2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)
Madrid, Spain, October 1-5, 2018

Distributed Sensing Subject to Temporal Logic Constraints

Zachary Serlin[1], Kevin Leahy[2], Roberto Tron[1], and Calin Belta[1]

Abstract— **This paper considers the combination of temporal logic (TL) specifications and local objective functions to create online, multiagent, motion plans. These plans are guaranteed to satisfy a persistent mission TL specification and locally optimize an objective function (e.g. in this paper, a cost based on information entropy). The presented approach decouples the two tasks by assigning sub-teams of agents to fulfill the TL specification, while unassigned agents optimize the objective function locally. This paper also presents a novel decoupling of the classic product automaton based approach while maintaining satisfaction guarantees. We also qualitatively show that optimality loss in the local greedy minimization due to the TL constraints can be approximated based on specification complexity. This approach is evaluated with a set of simulations and an experiment of 6 robots with real sensors.**

I. Introduction

In this work, we consider a team of agents concurrently tasked with a persistent mission given as a temporal logic (TL) formula over regions in their workspace and an objective function to optimize locally. Objective functions, such as information gathering, often dictate local behaviors that are difficult to optimize in the formal framework of TL planning. This problem arises in scenarios ranging from autonomous crop monitoring to disaster zone situational awareness. In a disaster zone, for example, a team of drones can be tasked with a persistent mission of "Take pictures of hospital A and home B infinitely often, do not visit hospital A until hospital C has been visited, and estimate flood water levels throughout the area." The first sections of this task are well suited for TL specifications but, it is not clear how estimating flood water levels can be stated as a TL specification. When posed as an information gathering task however, estimating flood water levels becomes straightforward.

This work builds upon both [1] and [2], where [1] creates a framework for subteams satisfying TL specifications and [2] considers a single agent combining temporal logic and objective function based tasks in a receding horizon approach. In [1], subteams are determined a priori, meaning agents tasked with satisfying the TL specification may be suboptimally positioned (i.e. they must waste considerable resources to reach desired regions). One goal of this work is to allocate subteams online to satisfy parts of a TL specification in a way that minimizes the total distance traveled by the subteam. [2] integrates both sensing and TL specifications in a receding horizon manner for a single agent; however, it does not consider multiple agents or more general objectives.

[1]Z. Serlin, R. Tron, and C. Belta are with the Department of Mechanical Engineering at Boston University, Boston, MA USA. Email: `zserlin@bu.edu, tron@bu.edu, & cbelta@bu.edu`

[2]K. Leahy is with the Massachusetts Institute of Technology Lincoln Laboratory, Lexington, MA USA. Email:`kevin.leahy@ll.mit.edu`

Prior approaches have leveraged knowledge from the TL specification (specifically the specification automaton that encodes the TL constraint) to reduce the computational and memory complexities of classic (product automaton based) approaches. Both [3] and [4] use sampling-based methods to generate an abstracted workspace with paths that satisfy the TL specification. This is done by guiding the sampling process, using a search heuristic based on regions from the specification, to choose samples that satisfy the specification. These approaches are computationally efficient, but do not consider known environment abstractions, multiple agents, or secondary objectives.

Much of the distributed sensing problem can be solved using approaches from decentralized partially observable Markov decision process (DEC-POMDP) literature [5], [6], but these approaches can be extremely computationally complex, and known to be NEXP-complete (and therefore intractable to calculate) in the worst cases [7]. Recently, progress was made in splitting DEC-POMDPs into manageable, local, actions [5]; however, they still do not provide guarantees over mission plans like TL specifications. Beyond DEC-POMDPs, many approaches have been proposed for similar local objective function optimizations, including receding horizon control [1], [2], [8], gradient ascent/decent [9], single-step horizon planning [10], and sampling based methods [11], [12]. The distributed sensing approach used in this work is inspired from [9], where information entropy is minimized in a distributed and locally optimal way.

In our formulation, we assume all agents in the workspace are not required at all times to satisfy the TL specification. This allows agents to distribute the TL task in real-time, while preforming a locally optimal objective function optimization if a more specific task is not required. We assume agents maintain connectivity; which is realistic for many current robotic systems, such as the previously mentioned disaster relief scenario or in ADS-B systems for UAVs [13]. Combined, these assumptions allow us to change the approach taken in previous TL planning literature, such as [1], [2], [14], [15], [16]. In these and similar works, TL planning is performed top down, in a computationally intense manner, and includes planning for information gathering. Here, our assumptions allow us to relax the computation required for TL planning, allowing for more flexible information gathering strategies. Thus, we can employ existing distributed information gathering algorithms while satisfying TL constraints.

The primary contributions of this work are: First, we present an approach for efficiently distributing a TL specification, defined over regions in the workspace, among a team of robots such that, when employing a secondary goal of local

