Available online at www.sciencedirect.com

ScienceDirect

IFAC Papers Online CONFERENCE PAPER ARCHIVE

IFAC-PapersOnLine 48-22 (2015) 174–179

Distributed Multi-Agent Persistent Surveillance Under Temporal Logic Constraints ⋆

Derya Aksaray * **Kevin Leahy** * **Calin Belta** *

* *Department of Mechanical Engineering, Boston University, Boston, MA 02215, USA (e-mail: daksaray, kjleahy, cbelta@ bu.edu).*

Abstract: We consider the problem of finding optimal agent trajectories for persistent surveillance missions subject to temporal logic (TL) constraints. Specifically, we aim to minimize the time between two consecutive visits to regions of interest in a partitioned environment while satisfying each agent's TL specification. We formulate a distributed optimization problem, where each agent plans its trajectory based only on local information. We use a formal methods approach to show that any trajectory resulting from the proposed controller satisfies the corresponding TL constraints. The results are illustrated through simulations by comparing the proposed strategy with a joint planner and a planner without communication.

Keywords: Distributed control, formal methods, autonomous vehicles

1. INTRODUCTION

In persistent surveillance missions, the agents continuously and repetitively patrol an area to provide situational awareness. A challenging objective in such problems is to find the optimal agent trajectories that optimize a global performance measure. A prominent approach is to partition (or discretize) the surveillance area and to define the performance measure as a weighted sum of the ages of the regions. In this context, the age of a region is defined as the amount of time that has passed since the last visit to the region by an agent. In the literature, various strategies are used to solve persistent surveillance problems. For instance, virtual pheromones are used in (Fu and Ang, 2009), where agents are probabilistically guided towards the areas that have not been visited for a long time. A vehicle routing problem, (Bertsimas and Van Ryzin, 1991), is solved to minimize the age of a particular point in (Stump and Michael, 2011). An optimal control problem is formulated in (Cassandras et al., 2013), where a metric of uncertainty growing due to uncovered areas is minimized. Alternatively, some auction algorithms are used in (Nigam and Kroo, 2008) to achieve region assignment among the agents to minimize the maximum age in the environment. Moreover, the authors of (Elmaliach et al., 2009) find a minimal Hamiltonian cyclic path and locate the robots properly on the path to obtain uniform frequency of visiting the viewpoints.

In some surveillance missions, the agents may be required to optimize a performance measure subjected to trajectory constraints such as visiting the regions in a specific order. For instance, if there exist special locations for uploading data, the agents should gather information before visiting these locations. Alternatively, some regions might need to be visited sequentially so that the data collected overall can be processed more effectively. These types of constraints represent complex mission specifications, and it is generally hard to formulate them in a classical optimization setup. As such, Temporal Logics (TL) have been proposed to specify such complex constraints. TLs are rich and expressive specification languages, whose syntax and semantics are well-defined, (Baier and Katoen, 2008). For example, Linear Temporal Logic (LTL) can be used to describe a persistent task as: "Visit regions A or B, then C, infinitely often. Never visit D before visiting C."

Motion planning subject to a TL formula has been studied in various studies including, but not limited to, (Kress-Gazit et al., 2009; Karaman and Frazzoli, 2011; Wongpiromsarn et al., 2012; Chen et al., 2012b; Smith et al., 2011). Typically, the high-level plan is found by model checking algorithms (Baier and Katoen, 2008), and this plan is then implemented by a low-level controller. In multi-agent systems, an essential task is to decompose a global formula into some local ones, each of which can be implemented by a single agent (Kloetzer et al., 2011; Ulusoy et al., 2013; Chen et al., 2012a). In this paper, we assume that the independent local formulas are already assigned to the agents as in (Guo and Dimarogonas, 2015).

In this paper, we study a multi-agent persistent surveillance problem, where each agent has limited energy and limited communication capability. Moreover, each agent has an individual LTL specification that not only expresses an order of visiting certain sites but also enforces periodic visits to the base for refueling. The main objective of the team is to minimize the summation of the ages of all regions (i.e. to maximize situational awareness) while satisfying their individual specifications. However, finding the optimal joint trajectories (without any specifications) is shown to be NP-hard by (Pasqualetti et al., 2012). Therefore, we formulate a distributed optimization prob-

⋆ This work was partially supported at Boston University by the NSF under grants CMMI-1400167 and NRI-1426907, and by the ONR under grant MURI N00014-09-1051.

10.1016/ifacol.2015.10.326

lem, where each agent computes its individual trajectory by minimizing the sum of age estimates while satisfying its own specification. Here, the age estimate refers to an agent's local estimation about the ages of each region, and we allow agents to exchange their age estimates based on their communication capabilities.

This work is closely related to (Ding et al., 2014) and (Guo and Dimarogonas, 2015). Creating progress constraints based on an energy function defined over a product automaton was introduced in (Ding et al., 2014). We use this idea in a distributed setting, where each agent has a local objective and an individual specification. Such a setting for the cooperative motion and task planning under local LTL specifications was also studied in (Guo and Dimarogonas, 2015). However, our work is different from (Guo and Dimarogonas, 2015) in two aspects: 1) we do not allow any relaxation of an LTL specification, and 2) we specifically focus on a persistent surveillance scenario, where energy-limited vehicles aim to minimize the sum of the ages of each region in a distributed fashion.

