AIAA SciTech Forum
11–15 & 19–21 January 2021, VIRTUAL EVENT
AIAA Scitech 2021 Forum

10.2514/6.2021-1123

Distributed Planning of Multi-Agent Systems with Coupled Temporal Logic Specifications

Ali Tevfik Buyukkocak*, Derya Aksaray†, Yasin Yazıcıoğlu‡
University of Minnesota, Minneapolis, MN, 55455

In this paper, we investigate the multi-agent mission planning and control problem under coordination constraints such as collision avoidance and spatial coherence. We propose a sequential trajectory planning algorithm that is run by each agent in a distributed manner. Mission requirements including servicing specific regions, avoiding static and dynamic obstacles (other agents), and flying close to the other agents are incorporated into the problem as Signal Temporal Logic (STL) constraints. The planning problem under STL constraints is formulated as a Mixed Integer Linear Program and an off-the-shelf tool is used to find the high-level trajectories. The resulting high-level trajectories are then tracked by the agents via Model Predictive Control. Illustrative simulation and experiment results are presented in addition to comparing the proposed method with a centralized solution.

I. Introduction

Recent advances in sensing, communication, and computation capabilities have widely expanded the application areas of multi-agent systems. In particular, there has been a significant research interest in the control of multi-agent systems (e.g., a team of quadrotors, robots) performing complex high level specifications in applications including but not limited to package delivery, reconnaissance and surveillance, warehouse servicing, and inspection. Design of scalable and provably correct algorithms for achieving collaboration among the agents such as formation (e.g., [1]), consensus (e.g., [2]), task allocation (e.g., [3]), coverage control (e.g., [4]), or persistent surveillance (e.g., [5]) has also gained significant attention in recent years.

In many applications, multi-agent systems may need to achieve tasks with complex requirements. For example, consider the following cooperative task: "a group of quadrotors with high resolution camera needs to visit a couple of goal regions within a minute by monitoring each goal region for at least 5 seconds while avoiding collisions with obstacles and other agents in the environment and flying close to each other during the mission." As the tasks get more complex by including spatial, temporal, and logical relationships, it gets harder to express them with algebraic equations. On the other hand, temporal logics [6] compactly represent such complex tasks. For example, the constraint or objective of each agent in a multi-agent system can be defined as a Linear Temporal Logic (LTL), and the multi-agent planning problem solves for each agent trajectory that guarantees the satisfaction of corresponding local LTL specification [7–11].

Multi-agent missions often require agents to service the specific regions for a certain amount of time such as picking up and delivering a package and monitoring a specific region. Using LTL or simple time or fuel optimal trajectories, one might be able to move an agent to the desired region [12]. However, time bounds and servicing duration cannot be explicitly incorporated along with the multistage complex tasks. For such cases, bounded temporal logics such as Time Window Temporal Logic (TWTL) [13], Metric Temporal Logic (MTL) [14], or Signal Temporal Logic (STL) [15] are more suitable languages to express rich spatial and temporal properties. For example, [16] solves a multi-agent planning problem with TWTL specifications, where agents achieve their individual TWTL tasks while avoiding collisions with each other. However, TWTL specifications can be used in problems with discretized state-space. On the other hand, STL can be used in continuous space problems. Moreover, it is different from other temporal logics by including predicates in the form of inequalities and endowing a robustness degree metric quantifying the degree of satisfaction. Such a metric enables one to encode an STL specification into some mixed integer equality and inequality constraints that can be used in formulating optimization problems such as Mixed-Integer Linear Program (MILP) [17].

In addition to respective complex mission specifications, the planning of multi-agent systems often includes some relative distance constraints. For example, one might desire agents not to arbitrarily spread over the environment, in

*Graduater Student, Department of Aerospace Engineering and Mechanics, AIAA Student Member.
†Assistant Professor, Department of Aerospace Engineering and Mechanics, AIAA Member.
‡Research Assistant Professor, Department of Electrical and Computer Engineering.

other words agents might be desired to fly with spatial coherence. By doing so, if an agent fails, there will always be some neighboring agents that could take over the mission in a redundant manner (e.g., [18]). Also, collaboration among agents may require uninterrupted communication among the agents which might be another reason for agents to fly close to each other. Another critical requirement for the safe operation of multi-agent systems is the avoidance of collisions with both other agents and static obstacles since the environment might be cluttered.

Relative distance constraints such as collision avoidance and coherence impose agents to consider the other agents in the planning of their trajectories. In other words, specifications of each agent have to depend on the states of the other agents. With such coupled temporal logic specifications, the complexity of centralized trajectory planning grows exponentially with the number of agents. For this reason, significant effort has been devoted to the design of distributed or decentralized algorithms. Various methods such as potential fields (e.g., [19]), control barrier functions (e.g., [20]), and sequential convex programming (e.g. [21]) have been employed in the literature for distributed multi-agent planning. These methods typically have not accommodated complex spatio-temporal requirements that can be expressed as temporal logics. Recent work using STL in multi-agent planning, on the other hand, accounts for such relative distance constraints only partially inside the temporal logic specifications [7, 22]. They rather handle them inside the agent dynamics or with local planners in both centralized and decentralized manners. In [23] multi-agent planning problem is solved by comprehensive STL specifications with trajectories sampled with different rates in a centralized way. In this study, we consider all the mission specifications regarding the servicing of regions with explicit time bounds, avoidance of collisions with static and dynamic obstacles, and spatial coherence inside compact and expressive STL specifications, and we solve for the trajectories that satisfy these coupled specifications in a distributed manner.

