Distributed Informative Path Planning under Temporal Logic Constraints

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

Abstract

In this work, 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. The mission constraints are distributed automatically among sub-groups of the agents. Each subgroup 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.

1. Introduction

In this paper, we consider a group of robots that are tasked with gathering information about a large environment, e.g. scanning a forest for signs of wildfire. In order to complete this task, the agents have to satisfy certain motion constraints such as “Always avoid obstacles” and “Visit a centralized station to upload gathered data.” 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 constraint formula among sub-groups of the agents such that if each sub-group 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.

Informative path planning is the problem of choosing trajectories for mobile sensors such that the information measured by those sensors is maximized. Maximally gathering information in a distributed manner is an example of a decentralized partially observable markov decision processes (DEC-POMDPs), 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-based methods for team configurations [7] and on rapidly-exploring random graphs [8]. Our work incorporates temporal logic constraints into the path planning problem. These constraints permit richer, more realistic constraints on the motion of the agent 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]. This method provides a broad framework in which teams of agents can act independently to ensure that global, cooperative behaviors are produced. In this paper, we extend this framework to include more realistic communication constraints and allow the agents to act according to reactive control policies rather than
follow pre-specified paths. Further, we show how to distribute specifications among sub-groups of agents, called cliques, rather than among individual agents.

2. Preliminaries

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

2.1. Information Theory

For a discrete random variable \( X \) with probability mass function (pmf) \( p \), (Shannon) entropy is a measure of uncertainty defined as

\[
H(X) = - \sum_{x \in R_X} p(x) \log p(x),
\]

where \( R_X \) is the range space of the random variable \( X \) [14]. For two random variables, \( X \) and \( Y \), with joint pmf \( p(x,y) \), the conditional entropy \( H(Y|X) \) is the uncertainty in \( Y \) given that the value of \( X \) is known, and is expressed as

\[
H(Y|X) = - \sum_{x \in R_X, y \in R_Y} p(x,y) \log p(y|x),
\]

where \( p(x,y) \) and \( p(y|x) \) are the joint and conditional pmfs of \( X \) and \( Y \), respectively [14].

2.2. Discrete Models

A deterministic transition system (TS) is a tuple \( TS = (Q,q^0,\text{Act},\text{Trans},\text{AP},|\cdot|) \), where \( Q \) is a set of states, \( q^0 \in Q \) is the initial state, \( \text{Act} \) is a set of actions, \( \text{Trans} \subseteq Q \times \text{Act} \times Q \) is a deterministic transition relation, \( \text{AP} \) is a set of atomic propositions, and \( |\cdot| \subseteq Q \times 2^{\text{AP}} \) is a satisfaction relation. We define \( AP_q = \{ p \in \text{AP} | (q,p) \in |\cdot| \} \) as the set of atomic propositions satisfied at state \( q \). A finite run of a transition system is a sequence of states \( q^0q^1 \ldots q^n \) such that \( \forall i \in \text{Act} \) such that \( (q^i,a,q^{i+1}) \in \text{Trans} \) \( \forall i = 0,1, \ldots \). An output trace of a run is a word \( w = w_0w_1 \ldots \) where \( w' = AP_{q^i} \).

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

A discrete time Markov Decision Process (MDP) is a tuple \( \text{MDP} = (S,s^0,\text{Act},P) \), where \( S \) and \( s^0 \) are defined as for an MC, \( \text{Act} \) is a set of actions, and \( P : S \times \text{Act} \times S \rightarrow [0,1] \) is a probabilistic transition relation with the probability of transitioning from state \( s \) to state \( s' \) under action \( a \) given by \( P(s,a,s') \). The set of actions \( a \) available at state \( s \) is \( \text{Act}(s) \subseteq \text{Act} \) such that \( \exists s' \in S \) with \( P(s,a,s') > 0 \). A sequence of states \( s^0a_1 \ldots a_i \) with \( P(s^0,a_1 \ldots a_i) > 0 \) for \( a_i \in \text{Act}(s_i) \) \( \forall i = 0,1, \ldots, l-1 \) is called a sample path of an MDP.