2. PRELIMINARIES

2.1 Linear Temporal Logic

Linear temporal logic (LTL) syntax is defined recursively over a set of atomic propositions AP as follows:

$$\phi = p|\neg p|\phi \wedge \psi|\phi \vee \psi|\phi \mathcal{U} \psi|\mathbf{X}\phi|\mathbf{F}\phi|\mathbf{G}\phi\,, \qquad (1)$$

where ϕ and ψ are LTL formulas and $p \in AP$ is an atomic proposition. LTL combines the Boolean operators negation ($\neg$), conjunction ($\wedge$), and disjunction ($\vee$) with the temporal operators until ($\mathcal{U}$), next ($\mathbf{X}$), eventually ($\mathbf{F}$), and always ($\mathbf{G}$). The semantics of LTL are defined over infinite words w from the alphabet 2^{AP}, i.e., the words consisting of sequences of symbols from the power set of atomic propositions. We denote the k^{th} position of a word as $w(k)$, e.g., $w(0)$ is the initial position. A word satisfies a formula ϕ, if ϕ is true at $w(0)$. A word satisfies $\mathbf{F}\phi$, if ϕ is true at some position $w(k)$. Furthermore, a word satisfies $\mathbf{G}\phi$, if ϕ is true at all positions in the word. The formula $\phi_1 \mathcal{U} \phi_2$ is satisfied by a word w, if ϕ_1 is true at all positions before ϕ_2 is true. Similarly, $\mathbf{X}\phi$ is satisfied by w if ϕ is true at $w(1)$. These operators can be combined to define complex mission specifications, such as "eventually visit region p_1 and visit region p_2 infinitely often, and do not visit region p_3 before visiting region p_1", which can be expressed as:

$$\mathbf{F}p_1 \wedge \mathbf{GF}p_2 \wedge \neg p_3 \mathcal{U} p_1. \qquad (2)$$

Definition 2.1. (Buchi Automaton) A *Buchi automaton* is a tuple $\mathcal{B} = (Q_\mathcal{B}, q_{\mathcal{B}0}, \Sigma, \Delta_\mathcal{B}, F_\mathcal{B})$, in which $Q_\mathcal{B}$ is a finite set of states, $q_{\mathcal{B}0}$ is the initial state, Σ is an alphabet, $\Delta_\mathcal{B} : Q_\mathcal{B} \times \Sigma \to Q_\mathcal{B}$ is a transition function, and $F_\mathcal{B} \subseteq Q_\mathcal{B}$ is the set of accepting states.

In this paper, we assume that $\Sigma = 2^{AP}$. A Buchi automaton accepts words from Σ^ω, the set of all infinite words from Σ, such that the states in $F_\mathcal{B}$ are visited infinitely often. The language of a Buchi automaton, i.e. $L_\mathcal{B}$, is the set of all such words. For every formula ϕ over AP, there exists a Buchi automaton $\mathcal{B}_\phi$ that accepts exactly the language that satisfies ϕ (i.e., the language $L_{\mathcal{B}_\phi} = \{w | w \models \phi\}$).

In this work, we consider a fragment of LTL, which we call the *surveillance fragment* of the following form

$$\mathbf{GF}\beta \bigwedge_{p\in\Phi_\mathbf{GF}} \mathbf{GF}p \bigwedge_{p\in\Phi_\mathbf{GF}} \mathbf{G}\Big(\beta \Rightarrow (\beta \mathcal{U} (\neg\beta \mathcal{U} p)) \wedge \psi_O \wedge \psi_S \wedge \psi_R\Big), \quad (3)$$

where $\beta \in 2^{AP}$ is a task to be performed infinitely often (e.g., visiting a base to refuel), and $\Phi_\mathbf{GF} \subseteq AP$ is a set of atomic propositions (or surveillance tasks) to be performed infinitely often. The term $\beta \Rightarrow (\beta \mathcal{U} (\neg\beta \mathcal{U} p))$ requires that an agent satisfying β must then satsify each $p \in \Phi_\mathbf{GF}$ before satisfying β again. The terms ψ_O,ψ_S, and ψ_R enforce ordering, safety, and reactivity constraints on $\Phi_\mathbf{GF}$, respectively. For more details on this fragment of LTL, the reader is directed to (Chen et al., 2012b).

2.2 Graph Theory

An undirected graph, $\mathcal{G} = (V, E)$, consists of a set of nodes, V, and a set of undirected edges, E. For any set of the nodes, $X \subset V$, $\mathcal{G}_X$ refers to the *subgraph* induced by the nodes in X (i.e., $\mathcal{G}_X$ consists of all nodes in X and all the edges between those nodes). In a graph, a k-length *path* is a sequence of nodes $\mathbf{q} = (v_0, v_1, ..., v_k)$ such that the edge between any v_i and v_{i+1} belongs to E, and its length is denoted by $|\mathbf{q}| = k$. An undirected graph, $\mathcal{G}$, is *connected* if there exists a path between any two nodes of the graph. The *connected components* of a graph $\mathcal{G}$ are the set of largest subgraphs of $\mathcal{G}$ that are each connected.

Let v_j and v_k be any two nodes in $\mathcal{G}$. The *distance* between v_j and v_k is denoted as $d(v_j, v_k)$, and it is equal to the length of the shortest path between them. The neighbor set of node v_i, $\mathcal{N}_{v_i}$, is the set including all adjacent nodes that are connected to v_i (i.e. $\mathcal{N}_{v_i} = \{v_j \mid (v_i, v_j) \in E\}$).