When it is impossible to find trajectories that satisfy the original specification, it may be desired to satisfy some relaxed versions of the specification instead. In [24] slack variables are added to avoid infeasibilities but this can lead to endangering the safety. The framework in [25] gives feedback on why the specification is not satisfiable and how to modify it. The approach of [26] minimizes the violation by considering both hard and soft constraints, where the hard constraints such as collision avoidance must be satisfied. Alternatively, as in [21], our algorithm considers only soft constraints for all specifications in the mission via relaxations; however, it mandates a relaxation-free solution to the original specification by diminishing the intermediate relaxations through iterative trajectory adjustments until a feasible solution is found similar to diagnosis and repair scheme of [27].

The remainder of the paper is organized as follows: Section II presents some preliminaries regarding Signal Temporal Logic. Section III formulates the multi-agent coordination problem under STL constraints. Section IV introduces our solution approach, that is, sequential trajectory planning. Then, simulations for illustrative case studies for indoor monitoring missions implemented by quadrotor platforms are discussed in Section V together with an experimental study. Finally, Section VI concludes the paper and provides some future directions.

II. Signal Temporal Logic

We use a fragment of Signal Temporal Logic (STL) [15] to represent individual specifications of the agents in space and time. STL is a bounded time temporal logic that can express rich time series. For example, STL can compactly and rigorously express mission requirements such as: "a quadrotor should be visiting the region A in the first 30 seconds for at least 5 consecutive seconds" and "another quadrotor should be flying at most 5 meters away from the first one while keep visiting region B in every 15 seconds." Therefore, STL is quite suitable to represent multi-agent mission specifications including coordination constraints among the agents as well as other individual tasks. STL syntax is given as

$$\phi ::= \top \mid \mu \mid \neg\phi \mid \phi_1 \wedge \phi_2 \mid F_{[a,b]}\phi, \tag{1}$$

where $a, b \in \mathbb{R}_{\geq 0}$ are the time bounds with $a \leq b$; $\neg$, $\wedge$, and F are the negation, conjunction (i.e., and), and finally (i.e., eventually) operators, respectively; ϕ, ϕ_1, and ϕ_2 are STL specifications, $\top$ is the Boolean True, and μ is a predicate in an inequality form such as $\mu = f(\mathbf{x}) \sim w$ with a scalar $w \in \mathbb{R}$, a signal $\mathbf{x} : \mathbb{R}_{\geq 0} \to \mathbb{R}^n$, a function $f : \mathbb{R}^n \to \mathbb{R}$, and a relation $\sim \in \{\leq, \geq\}$. Remaining useful operators are generated from the others as follows: $G_{[a,b]}\phi = \neg F_{[a,b]}\neg\phi$ is the globally (i.e., always) operator, and $\phi_1 \vee \phi_2 = \neg(\neg\phi_1 \wedge \neg\phi_2)$ is the disjunction (i.e., or) operator.

For any signal $\mathbf{x}$ representing the trajectory of the system, let $\mathbf{x}_t$ denote the value of $\mathbf{x}$ at time t and $(\mathbf{x}, t)$ be the part of the signal that is a sequence of $\mathbf{x}_{t'}$ for $t' \in [t, \infty)$. The satisfaction of an STL specification by the signal $(\mathbf{x}, t)$ is then

determined as

$$
\begin{aligned}
&(\mathbf{x}, t) \vDash \mu \iff f(\mathbf{x}_t) \sim w, \\
&(\mathbf{x}, t) \vDash \neg\mu \iff \neg\big((\mathbf{x}, t) \vDash \mu\big), \\
&(\mathbf{x}, t) \vDash \phi_1 \wedge \phi_2 \iff (\mathbf{x}, t) \vDash \phi_1 \text{ and } (\mathbf{x}, t) \vDash \phi_2, \\
&(\mathbf{x}, t) \vDash \phi_1 \vee \phi_2 \iff (\mathbf{x}, t) \vDash \phi_1 \text{ or } (\mathbf{x}, t) \vDash \phi_2, \\
&(\mathbf{x}, t) \vDash G_{[a,b]}\phi \iff \forall\, t' \in [t+a, t+b],\ (\mathbf{x}, t') \vDash \phi, \\
&(\mathbf{x}, t) \vDash F_{[a,b]}\phi \iff \exists\, t' \in [t+a, t+b],\ (\mathbf{x}, t') \vDash \phi.
\end{aligned}
\tag{2}
$$

While the expressions of $(\mathbf{x}, t) \vDash F_{[a,b]}\phi$ implies that ϕ is satisfied at some time instant in $[t+a, t+b]$, $(\mathbf{x}, t) \vDash G_{[a,b]}\phi$ denotes that ϕ is satisfied at all time instants between $[t+a, t+b]$. We assume finite-time STL specifications in this study, i.e., $a, b < \infty$. We also consider $t = 0$ as the initial time for the specifications, and denote $(\mathbf{x}, 0)$ with $\mathbf{x}$ for simplicity.