2.3. Automata

In this work, we consider missions that can be specified using syntactically co-safe Linear Temporal Logic (scLTL) [15]. Given a set of atomic propositions \( AP \), an scLTL formula is defined inductively as:

\[
\phi = p | \neg p | \phi_1 \lor \phi_2 | \phi_1 \land \phi_2 | \phi_1 \Rightarrow \phi_2 | \bigcirc \phi_1 | \Diamond \phi_1,
\]

where \( p \in AP \), and \( \phi_1 \) and \( \phi_2 \) are scLTL formulae, \( \neg, \lor, \land \), and \( \Rightarrow \) are Boolean negation, conjunction and disjunction, respectively, and \( \bigcirc, \bigodot \), 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 scLTL formula. We denote the set of all words that satisfy \( \phi \) as the language of \( \phi \), denoted \( L(\phi) \) [15].

A finite state automaton (FSA) is a tuple \( A = (X,\Pi,x^0,F,\rightarrow_A) \), where \( X \) is a set of states, \( \Pi \) 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 \( \phi \), there exist off-the-shelf tools which can construct an automaton \( A_\phi \) with input language \( 2^{AP} \) such that \( L(\phi) = L(A_\phi) \).

The synchronous product of a set of automata \( A_i = (X_i,\Pi_i,x^0_i,F_i,\rightarrow_{A_i}) \) for \( i \) in index set \( I \) is the automaton \( A_P = \prod_{i \in I} A_i = (X,\Pi_P,x_P,F_P,\rightarrow_{A_P}) \) where \( X_P = \prod_{i \in I} X_i \), \( \Pi_P = \bigcup_{i \in I} \Pi_i \), \( x_P = (x^0_i)_{i \in I} \), 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 \( ((q_i)_{i \in I},(a_i)_{i \in I}) \in \rightarrow_{A_P} \iff \forall j \in I \) such that \( \pi_j \in \Pi_j \), \( (q_i,\pi_j,a_j) \in \rightarrow_{A_i} \), and \( \forall k \in I \), \( q_k = a_k \).

A product automaton between a transition system \( TS = (Q,q^0,\text{Act},\text{Trans},\text{AP},|\cdot|) \) and an FSA \( A_\phi = (X,\Pi,\chi^0,\chi,X,\phi,\rightarrow_\phi) \) is an FSA \( \phi = TS \times A_\phi = (X_{\phi},\Pi,\chi^0,\chi,X_{\phi},\phi,\rightarrow_\phi) \).

\( X_{\phi} \subseteq Q \times X \) is the state space of \( \phi \), \( \chi^0 = (q^0,\sigma_0) \) is
the initial state, and $F_\phi \subseteq Q \times F$ is the set of accepting states. The transition relation is defined as $\xrightarrow{\cdot} = \{(q,x),p,(q',x') | (q,p,q') \in Trans, (x,AP_q,x') \in \Delta\}$. The state of $\mathcal{P}$ at time $k$ is $(q^k,x^k)$, which we denote as $\chi^k$ for brevity. If $\chi^{0:t}$ is an accepting run on $\mathcal{P}$, then the associated run $q^{0:t}$ satisfies $\phi$.

The distance to acceptance, $W : \chi \to \mathbb{Z}^+$ is the minimum number of transitions required to take an FSA from state $\chi$ to a final state. $W(\chi) = 0$ if $\chi$ is a final state, and $W(\chi) = \infty$ if no final states are reachable from $\chi$. The $k$-step boundary, $\partial(\chi,k)$ is the set of states that can be reached from state $\chi$ in exactly $k$ steps.

For a set $\Sigma$, we call the set of subsets $\{\Sigma_i \subseteq \Sigma, i \in I\}$, a distribution $\Delta$ of $\Sigma$ 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 $\omega$ onto $\Sigma_i$, written $\omega|_{\Sigma_i}$ is obtained by removing all symbols in $\omega$ that are not in $\Sigma_i$. For a language $L \subseteq \Sigma^*$ and a subset $\Sigma_i \subseteq \Sigma$, the projection of $L$ onto $\Sigma_i$, $L|_{\Sigma_i}$ is the set $\{\omega|_{\Sigma_i} | \omega \in L\}$.