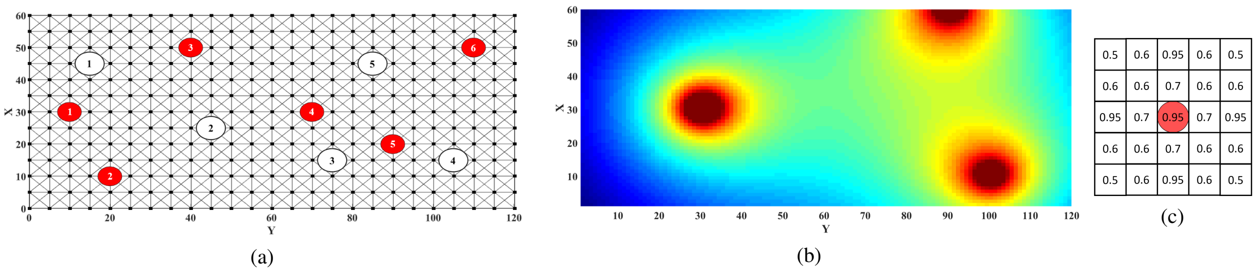


Fig. 1. (a) Finite environment Q and agent positions $x_i[0] \forall i \in \mathcal{I}$. Vertices labeled from Π are shown as white circles and agent position vertices are shown as red circles. Vertices in $\ell^{-1}(\epsilon)$ are the small black nodes. (b) Time-invariant scalar function $F(v) \forall v \in V$ (c) Sensor array and corresponding probabilities for a single agent. The red circle is again the agent position.

optimization of an objective function, global optimality loss is predictable and approximately proportional to the TL specification complexity. Second, a strategy is presented for replacing the classic product automaton in TL planning with planning on a simpler automaton (in this case a Büchi automaton) that then uses the A* planning algorithm to generate motion plans in the workspace that maintain satisfiability guarantees of classic approaches.

This paper is organized into three sections. In Sec. II, we formulate the problem of deploying a multi-agent team to both satisfy a TL specification and optimize a local objective function. Our solution is presented in Sec. III where we formalize how the TL specification is satisfied, sub-team agents are chosen, and agents minimize information entropy locally. In Sec. IV, we present simulations that verify our approach's impact on the optimality of the sensing algorithm based on specification complexity and a full scale experiment with 6 ground robots estimating the RGB values of a projected image.

II. Problem Formulation

In this section, we formulate the general case for combining TL specifications with a local objective function optimization. We first consider a general formulation of the problem and then detail one variant in Sec. III.

Notation: For a set Ω, $|\Omega|$ and 2^{Ω} are its cardinality and power set, respectively. For a set of symbols Σ, a word is a sequence $w = \sigma_1, \sigma_2, \ldots$ of symbols $\sigma_j \in \Sigma$ and Σ^{ω} is the set of all infinite words over Σ. The Cartesian product of two sets, A and B, is $A \times B$, and the product taken n times is $A^n = A \times A \ldots \times A$. The set difference between sets A and B is denoted $A \setminus B$, and the union is $A \cup B$. For an undirected graph $C = (V, \mathcal{E}_C)$, V is a set of vertices, $\mathcal{E}_C \subseteq V \times V$ is the set of edges (which is symmetric), and $\mathcal{N}(v) = \{u \in V : uv \in \mathcal{E}_C\}$ is the neighborhood of adjacent vertices to a vertex v. For an observation $y_i[k]$ or position $x_i[k]$, $[k]$ denotes the time step index, and subscript i denotes the agent index.

A. Environment

We model agent motion in its workspace as a finite graph. Such abstractions, although conservative, exist for realistic robot motion dynamics [17]. This is a common technique for simplifying complex, realistic, robot dynamics [18]. Under this assumption, the workspace is an undirected graph $Q = (V, \mathcal{E}_Q)$. We define a set of labels $\overline{\Pi} = \{\Pi, \epsilon\}$, where $\Pi = \{\pi_1, \pi_2, \ldots, \pi_n\}$ is a set of properties and ϵ is an empty property, that are mapped to the vertices in V with a labeling function $\ell : V \to \overline{\Pi}$. In other words, $\ell(v) = \pi_i$ means v is labeled with (and therefore satisfies when visited) property π_i, while $\ell(v) = \epsilon$ means visiting v does not satisfy any property in Π. The inverses of this function, $\ell^{-1}(\cdot)$, returns the set of vertices in V that satisfy the specified property. In this paper, the symbols in Π are simply numeric labels; however, Π can be expanded to include semantic labels (e.g. "charge," "obstacle," or "take a picture"). We make two key assumptions for the graph Q.

Assumption 1: All vertices in $\ell^{-1}(\epsilon)$ are connected.

Assumption 2: All vertices in $\ell^{-1}(\Pi)$ are reachable from $\ell^{-1}(\epsilon)$.

These assumptions are illustrated in Fig. 1, where any vertices with labels from Π (shown in red) can be reached from vertices with no labels (shown as black dots).

B. Agents

We consider a team of m agents labeled from a set $\mathcal{I} = \{1, 2, \ldots, m\}$ moving on the undirected graph Q. The location of each agent i at time k is $x_i[k] \in V$ and the position of all agents at a given time is $x[k] \in V^m$. We assume each agent can move to any adjacent node $\mathcal{N}(x_i[k])$ in one time step and all agents move synchronously. Each agent is initialized to $x_i[0] \in \ell^{-1}(\epsilon) \forall i \in \mathcal{I}$. The trajectory of agent $i : x_i[0]x_i[1] \ldots \in V^{\omega}$ generates a word over $\overline{\Pi}$: $\ell(x_i[0])\ell(x_i[1]) \ldots \in \overline{\Pi}^{\omega}$.

Assumption 3: Agents communicate in an undirected manner and remain connected to all other agents in $\mathcal{I}$, through at least one other agent, at all times.

Assumption 4: All agents can satisfying any property in Π and only one agent is required to visit $\ell^{-1}(\pi_i)$ for property π_i to be satisfied.