Apart from other temporal logics, most of which denote the satisfaction of a specification as either `True` or `False`, STL has a notion of *robustness degree* to quantify how well an STL specification is satisfied by measuring the distance to the violation, and it can be formally and recursively defined as [28],[29]:

$$
\begin{aligned}
\rho(\mathbf{x}, f(\mathbf{x}) \geq w, t) &= f(\mathbf{x}_t) - w, \\
\rho(\mathbf{x}, \neg(f(\mathbf{x}) \geq w), t) &= -\rho(\mathbf{x}, f(\mathbf{x}) \geq w, t), \\
\rho(\mathbf{x}, \phi_1 \wedge \phi_2, t) &= \min\big(\rho(\mathbf{x}, \phi_1, t), \rho(\mathbf{x}, \phi_2, t)\big), \\
\rho(\mathbf{x}, \phi_1 \vee \phi_2, t) &= \max\big(\rho(\mathbf{x}, \phi_1, t), \rho(\mathbf{x}, \phi_2, t)\big), \\
\rho(\mathbf{x}, F_{[a,b]}\phi, t) &= \max_{t' \in [t+a, t+b]} \rho(\mathbf{x}, \phi, t'), \\
\rho(\mathbf{x}, G_{[a,b]}\phi, t) &= \min_{t' \in [t+a, t+b]} \rho(\mathbf{x}, \phi, t').
\end{aligned}
\tag{3}
$$

While positive robustness degree indicates the satisfaction of specification ($\rho(\mathbf{x}, \phi, t) > 0 \Rightarrow (\mathbf{x}, t) \vDash \phi$), negative one represents a violation ($\rho(\mathbf{x}, \phi, t) < 0 \Rightarrow (\mathbf{x}, t) \nvDash \phi$). Although a zero robustness degree is inconclusive, we consider it as a satisfying case in this paper, i.e., $\rho(\mathbf{x}, \phi, t) \geq 0 \Rightarrow (\mathbf{x}, t) \vDash \phi$.

Example 1 *Consider a signal y. Assume that we have some constraints on this signal such that it needs to be* always *between* $0.8 \leq y \leq 1.2$ *along a mission of* 10 *s. This specification is represented as an STL specification, φ_1, in Fig. 1. Note that this STL specification is violated since the signal is out of the desired region for $t \in (2, 5)$. Moreover, the robustness degree is* $\rho(y, \varphi_1, 0) = \min_{t \in [0,10]} \big[\min\big((1.2 - y(t)), (y(t) - 0.8)\big)\big] \approx -0.4 < 0$ *which is determined by the furthest distance to the boundary (happening at $t = 3$). Now let another specification, φ_2, be "eventually the signal must satisfy $y \geq 1.2$ for any two consecutive time units along a mission horizon of* 10 *s." This requirement can be expressed with a combination of* eventually *and* globally *operators with the given predicate. As shown in Fig 1, the signal satisfies $y \geq 1.2$ for $t \in [2, 5]$ which implies the satisfaction of the specification with a robustness degree of* $\rho(y, \varphi_2, 0) = \max_{t \in [0,8]} \big(\min_{t' \in [t+0, t+2]}(y(t') - 1.2)\big) \approx 0.13 \geq 0$.

The combination of multiple linear inequalities (predicates) can be used in STL specifications to define regions of interest in the form of polygons. Let a region bounded by the lines of $x = 1$, $x = 2$, $y = 3$, and $y = 4$ which forms actually a square. Then we can use *conjunction* to define the region as

$$
Region = \big(x \geq 1 \wedge x \leq 2 \wedge y \geq 3 \wedge y \leq 4\big). \tag{4}
$$

Suppose that a quadrotor has a specification of $\phi_1 = G_{[0,\tau]}\big(Region\big)$, this would require the quadrotor to stay inside the region, i.e., satisfy the inequalities in Eq. (4), for all the time instants between $t = 0$ and $t = \tau$. Now going further let another specification be defined as $\phi_2 = F_{[0,T]}\phi_1$, i.e., $\phi_2 = F_{[0,T]}G_{[0,\tau]}\big(Region\big)$. This requires the quadrotor to visit region A for the next consecutive τ seconds at any time between $t = 0$ and $t = T$. Horizon of such a specification would be $hrz(\phi_2) = T + \tau$ which is mainly the time required to decide whether the specification is satisfied [30]. In many applications of cyber-physical systems, more than one region of interest is tried to be serviced. Assuming we have three regions of interest, namely A, B, and C with the respective servicing requirements, one compact STL specification to represent all spatio-temporal goals can be defined by the conjunction operators as

$$
\phi_{service} = F_{[0,T]}G_{[0,\tau_1]}\big(Region\ A\big) \wedge F_{[0,T]}G_{[0,\tau_2]}\big(Region\ B\big) \wedge F_{[0,T]}G_{[0,\tau_3]}\big(Region\ C\big). \tag{5}
$$

