond\pi_1 \wedge \Diamond\pi_2 \wedge \Diamond\pi_3 \wedge \Diamond\pi_4 \wedge (\neg\pi_3 \mathcal{U} \pi_2)\,. \tag{7}$$

Specification ϕ is interpreted as "eventually service regions 1 through 4, and service region 2 before servicing region 3." The deadline imposed on satisfaction is $T = 20$.

IV. Solution

The solution to Problem III.1 is summarized in Alg. 1. First, the team of agents splits into sub-teams according to their capabilities (IV-A). Next, the specification is checked for distributability among the sub-teams (IV-B). The specification is distributable if it may be separated into multiple local specifications such that if each sub-team satisfies its local specification, the global specification is satisfied. The sub-teams then independently execute a receding horizon algorithm to gather information and satisfy their local specifications. Finally, they return to a starting region and share their measurements (IV-C). Distributing the specification may be implemented offline pre-deployment, while sub-teams perform the receding horizon algorithm online during mission execution.

Algorithm 1 Solution Outline

Input: An scLTL formula ϕ over Σ, a distribution $\Delta = \{\Sigma_i \subseteq \Sigma, i \in I\}$ of Σ, a set of TS, $Robot_i$, $i \in I$, a deadline T, a lookahead horizon h, and an action horizon n

1: Build sub-teams $\mathbb{C}$, distribution $\Delta_{\mathbb{C}}$, and $\{TS_{c_i}\}_{i\in I_{\mathbb{C}}}$
2: $w_i \forall i \in I_{\mathbb{C}} = \text{GetLocalWords}\,(\phi, \Delta_{\mathbb{C}}, TS_{c_i}\ \forall i \in I_{\mathbb{C}})$
3: **for** $c_i \in \mathbb{C}$ **do**
4: Construct automata A_i^{loc} accepting w_i
5: Construct product $\mathcal{P}_i$
6: RecedingHorizonDP($\mathcal{P}_i, \chi^0, b^0, h, n, T$)
7: Share measurements with other sub-teams
8: Calculate $\boldsymbol{b}^T$

A. Sub-teams

As noted in Section III-A, if a request σ appears in the capabilities of more than one agent, according to the semantics of scLTL, those agents who own the request must service it simultaneously in order for it to be satisfied. Therefore, we break up the team of m agents into subsets called sub-teams according to the distribution of their capabilities. These sub-teams are the smallest groups of agents who must cooperate to satisfy the TL specification. These sub-teams may operate independently to satisfy a portion of the mission specification, as explained in sections IV-B & IV-C.

The *specification alphabet*, denoted Σ_ϕ, is the set of all atomic proposition appearing in ϕ. The team is split into a set of sub-teams $\mathbb{C} = \{c_1, \ldots, c_n\}$, where each sub-team c_i is made up of one or more agents from I and $n \leq m$. An agent may not be in more than one sub-team. The *sub-team alphabet* is given by $\Sigma_{c_i} = \bigcup_{j\in c_i} \Sigma_j$. In other words, the sub-team alphabet is the union of the capabilities of the members of the sub-team. Membership in sub-teams is such that given two sub-teams, c_i and c_j,

$$\Sigma_{c_i} \restriction_{\Sigma_\phi} \cap \Sigma_{c_j} \restriction_{\Sigma_\phi} = \emptyset,\ \forall c_i, c_j \in \mathbb{C}\,, \tag{8}$$

i.e., the capabilities of agents in any two sub-teams are distinct, while the capabilities of agents in the same sub-team overlap (although not necessarily with all agents in the sub-team). We may also define a distribution $\Delta_{\mathbb{C}}$ with respect to the sub-teams where $\bigcup_{c_i\in\mathbb{C}} \Sigma_{c_i} = \Sigma_\phi$. This distribution captures the capabilities of the sub-teams, by combining the capabilities of each of the agents in the sub-team. The set of sub-teams $\mathbb{C}$ is indexed by a set denoted $I_{\mathbb{C}}$.