3. PROBLEM FORMULATION

3.1 Environment Model

Suppose that a set of agents operate in an environment that contains multiple obstacles, a base, and multiple regions of interest. We abstract such an environment as an undirected graph, $\mathcal{G}^{env} = (V, \mathcal{E})$, where V is the set of nodes representing n regions of interest as well as the base, while $\mathcal{E}$ is the set of edges representing the feasible travel in one time step. An agent on $v_1 \in V$ at time t can reach $v_2 \in V$ at $t+1$, only if $(v_1, v_2) \in \mathcal{E}$. Let v_B denote the base, and let v_i for $i = 1, \ldots, n$ represent each of the n nodes. Let $\mathcal{L} : V \to AP$ denote a labeling function, where AP is the set of atomic propositions as defined in (1). As such, $\mathcal{L}$ labels the regions in the environment with the atomic propositions that are satisfied by visiting that region. For simplicity of presentation, we assume that $AP = V$ so that we consider formulas over the set of vertices. Fig. 1(a) illustrates an example of an abstracted environment.

In this setting, each node in the graph has a time-varying value, i.e. *age*, which denotes the duration of time a node is not visited by an agent. As such, the age of node v_i at time t, $\alpha_i(t)$, has the following dynamics:

$$\alpha_i(t) = \begin{cases} 0 & \text{if } \exists j \text{ s.t. } x_j(t) = v_i, \\ \alpha_i(t-1) + 1 & \text{otherwise,} \end{cases} \qquad (4)$$

where $x_j(t) \in V$ is the position of agent j on the graph. Moreover, the overall situational awareness of the

environment at time t can be quantified by the summation of the node ages $\sum_{i=1}^{n} \alpha_i(t)$, whose smaller values indicate more situational awareness.

3.2 Agent Model

Given $\mathcal{G}^{env} = (V, \mathcal{E})$, the motion of an agent is modeled as a deterministic transition system. A *transition system* is a tuple $\mathcal{T} = \big(X, x^0, Act, \Delta_{\mathcal{T}}, AP, \models\big)$, where $X \subseteq V$ is the set of states, $x^0 \in X$ is the initial state, Act is the set of available actions, $\Delta_{\mathcal{T}} \subseteq X \times Act \times X$ is the set of transitions, AP is a set of atomic propositions, and $\models$ is a satisfaction relation. For $x \in X$ and $p \in AP$, $(x, p) \in \models$ if and only if $\mathcal{L}(x) = p$, where $\mathcal{L}$ is the labeling function defined in Sec. 3.1. Moreover, the agents move synchronously on the graph so that any state transition occurs at the same time. The control policy of agent j is denoted by μ_j, which is a sequence of states over $\mathcal{T}$. Accordingly, if agent j occupies the node $x_j(t)$ at time t, $\mu_j = (x_j(t+1), x_j(t+2)\dots)$ determines the target nodes agent j needs to be in the future time steps.

Each agent has a limited energy capacity, and we use τ_j^* to denote the maximum operation time of agent j after it is refueled. Assuming that the travel time between any two adjacent nodes is 1, $\tau_j^* \in \mathbb{N}$ corresponds to the number of edges agent j may travel before returning to the base for refueling. Let $\tau_j(t)$ be the remaining travel capacity of agent j at time t. Then, $\tau_j(t)$ has the following dynamics:

$$\tau_j(t) = \begin{cases} \tau_j^* & \text{if } x_j(t) = v_B, \\ \tau_j(t-1) - 1 & \text{otherwise,} \end{cases} \tag{5}$$

where v_B denotes the base. In addition to limited energy, each agent has a limited communication capability. We assume that an agent occupying node v_i can only communicate with another agent occupying node v_j if $v_j \in \mathcal{N}_{v_i}$. Based on the preceding assumptions, the communication graph of the agents becomes a subgraph of $\mathcal{G}^{env}$.

For example, six agents are located on the nodes $Y = \{v_2, v_3, v_4, v_8, v_{10}, v_{12}\}$ of the graph $\mathcal{G}^{env}$ in Fig. 1(b), where the communication graph of the agents is a subgraph of $\mathcal{G}^{env}$ that is induced by the nodes in Y and their adjacent edges. In this setting, we assume that agents can instantaneously share information in the connected component they belong to. For example, the communication graph in Fig. 1(b) has three connected components, i.e. $\{1, 2, 3\}, \{4\}, \{5, 6\}$.

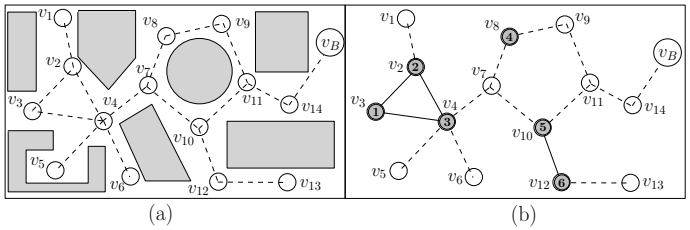

Fig. 1. (a) The abstraction of a surveillance area as a graph $\mathcal{G}^{env}$ (the obstacles are illustrated by the gray areas). (b) Six agents are located on the nodes of $\mathcal{G}^{env}$.