C. Specifications

Temporal logic constraints guarantee specific agent behaviors (e.g. visiting multiple vertices simultaneously or avoiding a set of vertices). The motion of the agents is constrained by a mission specified using linear temporal logic (LTL) formulas over Π. Given the set Π, an LTL formula is defined inductively as

$$\phi ::= \pi | \neg\pi | \phi_1 \wedge \phi_2 | \phi_1 \vee \phi_2 | \phi_1 \mathcal{U} \phi_2 | \Box\phi_1 | \Diamond\phi_1 | \bigcirc\phi_1 \ ,$$

where $\pi \in \Pi$ is an atomic proposition, ϕ_1 and ϕ_2 are LTL formulas, and $\wedge$ (conjunction), $\vee$ (disjunction), $\mathcal{U}$ (until), $\Box$ (always), $\Diamond$ (eventually), and $\bigcirc$ (next) [15], [19]. The semantics of LTL is defined over infinite words, $w \in (2^{\Pi})^{\omega}$. A complete description of the semantics of LTL is outside the scope of this work, but can be found in [19].

D. Sensing

Agents collectively perform a secondary distributed sensing task that does not involve properties from Π, but is focused on estimating an a priori unknown, time-invariant, scalar function $F : V \rightarrow \mathbb{R}$ that returns values in a finite set $Y \subseteq \mathbb{R}$. At each time k, each agent measures $F(x_i[k])$ with a noisy sensor, generating an observation $y_i[k] \in Y$, based on the sensing probability $P(y_i[k]|F(x_i[k])$, which is intrinsic to each sensor. We denote all sensor observations at a given time step as $y[k]$ and the estimate of $F(v) \forall v \in V$ based on all past measurements $y[0:k-1]$ as $F_e[k]$ (see Sec. III-E). If an agent has a sensor array covering multiple vertices, as shown in Fig. 1-C, an observation is generated at each vertex based on the corresponding sensing probabilities. The agents locally optimize a sensing function

$$h\left(V, y[0:k-1]\right) \ , \tag{1}$$

where $h(\cdot)$ is an objective function, such as mutual information or Shannon entropy, that depends on past measurements. In (1), h is a function of the teams' discrete observations. This paper focuses on a distributed sensing task, however h can be generalized to other objective functions, such as potential functions.

Example 1: Consider six robots operating in a rectangular, undirected, and 8-way connected, lattice graph (see Fig. 1-a), with five labeled "regions" or vertices. In this case, $\mathcal{I} = \{1, ..., 6\}$, $\Pi = \{1, ..., 5\}$, and the agents are initialized to $x_i[0] \in \ell^{-1}(\epsilon) \forall i \in \mathcal{I}$ as shown. Fig. 1-b shows the true $F(v) \forall v \in V$ scalar function that is initially unknown by the agents. Finally, Fig. 1-c shows $P(y_i[k]|F(x_i[k])$ for a single agent. Agents are tasked with the LTL formula

$$\phi = \Diamond \pi_1 \wedge (\neg \pi_2 \mathcal{U} (\pi_1 \wedge \pi_5)) \wedge \Box \Diamond \pi_3 \wedge \Box \neg \pi_4, \tag{2}$$

which states "eventually visit 1, do not visit 2 until you have visited 1 and 5, visit 3 infinitely often, and never visit 4."

Agents sample $F(x_i[k]) \forall i \in \mathcal{I}$, and receive measurements, $y[k]$, between $\{0, \ldots, 255\}$ (i.e. $Y = \{0, \ldots, 255\}$). The probability of $y_i[k]$ being correct is shown in Fig. 1-c and represents a binary "sensed correctly" outcome. ■

E. Problem Statement

Problem 1: Given a team of agents $\mathcal{I}$ with the capabilities described in Sec. II-A – II-D that operate in an environment graph Q, solve

$$\min_{x[0]:x[k-1]} h\left(V, y[0:k-1]\right) \ , \tag{3}$$

at each time k, such that LTL formula ϕ over Π is also satisfied.

III. Solution

This solution consists of two components: satisfying the LTL specification ϕ and greedy optimization of h. We propose locally decoupling these components by allowing agents to follow information gathering strategies when not actively making progress towards satisfying ϕ.

To accomplish this, agents located nearest to $\ell^{-1}(\pi_i)$ for an i that makes progress in ϕ are assigned to a sub-team. Agents use an online, distributed, election algorithm, similar to FloodMax [20] and the Hungarian algorithm [21], to determine the best set of agents to make progress toward satisfying ϕ (see Sec. III-C). The distance to acceptance metric from [8] (see Sec. III-A) is used to determine how to make progress toward satisfying ϕ. The remaining agents locally minimize h and, to prevent visiting unintended regions, avoid all vertices in $\ell^{-1}(\Pi)$. Here, agents minimize information entropy, as in [9], to gather information about the environment in a locally optimal way. All agents take sensor measurements at every time step regardless of objective. Once progress towards satisfaction has been made, the election algorithm begins again, and the process is repeated infinitely.

A. Satisfying LTL constraints

The LTL specification ϕ is translated to a *non-deterministic Büchi automaton*[1](NBA) that accepts the language satisfying ϕ. A Büchi automaton is a tuple $\mathcal{B} = (S, S_0, \Pi, \delta, \mathcal{F})$, where S is a finite set of states, S_0 is the finite set of initial states, Π is an input alphabet, $\delta : S \times 2^{\Pi} \rightarrow 2^{S}$ is a transition relation, and $\mathcal{F} \subseteq S$ is the set of accepting states. The NBA takes as input a word $w = \sigma_1, \sigma_2, \sigma_3, \ldots$, with $\sigma \in 2^{\Pi}$, inducing a run of states $s_1, s_2, s_3, \ldots$ from S starting in S_0, and $\delta\left(s_j, \sigma_j\right) = s_{j+1}$, where j is the index of output state sequence. Such a sequence of symbols is accepted by $\mathcal{B}$ if $\mathcal{F}$ is visited infinitely often.