Example 4. Given the distribution $\Sigma_1 = \{\pi_1, \pi_4\}$, $\Sigma_2 = \{\pi_2, \pi_3\}$, and $\Sigma_3 = \{\pi_2, \pi_3\}$, we construct the set $\mathbb{C} = \{c_1, c_2\}$, where c_1 contains agent 1, and c_2 contains agents 2 and 3. As such, $\Sigma_{c_1} = \{\pi_1, \pi_4\}$ and $\Sigma_{c_2} = \{\pi_2, \pi_3\}$, since agent 1 has capabilities π_1 and π_4 and agents 2 and 3 have capabilities π_2 and π_3, and each of these propositions appears in ϕ as given by Equation 7.

For each sub-team $c_i \in \mathbb{C}$, we construct a product transition system $TS_{c_i} = \left(Q_{c_i}, q^0_{c_i}, Act_{c_i}, Trans_{c_i}, \Sigma_{c_i}, \models_{c_i}\right)$, consisting of $|c_i|$ copies of $Robot$, where $Q_{c_i} \subseteq \prod_{i\in c_i} Q_i$, $q^0_{c_i} = (q^0_i)_{i\in c_i}$, $\Sigma_{c_i} = \bigcup_{i=1}^{|c_i|} \Sigma_i$, and $\models_{c_i} = \bigcup_{i=1}^{|c_i|} \models_i$. The set of transitions at state $q^k_{c_i}$ is defined as $Trans_{c_i} \subseteq Q_{c_i} \times Act_{c_i} \times Q_{c_i}$ such that $\forall q^k_i \in q^k_{c_i}$, $\exists q'_i, a_i$ such that $\left(q^k_i, a_i, q'_i\right) \in Trans_i$ and $\exists j \in c_i$, $j \neq i$ such that $(q_j, q_i) \in Q_{Comm}$. Thus, the transition system includes only actions that do not disconnect the group communication graph.

B. Task Distribution

This section deals with determining if the mission specification may be distributed among the sub-teams as created in IV-A. If distribution is possible, we present a method for finding a local task for each sub-team which guarantees satisfaction of the specification. This process is summarized in Algorithm 2. Algorithm 2 is inspired by [10], with modifications to permit a more restrictive communication model as well as accommodating online motion planning. Correctness of the algorithm is based on the same concepts as [10], and as such details and proofs are omitted here.

First, the specification ϕ is converted to an FSA $A_\phi$such that $L(A_\phi) = L(\phi)$. Sub-team-specific FSAs, A_i $\forall i \in I_{\mathbb{C}}$, are created by projecting A_ϕ onto the capabilities of each sub-team, such that $L(A_i) = L(A_\phi) \restriction_{\Sigma_{c_i}}$. Next, we add the empty string, ϵ and self transitions to A_i to create $\hat{A}_i$. For each sub-team, a product FSA P_i is constructed from each $\hat{A}_i$ and its corresponding product transition system, TS_{c_i}.

The product FSAs capture the behavior of each sub-team and its ability to satisfy requests from ϕ while remaining in communication.

Once the sub-team capabilities are captured in their corresponding product FSAs P_i, they are converted to minimal, deterministic representations A_i^ϵ using the subset construction algorithm outlined in [17]. This means that A_i^ϵ captures all possible words that the behavior of sub-team i can produce. Taking the product of these FSAs, $\|_{i\in I_\mathbb{C}} A_i^\epsilon$, captures the possible interleaving behavior of the sub-teams.

If the language of the original FSA, A_ϕ, is trace closed and the language of the product of A_ϕ with the product of all of the A_i^ϵ, $\|_{i\in I_\mathbb{C}} A_i^\epsilon$ is non-empty, a satisfying word can be found using backwards reachability [18]. This word is projected onto the sub-teams' capabilities to find local words for each sub-team. These words are guaranteed to satisfy ϕ when executed by the sub-teams. The complexity of checking for trace-closedness of A_ϕ is bounded by $O\left(|X|\cdot|\Sigma|\right)$, construction of A_i^ϵ by $O\left(|Trans_{c_i}|\cdot|Q|\right) + O\left(|Trans_{c_i}|log|Trans_{c_i}|\right)$, and construction of $\|i\in I_\mathbb{C} A_i^\epsilon$ by $O\left(\left(\prod_{i\in I_\mathbb{C}}|Trans_{c_i}|\right)^2\cdot|\Sigma|\right)$. Details on the proof of correctness and complexity can be found in [10].

