2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)
October 25-29, 2020, Las Vegas, NV, USA (Virtual)

Decentralized Safe Reactive Planning under TWTL Specifications

Ryan Peterson, Ali Tevfik Buyukkocak, Derya Aksaray, and Yasin Yazıcıoğlu

***Abstract*— We investigate a multi-agent planning problem, where each agent aims to achieve an individual task while avoiding collisions with others. We assume that each agent's task is expressed as a Time-Window Temporal Logic (TWTL) specification defined over a 3D environment. We propose a decentralized receding horizon algorithm for online planning of trajectories. We show that when the environment is sufficiently connected, the resulting agent trajectories are always safe (collision-free) and lead to the satisfaction of the TWTL specifications or their finite temporal relaxations. Accordingly, deadlocks are always avoided and each agent is guaranteed to safely achieve its task with a finite time-delay in the worst case. Performance of the proposed algorithm is demonstrated via numerical simulations and experiments with quadrotors.**

I. Introduction

Collision avoidance is critical for the safe operation of multi-agent systems. Since the complexity of joint planning grows exponentially with the number of agents, significant effort has been devoted to the design of decentralized algorithms that avoid deadlocks and ensure safety. Methods such as potential fields, control barrier functions, or differential games have been used for decentralized planning (e.g., [1], [2], [3]). However, these methods typically do not accommodate complex spatio-temporal requirements.

Recently, there has been a significant interest in the analysis and control of dynamical systems under complex spatio-temporal specifications that can be expressed by temporal logics. For instance, linear temporal logic (LTL) [4] has been extensively used in motion planning and control of robots (e.g., [5], [6], [7]). However, LTL cannot express tasks with time constraints such as "visit A for 1 time unit within 5 time units interval, and after this visit B for 3 time units within 10 time units interval, and visiting C must be performed after visiting both A and B twice". Such tasks with time constraints can be expressed via bounded temporal logics (e.g., [8], [9], [10]).

In this paper, we introduce a decentralized algorithm for multi-agent systems to satisfy time-window temporal logic (TWTL) [10] specifications while operating in shared environments. In this setting, each agent has an individual TWTL specification. Agents can communicate with the other agents in their local neighborhoods to plan collision-free paths in a receding horizon manner based on the energy functions [11] computed over their individual product automata. Since it may be sometimes impossible to find safe paths which satisfy the exact specifications, we allow for the temporal relaxation of TWTL specifications and show that, if the environment is sufficiently connected, the proposed approach ensures the completion of all TWTL specifications with finite relaxation.

This work is closely related to [5], [12], [13]. In [5], a multi-agent receding horizon approach is proposed to generate each agent's path independently using only local information. Their method ensured LTL satisfaction using the idea of energy function defined over a product automaton [11]. However, our work differs from [5] by 1) considering tasks with explicit time constraints (TWTL) and allowing its relaxation when necessary, and 2) guaranteeing collision avoidance under mild assumptions on graph connectivity. In [12], collision avoidance was ensured by penalizing transitions in the centralized graph which captures the environment and TWTL specifications for all agents. However, this centralized approach is not scalable as the number of agents increases. Moreover, in [12], if a safe path satisfying the TWTL cannot be found, the algorithm terminates and does not allow for relaxations of the TWTLs. Finally, the work in [13] considered a global task that needs to be achieved by a multi-agent system and allowed for temporal relaxation of TWTL specifications. However, collision avoidance was not incorporated in path planning and practically achieved by the assumption of quadrotors flying at different altitudes.

II. Preliminaries

A. Time Window Temporal Logic (TWTL)

Syntax and Semantics: The syntax of TWTL is defined as:

$$\phi := s \mid \phi_i \wedge \phi_j \mid \phi_i \vee \phi_j \mid \neg\phi_i \mid \phi_i \cdot \phi_j \mid \mathrm{H}^d s \mid [\phi_i]^{[a,b]},$$

where ϕ is a TWTL formula; $s \in \mathcal{S}$ is a site label and $\mathcal{S}$ is the set of site labels; $\wedge$, $\vee$, and $\neg$ are the Boolean operators for conjunction, disjunction, and negation, respectively; $\cdot$ is the concatenation operator such that $\phi_i \cdot \phi_j$ specifies that first ϕ_i and then immediately ϕ_j must be satisfied. The semantics are defined with respect to finite output words $\mathbf{o}$ over $\mathcal{S}$ where $\mathbf{o}(k)$ denotes the kth element on $\mathbf{o}$. The *hold* operator $\mathrm{H}^d s$ specifies that a region $s \in \mathcal{S}$ should be visited for d time units (i.e., $\mathbf{o} \models \mathrm{H}^d s$ if $\mathbf{o}(t) = s \; \forall t \in [0, d]$), while the *within* operator $[\phi]^{[a,b]}$ bounds the satisfaction of ϕ within a time window $[a, b]$. For instance, consider the TWTL formula:

$$\phi = \neg C^{[0,5]} \cdot \left[\mathrm{H}^2 A \wedge [\mathrm{H}^2 B]^{[0,6]}\right]^{[1,10]}, \tag{1}$$

which means "Do not visit C within the time bound [0,5], and immediately after this, service A for 2 time units and service B for two time units within [0,6], both of which must be performed within [1,10]."

*This work was supported by Honeywell Aerospace and MnDRIVE.