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

3. Problem Formulation

In this section, we give the motion, communication, and sensing models considered in this work and formulate the problem under consideration.

3.1. 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 $\mathcal{G} = (V, \rightarrow_{\mathcal{G}})$, where $V$ is a set of states and $\rightarrow_{\mathcal{G}} \subseteq V \times V$ is a set of edges. Such a discrete graph may be constructed as the quotient graph of a partitioned continuous environment.

We define a labeling function, $\mathcal{L} : V \rightarrow 2^{AP}$, 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 $\mathcal{G}$ is modeled by a transition system $\mathcal{R}_i = (Q_i, q_0^i, Act_i, Trans_i, \Sigma_i, \rightarrow_i)$, where $Q_i \subseteq V$ are the set of states that $\mathcal{R}_i$ can occupy, $q_0^i \in Q_i$ is the initial state, $Act_i$ is the set of actions the robots can take, $\Sigma_i \subseteq AP \cup \{\varepsilon\}$ are the robot’s service capabilities, and $\rightarrow_i : Q_i \times Q_i$ captures how atomic propositions may be satisfied by agent $i$ at the states, where $(q,\varepsilon) \rightarrow_i (q',\sigma)$ for all $q \in Q$ and $(q,\sigma) \rightarrow_i (q',\sigma)$ for all $\sigma \in \Sigma_i$, if and only if $\sigma \in \mathcal{L}(q)$. The model uses a discrete clock $k$ which is initialized to zero and increments by 1 every time $\text{Robot}_i$ takes an action. The state of $\text{Robot}_i$ at time $k$ is $q_i^k$, and the action executed at time $k$ is $a_i^k$. We write $a_i$ to represent the vector of actions taken by the team at time $k$.

Each run $r_i = q_0^i a_1^i \ldots$ of $\mathcal{R}_i$ generates a corresponding output word $\omega_i = a_0^i a_1^i \ldots$. Each symbol in $\omega_i$ comes from the alphabet of $\mathcal{R}_i$ such that $(q_i^0, a_0^i) \in \rightarrow_i$. For the team as a whole, the output word $\omega_{\text{team}} = a_0^0 a_1^0 \ldots$ is generated such that $\omega_{\text{team}} = \bigcup_{i=1}^m \omega_i^k$ is the union of all propositions serviced at time $k$.

A distribution $\Delta$ 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 $\sigma$ appears in more than one set $\Sigma_i$, 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 Figure 1. This environment is modeled as a graph with 64 nodes, which are inherited by the transition systems $\{\mathcal{R}_i\}_{i=1,2,3}$. 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 $\pi_H$ and has motion primitives $\{N,S,E,W\}$, corresponding to each of the four directions on the grid. Obstacles in the environment are labeled $\pi_O$. The distribution $\Delta$ of agent capabilities is $\Sigma_1 = \{\pi_1, \pi_4\}$, $\Sigma_2 = \{\pi_2, \pi_3\}$, and $\Sigma_3 = \{\pi_2, \pi_3\}$. 

![Figure 1: Environment for case study](image-url)
3.2. Sensing Model

The team of robots is tasked with estimating a feature of the environment that evolves stochastically over time. This feature is modeled as a Markov Chain \( Targ = (S, s^0, P) \) which evolves synchronously with the Robot. 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, 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 \( y^k \). 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 environment \( s^k \), the position of the robot taking the measurement \( q_i^k \), and the statistics of the sensor. We encapsulate this with the measurement likelihood function

\[
h(y, s, q) = Pr[\text{the measurement is } y \mid Targ \text{ in state } s, Robot \text{ in state } q]. \tag{4}
\]