Instead of keeping track of the true age of each node through a central authority or a complete communication graph, each agent maintains a local knowledge about the age of each node based on their visits and communication with other agents. Let $\alpha_i^j(t)$ be the age of node i at time t according to agent j, which we call the *age estimate*. Then, the dynamics of $\alpha_i^j(t)$ is as follows:

$$\theta_i^j(t) = \begin{cases} 0 & \text{if } x_j(t) = v_i, \\ \theta_i^j(t-1) + 1 & \text{otherwise.} \end{cases} \tag{6}$$

$$\alpha_i^j(t) = \min_{k \in C_j(t)} \theta_i^k(t), \tag{7}$$

where $C_j(t) = \mathcal{N}_{x_j(t)} \cup j$ is the communication set of agent j. Note that if $|C_j(t)| \geq 2$, agent j updates its age estimates via exchanging information in its neighborhood.

Fact 1. Let $\alpha_i(t)$ be the true age of v_i, and let $\alpha_i^j(t)$ be the age estimate of v_i according to an arbitrary agent j. Let $\alpha_i(0) = \alpha_i^j(0) = 0$ for all i, j. Then, $\alpha_i(t) \leq \alpha_i^j(t)$.

3.3 Problem Definition

We first introduce the global optimization problem for a general multi-agent persistent surveillance scenario.

Problem 3.1. (Minimize True Age) Let $\pi = (\mu_1, ..., \mu_m)$ be the joint control policy for m agents. Given a graph environment $\mathcal{G}^{env}$ with n nodes, m transition systems $\mathcal{T}_1, ..., \mathcal{T}_m$, m local LTL specifications (ϕ_j for each agent j), and a discount factor $\gamma \in (0, 1)$, find a control policy

$$\pi^* = \arg\min_{\pi} \sum_{t=1}^{\infty} \gamma^t \sum_{i=1}^{n} \alpha_i(t) \tag{8}$$

such that each ϕ_j is satisfied infinitely often.

Solving Problem 3.1 requires the aggregate states of each agent to calculate $\alpha_i(t)$. Note that such a solution is not scalable and practical due to the dependence on global information and coordination. Based on Fact 1, we propose a distributed solution to Problem 3.1 by considering each agent's policy independently. Accordingly, we aim to solve the following distributed optimization problem:

Problem 3.2. (Minimize Age Estimate) Let μ_j be a control policy for agent j. Given a graph environment $\mathcal{G}^{env}$ with n nodes, a transition system $\mathcal{T}_j$, an LTL specification ϕ_j, and a discount factor $\gamma \in (0, 1)$, find a control policy

$$\mu_j^* = \arg\min_{\mu} \sum_{t=1}^{\infty} \gamma^t \sum_{i=1}^{n} \alpha_i^j(t) \quad \forall j \tag{9}$$

such that ϕ_j is satisfied infinitely often.

Note that the problem of minimizing the sum of the true ages via a team of robots without any LTL constraints has already been considered in the literature, e.g., (Pasqualetti et al., 2012). In this paper, we define a notion called *age estimate* and use it to formulate a distributed optimization problem. Accordingly, solving (9) is an approximate, but a scalable, approach such that each agent computes its trajectory independently based on minimizing the sum of the age estimates and satisfying its own LTL specification.

4. DISTRIBUTED CONTROL SYNTHESIS UNDER LTL CONSTRAINTS

4.1 Energy Function

In this section, we first describe the product between a transition system and a Buchi automaton. Then, we define

an energy function on the states of the product automaton. Such an energy function is used in (Ding et al., 2014) to enforce the satisfaction of an LTL formula. Similarly, we will use the energy function as a progress constraint in Problem 3.2.

Definition 4.1. (Product Automaton) Given a transition system $\mathcal{T} = \big(X, x^0, Act, \Delta_{\mathcal{T}}, AP, \models\big)$ and a Buchi automaton $\mathcal{B} = (Q_{\mathcal{B}}, q_{\mathcal{B}0}, \Sigma, \delta_{\mathcal{B}}, F_{\mathcal{B}})$, their product automaton denoted by $\mathcal{P} = \mathcal{T} \times \mathcal{B}$ is a tuple $\mathcal{P} = (Q_{\mathcal{P}}, q_{\mathcal{P}0}, \Delta_{\mathcal{P}}, F_{\mathcal{P}})$, where $Q_{\mathcal{P}} = X \times Q_{\mathcal{B}}$ is the set of states; $q_{\mathcal{P}0} = \{x^0\} \times q_{\mathcal{B}0}$ is the initial state; $\Delta_{\mathcal{P}} \subseteq Q_{\mathcal{P}} \times Q_{\mathcal{P}}$ is the set of transitions defined by: $\big((x,q),(x',q')\big) \in \Delta_{\mathcal{P}}$ iff $x \to_{\mathcal{T}} x'$ and $q \overset{\mathcal{L}(x)}{\to}_{\mathcal{B}} q'$; and $F_{\mathcal{P}} = X \times F_{\mathcal{B}}$ is the set of accepting states on $\mathcal{P}$.