R. Peterson, A.T. Buyukkocak, and D. Aksaray are with the Department of Aerospace Engineering and Mechanics, University of Minnesota, Minneapolis, MN, 55455, pete9936@umn.edu, buyuk012@umn.edu, daksaray@umn.edu, and Y. Yazıcıoğlu is with the Department of Electrical and Computer Engineering, University of Minnesota, Minneapolis, MN, 55455, ayasin@umn.edu,

Temporal Relaxation: Given a TWTL specification, its temporal relaxation is written by adding slack variables to the time windows of each hold operator to represent shorter or longer time windows. The relaxed version of ϕ in (1) is:

$$\phi(\boldsymbol{\tau}) = \neg C^{[0,(5+\tau_1)]} \cdot \left[\mathrm{H}^2 A \wedge [\mathrm{H}^2 B]^{[0,(6+\tau_2)]}\right]^{[1,(10+\tau_3)]} \quad (2)$$

where $\boldsymbol{\tau} = (\tau_1, \tau_2, \tau_3) \in \mathbb{Z}^3$ is the temporal relaxation vector. Overall, the temporal relaxation of $\phi(\boldsymbol{\tau})$ is quantified by $|\boldsymbol{\tau}|_{TR} = \max_j(\tau_j)$ [10].

For any TWTL specification, ϕ, a finite deterministic state automaton (dFSA) can be constructed that captures all temporal relaxations of ϕ.

Definition 1. *(Deterministic Finite State Automaton) [10] A dFSA that encodes all temporal relaxations of a TWTL formula is a tuple $\mathcal{A}_\infty = (S_{\mathcal{A}_\infty}, s_0, \Sigma, \delta_{\mathcal{A}_\infty}, F_{\mathcal{A}_\infty})$[1]: $S_{\mathcal{A}_\infty}$ is a finite set of states; $s_0 \in S_{\mathcal{A}_\infty}$ is the initial state; Σ is the input alphabet; $\delta_{\mathcal{A}_\infty} : S_{\mathcal{A}_\infty} \times \Sigma \to S_{\mathcal{A}_\infty}$ is the transition function; $F_{\mathcal{A}_\infty} \subseteq S_{\mathcal{A}_\infty}$ is the set of accepting states.*

B. Graph Theory

A weighted directed graph is a tuple $\mathcal{G} = (X, \Delta, w)$ where X is a set of nodes, $\Delta \subset X \times X$ is a set of edges between the nodes, and $w : \Delta \to \mathbb{R}^+$ denotes the weight function, which is a mapping from the edge set to the set of positive real numbers. A node $x_j \in X$ is said to be an out-neighbor to another node $x_i \in X$ if $(x_i, x_j) \in \Delta$, this can also be denoted by $x_j \in \mathcal{N}_i$ where $\mathcal{N}_i$ is the set of out-neighboring nodes to x_i. For brevity, we will use the term "neighbor" to refer to "out-neighbor" in the rest of the paper. A path $\mathbf{p}$ on a graph $\mathcal{G}$ is a sequence of adjacent nodes, i.e, there exists an edge from any node in $\mathbf{p}$ to the next node in the sequence. We use $|\mathbf{p}|$ to denote the path length (the total number of edges traversed on $\mathbf{p}$). The weighted graph distance between the nodes, $d(x_i, x_j)$, is equal to the cumulative weight of edges traversed along the shortest (minimum cumulative weight) path from x_i to x_j. We use $\mathcal{N}_i^h$ to denote the set of all nodes that are reachable from x_i in at most h-hops (via a path of length at most h). A graph is strongly connected if there exists a path from any node x_i to any other node x_j. The graph $\mathcal{G}$ is said to be k-connected if it remains connected whenever fewer than k nodes are removed.

III. Problem Statement

A. Agent Model

Dynamics: We assume that an agent moves in a 3D environment, whose abstraction is initially given as a graph $\mathcal{G} = (X, \Delta, w)$. In general, several methods (e.g., [14], [15]) can be used to construct such an abstraction; however, the construction of the abstraction is beyond the scope of this paper. Given an environment graph, we model the dynamics of each agent as a deterministic weighted transition system. Moreover, the agents move synchronously on $\mathcal{G}$ meaning state transitions for any agent happens at the same time.

Definition 2. *(Deterministic Weighted Transition System) A deterministic weighted transition system (dWTS) is a tuple $\mathcal{T} = (X, x_0, \Delta, w, AP, l)$ where X is a finite set of states; $x_0 \in X$ is the initial state; $\Delta \subseteq X \times X$ is the set of transitions; $w : \Delta \to \mathbb{R}^+$ is the weight function; AP is a finite set of atomic propositions; $l : X \to 2^{AP}$ is the labeling function.*

[1]The subscript ∞ is used to indicate all temporal relaxations.

We assume that the transition system $\mathcal{T}$ is (strongly) connected, that is, for each $x \in X$, there exists $x' \in X$ that can be reached in a finite number of transitions. A path (or run) of the system is a sequence of states $\mathbf{x} = x_0 x_1 \ldots$. This path $\mathbf{x}$ generates an output word $\mathbf{o} = o_0 o_1 \ldots$, where $o_t = l(x_t)$, $\forall t \geq 0$. The language corresponding to a transition system $\mathcal{T}$ is the set of all generated output words, denoted by $\mathcal{L}(\mathcal{T})$. The weight of a transition can be defined by the time and/or fuel cost required to traverse the transition. Without loss of generality, we will consider normalized weights, i.e., $w(x, x') \in (0, 1]$ for all $(x, x') \in \Delta$.