Algorithm 2 Find local words from a global specification

1: **function** GETLOCALWORDS(ϕ,$\Delta_\mathbb{C}$,$TS_{c_i}\ \forall i\in I_\mathbb{C}$)
2: Construct A_ϕ and $\{A_i = A_\phi \restriction_{\Sigma_{c_i}},\ i\in I_\mathbb{C}\}$
3: Construct $\hat{A}_i$ from $A_i \forall i\in I_\mathbb{C}$ by adding ϵ
4: Construct product FSA $P_i = \hat{A}_i \times TS_{c_i}\ \forall i\in I_\mathbb{C}$
5: Construct A_i^ϵ via subset construction algorithm
6: Construct $\|_{i\in I_\mathbb{C}} A_i^\epsilon$ and verify $L(\|_{i\in I_\mathbb{C}} A_i^\epsilon)\neq\emptyset$
7: **if** $L(A_\phi)$ is trace closed **then**
8: $A_G = A_\phi \times \|_{i\in I_\mathbb{C}} A_i^\epsilon$
9: **else**
10: $A_G = \neg\left(\|_{i\in I_\mathbb{C}}\left(\left(\|_{i\in I_\mathbb{C}} A_i^\epsilon \times (\neg A_\phi)\right)\restriction_{\Sigma_i}\right)\right) \times \|_{i\in I_\mathbb{C}} A_i^\epsilon$
11: **if** $L(A_G)=\emptyset$ **then**
12: **return** No solution
13: **else**
14: Find satisfying word $w_g\in L(A_G)$
15: Find local words $w_i = w_g\restriction_{\Sigma_{c_i}},\ \forall i\in I_\mathbb{C}$
16: **return** $w_i\ \forall i\in I_\mathbb{C}$

C. Dynamic Programming

Once a local word w_i has been found for each sub-team $c_i\in\mathbb{C}$, the agents are deployed in the environment. At this point, each sub-team separately executes a receding horizon planner that locally maximizes information gathering while guaranteeing the satisfaction for their assigned local word (if the local word is satisfiable). Two agents may occupy the same region without collision and members of each sub-team share observations in real-time during execution.

To initialize the planner, each sub-team constructs an automaton A_i^{loc} which accepts the local word w_i as computed in Section IV-B. We add the constraint that agents must return to the starting region (π_H), as well as obstacle avoidance constraints. In general, the constraint on obstacle avoidance does not distribute across the entire team, but including this constraint for each sub-team after distributing the formula ensures that all agents successfully avoid obstacles. Returning to the starting region permits sub-teams to share measurements after all planners have completed. Further, returning to a starting region allows for the team to redistribute itself according to a new specification. That is, the specifications considered in this paper are "stepping-stones" to missions over long horizons in which agents periodically meet to share measurements. A transition is added from the set of final states in A_i^{loc} to a new final state, so that the mission terminates only if the agents have returned to the starting region.Next, a product automaton $\mathcal{P}_i$ is constructed from A_i^{loc} and TS_{c_i}.

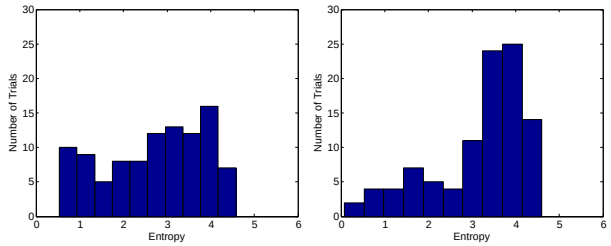
The receding horizon algorithm (Alg. 3) uses $\mathcal{P}_i$, the initial belief b^0, a lookahead horizon h, an action horizon n, and a deadline T to plan the motion of each sub-team on-line. The algorithm first finds the set of states that are reachable in the time left before the deadline (line 4). This produces h sets of states autStates[i] such that each state $\chi\in$ autStates[i] is reachable in i steps from the current state of the automaton and can reach an accepting state within the remaining budget $T-k-i$. Next, this set of states is used to construct a finite MDP in the belief space.This algorithm combines the motion and budget constraints and applies the Bayes filter for each possible sequence of h actions and h observations that can be realized from the current state. Finally, a policy μ is generated using Bellman iteration to minimize expected entropy over the horizon h (line 6). This policy is followed for the duration of the action horizon, n, at which point the planning process is repeated. The algorithm terminates when a final state is reached. The time complexity of this algorithm is $O\left(|Act_{c_i}|^h|R_Y|^{h|c_i|}|S|\lceil\frac{T}{n}\rceil\right)$. Details of this algorithm, including a proof that it is guaranteed to satisfy the given scLTL specification, are available in [9].