Definition 4.2. (Energy function in $\mathcal{P}$) The energy function $V(q_{\mathcal{P}})$, $q_{\mathcal{P}} \in Q_{\mathcal{P}}$, is defined as

$$V(q_{\mathcal{P}}) = \begin{cases} \min\limits_{q'_{\mathcal{P}} \in F_{\mathcal{P}^*}} d(q_{\mathcal{P}}, q'_{\mathcal{P}}), & \text{if } q_{\mathcal{P}} \notin F_{\mathcal{P}^*} \\ 0, & \text{if } q_{\mathcal{P}} \in F_{\mathcal{P}^*} \end{cases}, \qquad (10)$$

where $F_{\mathcal{P}^*} \subseteq F_{\mathcal{P}}$ is the set of self-reachable accepting states from $\mathcal{P}$, and $d(q_{\mathcal{P}}, q'_{\mathcal{P}})$ is the graph distance between $q_{\mathcal{P}}$ and $q'_{\mathcal{P}}$. If $F_{\mathcal{P}^*}$ is not reachable from a state $q_{\mathcal{P}} \in Q_{\mathcal{P}}$, then $V(q_{\mathcal{P}}) = \infty$.

4.2 Design of a Receding Horizon Controller

This section presents the design of a state-feedback controller for each agent that solves (9). The proposed controller is designed over the product automaton, which captures both the motion of an agent and the satisfaction of an LTL specification. Let $\mathcal{P}_j$ be the product automaton of agent j. We denote the product automaton state of agent j at time t by $s_j(t)$. Note that any $s_j(t) \in Q_{\mathcal{P}_j}$ corresponds to an energy $V(s_j(t))$, i.e. the distance to a self-reachable accepting state at time t, as defined in (10). Moreover, agent j has a remaining movement capacity at time t as $\tau_j(t)$, i.e. the remaining number of transitions it can pursue before returning to the base. Accordingly, we define $J_j\big(s_j(t), s_j(t-1), C_j(t)\big)$ that is the sum of the age estimates at time t with respect to agent j under the transition from $s_j(t-1)$ to $s_j(t)$ over $\mathcal{P}_j$ based on the shared information among the agents in $C_j(t)$. Then, (9) can be reformulated for all j as

$$\mu_j^* = \arg\min_{\mu} \sum_{t=1}^{\infty} \gamma^t J_j\big(s_j(t), s_j(t-1), C_j(t)\big) \quad (11)$$
$$s.t.\, V\big(s_j(t)\big) \le \tau_j(t),$$

where the constraint in (11) restricts the agent not to move to a product automaton state that has a higher energy (longer distance to satisfaction) than its movement capacity. Note that the set $C_j(t)$ depends on the states of all agents at t. Instead of predicting the elements of this set for future time steps, we design a controller for agent j that can immediately utilize the information shared in $C_j(t)$ to compute the next control action. Therefore, we propose a receding horizon controller to solve the problem in (11) by formulating a fixed horizon optimization as:

$$\mu_j^{N^*} = \arg\min_{\mu} \sum_{t'=t+1}^{t+N} \gamma^{t'-t} J_j\big(s_j(t'), s_j(t'-1)\big) \quad (12)$$
$$s.t.\, V\big(s_j(t')\big) \le \tau_j(t'),$$

where N is the look-ahead horizon, and $\mu_j^{N^*}$ results in a state trajectory over $\mathcal{P}_j$ as $\mathbf{q}^{j*}_{t+1:t+N} = (s_j(t+1), \ldots, s_j(t+N))$. Agent j moves from $s_j(t)$ to $s_j(t+1)$, which is also mapped to $\mathcal{T}_j$ to update the age estimates according to (7). Then, agent j solves (12) to obtain $\mathbf{q}^{j*}_{t+2:t+N+1}$, and it moves to $s_j(t+2)$ from $s_j(t+1)$.

We introduce Alg. 1 as the receding horizon controller of agent j. Alg. 1 starts with creating a Buchi automaton $\mathcal{B}_j$ from the agent's specification ϕ_j. Then, a product automaton $\mathcal{P}_j$ is constructed from $\mathcal{B}_j$ and $\mathcal{T}_j$. Each state $q_{\mathcal{P}_j} \in Q_{\mathcal{P}_j}$ is labeled with its energy, $V(q_{\mathcal{P}_j})$. Moreover, we construct $\mathcal{F}_{\mathcal{P}^*} \subseteq \mathcal{F}_{\mathcal{P}}$, which contains the accepting states that are self-reachable through a τ_j^*-length path. Note that τ_j^* is the maximum movement capacity of agent j. Given the initial automaton state $s_j(0)$, if $V\big(s_j(0)\big) > \tau_j^*$, then it is not possible to find a trajectory originating from $s_j(0)$ and satisfying ϕ_j in τ_j^* transitions. Thus, the algorithm returns no solution. Otherwise, at each time t, agent j updates its age estimates via (7). Then, it computes the set of all feasible N-length paths from its current state $s_j(t)$, which we denote $\psi_j(s_j(t), N)$. For any $\mathbf{q}^j \in \psi_j(s_j(t), N)$, the feasibility implies that each state in $\mathbf{q}^j = (s_j(t+1), \ldots, s_j(t+N))$ does not violate the energy constraint in (12). Moreover, every $\mathbf{q}^j$ has a corresponding cost, i.e., $\sigma(\mathbf{q}^j)$, calculated via (12). Accordingly, the optimal path $\mathbf{q}^{j*}$ is found via an auction-based selection to assign each agent with the lowest (possible) cost paths. In particular, each agent first selects $\mathbf{q}^{j*} = \arg\min_{\mathbf{q}^j \in \psi_j} \sigma(\mathbf{q}^j)$ and determines the destination node at the next time step. If multiple agents plan to travel to the same node v_k, the agent whose path has the lowest cost, travels to v_k. Then, the other agents filter the paths starting with v_k from their feasible path sets, i.e. $\psi_j(s_j(t), N)$, and select a path with the next lowest cost by sequentially modifying $\psi_j(s_j(t), N)$. Finally, when the agent finds $\mathbf{q}^{j*} = (s_j(t+1), \ldots, s_j(t+N))$, it implements the transition from $s_j(t)$ to $s_j(t+1)$ and repeats the algorithm.