Communication: All agents are assumed to communicate with the agents within their $2H$-hop neighborhood where $H \geq 1$ is the length of planning horizon. When agents communicate, each shares its progress towards its individual task (a notion to measure progress will be defined in Sec. IV), the next H-hop path, and an indicator for planning update.

Specification: Each agent i aims to satisfy a TWTL ϕ_i that is defined over the atomic proposition set AP of the transition system $\mathcal{T}$. It is assumed that agents do not know about the other agents' specifications. In presence of violations, instead of terminating the mission, it will be allowed to satisfy a temporally relaxed version of ϕ_i, i.e., $\phi_i(\boldsymbol{\tau}^i)$.

B. Problem Statement

Suppose that there are n identical agents, each of which has the same transition system $\mathcal{T}$. We address the problem of planning paths for n agents, each of which is required to satisfy an individual TWTL specification while avoiding collisions with each other. In order to enforce collision avoidance, the collision cases are first identified.

Occupying the same state: If agent i occupies the same state as another agent j, then there exists $t \geq 0$ such that $x_t^i = x_t^j$.

Traversing the same transition: Let agents i and j move according to the transitions denoted as (x_t^i, x_{t+1}^i) and (x_t^j, x_{t+1}^j), respectively. If $(x_t^i, x_{t+1}^i) = (x_{t+1}^j, x_t^j)$, then the agents traverse the same transition (i.e., swapping).

Finally, a safe path can be formally defined as:

Definition 3. *(Safe path) The path of agent i, $\mathbf{x_i} = x_0^i x_1^i \ldots$, is safe if the following statements are true for all $t \geq 0$ and for all $j \neq i$: 1) $x_t^i \neq x_t^j$, and 2) $(x_t^i, x_{t+1}^i) \neq (x_{t+1}^j, x_t^j)$.*

In this paper, we aim to solve a multi-agent path planning problem, which results in agent paths always ensuring collision avoidance and satisfying the TWTL specifications (or minimally relaxed versions of the original TWTLs).

Problem 1. *Let a multi-agent system consist of n identical agents each of which has a transition system $\mathcal{T}$ and individual TWTL specifications ϕ_i. Find safe agent paths $\mathbf{x_1}, \ldots, \mathbf{x_n}$ that satisfy minimally relaxed TWTL specifica-*

tions $\phi_1(\boldsymbol{\tau}^1), \ldots, \phi_n(\boldsymbol{\tau}^n)$, i.e.,

$$\begin{aligned} \min_{\mathbf{x}_1,\ldots,\mathbf{x}_n} & \sum_{i=1}^{n} |\boldsymbol{\tau}^i|_{TR} \\ s.t. \;\; & x_t^i \neq x_t^j, \quad \forall j \neq i \\ & (x_t^i, x_{t+1}^i) \neq (x_{t+1}^j, x_t^j), \;\; \forall j \neq i \\ & \mathbf{o}_i \models \phi_i(\boldsymbol{\tau}^i), \quad \forall i \in \{1, ..., n\}. \end{aligned} \tag{3}$$

where $\mathbf{o}_i$ is the output word associated with the state path $\mathbf{x_i} = x_0^i x_1^i \ldots$ over $\mathcal{T}$, and $|\boldsymbol{\tau}^i|_{TR} \in \mathbb{Z}$ is the temporal relaxation of ϕ_i.

C. Challenges of a Centralized Solution

The multi-agent problem defined in (3) can be solved via an automata-theoretic approach. To this end, a product automaton $\mathcal{P}$ can be constructed for each agent given its transition system $\mathcal{T}$ and its automaton $\mathcal{A}_\infty$. The purpose of product automaton is to encode all possible satisfactory cases given the feasible movement on the transition system.

Definition 4. *(Product Automaton) Given a transition system $\mathcal{T} = (X, x_0, \Delta, w, AP, l)$ and a finite state automaton $\mathcal{A}_\infty = (S_{\mathcal{A}_\infty}, s_0, \Sigma, \delta, F_{\mathcal{A}_\infty})$ capturing all temporal relaxations of a TWTL specification, a (weighted) product automaton $\mathcal{P} = \mathcal{T} \times \mathcal{A}_\infty$ is a tuple $\mathcal{P} = (S_\mathcal{P}, p_0, \Delta_\mathcal{P}, w_p, F_\mathcal{P})$, where*

- *$S_\mathcal{P} = X \times S_{\mathcal{A}_\infty}$ is the finite set of states;*
- *$p_0 := (x_0, s_0) \in S_\mathcal{P}$ is the initial state;*
- *$\Delta_\mathcal{P} \subseteq S_\mathcal{P} \times S_\mathcal{P}$ is the set of transitions;*
- *$w_p : \Delta_\mathcal{P} \to \mathbb{R}^+$ is the weight function defined as: $w_p\big((x,s),(x',s')\big) = w\big((x,x')\big)$;*
- *$F_\mathcal{P} = X \times F_{\mathcal{A}_\infty}$ is the set of accepting states.*

Let $p = (x, s) \in S_\mathcal{P}$ and $p' = (x', s') \in S_\mathcal{P}$ be states in product automaton $\mathcal{P}$. A transition from p to p', i.e., $(p, p') \in \Delta_\mathcal{P}$, implies a transition $(x, x') \in \Delta$ and $\delta(s, l(x')) = s'$. The notions of path and acceptance are the same as in dFSA. A satisfying run of $\mathcal{T}$ with respect to ϕ can be obtained by computing a path from the initial state to an accepting state over $\mathcal{P}$ and projecting the path onto $\mathcal{T}$.