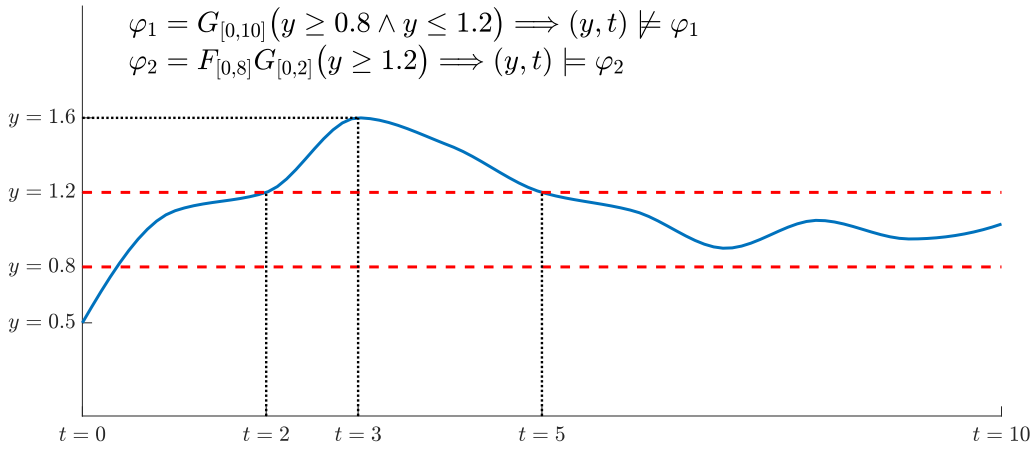


Fig. 1 Evolution of signal y (in blue) in time along with the linear constraints in red that constitute the predicates of the STL specifications. While φ_1 constrains y to stay between the bounds all the time (violated), φ_2 requires y $\geq$ 1.2 for at least two consecutive time units which is achieved successfully.

III. Problem Statement

Let $\mathrm{x} \in \mathbb{R}^n$ be the state of an agent and $u \in \mathbb{R}^m$ be the control input. Motion planning of an agent with the dynamics of $\dot{\mathrm{x}} = f(\mathrm{x}, u)$ requires the finding of a respective trajectory **x** optimizing a desired objective and ensuring the satisfaction of desired constraints. Any continuous-time dynamical system can be converted to a discrete-time model. We assume that the quadrotor has a linearized dynamics as

$$\mathrm{x}^+ = A\,\mathrm{x} + B\,u, \tag{6}$$

with $A \in \mathbb{R}^{n\times n}$, $B \in \mathbb{R}^{n\times m}$ and consider the state trajectory **x** as the signal we defined in Sec. II. Complex specifications and requirements on the trajectory can be incorporated as STL constraints. In addition to servicing demands given in Eq. (5), multi-agent operations often require coordination between agents. In the presence of more than one agent in a shared environment, such coordination can be achieved by incorporating the state information of other agents into the respective STL specification of each agent. For example, $G_{[0,t_f]}\big(d(i,j) \geq r_{min}\big)$ implies a coupled temporal logic such that the Euclidean distance $d(i,j) \in \mathbb{R}$ between two agents i and j needs to be greater than or equal to r_{min} always between $t = 0$ and $t = t_f$ to enforce collision avoidance. Furthermore, in multi-agent missions, one might desire the agents to operate close to each other for spatial coherence. In other words, the maximum distance between the agents can be limited to r_{max}. Similar to collision avoidance, such a requirement can also be expressed via STL as $G_{[0,t_f]}\big(d(i,j) \leq r_{max}\big)$. Moreover, there may be several static obstacles to avoid in the environment as well. As in Eq. (4), we can define obstacles as the regions that are bounded by line equations. These lines are combined by *disjunction* operators this time, since satisfying only one of them is enough for collision avoidance. To illustrate, let an obstacle be bounded by the lines $x = -2$, $x = -1$, $y = -0.5$, and $y = 0.5$. Then the predicates that define the obstacle are combined as follows:

$$Obs = \big(x \leq -2 \vee x \geq -1 \vee y \leq -0.5 \vee y \geq 0.5\big). \tag{7}$$