Agents tasked with making progress in ϕ form a temporary sub-team of agents $\tau \subseteq \mathcal{I}$. Agents in τ move to generate a single symbol $\sigma_j \in 2^{\Pi}$. If multiple propositions form a single symbol (i.e. $\sigma_j = \{\pi_1, \pi_2\}$) then σ_j is generated when all specified vertices are visited (or not) simultaneously); more formally, when $\bigcup_{i \in \tau} \ell(x_i[k]) = \sigma_j$.

Remark 1: Classically, a product automaton is created from the Cartesian product of some transition system for the workspace and the above Büchi automaton [1], [15]. Planning is done on this product automaton, which can become intractably large in systems with multiple agents or large environments. We attempt to avoid this intractability by splitting the specification and workspace planning. This split is possible by *Assumptions 1 – 4*, which form the basis for *Proposition 1*. The costs of these assumptions are restrictions on both agent heterogeneity (*Assumption 4*), and the graph Q (*Assumptions 1-2*).

Example 1 (cont.): Below, in Fig. 2, is the Büchi automaton corresponding to (2), generated from an off-the-shelf conversion tool [22]. Starting in s_0, the symbol enabling a transition to s_2, σ_1, would be represented as $\sigma_1 = \{\pi_1, \pi_5\} = \{1, 5\}$

[1]There exist off-the-shelf tools to perform this translation, see [22], [23].

and the corresponding agents designated to service σ_1 could be $\tau = \{1, 4\}$ (see Fig. 1-A). Symbol σ_1 would be generated when $\bigcup\limits_{i \in \tau} \ell(x_i[k]) = \sigma_1$. ■

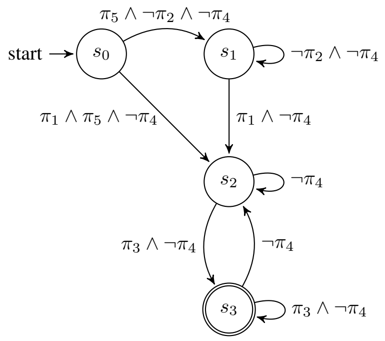


Fig. 2. Büchi Automaton for (2)

At each state in $\mathcal{B}$, transitions exist that are enabled by the generation of symbols. The choice of which symbols to generate is decided using the distance to acceptance (DTA) metric employed in [2], [8]. The DTA represents the minimum number of states required to reach an accepting state $s_a \in \mathcal{F}$ from a given state s_j and is a local minimum at states in $\mathcal{F}$. It is shown in [8] that this potential-like function, if minimized over transitions, will drive $\mathcal{B}$ to visit a satisfying state infinitely often (thus satisfying ϕ). The distance in $\mathcal{B}$ between any two states is defined as

$$d(s, s') = \begin{cases} \min\limits_{\mathbf{p} \in \mathcal{D}(s,s')} \mathcal{L}(\mathbf{p}), & \text{if } \mathcal{D}(s, s') \neq \emptyset \\ \infty, & \text{if } \mathcal{D}(s, s') = \emptyset \end{cases}, \tag{4}$$

where $\mathcal{D}$ is the set of states between states s and s', $\mathbf{p}$ is a single path through $\mathcal{D}$ ($\{\mathbf{p} = s_1 s_2 s_3 \dots s_n | s_1 = s, s_n = s', s_j \xrightarrow{\sigma} s_{j+1}, \forall n \geq 0, \forall j \in 0, n-1\}$), and $\mathcal{L}(\mathbf{p})$ is the number of states in $\mathbf{p}$. DTA is then found, offline, for each state in $\mathcal{B}$, by

$$d_a(s_j) = \min_{s_a \in \mathcal{F}} d(s_j, s_a). \tag{5}$$

When agents must decide which state to transition to, they choose the next state with the lowest DTA. More formally,

$$s_{j+1} = \arg\min_{s} d_a(s) \quad \forall s \in \mathcal{N}(s_j). \tag{6}$$

Once s_{j+1} is determined, the corresponding σ_j dictates the next generated symbol (where $\delta(s_j, \sigma_j) = s_{j+1}$). If two possible states have the same DTA, the state with the lower transition cost is selected. This cost depends on the agents assigned to τ and is detailed in Sec. III-C.

B. Motion Planning

The importance of a travel cost for our solution is two-fold. Agents require both a path through graph Q to a desired vertex, and a reliable estimate of the cost of traveling there. This is a mature area of research, and there exist many planning tools to accomplish this task [24], [18]. A brief overview of the employed cost estimate is provided below.

Given graph Q, we assume an agent can move to a vertex subject to cost

$$D_{i,n} := L(x_i, n), \tag{7}$$

where $D_{i,n}$ is the cost for agent i to move to n, the nearest vertex in $\ell^{-1}(\pi_n)$, and L is the shortest path length between two vertices in graph Q. In this work, L is computed using the A* algorithm [24] with a heuristic $f(v)$ given as

$$f(v) = g(x_i, v) + j(v, n), \tag{8}$$

where $v \in V$, $g(x_i, v)$ is the path length from the position of agent i to vertex v, and $j(v, n)$ is the Euclidean distance between v and the nearest node in $\ell^{-1}(\pi_n)$. When computed for a specific agent-region pair, the algorithm considers all other regions as obstacles. The cost $D_{i,n}$ of $|\mathcal{I}|$ agents to $|\Pi|$ "regions" is denoted as D.