The centralized solution of (3) requires to construct an aggregated product automaton $\mathcal{P}_{full} = \mathcal{T}_{full} \times \mathcal{A}_{\infty,full}$ where $\mathcal{T}_{full}$ is the cartesian product of $\mathcal{T}$ for n times, and $\mathcal{A}_{\infty,full} = \mathcal{A}_{\infty,1} \times \mathcal{A}_{\infty,2} \times \ldots \times \mathcal{A}_{\infty,n}$. Such an aggregated product automaton captures all possible agent movements and enable to find safe paths that satisfy the TWTL specifications (or their temporal relaxations). While such an approach can result in optimal agent trajectories, the complexity of constructing $\mathcal{P}_{full}$ exponentially grows as the number of agents n increases. Hence, we propose a decentralized approximate solution to Problem 1 by constructing individual product automaton of each agent and solve a planning problem over individual product automata by using only local neighborhood information.

IV. Receding Horizon Safe Path Planning with TWTL Satisfaction

The proposed decentralized approach is comprised of two parts. In the offline part, a product automaton $\mathcal{P}_i$ is constructed for each agent i given its transition system $\mathcal{T}$ and automaton $\mathcal{A}_{\infty,i}$ (representing all temporal relaxations of TWTL ϕ_i). Based on $\mathcal{P}_1, \ldots, \mathcal{P}_n$, each agent computes its nominal plan (from its initial product automaton state to an accepting state in $F_{\mathcal{P}_i}$ via a Dijkstra-based algorithm) irrespective of other agents' paths. Note that the path computed over $\mathcal{P}$ is $\mathbf{p_i} = (x_t^i, s_t^i)(x_{t+1}^i, s_{t+1}^i) \ldots$, so the corresponding path $\mathbf{x_i}$ on $\mathcal{T}$ can always be extracted from $\mathbf{p_i}$.

In the online portion, agents move according to the path $\mathbf{p_i}$. Whenever an agent encounters other agents in its local neighborhood, a negotiation protocol, i.e., priority assignment to the agents, is performed. This protocol is used to decide which agents are "yielded" to by considering their respective paths for collision avoidance. Such a priority assignment is achieved by the energy function defined over the product automaton states, which as a measure of "progress" towards satisfying the given TWTL specification.

Definition 5. *(Energy Function) Given a product automaton $\mathcal{P}$, the energy of each product automaton state $p \in S_\mathcal{P}$ is defined similar to [11] as*

$$E(p, \mathcal{P}) = \begin{cases} \min\limits_{p' \in F_\mathcal{P}} d\big(p, p'\big) & \text{if } p \notin F_\mathcal{P} \\ 0 & \text{otherwise} \end{cases} \tag{4}$$

where $d\big(p, p'\big)$ is the weighted graph distance between p and p'. If there is no such reachable accepting state $p' \in F_\mathcal{P}$, the energy of p is $E(p, \mathcal{P}) = \infty$. Any states in $\mathcal{P}$ with infinite energy, $E(p, \mathcal{P}) = \infty$, are pruned from $\mathcal{P}$ to ensure the satisfiability of the specification.

Each agent i is given an initial priority based on the energy of its current state, $E(p_t^i, \mathcal{P}_i)$. An agent with lower energy is given priority over the higher energy agents in its local neighborhood in order to ensure finite relaxation of all TWTL specifications as discussed in Sec. IV-B. If agents have equivalent energy then the higher priority agent is chosen at random between these agents. Note that the highest priority agent does not consider the other agents and is never required to do replanning; therefore, we can guarantee progress toward satisfying its TWTL specification.

For a local horizon path that consists of H-hops, a minimum communication neighborhood of $2H$ is required to detect collisions along the H-length path. We define this communication neighborhood of $2H$ as the local neighborhood. For an agent i, we consider the set of its $2H$-hop neighbors as $\mathcal{N}_i^{2H}$. For each agent i in the environment, the proposed method requires that all agents in its local neighborhood, $j \in \mathcal{N}_i^{2H}$, communicate their H-horizon path information, $\mathbf{p}_j = p_t^j, p_{t+1}^j, \ldots, p_{t+H}^j$ with agent i at any time instant t. The method also requires sharing of an indicator that states if the agent's path has been updated with respect to its local neighborhood. We denote this "update flag" as U_{flag} which is required for a proper path update protocol discussed below.

A. Algorithm Descriptions

The outline of the proposed method is shown in Fig. 1 for a single agent. Note that the proposed method runs

independently (for each agent) in a distributed manner.