It is worth to mention that the obstacles can also be defined as the negated inner polygons, e.g., negation of a region defined as in Eq. (4) combined with conjunction operators. However, since the negation operator duplicates the number of constraints and variables that will be discussed in Sec. IV, we represent static obstacle avoidance by the disjunction as in (7). Note that such safety requirements need to be met all the time during the mission, i.e., between $t = 0$ and $t = t_f$ where t_f is the final mission time. Therefore the *globally* operator of STL is quite convenient to use while enforcing such specifications from agent i's perspective to a system including n agents as

$$\phi_{coll.avoid.} = G_{[0,t_f]}\Big(\bigwedge_{j=1\neq i}^{n} \|\mathrm{x_i} - \mathrm{x_j}\| \geq r_{min}\Big), \tag{8a}$$

$$\phi_{coherence} = G_{[0,t_f]}\Big(\bigwedge_{j=1\neq i}^{n} \|\mathrm{x_i} - \mathrm{x_j}\| \leq r_{max}\Big), \tag{8b}$$

$$\phi_{obstacle} = G_{[0,t_f]}\big(Obstacles\big), \tag{8c}$$

where we denote $Obstacles = Obs_1 \wedge Obs_2 \wedge \dots$ as the combination of static obstacle avoidance constraints. While servicing of goal regions and static obstacle avoidance specifications depend only on the respective agent's state and static positions of the goal and obstacle regions, collision avoidance and coherence specifications require the variable state information of other agents in the environment. In this manner, we can categorize the specifications as *uncoupled* and *coupled* and represent the overall STL specification for a single agent as

$$\Phi = \phi_{u/coup.} \wedge \phi_{coup.}, \tag{9}$$

where $\phi_{u/coup.} = \phi_{service} \wedge \phi_{obstacle}$ and $\phi_{coup.} = \phi_{coll.avoid.} \wedge \phi_{coherence}$. Using the STL specification in Eq. (9), the optimal control problem we aim to solve in this paper is optimizing an objective function (e.g., minimizing the total agent movement) subject to the agent dynamics and the satisfaction of Φ. Accordingly, we formally define the problem as follows:

Problem 1 *Suppose a multi-agent system comprising n agents moving in a shared environment according to* $\mathrm{x}_i(t+1) = A\,\mathrm{x}_i(t) + B\,u_i(t)$ *and required to satisfy an individual STL specification* Φ_i *for* $i = 1, \dots, n$. *Find optimal policies* $\boldsymbol{u}_1^*, \dots, \boldsymbol{u}_n^*$ *to generate trajectories* $\mathbf{x}_1, \dots, \mathbf{x}_n$ *satisfying given STL specifications where* $\boldsymbol{u}_i^* = \big[u_i^*(0)\ u_i^*(1)\ \cdots\ u_i^*(H-1)\big] \in \mathbb{R}^{m\times H}$ *and* $\mathbf{x}_i = \big[\mathrm{x}_i(0)\ \mathrm{x}_i(1)\ \cdots\ \mathrm{x}_i(H)\big] \in \mathbb{R}^{n\times(H+1)}$.

$$\begin{aligned} \{\boldsymbol{u}_1^*, \dots, \boldsymbol{u}_n^*\} = \arg\min \sum_{i=1}^{n} \sum_{t=0}^{H-1} \mathcal{J}_u\big(u_i(t)\big) \\ s.t. \quad \mathrm{x}_i(t+1) = A\,\mathrm{x}_i(t) + B\,u_i(t),\ u_{min} \leq u_i(t) \leq u_{max},\ \forall t, \\ \rho(\mathbf{x}_i, \Phi_i, 0) \geq 0 \quad for\ i = 1, \dots, n. \end{aligned} \tag{10}$$

where $H \geq \max(hrz(\Phi_i))$ is the overall mission horizon which must be long enough to decide the satisfaction of the specifications, i.e., longer than the horizon of any specification, $\mathcal{J}_u\big(u_i(t)\big) \in \mathbb{R}$ is a running cost as a function of control input, and $\rho(\mathbf{x}_i, \Phi_i, 0) \geq 0$ enforces the satisfaction of the STL constraint by requiring the robustness degree to be nonnegative.

Solution of an aggregated optimization problem (Eq. (10)) that includes all agents in the environment enables to find optimal trajectories (if any) satisfying the STL specifications. While such a centralized approach can result in optimal agent trajectories, the time complexity exponentially grows as the number of agents increases. Hence, we propose an approximate solution to Eq. (10) by constructing individual optimization problems of each agent and solve a sequential planning problem in a distributed fashion.