This section assumes that multiple agents can occupy the same node in graph Q, which is unrealistic in most situations. This problem is addressed later in the Sec. IV.

C. Sub-team Election

Once the next state s_{j+1} and transition σ_j are determined using (6), agents must elect (online) the nearest sub-team τ to service regions in σ_j to minimize cost J subject to $\bigcup\limits_{i \in \tau} \ell(x_i[k]) = \sigma_j$. To do this, all agents in $\mathcal{I}$ must first distribute the updated cost information D.

To distribute D, agents individually determine local costs J and share them with neighboring agents in a manner similar to the FloodMax algorithm [20]. Agents share both their calculated D, and all information they have received from other agents. This approach, given *Assumption 3*, guarantees agents will receive a fully updated D matrix in maximum $|\mathcal{I}|$ rounds of information sharing [20].

Once σ_j and D have been updated, each agent uses the Hungarian algorithm [21] to elect the nearest agents to regions of interest in σ_j to τ (i.e. if $\sigma_j = \{\pi_1, \pi_2\}$ then π_1 and π_2 are the regions of interest to be serviced). The cost of a transition for a given σ and τ pair is

$$C(\sigma, \tau) = \sum_{i=1}^{|\tau|} J_{\tau^i, \sigma^i}, \tag{9}$$

where τ^i and σ^i are the i^{th} element in the sets. If the costs are identical, the lower state index is chosen. Agents use the same algorithms and data in their calculations, so they arrive at the same result in a distributed manner. Each agent checks if its index appears in τ, and if so, uses A* to determine the exact path sequence to its assigned region. Each agent in τ evaluates the path length for all agents in τ and, if they are not equivalent, waits for $D(x_i, n) - \min(D(\tau, \sigma_j))$ time steps.

D. Satisfiability

Proposition 1: Generation of a satisfying run in $\mathcal{B}$ is guaranteed if $|\mathcal{I}| \geq \max |\sigma_j| \forall j$, DTA is used to choose σ_j, and the A* algorithm plans agent trajectories in a graph Q with *Assumptions 1-2*.

Proof: ϕ is satisfied if a run of states in $\mathcal{B}$ visits an accepting state infinitely often. Using DTA (6) to guide the generation of symbols is guaranteed to induce such a run. To generate each symbol, all agents in τ must reach their assigned vertices simultaneously ($\bigcup_{i \in \tau} \ell(x_i[k]) = \sigma_j$). This can only be possible for all σ_j if $|\mathcal{I}| \geq \max|\sigma_j| \forall j$ and *Assumptions 1-2* hold for graph Q. This implies that any τ can reach any vertices in $\ell^{-1}(\Pi)$ without generating unintended symbols if each agent is able to find a path through graph Q. A* is guaranteed to find such a path, if one exists, since $j(v,n) \leq g(x_i,v) \forall v$ then $j(v,n)$ [24]. ϕ is therefore guaranteed to be satisfied. ■

E. Information gathering

When agents are not part of τ, they minimize h locally. As posed in Sec. II, h is plagued with "the curse of history." [9] This means as $k \to \infty$, so does the size of $y[0:k]$, making the problem computationally intractable. To solve this known problem, we employ an approach adapted from [9] to minimize conditional entropy H using the Markovianity of the process. For this paper, $h = 1 - H(F_e[k])$, where $F_e[k]$ is the estimate of scalar function $F(v) \forall v \in V$ at time k and $F_e(x_i)[k]$ is the estimate of the scalar function at vertex x_i. Below is a brief overview of [9], using the above notation, which has been shown to be locally optimal at gathering information. To compute the entire system's entropy field, agents must share their $P(y_i[k]|F(x_i)[k])$, positions $x_i[k]$, and $y_i[k]$ values at each time k. This is done in the same way the D matrix is shared (see Sec. III-C). From Bayes' Rule, the joint observation and the system's prior distribution, $P(F_e[k])$, can be used to compute the system's posterior distribution $P(F(x_i)[k]|y[k])$. We assume the robot positions are known and therefore the prior distribution prediction step is denoted $P(F_e[k+1]|y[k])$. These steps constitute a classic recursive Bayesian estimator of $F(v) \forall v \in V$ [18]. The probability that the sensor estimate is equal to some $E \in Y$ is therefore

$$\begin{aligned} &P(F_e(x_i)[k+1] = E) = \\ &\quad P(F_e(x_i)[k+1]|y_i[k]) P(F_e(x_i)[k]|y(x_i)[k]). \quad (10) \end{aligned}$$

The information entropy is then defined as

$$\begin{aligned} &H(F_e(x_i)[k]) = \\ &\quad -\sum_{E \in Y} P(F_e(x_i)[k] = E) \log_2 P(F_e(x_i)[k] = E). \quad (11) \end{aligned}$$

The resulting entropy field is then used locally to plan for each sensing agent to move in the direction of highest entropy (or lowest information), meaning

$$h = 1 - H(F_e(x_i)[k]), \quad (12)$$

because the goal is maximizing entropy using a minimization. The motion plan is then,

$$x_i[k+1] = \underset{x \in \mathcal{N}(x_i[k])}{\arg\min}\ h(x), \quad (13)$$

where ties are broken by picking one of the lowest valued neighbors randomly.