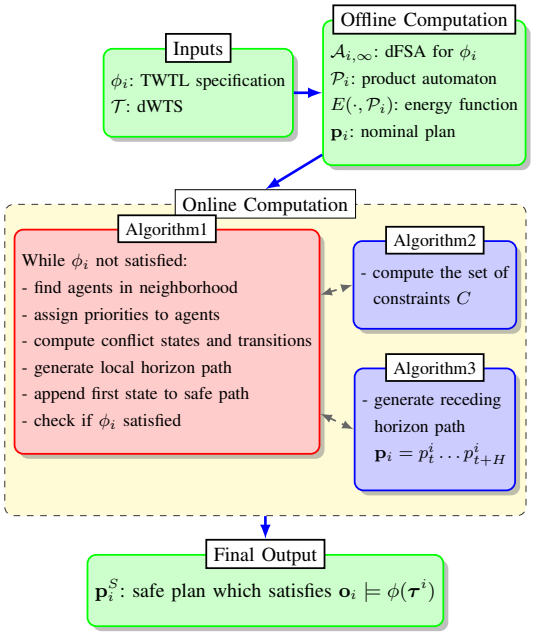

Fig. 1. Outline of the proposed method.

In the offline part, the product automaton and a nominal plan (without considering any other agents) are computed. The online part includes Algorithms 1-3. First, Alg. 1 takes in the initial information computed offline and executes until an accepting product automaton state is reached (i.e., satisfying either the original or the temporally relxaed TWTL formula, ϕ_i) (line 3). In each iteration of the while loop, the set of all agents within agent i's local neighborhood, $\mathcal{N}_i^{2H}$, is found in line 4. Next, the agent priorities are determined for all neighboring agents including the agent i itself (line 5). Note that agent priority depends on the energy associated with the agent's current state, $E(p_t^i, \mathcal{P}_i)$. Then, the set of agents which have a higher priority than agent i, i.e., HP_i is generated (line 6). Note that the set HP_i will be used by Alg. 2 to determine the set of conflicting states and transitions. Now, line 7 in Alg. 1 ensures that all agents in HP_i have been updated based on their local neighborhoods before agent i's path can be updated. This step is required to ensure that the current finite horizon path information obtained in line 8 from all agents in $\mathcal{N}_i^{2H}$ is correct (line 9) and the set of conflicting states and transitions, C, can be properly computed in (line 9). The, the set C is used in line 10 in order to find a finite horizon safe path starting from the current state (according to Def. 3). With an updated safe path, the finite-path information and update flag, which is now set to *true*, are shared with the neighboring agents (line 11). Note that line 12 ensures that the updated path is not executed until all agents in its local neighborhood have updated their own paths to account for potential collisions with higher priority agents. As in conventional receding horizon approaches, only the next state $\mathbf{p}_i(1)$ of the generated path is appended to the safe plan $\mathbf{p}_i^s$ (line 13). Lastly, line 14 checks if $\mathbf{p}_i(1)$ is an accepting product automaton state, and if so the agent has satisfied its TWTL specification, $\phi(\boldsymbol{\tau}^i)$, and is assigned with the lowest priority (line 15). The agent's path is advanced one step and the update flag is set back to *false* (line 16).

Algorithm 1: Online Safe Path Planning for Agent i

Input: $\mathcal{P}_i$, $\mathbf{p}_i$ Product Automaton and Nominal Plan
1 **Note:** $p_t^i = (x_t^i, s_t^i)$ is an element of the finite horizon path $\mathbf{p}_i = p_t^i, \ldots, p_{t+H}^i$;
2 **Initialization:** $x_0^i = p_{x,0}^i; \mathbf{p}_i^s = p_0^i; U_{flag} = false$;
3 **while** $p_t^i \notin F_{\mathcal{P}}$ **do**
4 Find the set of neighboring agents, $\mathcal{N}_i^{2H}$;
5 Compute agent priorities (including self);
6 Generate HP_i, the set of neighboring agents with higher priority;
7 Wait until all agents' U_{flag} in HP_i set to *true*;
8 Obtain H-hop path $\mathbf{p}_j$ from agents $\forall j \in \mathcal{N}_i^{2H}$;
9 $C \leftarrow \mathbf{algorithm2}(HP_i, \mathcal{P}_i, \mathbf{p}_i, \mathbf{p}_j | \forall j \in HP_i)$
10 $\mathbf{p}_i, U_{flag} \leftarrow \mathbf{algorithm3}(\mathcal{P}_i, p_t^i, C)$
11 Share path $\mathbf{p}_i$ and U_{flag} with neighboring agents, $\mathcal{N}_i^{2H}$;
12 Wait until all agents' U_{flag} in $\mathcal{N}_i^{2H}$ set to *true*;
13 $\mathbf{p}_i^s = \mathbf{p}_i^s, \mathbf{p}_i(1)$
14 **if** $\mathbf{p}_i(1) \in F_{\mathcal{P}}$ **then**
15 $HP_i = \mathcal{N}_i^{2H}$
16 $p_t^i = \mathbf{p}_i(0)$ and $U_{flag} = false$;

In Alg. 2, we account for two types of conflicts in order to guarantee a safe update in Alg. 3. The first type of conflict, C_X, enforces that agent i cannot occupy the same state with an agent of higher priority at the next time step as required by the first constraint in (3) (lines 4-5). The second type of conflict, C_Δ, enforces that no two agents' traverse the same transition (i.e., swapping) as the second constraint in (3) (lines 6-7). If any conflicts, C_X or C_Δ, are found then they are added to the set of conflicting states and transitions, C, with the hop h associated with them (line 8). Finally, lines 9-10 in Alg. 2 ensures that the highest priority agent cannot transition to higher energy states in the first hop. This is required in order to guarantee progress toward TWTL satisfaction discussed in Sec. IV-B-Thm. 2.