IV. Solution Approach

A. MILP Encoding

The STL specification is enforced in Eq. (10) by the inequality constraint on the robustness degree metric, which is computed via the recursive definitions of min and max functions as described in Eq. (3). Since the robustness degree constraint includes piece-wise continuous and non-convex functions, it brings additional complexity to the optimization problem in Eq. (10). An alternative way to represent the constraint of satisfying an STL specification is encoding it as a set of constraints with binary variables in the form of $z^\phi \in \{0, 1\}$. While $z^\phi = 1$ implies the satisfaction, $z^\phi = 0$ denotes the failure of the specification ϕ. The big-M approach described in [17] and [31] can be used to generate such constraints. For each predicate in the STL specification, there exists a couple of big-M constraints containing the binary variables. For a predicate μ in the form of inequality, the big-M constraints are defined as

$$\mu = f(\mathbf{x}) \geq w \quad \begin{cases} f(\mathbf{x}) - w \geq M(z^\mu - 1), \\ f(\mathbf{x}) - w \leq M z^\mu. \end{cases} \tag{11}$$

where $M \in \mathbb{R}^+$ is a sufficiently large number. An STL specification depends on the predicates inside it. Therefore, the binary constraints corresponding to a general STL specification can be built into each other starting from the predicates.

The remaining connections of Boolean operators with other temporal operators and predicates can then be constructed with the following rules and encoded as integer constraints for STL specifications ϕ and φ [17].

Negation: $\phi = \neg\varphi$

$$z^{\phi}(t) = 1 - z^{\varphi}(t). \tag{12}$$

Conjunction: $\phi = \bigwedge_{i=1}^{m} \varphi_i$

$$\begin{aligned} &z^{\phi}(t) \leq z^{\varphi_i}(t), \quad i = 1, \ldots, m, \\ &z^{\phi}(t) \geq 1 - m + \sum_{i=1}^{m} z^{\varphi_i}(t). \end{aligned} \tag{13}$$

Disjunction: $\phi = \bigvee_{i=1}^{m} \varphi_i$

$$\begin{aligned} &z^{\phi}(t) \geq z^{\varphi_i}(t), \quad i = 1, \ldots, m, \\ &z^{\phi}(t) \leq \sum_{i=1}^{m} z^{\varphi_i}(t). \end{aligned} \tag{14}$$

Globally: $\phi = G_{[a,b]}\varphi$

$$z^{\phi}(t) = \bigwedge_{\tau=t+a}^{t+b} z^{\varphi}(\tau). \tag{15}$$

Finally: $\phi = F_{[a,b]}\varphi$

$$z^{\phi}(t) = \bigvee_{\tau=t+a}^{t+b} z^{\varphi}(\tau). \tag{16}$$

The satisfaction of a nested STL specification can then be determined by a unique z value which is constructed by the combinations of Eqs. (12), (13), (14), (15), and (16). For example, consider the previous samples of specifications $\phi_1 = G_{[0,\tau]}\big(Region\ A\big)$ and $\phi_2 = F_{[0,T]}G_{[0,\tau]}\big(Region\ A\big)$. We can construct $z^{\phi_2}(0) = \bigvee_{t=0}^{T} z^{\phi_1}(t)$ where $z^{\phi_1}(t) = \bigwedge_{t'=t+0}^{t+\tau} z^{A}(t')$ with $z^{A}(t') = 1$ when the agent is inside the Region A at time t'. Similarly, the satisfaction of the respective agent specifications Φ_i can also be denoted with a binary variable $z^{\Phi_i}(t)$. Then the satisfaction of the STL specification which is previously implied by $\rho(\mathbf{x}_i, \Phi_i, 0) \geq 0$, now reduces to $z^{\Phi_i}(0) = 1$.

B. Relaxation of Individual Optimization Problems

While the number of the uncoupled predicates such as service and static obstacle avoidance predicates in Eqs. (5) and (8c) is designated by $p_{u/coup.}$, the number of other predicates that are coupled with other agents' states as in the collision avoidance and coherence cases in Eqs. (8a) and (8b) is represented by $p_{coup.}$. Then the total number of predicates is found as $p_{total} = p_{u/coup.} + p_{coup.}$. We designate the predicate indices with $k \in \{1, \ldots, p_{total}\}$

Each predicate μ_k can be relaxed by modifying Eq. (2) as $f(\mathbf{x}) + \boldsymbol{\delta_k} \geq w_k$ to avoid any infeasibility in the solution with an amount of a distinct and time dependent relaxation variable $\boldsymbol{\delta_k} = [\delta_k(0)\ \delta_k(1) \cdots \delta_k(H)]^T \in \mathbb{R}^{H+1}$. Here $\delta_k(t) \in \mathbb{R}$ denotes the relaxation variable of the predicate (i.e., inequality) k at time t. All predicates used inside the STL specification are relaxed in a similar fashion. Such relaxation values are then penalized in addition to the cost of the optimization problem (Eq. (10)) with a great amount of penalty weights relative to the original cost. An additional constraint of $\delta_k(t) \geq 0,\ \forall\{k, t\}$ is added so that with appropriate penalty weights, the argument of the Eq. (17) can yield $\forall\delta_k(t) = 0$ if possible without being too conservative, and small $\delta_k(t)$'s if not. The main reason for defining relaxation predicates time dependent is to avoid unnecessary relaxations over the time steps other than the particular time step in which the relaxation is required. Consider the collision avoidance requirement, suppose that the same relaxation that is required only at a single time step is applied all over the trajectory, i.e., agents have to be dangerously close at a single time step but they are now flexible to do so during the whole mission. Then we could not obtain trajectories as separated as possible but the ones that are already relaxed at all time steps and may be in conflict all the time. The maximal separation of trajectories via time dependent relaxations are desired to be able to find feasible trajectories, which are assumed to exist throughout the paper, by the sequential detect and repair scheme that will be discussed in detail in Sec. IV.C.