Algorithm 1 Receding Horizon Controller for Agent j

input : A transition system $\mathcal{T}_j$, specification ϕ_j, maximum operation time τ_j^*, and lookahead horizon N

```
Construct Buchi automaton B_j from φ_j
Take product P_j of B_j and T_j
Compute V(q_{P_j}) for all q_{P_j} ∈ Q_{P_j}
if V(s_j(0)) > τ_j^* then
    return no solution
else
    α_i^j(0) ← 0 for all i = 1, ..., n
    for t ← 0 to ∞ do
        Update α_i^j(t) according to (6) and (7)
        Compute the set of feasible paths, ψ_j(s_j(t), N)
        Compute the cost of each path σ(q^j) ∀q^j ∈ ψ_j(s_j(t), N)
        Find q^{j*} via Auction(σ(q^j), C_j(t))
        Implement the transition from s_j(t) to s_j(t+1) on P_j and
        the corresponding transition on T_j
        Update τ_j(t) based on (5)
        t ← t + 1
```

In the following theorem, we present a sufficient condition that ensures the infinitely often satisfaction of a given LTL specification. To this end, we present an upper bound for

the energy of the initial product automaton state with respect to the vehicle capacity.

Theorem 4.1. Given an LTL formula ϕ_j as (3), Alg. 1 produces an infinite trajectory satisfying ϕ_j if $V\big(s_j(0)\big) \leq \tau_j^*$.

Proof. If $V\big(s_j(0)\big) \leq \tau_j^*$, then there exists a path (of at most τ_j^*-length) from $s_j(0)$ to the set of self-reachable accepting states $\mathcal{F}_{\mathcal{P}^*}$. Moreover, the path originated from $s_j(0)$ only contains states satisfying $V\big(s_j(t)\big) \leq \tau_j(t) \leq \tau_j^*$ for all $t > 0$ because only the paths with feasible states are taken into account (line 10). Now, we will show that $V\big(s_j(t)\big)$ goes to zero infinitely often. Since $\tau_j(t)$ is a decreasing function, (5), while the agent is not at the base, $V\big(s_j(t)\big)$ will eventually go to zero in at most $\tau_j(t)$ time steps. In particular, if the agent is patrolling in the environment at $t = t' \geq 0$, then there always exists k_i such that $t' + \tau_j(t') \geq k_i > t'$ and $V\big(s_j(k_i)\big) = 0$ (i.e., $s_j(k_i) \in \mathcal{F}_{\mathcal{P}^*}$). Note that k_i is finite, thus the time sequence $(k_1, k_2, \dots)$ is infinite in the repeated run of Alg. 1. Hence, Alg. 1 produces an infinite trajectory $(s_j(0), s_j(1), \dots)$ such that $V\big(s_j(t)\big) \leq \tau_j(t)$ for all $t \geq 0$ and $V\big(s_j(k_i)\big) = 0$ for an infinite sequence $(k_1, k_2, \dots)$, where $k_i > 0$. Consequently, the resulting infinite trajectory satisfies ϕ_j.

4.3 Complexity

The offline computations in Alg. 1 are constructing the Buchi automaton ($\mathcal{B}$) from an LTL formula ϕ, generating the product automaton ($\mathcal{T} \times \mathcal{B}$), and computing the energy of each product automaton state (lines 1-3). The complexity of the offline part greatly depends on the length of the LTL formula, i.e. $|\phi|$, since the size of the product automaton is bounded by $|X| \times |\phi| \times 2^{|\phi|}$ (Ding et al., 2014), where $|X|$ is the number of states in $\mathcal{T}$, and $|\phi| \times 2^{|\phi|}$ is the maximum number of states in $\mathcal{B}$ created from ϕ. Note that the offline part of Alg. 1 may have arbitrarily high computation cost, but it is executed only once.

Theorem 4.2. Given an LTL formula ϕ_j as (3), the number of online operations to be performed by each agent via Alg. 1 scales with $O\big(\Delta_{\mathcal{P}}(s_j)^N + |C_j|\big)$, where $\Delta_{\mathcal{P}}(s_j)$ is the number of transitions that can be taken at s_j, N is the horizon length, and C_j is the communication set.

Proof. At each time step, each agent j computes the N-length feasible paths (with the corresponding costs) from its current product automaton state s_j (lines 10-11). This process is similar to a depth-first search so the maximum number of operations is bounded by $\Delta_{\mathcal{P}}(s_j)^N$. Moreover, the optimal path is found by an auction-based selection in a sequential way so that the maximum number of decision-making among C_j is at most $|C_j|$. Hence, the overall online operations performed by agent j is in the order of $O\big(\Delta_{\mathcal{P}}(s_j)^N + |C_j|\big)$.