Algorithm 3 Receding horizon DP algorithm

1: **function** RECEDINGHORIZONDP($\mathcal{P}_i$,χ^0,b^0,h,n,T)
2: $\chi=\chi^0; b=b^0; k=0$
3: **while** $\chi\notin F_{\mathcal{P}_i}$ **do**
4: autStates = REACHABLE $(\mathcal{P}_i,\chi,h,T-k)$
5: MDP = BUILDMDP $(\mathcal{P}_i,\chi,b,$ autStates$)$
6: μ = BELLMANITERATION (MDP)
7: **if** $k\geq T-h$ **then**
8: $n=T-k$
9: **for** $i=1$ to n **do**
10: $(\chi,b)=$ result from applying $\mu\left(i,(\chi,b)\right)$
11: $k++$

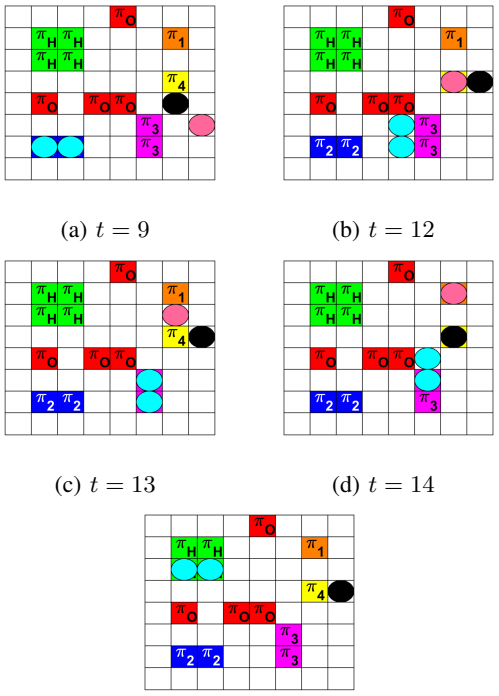
V. SIMULATION AND RESULTS

We simulated our running example on a PC with a 1.9 GHz processor and 8 GB of memory. We ran 100 simulations each using our receding horizon algorithm implementation and using a random walk on paths guaranteed to satisfy the specification. For both the receding horizon and the random walk simulations, the target was initialized randomly. The total time to run 100 receding horizon simulations was 2057



(a) (b)

Fig. 1: Histogram of entropy results for 100 simulations for (a) receding horizon and (b) random walk approaches.



(a) $t = 9$ (b) $t = 12$

(c) $t = 13$ (d) $t = 14$

(e) $t = 20$

Fig. 2: Sample run from $t = 0$ to $t = 20$. Pink circle is agent 1, blue circles are agents 2 and 3, and black circle is target.

seconds, or about 21 seconds per simulation. The product automata for all agents were computed before running the simulations. The product automaton for agent 1 had 320 nodes and 1008 edges and took 20 seconds to compute. The product for agents 2 and 3 had 1800 nodes and 14,422 edges and required 3485 seconds to compute. Simulation results are summarized in Fig. 1. Fig. 1a shows results for the receding horizon approach, having average entropy of 2.71 bits. The random walk results are in Fig. 1b, with average entropy of 3.17 bits. The difference in mean entropy was statistically significant, with a Student's t-test with 198 degrees of freedom yielding a p-value of 0.004 ($t(198) = -2.90$, $p = 0.004$). Thus the receding horizon algorithm shows better performance on accepting runs on the product automaton than the random walk approach.