We define first the uncoupled predicates and then the coupled ones such that $\boldsymbol{\delta_k}$ with the indices $k \in \{1, \ldots, p_{u/coup.}\}$ are uncoupled, and $\boldsymbol{\delta_k}$ for $k \in \{p_{u/coup.} + 1, \ldots, p_{total}\}$ are defined as the coupled predicates. In this work, we also consider minimizing agent movement over the environment while satisfying the desired STL specifications. To this end, we define the running cost for any agent i as the l_1-norm of the control effort, $\mathcal{J}_u\big(u_i(t)\big) = \big\|u_i(t)\big\|$. The optimization problem previously presented in Eq. (10) with the nonnegative robustness degree constraint can now be reformulated for each agent with the integer constraints and the cost penalizing agent movement and the relaxation of predicates as follows:

$$
\begin{aligned}
\mathbf{u}^* = \arg\min \ & \left(\alpha_1 \, \boldsymbol{\gamma}_{u/coup.} + \alpha_2 \, \boldsymbol{\gamma}_{coup.} + \sum_{t=0}^{H-1} \|u(t)\|\right) \\
s.t. \quad & \mathrm{x}(t+1) = A\,\mathrm{x}(t) + B\,u(t), \ u_{min} \le u(t) \le u_{max} \ \forall t, \\
& z^{\Phi}(0) = 1, \ \forall \delta_k(t) \ge 0 \, for \, k \in \{1, \ldots, p_{total}\}.
\end{aligned}
\tag{17}
$$

where $\boldsymbol{\gamma}_{u/coup.} = \sum_{k=1}^{p_{u/coup.}} \sum_{t=0}^{H} \delta_k(t) \in \mathbb{R}$ represents the summation of relaxation variables of the uncoupled predicates. Also $\boldsymbol{\gamma}_{coup.} = \sum_{k=p_{u/coup.}+1}^{p_{total}} \sum_{t=0}^{H} \delta_k(t) \in \mathbb{R}$ denotes the summation of the relaxation variables of the coupled predicates that depend upon the states of the other agents. The overall objective function to be minimized in Eq. (17) can be represented as $\mathcal{J}\left(\mathbf{u}, \boldsymbol{\delta_1}, \ldots, \boldsymbol{\delta_{p_{total}}}\right)$.

In sequential trajectory planning, which is discussed in detail throughout Sec. IV.C, agents are able to adjust their trajectories to resolve any conflict with other agents. However, if an agent diverges from its respective goal regions or passes through a static obstacle due to any conflict with other agents during an iteration, remaining agents would act according to this unrealistic trajectory of that agent and not necessarily make any correction in their trajectories. Therefore, we penalize $\boldsymbol{\gamma}_{u/coup.}$ and $\boldsymbol{\gamma}_{coup.}$ oddly as $\alpha_1 \gg \alpha_2 \gg \sum_{t=0}^{H-1} \|u_{max}\| \in \mathbb{R}$ for agents to succeed in their tasks with static regions primarily including the service and obstacle areas, and then adjust those trajectories within the next iterations if they have conflicts with the other agents. The resulting constrained optimization problem in Eq. (17) takes the form of a standard Mixed-Integer Linear Program. There are several off-the-shelf tools such as MATLAB *intlinprog* and Gurobi [32] to handle such problems.

C. Sequential Trajectory Planning

The proposed approach for multi-agent trajectory planning consists of two parts: 1) Each agent's trajectory is found first without considering any interaction with other agents, i.e., the aim is just to service the goal region(s) and to avoid static obstacle(s). 2) Agents start to solve for their respective trajectories sequentially from agent 1 to n with the currently available information of all the other agent trajectories. Whenever an agent i determines its trajectory with respect to other agents, it shares its updated trajectory, and the remaining agents $i+1$ to n consider the updated trajectory of that agent in their calculations to avoid multiple resolution efforts for the same pairwise conflicts. In detail, from agent i's perspective, Alg. 1 is initialized with the inputs of initial state information and the respective STL specification. All agents generate their trajectories without considering other agents in line 1. In other words, they initially find a trajectory that just visits the desired goal regions within given time intervals while avoiding static obstacles. If the agent has no uncoupled specification, then the initial trajectory of it is fixed to the initial condition (line 2). The initial trajectories of the agents are shared among them in lines 3 and 4 to constitute an initial guess for all. Between lines 5 and 13, an iterative method is implemented which runs until all agents are able to compute their respective trajectories without any relaxation in their STL specifications. First, the iteration number h is defined, and available current trajectories of other agents are collected in lines 6 and 7. In line 8, the agent solves for its trajectory achieving uncoupled specifications by minimally violating the coupled ones using the current information received from other agents. An indicator $Flag_i$ is used to communicate that the agent successfully computes its trajectory without relaxing its specification which is determined and broadcast in lines 9-11.