5. CASE STUDY

Consider two agents that move on the graph displayed in Fig. 1. The agents are deployed simultaneously from the base. The first agent must satisfy the specification

$$\phi_1 = \mathbf{GF} v_b \wedge \mathbf{GF} v_2 \wedge \mathbf{GF} v_{10} \wedge \mathbf{G}\Big(v_b \Rightarrow v_b \mathcal{U} \left(\neg v_b \mathcal{U} v_2\right) \wedge \left(\neg v_b \mathcal{U} v_{10}\right)\Big), \quad (13)$$

while the second agent must satisfy the specification

$$\phi_2 = \mathbf{GF} v_b \wedge \mathbf{GF} v_3 \wedge \mathbf{GF} v_8 \wedge \mathbf{G}\Big(v_b \Rightarrow v_b \mathcal{U} \left(\neg v_b \mathcal{U} v_3\right) \wedge \left(\neg v_b \mathcal{U} v_8\right)\Big). \quad (14)$$

Note that ϕ_1 expresses "visit the base, node 2, and node 10 infinitely often, and never return to the base before visiting nodes 2 and 10". ϕ_2 has the similar structure as ϕ_1, but the second agent must visit nodes 3 and 8 instead of nodes 2 and 10. The maximum operation time of both agents is 20, i.e., $\tau_1^* = \tau_2^* = 20$, the planning horizon is 3, i.e., $N = 3$, and the discount factor is 0.8, i.e., $\gamma = 0.8$. The simulations were implemented in MATLAB by using a laptop with a 2.6 GHz processor and 8 GB memory.

In order to evaluate the performance of the proposed algorithm, we compare it with respect to two other strategies. The first one is a joint planner that computes the feasible paths for the two agents jointly based on the true age. This planner can also be thought of as a centralized planner that decides on actions for both agents to globally minimize the sum of the ages in the given horizon. The second strategy allows the agents to act independently without any communication. In other words, this strategy is similar to Alg. 1; however, the agents are not able to communicate with each other, i.e. $C_j(t) = \{j\}$, $\forall j, t$. In the results, we refer to these planners as Communication (Alg. 1), Joint, and No Communication.

The instantaneous sums of the node ages for three strategies are illustrated in Fig. 2. In Fig. 2(a), the results of the Joint planner are presented, where the true age and the age estimates are exactly the same. In Fig.s 2(b) and (c), the results of the Communication and No Communication strategies are presented, where the blue line corresponds to the true age whereas the yellow and the red lines illustrate the age estimates of both agents. Also, the straight line in all figures shows the steady-state average sum of the true ages, which can represent the average situational awareness of the mission. In all figures, first a transient behavior is observed, then all ages settle into a periodic steady state behavior. The results indicate that the age estimates are closer to the true age with communication than without communication. In Fig. 3(a), the discounted sum of the true ages is shown for three strategies. As expected, the Joint planner has the lowest age sum (the highest situational awareness); however, it has a significantly higher computational cost than the other two strategies. Specifically, the discounted age sum via the Communication strategy is only 4% higher than the Joint planner, but its computation time is less than half of the joint planner as seen from Tab. 1. Moreover, if N is increased from 3 to 5, the computation complexity of the Joint planner dramatically increases. Based on the results, the Alg. 1 exhibits a sufficiently good performance with a much lower complexity as shown in Tab. 1. Finally, Fig. 3(b) illustrates the energy function that goes to zero periodically for both agents. This implies that agents periodically satisfy their LTL specifications. Note that the energy functions are not strictly decreasing. The small jumps in Fig. 3(b) indicate that the agents may jump the higher energy states (getting further away from the satisfaction) to minimize the ages as long as their capacity allows it (i.e. $V\big(s_j(t)\big) \leq \tau_j(t)$).

Table 1. The results for 500 steps with $\gamma = 0.8$.

Planner	Horizon (N) = 3		Horizon (N)= 5	
	Comp. Time (sec)	Cost	Comp. Time (sec)	Cost
Joint	29.07	49842	1951.08	42605
w/Comm.	13.68	51935	50.87	48331
No Comm.	8.94	54799	42.76	55502

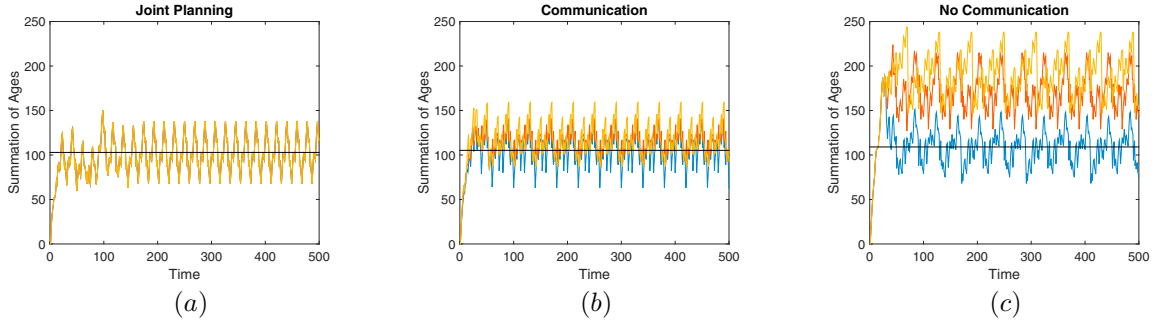

Fig. 2. The sum of true ages is shown in blue, while the sum of age estimates are shown in red and yellow. The straight line indicates the steady-state average sum of true ages. $N = 3$ and $\gamma = 0.8$. (a) Joint planner (true age and age estimate overlap); (b) Planner with communication (Alg. 1); (c) Planner without communication.