F. Qualitative Loss of Optimality Approximation

To compare the impact of various LTL specifications on the distributed sensing task, we find a qualitative approximation of the optimality loss based on LTL specification complexity. We begin by defining the change in h per time step k as

$$\begin{aligned} \zeta[k+1] &= \sum_{x \in V} h[k+1] - h[k] \\ &= \sum_{i=1}^{m} h(x_i[k+1])[k+1] - h(x_i[k+1])[k], \quad (14) \end{aligned}$$

where $\zeta[k+1]$ is the sum of changes in h based on only agent measurements. We denote $\zeta^\star[k+1]$ the greedy optimal policy (for one time step, holding h constant, and where all agents follow (13)). The optimality loss, $(\zeta^\star - \zeta)$, arises from the LTL constraints forcing agents to behave sub-optimally and can be approximated by

$$\zeta_R[k] = \zeta^\star[k] - \zeta[k] \gtrapprox \zeta^\star[k] \frac{\max|\inf(w)|}{m}, \quad (15)$$

where $\zeta_R[k]$ is the optimality loss, or regret, and $\inf(w)$ refers to the set of infinitely repeated symbols in a word that satisfies ϕ. We assume each agent can make the same contribution to the decrease in the total uncertainty. As $k \to \infty$, $\max|\inf(w)|$ is the maximum number of agents removed from the optimal policy. Equation (15) does not strictly bound ζ_R due to sensor measurement uncertainty, causing ζ and $\zeta^\star$ to chatter around zero as $F_e[k]$ approaches $F(v) \forall V$. As $k \to \infty$, $F(v) \forall v \in V$ will be measured infinitely many times, meaning ζ_R will fluctuate, from sensor noise, around zero. Therefore, to compare different LTL specifications' impacts on sensing, we evaluate the average change in h per time step per agent over a finite time window, given by

$$\overline{\zeta} = \frac{\sum_{k=0}^{K} \zeta[k]}{Km}, \quad (16)$$

where K is some finite time. From (15) and (16), $\overline{\zeta}$ can be approximated, based on LTL specification complexity, as

$$\overline{\zeta} \approx \overline{\zeta^\star} \frac{m - \max|\inf(w)|}{m}, \quad (17)$$

where $\overline{\zeta^\star}$ is (16) evaluated for $\zeta^\star$. Fig. 4 plots both this proposed approximation and the actual simulation data for $K = 1000$. This approximation is conservative as it assumes agents that are satisfying LTL constraints are not making sensor measurements (which is not the case). While our current analysis is qualitative, we plan to investigate more rigorous bounds in the future.

IV. Simulations and Experiments

The above algorithms are implemented in both simulation and in an experimental team of robots. Above, agents are allowed to occupy the same position, however this is not possible in practice. For agents in τ to avoid collision, each looks one time step ahead for possible collisions with other agents in τ. If a collision is imminent, the agent with the

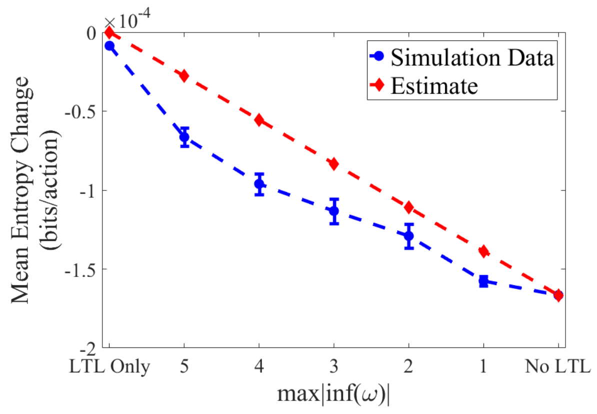


Fig. 3. Mean entropy change per action based on algorithm and specification choice. The x-axis represents changes in LTL specification complexity, starting with requiring all agents for the LTL task on the left ($\max|\inf(w)|= m$) and going to all agents are following information entropy minimization on the right ($\max|\inf(w)|= 0$). The estimate from (17) is also shown.

lower label index in $\mathcal{I}$ replans its route with the higher index agent as an obstacle. The higher indexed agent then waits until the lower indexed agent is more than one node away before continuing. τ agents must compensate for longer paths accordingly. Agents not in τ use their knowledge of all agent positions to inflate their estimate field. The vertex position of each agent, and all of its neighboring vertices are artificially set to a value greater than one before (13) is calculated. This turns agents into obstacles and they are avoided. The agents then reset the inflated values and proceed. Taken together, these approaches guarantee there are no collisions.

A. Simulations

The test case (Example 1) is a 5 region and 6 agent case with an LTL formula as shown in (2). The initial agent positions are shown in Fig. 1-A, and the unknown scalar function $F(v)$ is shown in Fig. 1-B. Five hundred trials were conducted for the above algorithms with variations of (2), such that $\max|\inf(w)|= \{1, 2, 3, 4, 5\}$, the LTL algorithms without distributed sensing, and the distributed sensing algorithms without LTL constraints. The information-gathering-only algorithm treats the regions from the prior LTL specification as obstacles and the LTL constraints only algorithm hold unused agents stationary. For each trial, $\bar{\zeta}$, in (bits/action), is recorded with $K = 1000$ and the average, upper, and lower values from these trials are shown in Fig. 3. Each trial is run for a thousand time steps and the initial agent locations and region locations remain stationary between trials. The approximation from (17) is also shown in Fig. 3.