Algorithm 2: Find Conflicts for Agent i

Input: $HP_i, \mathcal{P}_i$ - Set of higher priority agents than agent i in local neighborhood, product automaton
Input: $\mathbf{p}_i, \mathbf{p}_j | \forall j \in HP_i$ - finite horizon paths considered for conflicts (Recall $\mathbf{p}_i = p_t^i, \ldots, p_{t+H}^i$)
Output: C - Set of conflict states and transitions
1 **Initialization:** $C_X = \emptyset, C_\Delta = \emptyset$, $C = (C_X, C_\Delta)$, $x_0^i = x_t^i$, $x_0^j = x_t^j$
2 **for** $h = 1 : H$ **do**
3 **for** *each agent j in* HP_i **do**
4 **if** $x_h^i = x_h^j$ **then**
5 $C_{X,h} = x_h^j$
6 **if** $(x_t^i, x_{t+1}^i) \neq (x_{t+1}^j, x_t^j)$ **then**
7 $C_{\Delta,h} = (x_{h-1}^i, x_h^i)$
8 $C = C \cup (C_{X,h}, C_{\Delta,h})$
9 **if** $HP_i == \emptyset$ *and* $E(p_t^i, \mathcal{P}_i) > 0$ **then**
10 $C_{\Delta,1} = \{(x_t^i, x') \text{ from } (p_t^i, p') \in \Delta_{\mathcal{P}_i} | E_i(p', \mathcal{P}_i) \geq E(p_t^i, \mathcal{P}_i)\}$
11 **return** $C = (C_X, C_\Delta)$

The objective of Alg. 3 is to generate a local H-hop path over the agent's product automaton, $\mathcal{P}_i$. Alg. 3 essentially performs a depth-first-like search from the current state p_t^i on $\mathcal{P}_i$ to find a conflict-free H-length path that has the minimum the sum of energies. To this end, the set of H-length safe paths originating from p_t^i is found in lines 1 and 2 by making sure that each $p_{t+h}^i = (x_h^i, s_h^i)$ over an H-length safe path satisfies $x_h^i \notin C_{X,h}$ and $(x_{h-1}^i, x_h^i) \notin C_{\Delta,h}$ for all $h = \{1, \ldots, H\}$. Then, the cost of each path is computed as the sum of energies of each state in the path (line 3). Finally, the path with lowest cumulative energy is chosen (line 4) and the update flag is set to *true* (line 5). This path and update flag are then returned to line 10 of Alg. 1.

Algorithm 3: Receding Horizon Plan for Agent i

Input: $\mathcal{P}_i, p_t^i$ - Product automaton and current state
Input: C - Set of conflict states and transitions
Output: $\mathbf{p}_i, U_{flag}$ - Conflict-free path; Update flag

1 Find the set of all H-length paths starting from p_t^i, $\mathbf{P}^H$;
2 Find safe paths $\mathbf{P}_s \subseteq \mathbf{P}^H$ by removing the paths that contain conflicts based on $C = (C_X, C_\Delta)$;
3 Calculate the cumulative energy E_{path}^k of each $\mathbf{p}^k \in \mathbf{P}_s$;
4 $\mathbf{p}_i = arg \min_{\mathbf{p}^k \in \mathbf{P}_s} (E_{path}^k)$;
5 $U_{flag} = true$
6 **return** $\mathbf{p}_i, U_{flag}$

B. Theoretical Results

The proposed method guarantees the generation of a safe path which satisfies the corresponding agent's TWTL formula or a finite relaxation of it, $\phi(\boldsymbol{\tau})$. The proofs of Theorems 1 and 2 are omitted due to space limitations and can be found in [16].

Theorem 1. *(Safety) If the environment graph $\mathcal{G} = (X, \Delta, w)$ is n-connected, then Alg. 1 always produces a safe path $\mathbf{p}_i$ which satisfies the properties in Def. 3.*

Theorem 2. *(Satisfaction of TWTL with Finite Relaxation) If agent i has an initial product automaton state p_0^i such that $E(p_0^i, \mathcal{P}_i) < \infty$, then Alg. 1 produces a finite path which satisfies the given TWTL specification or a relaxed version of the specification, $\phi(\boldsymbol{\tau}^i)$.*

V. Results

The code for our simulations and experiments is derived from the PyTWTL package (`hyness.bu.edu/twtl`) which handles the construction of $\mathcal{A}_\infty$ corresponding to a given TWTL ϕ, and the creation of the product automaton $\mathcal{P}$ given a transition system $\mathcal{T}$. The algorithms presented in this paper can be found at `https://github.com/pete9936/pyTWTL_ObsAvoid`. All simulations were carried out on a desktop computer with 4 cores running Ubuntu 16.04, 4.0GHz CPU, and 32GB of RAM.