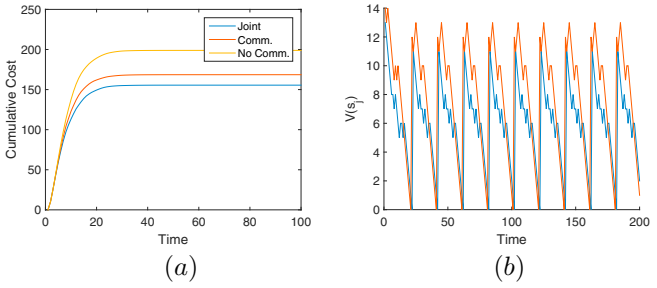

Fig. 3. (a) The discounted sum of the true ages, $N = 3$ and $\gamma = 0.8$. (b) Energy function, $V(s_j)$, of 2 agents.

6. CONCLUSION

We considered a multi-agent persistent surveillance problem, where agents walk on a graph to minimize the sum of the node ages while satisfying their own LTL specifications. Instead of planing the joint team trajectories by minimizing the sum of the true ages, we defined an objective function based on the age estimates, which enabled to design a receding horizon controller to plan each agent's trajectory independently by using only local information. We used an automata-theoretic approach to show that any trajectory resulting from the proposed controller satisfies the corresponding LTL specification. We also showed by simulations that the proposed controller performs sufficiently good with a low computational complexity.

REFERENCES

Baier, C. and Katoen, J. (2008). *Principles of model checking*, volume 26202649. MIT press Cambridge.

Bertsimas, D. and Van Ryzin, G. (1991). A stochastic and dynamic vehicle routing problem in the euclidean plane. *Operations Research*, 39(4), 601–615.

Cassandras, C., Lin, X., and Ding, X. (2013). An optimal control approach to the multi-agent persistent monitoring problem. *IEEE Trans. on Automatic Control*, 58(4), 947–961.

Chen, Y., Ding, X.C., Stefanescu, A., and Belta, C. (2012a). Formal approach to the deployment of distributed robotic teams. *IEEE Trans. on Robotics*, 28(1), 158–171.

Chen, Y., Tumova, J., and Belta, C. (2012b). LTL robot motion control based on automata learning of environmental dynamics. In *Int. Conference on Robotics and Automation (ICRA)*, 5177–5182. IEEE.

Ding, X., Lazar, M., and Belta, C. (2014). LTL receding horizon control for finite deterministic systems. *Automatica*, 50(2), 399–408.

Elmaliach, Y., Agmon, N., and Kaminka, G.A. (2009). Multi-robot area patrol under frequency constraints. *Annals of Mathematics and Artificial Intelligence*, 57(3-4), 293–320.

Fu, J. and Ang, M. (2009). Probabilistic ants (pants) in multi-agent patrolling. In *Int. Conference on Advanced Intelligent Mechatronics*, 1371–1376. IEEE/ASME.

Guo, M. and Dimarogonas, D.V. (2015). Multi-agent plan reconfiguration under local LTL specifications. *Int. Journal of Robotics Research*, 34(2), 218–235.

Karaman, S. and Frazzoli, E. (2011). Linear temporal logic vehicle routing with applications to multi-UAV mission planning. *Int. Journal of Robust and Nonlinear Control*, 21(12), 1372–1395.

Kloetzer, M., Ding, X.C., and Belta, C. (2011). Multi-robot deployment from LTL specifications with reduced communication. In *Conference on Decision and Control and European Control Conference*, 4867–4872.

Kress-Gazit, H., Fainekos, G.E., and Pappas, G.J. (2009). Temporal-logic-based reactive mission and motion planning. *IEEE Trans. on Robotics*, 25(6), 1370–1381.

Nigam, N. and Kroo, I. (2008). Persistent surveillance using multiple unmanned air vehicles. In *Aerospace Conference*, 1–14. IEEE.

Pasqualetti, F., Franchi, A., and Bullo, F. (2012). On cooperative patrolling: Optimal trajectories, complexity analysis, and approximation algorithms. *IEEE Trans. on Robotics*, 28(3), 592–606.

Smith, S., Tumova, J., Belta, C., and Rus, D. (2011). Optimal Path Planning for Surveillance with Temporal Logic Constraints. *Int. Journal of Robotics Research*, 30(14), 1695–1708.

Stump, E. and Michael, N. (2011). Multi-robot persistent surveillance planning as a vehicle routing problem. In *Conference on Automation Science and Engineering (CASE)*, 569–575. IEEE.

Ulusoy, A., Smith, S.L., Ding, X.C., Belta, C., and Rus, D. (2013). Optimality and Robustness in Multi-Robot Path Planning with Temporal Logic Constraints. *Int. Journal of Robotics Research*, 32(8), 889–911.

Wongpiromsarn, T., Topcu, U., and Murray, R. (2012). Receding horizon temporal logic planning. *IEEE Trans. on Automatic Control*, 57(11), 2817–2830.