The information gathering algorithm, which has been shown to be optimal in distributively minimizing information entropy [9], has the most efficient reduction in entropy per action. Satisfying the LTL constraints without minimizing information entropy, as expected, had a near zero reduction in information entropy. The increase in $\max|\inf(w)|$ reduces ζ, and the approximation from (17) allows for different LTL specifications to be compared for their impacts on $\bar{\zeta}$.

B. Experiments

We further verified our approach with a full-scale experiment using a 6 robot team of identical, modified, iRobot Create2 ® ground robots (see Fig. 4-B). These robots are each controlled with a Raspberry Pi 3 Model B with a 64-bit, 1.2GHz, processor running Ubuntu 16.04 MATE and ROS Kinetic. They each have a five RGB color sensor array mounted on their top plate aligned in a pattern identical to the simulation sensor configuration in Fig. 1-C (see Fig. 4-B).

Each robot is tracked using an OptiTrack Motion Capture System in a 6 by 9 meter arena (shown in Fig. 4-A) that provides perfect position and heading information. The agents communicate over a wifi network and move via waypoint following (using a PD controller). The arena is surrounded by a set of 6 projectors that project images over the arena floor. This area (3 by 6 meters) is shown in Fig. 4 and Fig. 5-A and contains blue marked tiles in the same locations as the regions in Fig. 5-B. In this experiment, the graph Q is a 60 by 120 node lattice graph (identical in configuration to Fig. 1-A but much denser).

The projectors create a light field (an RGB image) that is sensed as the robots move. Using the above algorithms, the robots satisfy (2) while minimizing information entropy. The projected image is shown in Fig. 5-A, and the experimental sensor result after 1000 time steps (approximately 3 hours) is shown in Fig. 5-B.

The estimated sensor values in Fig. 5-B are estimated using five TCS34725 RGBC sensors mounted on the top of each robot. These sensors have an IR filter, measure levels of red (R), green (G), blue (B), and white (C) light, and have the approximate sensor accuracies shown in Fig. 1-C. The limiting factor for run-time in this experiment was the time required for each sensor to gather photons and return measurements (~ 0.75 seconds per sensor). The sensor locations at any time were determined using nearest neighbor approximation onto graph Q meaning each sensor value is accurate to within 2.5

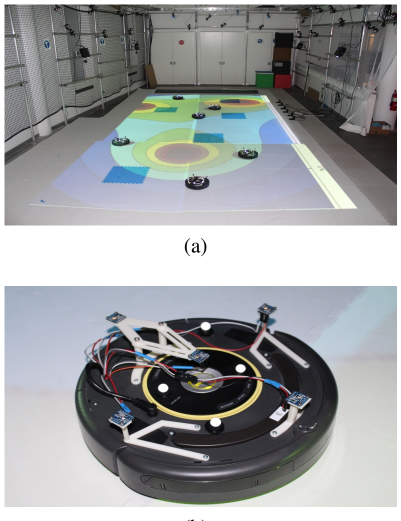
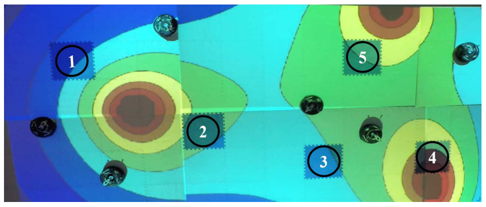
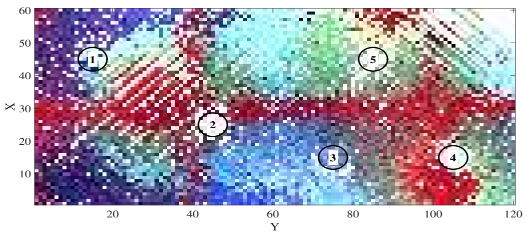


Fig. 4. (a) The experimental arena with 6 robots, the projected light field, and OptiTrack motion capture system. (b) Modified iRobot Create2 ® robot with array of five TCS34725 RGBC light sensors and motion capture tracking markers.



(a)



(b)

Fig. 5. Experimental results after 1000 time steps. (a) The experimental arena with projected light field (RGB image). (b) The experimental sensor estimates of the projected light field at k = 1000.

centimeters. To determine the appropriate RGB value shown in Fig. 5-B, the RGB values are corrected for saturation by dividing by the (C) value from the sensor. A noticeable band of darker color at the edges of each projector appears in Fig. 5-B because the angle of incidence on the sensors is large there. This effect is further accentuated because the sensors are slightly raised above the robot, meaning they leave the projection envelope where two projectors meet. The projected field estimate is shown in Fig. 5-B, and correlates well with the projected image (Fig. 5-A).

V. Conclusion

In this paper we consider a multiagent persistent planning problem coupled with an optimization problem. The presented approach guarantees the mission satisfaction; while also providing a qualitative approximation of the distributed sensing optimality loss based on temporal logic specification complexity. This is primarily done by assigning agents to generate the necessary NBA state transitions based on minimal travel cost J and the distance to acceptance metric. This frees temporarily unused agents to accomplish a secondary objective function optimization constraint, which in this case is locally minimizing information entropy. The qualitative approximation of optimality loss in sensing based on temporal logic constraint complexity is shown to be conservative but reasonable based on simulation results. The approach is experimentally tested with six robots and a simple LTL specification that requires at most two robots to produce a given state transition. Overall, this approach explores a novel approach to enforcing temporal logic constraints, by splitting specification automaton and workspace planning, while also allowing for a secondary greedy objective function minimization.