In simulations, five agents moving in an environment shown in Fig. 2 are considered, where each agent is given a different TWTL specification defined over the environment.

$$\begin{aligned}
\phi_1 &= [\mathrm{H}^2 B]^{[0,(6+\tau_1^1)]} \cdot [\mathrm{H}^1 A]^{[0,(5+\tau_2^1)]}, \\
\phi_2 &= [\mathrm{H}^2 B]^{[0,(5+\tau_1^2)]} \cdot [\mathrm{H}^1 C]^{[0,(4+\tau_2^2)]}, \\
\phi_3 &= [\mathrm{H}^1 D]^{[0,(4+\tau_1^3)]} \cdot [\mathrm{H}^2 F]^{[0,(4+\tau_2^3)]}, \\
\phi_4 &= [\mathrm{H}^1 E]^{[0,(5+\tau_1^4)]} \cdot [\mathrm{H}^1 Base4]^{[0,(3+\tau_2^4)]}, \\
\phi_5 &= [\mathrm{H}^1 G]^{[0,(6+\tau_1^5)]} \cdot [\mathrm{H}^1 Base5]^{[0,(6+\tau_2^5)]},
\end{aligned} \tag{5}$$

where τ_1^i and τ_2^i refer to the temporal relaxations of the corresponding tasks of agent i. All five TWTL specifications are examples of servicing some regions in sequence and they are kept relatively simple for illustration purposes. Note that the proposed approach can also be used for more complex tasks due to the rich expressiveness of the TWTL language.

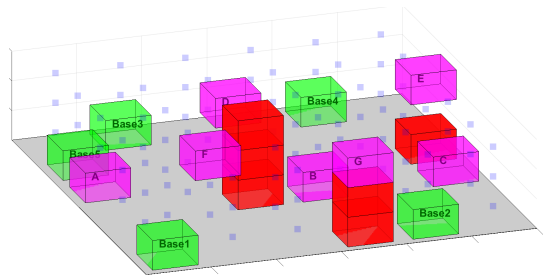

Fig. 2. The 6 x 6 x 3 discretized environment shared by 5 agents. Initial positions are given by the green nodes. Obstacles are shown in red, and seven regions of interest (A, B, C, D, E, F, G) are shown in magenta. Agents can stay put or move to any neighboring cell (non-obstacle) in one time step.

For each agent, the temporal relaxations of the nominal and collision-free paths are highlighted in Tab. I. For the given scenario, Tab. I shows that only Agent 1 needs positive temporal relaxation. This is due to Agent 1 waiting Agent 2 to satisfy the "service B for 2 time units" part of its formula since Agent 2 has higher priority (lower energy) than Agent 1 as illustrated in Fig. 3. Notice that while Agent 1 is waiting between $t \in [2, 4]$, its energy is not decreasing (no progress towards task satisfaction) in Fig. 3. Also, negative relaxation in Tab. I implies that the formulae are satisfied within a shorter time window than the time window originally allotted.

TABLE I
Temporal Relaxation (τ) of paths.

	Nominal paths					Safe paths				
	$\mathbf{x_1}$	$\mathbf{x_2}$	$\mathbf{x_3}$	$\mathbf{x_4}$	$\mathbf{x_5}$	$\mathbf{x_1}^S$	$\mathbf{x_2}^S$	$\mathbf{x_3}^S$	$\mathbf{x_4}^S$	$\mathbf{x_5}^S$
τ_1	$\mathbf{-1}$	-1	-1	-1	-1	$\mathbf{+1}$	-1	-1	-1	-1
τ_2	-1	-1	0	0	-1	-1	-1	0	0	-1

For the $6 \times 6 \times 3$ environment shown in Fig. 2, each respective agent has a transition system $\mathcal{T}$ of $(102; 1594)$ states and transitions and a product automaton $\mathcal{P}_i$ of about (200; 3200) states and transitions. The offline part of our approach (generating $\mathcal{T}$, $\mathcal{P}_{1\ldots5}$, $E_{i\ldots}$, and the nominal paths $\mathbf{p}_1, \ldots, \mathbf{p}_5$) takes 3.90 seconds. Generating the collision-free

paths (the online portion of our algorithm) with a horizon length $H = 3$ takes 1.17 seconds. However, the metric of greater concern is the average time for an individual agent's receding horizon update which is approximately 0.0275 seconds, fast enough for real-time execution.

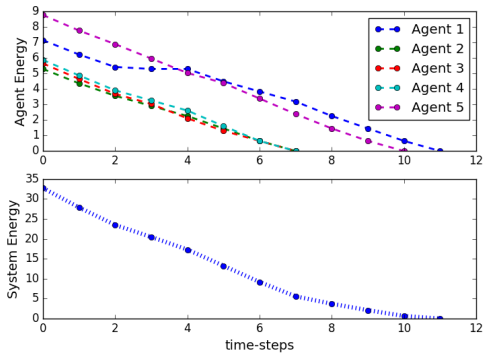

Fig. 3. The agent energy values at each state along the paths are shown as well as the collective energy of the system. These energy values correspond to each state along the path which leads to satisfaction of each agents' TWTL formula shown in (5).

We define the iteration time as the time consumed by a single agent to update its receding horizon path. Table II shows the total online run-time (i.e., the sum of all iteration times of all agents) and the average iteration time for varying lengths of horizon. As seen from Tab. II, the average iteration time (and the total run time) significantly grows when the horizon, H, increases. Note that the safe paths generated based on the TWTL formulae (5) trying to be satisfied in Fig. 2 were identical for all values of H, meaning that extended path information from neighboring agents did not affect the outcome of this scenario. However, in more constrained environments such as a narrow corridor with many agents, increasing the horizon will potentially influence the outcome as longer paths will include more information from the neighbors.