VI. Conclusion

The work presented in this paper expands on previous work in single agent informative path planning under temporal logic constraints. We present a novel method for distributing tasks among groups of agents with a restrictive communication model and give a framework for incorporating these methods into a single, flexible algorithm. While connectivity is not necessarily maintained during the entire mission, our formulation permits some global coordination by dividing the team into sub-teams that act independently. We consider a team of heterogeneous agents with homogeneous sensing, but heterogeneous sensing capabilities easily fit into our methodology. We plan to extend this work to a persistent monitoring setting, in which team behavior can be optimized over an infinite horizon. Other future areas for future work are methods of data fusion without requiring sharing of individual measurements and operation in unknown topology.

References

[1] 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.

[2] M. Schwager, P. Dames, D. Rus, and V. Kumar, "A multi-robot control policy for information gathering in the presence of unknown hazards," in *Proceedings of the International Symposium on Robotics Research (ISRR)*, 2011.

[3] J. Binney, A. Krause, and G. Sukhatme, "Informative path planning for an autonomous underwater vehicle," in *Robotics and Automation (ICRA), 2010 IEEE International Conference on*. IEEE, 2010, pp. 4791–4796.

[4] S. K. Gan, R. Fitch, and S. Sukkarieh, "Online decentralized information gathering with spatial–temporal constraints," *Autonomous Robots*, vol. 37, no. 1, pp. 1–25, 2014.

[5] A. Meliou, A. Krause, C. Guestrin, and J. M. Hellerstein, "Nonmyopic informative path planning in spatio-temporal models," in *AAAI*, vol. 10, no. 4, 2007, pp. 602–607.

[6] A. Singh, A. Krause, C. Guestrin, and W. J. Kaiser, "Efficient informative sensing using multiple robots," *Journal of Artificial Intelligence Research*, vol. 34, no. 2, pp. 707–755, 2009.

[7] B. Charrow, V. Kumar, and N. Michael, "Approximate representations for multi-robot control policies that maximize mutual information," *Autonomous Robots*, vol. 37, no. 4, pp. 383–400, 2014.

[8] G. A. Hollinger and G. S. Sukhatme, "Sampling-based robotic information gathering algorithms," *The International Journal of Robotics Research*, vol. 33, no. 9, pp. 1271–1287, 2014.

[9] A. Jones, M. Schwager, and C. Belta, "Technical report: A receding horizon algorithm for informative path planning with temporal logic constraints," *arXiv preprint arXiv:1301.7482*, 2013.

[10] Y. Chen, X. C. Ding, A. Stefanescu, and C. Belta, "Formal approach to the deployment of distributed robotic teams," *Robotics, IEEE Transactions on*, vol. 28, no. 1, pp. 158–171, 2012.

[11] A. Mazurkiewicz, "Introduction to trace theory," *The Book of Traces*, 1995.

[12] M. Kloetzer and C. Belta, "Automatic deployment of distributed teams of robots from temporal logic motion specifications," *IEEE Transactions on Robotics*, vol. 26, no. 1, pp. 48–61, 2010.

[13] M. Mukund, "From global specifications to distributed implementations," in *Synthesis and control of discrete event systems*. Springer, 2002, pp. 19–35.

[14] O. Kupferman and M. Y. Vardi, "Model checking of safety properties," *Formal Methods in System Design*, vol. 19, no. 3, pp. 291–314, 2001.

[15] T. Latvala, "Efficient model checking of safety properties," in *Model Checking Software*. Springer, 2003, pp. 74–88.

[16] D. Tardioli, "A wireless communication protocol for distributed robotics applications," in *Autonomous Robot Systems and Competitions (ICARSC), 2014 IEEE International Conference on*. IEEE, 2014, pp. 253–260.

[17] J. Hopcroft and J. Ullman, *Introduction to Automata Theory, Languages, and Computation*. Pearson Education, 2008. [Online]. Available: http://books.google.com/books?id=tzttuN4gsVgC

[18] S. LaValle, *Planning Algorithms*. Cambridge University Press, 2006. [Online]. Available: http://books.google.com/books?id=EQDrngEACAAJ