VI. Acknowledgements

This work was supported by the National Science Foundation under grants CMMI-1400167, NRI-1426907, NRI-1734454, and IIS-1717656.

References

[1] K. Leahy, A. Jones, M. Schwager, and C. Belta, “Distributed information gathering policies under temporal logic constraints,” in *2015 54th IEEE Conference on Decision and Control (CDC)*, Dec 2015.

[2] A. Jones, M. Schwager, and C. Belta, “Information-guided persistent monitoring under temporal logic constraints,” in *2015 American Control Conference (ACC)*, pp. 1911–1916, July 2015.

[3] J. McMahon and E. Plaku, “Robot motion planning with task specifications via regular languages,” *Robotica*, vol. 35, no. 1, pp. 26–49, 2017.

[4] E. Plaku, L. E. Kavraki, and M. Y. Vardi, “Hybrid systems: from verification to falsification by combining motion planning and discrete search,” *Formal Methods in System Design*, vol. 34, no. 2, 2009.

[5] S. Omidshafiei, A. AghaMohammadi, C. Amato, S. Liu, J. P. How, and J. Vian, “Decentralized control of multi-robot partially observable markov decision processes using belief space macro-actions,” *The International Journal of Robotics Research*, vol. 36, no. 2, pp. 231–258, 2017.

[6] C. Amato and S. Zilberstein, “Achieving goals in decentralized POMDPs,” in *Proceedings of the Eighth International Conference on Autonomous Agents and Multiagent Systems*, (Budapest, Hungary), pp. 593–600, 2009.

[7] D. S. Bernstein, S. Zilberstein, and N. Immerman, “The complexity of decentralized control of markov decision processes,” *CoRR*, vol. abs/1301.3836, 2013.

[8] X. Ding, M. Lazar, and C. Belta, “LTL receding horizon control for finite deterministic systems,” *Automatica*, vol. 50, no. 2, pp. 399 – 408, 2014.

[9] B. J. Julian, M. Angermann, M. Schwager, and D. Rus, “Distributed robotic sensor networks: An information-theoretic approach,” *The International Journal of Robotics Research*, vol. 31, no. 10, pp. 1134–1154, 2012.

[10] M. Schwager, P. Dames, D. Rus, and V. Kumar, *A Multi-robot Control Policy for Information Gathering in the Presence of Unknown Hazards*, pp. 455–472. Cham: Springer International Publishing, 2017.

[11] B. Charrow, V. Kumar, and N. Michael, “Approximate representations for multi-robot control policies that maximize mutual information,” *Autonomous Robots*, vol. 37, pp. 383–400, Dec 2014.

[12] A. Ruiz, T. Wiedemann, C. Manss, L. Magel, J. Mueller, D. Shutin, and L. Merino, “Decentralized multi-agent exploration with online-learning of gaussian processes,” in *IEEE International Conference on Robotics and Automation*, 2016.

[13] M. Guterres, S. Jones, G. Orrell, and R. Strain, “Ads-b surveillance system performance with small uas at low altitudes,” tech. rep., The MITRE Corporation, McLean, VA, 2017.

[14] Y. Chen, X. C. Ding, A. Stefanescu, and C. Belta, “Formal approach to the deployment of distributed robotic teams,” *IEEE Transactions on Robotics*, vol. 28, pp. 158–171, Feb 2012.

[15] C. Belta, B. Yordanov, and E. Gol, *Formal Methods for Discrete-Time Dynamical Systems*. Studies in Systems, Decision and Control, Springer International Publishing, 2017.

[16] M. Guo, J. Tumova, and D. V. Dimarogonas, “Cooperative decentralized multi-agent control under local ltl tasks and connectivity constraints,” in *53rd IEEE Conference on Decision and Control*, Dec 2014.

[17] K. Leahy, D. Zhou, C. Vasile, K. Oikonomopoulos, M. Schwager, and C. Belta, “Provably correct persistent surveillance for unmanned aerial vehicles subject to charging constraints,” in *2014 International Symposium on Experimental Robotics (ISER2014)*, June 2014.

[18] S. M. LaValle, *Planning Algorithms*. Cambridge, U.K.: Cambridge University Press, 2006.

[19] C. Baier and J.-P. Katoen, *Principles of Model Checking (Representation and Mind Series)*. The MIT Press, 2008.

[20] N. A. Lynch, *Distributed algorithms*. Morgan Kaufmann, 1996.

[21] H. W. Kuhn, “The hungarian method for the assignment problem,” *Naval Research Logistics Quarterly*, vol. 2, no. 1-2, pp. 83–97, 1955.

[22] P. Gastin and D. Oddoux, “Fast LTL to Büchi automata translation,” in *Proceedings of the 13th International Conference on Computer Aided Verification (CAV’01)* (G. Berry, H. Comon, and A. Finkel, eds.), vol. 2102 of *Lecture Notes in Computer Science*, (Paris, France), pp. 53–65, Springer, July 2001.

[23] A. Ulusoy and C.-I. Vasile, “LOMAP LTL Optimal Multi-agent Planner,” 2000–2004.

[24] P. E. Hart, N. J. Nilsson, and B. Raphael, “A formal basis for the heuristic determination of minimum cost paths,” *IEEE Transactions on Systems Science and Cybernetics*, vol. 4, pp. 100–107, July 1968.