TABLE II

HOW RECEDING HORIZON LENGTH IMPACTS EXECUTION TIME

	$H = 2$	$H = 3$	$H = 4$
Total Online Run-time (s)	0.15	1.17	68.1
Avg. Iteration time (s)	0.0037	0.0275	1.621

The proposed method is also verified experimentally on a team of five Crazyflies 2.0. All experiments are conducted in a testing environment of $3m \times 3m \times 1.5m$ motion-capture space identical to the simulation environment shown in Fig. 2, using a `VICON` camera system with 8 cameras. We use the Crazyswarm package [17] to perform the low-level control algorithms, communication, and interface of the `VICON` system with the Crazyflies. Note that in the experiments a "downwash zone" below each quadrotor was considered, since entering these regions would lead to loss of flight. This restriction was not formally considered in the presented method since this is both an environment and agent-dependent constraint. A video of the experiment can be found at `https://youtu.be/qQADyO1pYr8`.

VI. CONCLUSIONS AND FUTURE WORK

We present an automaton-theoretic approach for generating collision-free paths for a multi-agent system such that each agent satisfies an individual task encoded as a TWTL specification in finite time. The proposed approach guarantees both collision avoidance among agents (and static obstacles) and the satisfaction of the individual TWTL specifications (with a finite relaxation) given some mild assumptions on the environment.

As a future direction, we plan to extend the proposed algorithm to achieve safe planning in more cluttered environments with weaker connectivity properties. Furthermore, we are interested in extending this approach to safely serve cooperative tasks that require the coordination of multiple agents (e.g., [18], [19]).

REFERENCES

[1] L. Wang, A. D. Ames, and M. Egerstedt, "Safety barrier certificates for collisions-free multirobot systems," *IEEE Transactions on Robotics*, vol. 33, no. 3, pp. 661–674, 2017.

[2] C. E. Luis and A. P. Schoellig, "Trajectory generation for multiagent point-to-point transitions via distributed model predictive control," *IEEE Robotics and Aut. Letters*, vol. 4, no. 2, pp. 375–382, 2019.

[3] T. Mylvaganam, M. Sassano, and A. Astolfi, "A differential game approach to multi-agent collision avoidance," *IEEE Transactions on Automatic Control*, vol. 62, no. 8, pp. 4229–4235, 2017.

[4] C. Baier, J.-P. Katoen *et al.*, *Principles of Model Checking*. MIT press Cambridge, 2008, vol. 26202649.

[5] D. Aksaray, K. Leahy, and C. Belta, "Distributed multi-agent persistent surveillance under temporal logic constraints," *IFAC-PapersOnLine*, vol. 48, no. 22, pp. 174–179, 2015.

[6] G. E. Fainekos, A. Girard, H. Kress-Gazit, and G. J. Pappas, "Temporal logic motion planning for dynamic robots," *Automatica*, vol. 45, no. 2, pp. 343–352, 2009.

[7] E. M. Wolff, U. Topcu, and R. M. Murray, "Optimization-based trajectory generation with linear temporal logic specifications," in *IEEE Int. Conf. on Robotics and Automation*, 2014, pp. 5319–5325.

[8] R. Koymans, "Specifying real-time properties with metric temporal logic," *Real-Time Systems*, vol. 2, no. 4, pp. 255–299, 1990. [Online]. Available: http://dx.doi.org/10.1007/BF01995674

[9] I. Tkachev and A. Abate, "Formula-free Finite Abstractions for Linear Temporal Verification of Stochastic Hybrid Systems," in *Int. Conf. on Hybrid Systems: Computation and Control*, 2013.

[10] C.-I. Vasile, D. Aksaray, and C. Belta, "Time window temporal logic," *Theoretical Computer Science*, vol. 691, pp. 27–54, 2017.

[11] X. Ding, M. Lazar, and C. Belta, "LTL receding horizon control for finite deterministic systems," *Automatica*, vol. 50, no. 2, 2014.

[12] C.-I. Vasile and C. Belta, "An Automata-Theoretic Approach to the Vehicle Routing Problem," in *Proceedings of the Robotics: Science and Systems (RSS)*, Berkeley, California, USA, July 2014.

[13] D. Aksaray, C.-I. Vasile, and C. Belta, "Dynamic routing of energy-aware vehicles with temporal logic constraints," in *Int. Conference on Robotics and Automation (ICRA)*. IEEE, 2016, pp. 3141–3146.

[14] G. J. Pappas, "Bisimilar linear systems," *Automatica*, vol. 39, no. 12, pp. 2035–2047, 2003.

[15] A. Nikou, D. Boskos, J. Tumova, and D. V. Dimarogonas, "On the timed temporal logic planning of coupled multi-agent systems," *Automatica*, vol. 97, pp. 339–345, 2018.

[16] R. Peterson, A. T. Buyukkocak, D. Aksaray, and Y. Yazicioglu, "Decentralized safe reactive planning under TWTL specifications," *arXiv preprint arXiv:2007.12278*, 2020.

[17] J. A. Preiss, W. Honig, G. S. Sukhatme, and N. Ayanian, "Crazyswarm: A large nano-quadcopter swarm," in *Int. Conference on Robotics and Automation (ICRA)*. IEEE, 2017, pp. 3299–3304.

[18] R. Bhat, Y. Yazıcıoğlu, and D. Aksaray, "Distributed path planning for executing cooperative tasks with time windows," *IFAC-PapersOnLine*, vol. 52, no. 20, pp. 187–192, 2019.

[19] S. Seyedi, Y. Yazıcıoğlu, and D. Aksaray, "Persistent surveillance with energy-constrained uavs and mobile charging stations," *IFAC-PapersOnLine*, vol. 52, no. 20, pp. 193–198, 2019.