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 \( b_i^k(s) = Pr[s^k = s | y_i^1, ..., y_i^k] \). Each belief state \( b_i \) is initialized as an identical pmf \( h^0 \) which reflects any initial information about the state of \( s^0 \). For a clique of agents who are able to communicate, we denote the belief of the \( j \)th clique, which is identical among the agents in the team, as \( b_j^k \). As agents take measurements, they share those measurements with the team, and the team belief is updated as

\[
b_j^k(s) = \eta \Pr\left(y_j^k | s, q_j^k \right) \sum_{s' \in S} P(s') b_j^{k-1}(s') \tag{5}\]

where \( \eta \) is the appropriate normalization factor, \( y_j^k \) is the collection of measurements taken by the clique, and \( q_j^k \) describes the positions of all of the agents in the cliques. We assume that the measurements of the agents are conditionally independent\(^1\). This is a standard assumption in recursive filtering. The conditional distribution of the measurements \( y_j^k \) is

\[
\Pr\left(y_1^k, ..., y_m^k \mid s, q_1^k, ..., q_m^k \right) = \prod_{i=1}^m h\left(y_i^k | s, q_i^k \right). \tag{6}
\]

Each clique belief state \( b_j^k \) evolves according to an MDP \( E_{stj} = (B, b_0, P_{est}, Q_j) \). \( B \) 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 \( b'_j \) is the result of applying Equation 5 after measuring \( y_j^k \) in states \( q_j^k \), then \( P_{est}(b_j, q_j^k, b'_j) \) is the total probability of observing \( y_j^k \), i.e.

\[
P_{est}\left(b_j, q_j^k, b'_j \right) = \sum_{s_1, s_2 \in S} \prod_{i \in I} h\left(y_i^k | s_i, q_i^k \right) P(s_1, s_2) b_j(s_1). \tag{7}
\]

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.

3.3. 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, \( Q_{Comm} = \{ (q_i, q_l) | d(q_i, q_l) \leq CommDist \} \), where \( d(q_i, q_l) \) is the distance between \( q_i \) and \( q_l \) on the graph of the environment \( \mathcal{G} \). We denote the set of states where communication is impossible \( Q_{NoComm} \) such that \( Q \times Q = Q_{Comm} \cup Q_{NoComm} \).

For a team of \( m \) agents, we create a communication graph, \( C = (V_C, \rightarrow_C) \), to model the ability of agents to communicate with each other. In this graph, the vertices \( V_C \) represent the \( m \) agents, and there is an edge between \( v_1, v_2 \in V_C \) if \( d(v_1, v_2) \leq CommDist \). 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].

3.4. 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 \( \phi \) 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 \( a ^{0:T-1} \) that minimizes the uncertainty in the estimate of the state of \( Targ \) while satisfying the mission constraints in time. In other words,
Problem 3.1 (scLTL-constrained informative path planning). Given a team of $m$ agents each with model Robot operating in an environment $Targ$, an scLTL formula $\phi$ over $AP$, and a deadline $T$, solve

$$
\min_{a^0:T-1} E_{q^0:T} \left[ H \left( b^T | b^0, q^0:T \right) \right]
$$

subject to

$\phi$ is satisfied

(8)

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

$$
\phi = \Diamond \pi_1 \land \Diamond \pi_2 \land \Diamond \pi_3 \land \Diamond \pi_4 \land (- \pi_3 \lor \pi_2).
$$

(9)

The mission specified by $\phi$ is interpreted as "eventually service regions 1 through 4, and service region 2 before servicing region 3." The deadline imposed on satisfaction is $T = 20$.

4. Solution

The solution to Problem 3.1 is summarized in Alg. 1. First, the team of agents is split into cliques according to their capabilities (4.1). Next, the specification is checked for distributability among the cliques (4.2). The specification is distributable if it may be separated into multiple local specifications such that if each clique satisfies its local specification, the global specification is satisfied. The cliques then independently execute a receding horizon algorithm to gather information and satisfy their clique specifications. Finally, they return to a starting region and share their measurements (4.3).

**Algorithm 1 Solution Outline**

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

1. Build cliques $C$, distribution $\Delta_C$, and $\{TS_{c_i}\}_{i \in I_C}$
2. $w_i \forall i \in I_C = GETLOCALWORDS(\phi, \Delta_C, TS_{c_i}, \forall i \in I_C)$
3. for $c_j \in C$ do
4. Construct automata $A_{loc}^i$ accepting $w_i$
5. Construct product $\mathcal{P}_i$
6. RECEEDINGHORIZONDP($\mathcal{P}_i, \chi^0, b^0, h, n, T$)
7. Share measurements with other cliques
8. Calculate $b^T$

4.1. Cliques

As noted in Section 3.1, if a request $\sigma$ 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 cliques according to the distribution of their capabilities. These cliques are the smallest groups of agents who must cooperate to satisfy the TL specification. These cliques may operate independently to satisfy a portion of the mission specification, as explained in sections 4.2 & 4.3.

The specification alphabet, denoted $\Sigma_\phi$, is the set of all atomic proposition appearing in $\phi$. The team is split into a set of cliques $C = \{c_1, \ldots, c_n\}$, where each clique $c_i$ is made up of one or more agents from $I$ and $n \leq m$. An agent may not be in more than one clique. The clique alphabet is given by $\Sigma_{c_i} = \bigcup_{j \in c_i} \Sigma_j$. In other words, the clique alphabet is the union of the capabilities of the members of the clique. Membership in cliques is such that given two cliques, $c_i$ and $c_j$, $\Sigma_{c_i} \cap \Sigma_{c_j} = \emptyset$, $\forall c_i, c_j \in C$, i.e., the capabilities of agents in any two cliques are distinct, while the capabilities of agents in the same clique overlap (although not necessarily with all agents in the clique). We may also define a distribution $\Delta_C$ with respect to the cliques where $\bigcup_{c \in C} \Sigma_c = \Sigma_\phi$. This distribution captures the capabilities of the cliques, by combining the capabilities of each of the agents in the clique. The set of cliques $C$ is indexed by a set denoted $I_C$.

**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 $C = \{c_1, c_2\}$, 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 $\pi_1$ and $\pi_4$ and agents 2 and 3 have capabilities $\pi_2$ and $\pi_3$, and each of these propositions appears in $\phi$ as given by Equation 9.

For each clique $c_i \in C$, we construct a product transition system $TS_{c_i} = (Q_{c_i}, q^0_{c_i}, Act_{c_i}, Trans_{c_i}, \Sigma_{c_i}, |c|)$, consisting of $|c_i|$ copies of Robot, where $Q_{c_i} \subseteq \prod_{c \in C} Q$, $q^0_{c_i} = (q^0_i)_{i \in c_i}$, $\Sigma_{c_i} = \bigcup_{c \in c_i} \Sigma_c$, and $|c_i| = \bigcup_{c \in c_i} |c|$. The set of transitions at state $q_i$ is defined as $Trans_{c_i} = \sum_{c \in c_i} Act_{c_i} \times \sum_{c \in c_i} q^0_{c_i} \times \sum_{c \in c_i} \sum_{c \in c_i} |c_i|$. The set of transitions at state $q_i$ is defined as $Trans_{c_i}$ such that $\forall \bar{q_i} \in q^0_{c_i}$, $\exists q_i, a_i$ such that $(\bar{q_i}, a_i, q_i) \in Trans$ and $\exists j \in c_i$ such that $(q_j, q_i) \in Q_{comm}$. In other words, the transition system includes only those actions that do not disconnect the group communication graph.

4.2. Task Distribution

This section deals with determining if the mission specification may be distributed among the cliques as created in 4.1. If distribution is possible, we present a method for finding a local task for each clique which
guarantees satisfaction of the specification. This process is summarized in Algorithm 2. Algorithm 2 is largely based on [10], and as such details and proofs are omitted in this work. First, the specification φ is converted to an FSA Aφ (Fig. 2) such that \( L(Aφ) = L(φ) \). Clique-specific FSAs, \( A_i \) \( \forall i \in I_c \), are created by projecting \( Aφ \) onto the capabilities of each clique, such that \( L(A_i) = L(Aφ) \upharpoonright \Sigma_i \). Next, we add the empty string, \( ε \) and self transitions to \( A_i \) to create \( ˆA_i \). For each clique, a product FSA \( P_i \) is constructed from each \( ˆA_i \) and its corresponding product transition system, \( TS_{c_i} \). The product FSAs capture the behavior of each clique and its ability to satisfy requests from \( φ \) while remaining in communication.

Once the capabilities of the cliques are captured in their corresponding product FSAs \( P_i \), they are converted to minimal, deterministic representations \( A^f_i \) by following the subset construction algorithm outlined in [17]. This means that \( A^f_i \) captures the ability of each clique to satisfy elements in \( Σφ \) without taking into account any information about the environment. By taking the product of these FSAs, \( \prod_{i \in I_c} A^f_i \), we can capture the possible interleaving behavior of the cliques.

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 \( A^f_i \), \( \prod_{i \in I_c} A^f_i \) (Fig. 3), is non-empty, a satisfying word can be found using standard backwards product of all of the \( w \)satisfy the local word

\[ φ \]

is a word

\[ l \]

ows: if the language of

\[ A \]

closed and the language of the product of

\[ A \]

correctness and complexity can be found in [10].

### Algorithm 2 Find local words from a global specification

1: function GETLOCALWORDS(φ, \( A_C, TS_{c_i} \) \( \forall i \in I_C \))
2: \( Aφ \leftarrow A \)
3: \( \{A_i \leftarrow A_φ \ | \ \Sigma_i \} \forall i \in I_c \)
4: \( \text{Construct product FSA } P_i = A_i \times TS_{c_i} \forall i \in I_c \)
5: \( \text{Construct } A^f_i \text{ via subset construction algorithm} \)
6: \( \text{Construct } \prod_{i \in I_c} A^f_i \text{ and verify } L(\prod_{i \in I_c} A^f_i) \neq \emptyset \)
7: if \( L(Aφ) \) is trace closed then
8: \( A_G = Aφ \times \prod_{i \in I_c} A^f_i \)
9: else
10: \( A_G = -\bigl(\prod_{i \in I_c} \bigl(\prod_{i \in I_c} A^f_i \times (¬A_φ)\bigl) \upharpoonright \Sigma_i\bigr)\times \prod_{i \in I_c} A^f_i \)
11: if \( L(A_G) = \emptyset \) then
12: \( \text{No solution} \)
13: else
14: \( \text{Find satisfying word } w_f \in L(A_G) \)
15: \( \text{Find local words } w_i = w_f \upharpoonright \Sigma_i \forall i \in I_c \)
16: return \( w_i \forall i \in I_c \)

agents may occupy the same region without collision and during execution, each agent in a clique shares its observations with the other members of the clique in real-time.

To initialize the planner, each clique must construct an automaton \( A^{loc}_i \) which accepts the local word \( w_i \) as computed in Section 4.2. Additionally, we add the constraint that agents must return to the starting region \( (π_H) \), as well as obstacle avoidance constraints2. Returning to the starting region permits sharing measurements among cliques 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 the agents periodically meet to share measurements. A transition is added from the set of final states in \( A^{loc}_i \) to a new final state, so that the mission terminates only if the agents have returned to the starting region (Fig. 4). Next, a product automaton \( P_i \) is constructed from \( A^{loc}_i \) and \( TS_{c_i} \).

The receding horizon algorithm (Alg. 3) uses \( P_i \), the initial belief \( b^0 \), a lookahead horizon \( h \), an action horizon \( n \), and a deadline \( T \) to plan the motion of each clique on-line. The algorithm first finds the set of states that are reachable in the remaining time before the deadline. This produces \( h \) sets of states \( \text{autStates}[i] \) such that each state \( χ \in \text{autStates}[i] \) is reachable in \( i \) steps

\[ 2 \text{In general, the constraint on obstacle avoidance does not distribute across the entire team, but including this constraint for each clique after distributing the formula ensures that all agents successfully avoid obstacles.} \]

4.3. Dynamic Programming

Once a local word \( w_i \) has been found for each clique \( c_i \in C \), the agents are deployed in the environment. At this point, each clique 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).

Each clique executes its own planner separately. Two
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 by starting from the current state. Finally, a policy $\mu$ is generated using Bellman iteration to minimize expected entropy over the horizon $h$. 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. Details of this algorithm, including a proof that it is guaranteed to satisfy the given scLTL specification, are available in [9].

3 space of pmfs whose support is $S$
under temporal logic constraints. We present a novel method for distributing tasks among groups of agents 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. We consider a team of heterogeneous agents with homogeneous sensing, but heterogeneous sensing capabilities can easily be accommodated using our methodology. Other future areas for future work are methods of data fusion without requiring the sharing of individual measurements and operation in unknown topology.

References