With a slight abuse of notation, we denoted the set of relaxation variables for the predicates inside the Φ_i as $\boldsymbol{\delta^i} = \bigcup_{\boldsymbol{k}} \boldsymbol{\delta_k}$ in line 9 as $k \in \{1, \ldots, p_{total}\}$. Note that the total predicate number, p_{total}, may be different for each agent. We also use $\boldsymbol{\delta^{ij}} = \bigcup_{\boldsymbol{k_j}} \boldsymbol{\delta_{k_j}}$ to represent relaxation variables of the predicates inside Φ_i coupled with the state of agent j, where k_j denotes the index of such predicates, i.e., $\boldsymbol{\delta^{ij}} \subseteq \boldsymbol{\delta^i}$. To avoid oscillations and possible deadlocks that may occur when an agent has limited waypoint options, we adapt an approach that makes this agent more expensive for other agents to be in conflict with. We first define a conflict variable via an element-wise maximization such that $\beta_{ij} = \max \boldsymbol{\delta^{ij}} \in \mathbb{R}$, which formally denotes the worst (largest) relaxation at any time t that is needed for agent i to make during a conflict with agent j. Whenever there are two agents i and j that relax their bilaterally coupled specifications, i.e., $\beta_{ij}, \beta_{ji} > 0$, this implies that one of the agents has no other option, and the other agent's remaining options contain conflicts with other agents. However, under the assumption of the existence of at least one trajectory for each agent to travel that satisfies the local specification without needing a relaxation, we know that one of those two agents has an alternative trajectory to take which may possibly be in conflict with other agent that have more flexibility. Consequently, a chain of sequential adjustments of the agent trajectories results in finding feasible trajectories for all agents in the environment. Accordingly, in line 12, similar to [21], we make it expensive for such two agents to remain in conflict with each other by penalizing the relaxations of the predicates coupled with each other more in the objective function. We define a

higher penalty weight for the associated bilateral relaxation among the agents i and j such that $\bar{\alpha} = (\alpha_1 + \alpha_2)/2$. This penalty setting implies that any agent other than j becomes less expensive to be in conflict with while the primary aim of servicing the goal regions remains the same. The additional penalty term $\bar{\alpha}\boldsymbol{\gamma}^{ij}$ is then added to the nominal objective function $\mathcal{J}\left(\mathbf{u}, \boldsymbol{\delta_1}, \ldots, \boldsymbol{\delta_{p_{total}}}\right)$ where $\boldsymbol{\gamma}^{ij} = \sum_{k_j} \sum_{t=0}^{H} \delta_{k_j}(t) \in \mathbb{R}$. It is also worth to mention that the same penalty term is applied to agent j's cost as well during its respective iteration to penalize the conflict with the agent i more in the same way. Moreover, the objective function accumulates over the iterations such that once the conflicts between two agents are penalized, the high penalty terms is inherited from the previous iterations over the next iterations.

Whenever an agent determines its trajectory with respect to other agents, even though it has some relaxations which imply a violation of the original specification, it still updates its plan, and the remaining agents ($i+1$ to n) consider the updated trajectory of that agent in case the conflict (i.e., the reason for relaxation) can be solved by one of these agents within the same iteration. If there is at least one agent with a relaxed specification, each agent recalculates its trajectory in the next iteration. If not, then all agents successfully satisfy their respective specifications.

Algorithm 1: Sequential Trajectory Planning for Agent i

input : Initial position x_{i_0} and STL specification Φ_i

if $\phi_{i,u/coup.} \neq \emptyset$ **then** Solve Eq. (17) to get $\mathbf{x}_i^0$ with $\Phi_i = \phi_{i,u/coup.}$ via Eq. (9);

else $\mathbf{x}_i^0 = [\mathrm{x}_{i_0} \ \cdots \ \mathrm{x}_{i_0}]$;

Broadcast the resultant trajectory, $\mathbf{x}_i^0$;

Initialize $h = 0$; $Flag_j^h =$ `False` for $j \in \{1, \ldots, n\}$;

while $\exists Flag_j^h =$ `False` *for* $j \in \{1, \ldots, n\}$ **do**

  $h = h + 1$;

  Obtain current trajectories $\mathbf{x}_j^h$ from agents $j \in \{1, \ldots, i-1\}$, and $\mathbf{x}_j^{h-1}$ from agents $j \in \{i+1, \ldots, n\}$;

  Solve Eq. (17) including relative distance constraints to compute $\mathbf{x}_i$ and broadcast it;

  if $\nexists \delta \in \boldsymbol{\delta^i} > 0$ **then** Set $Flag_i^h =$ `True` ;

  else Set $Flag_i^h =$ `False`;

  Broadcast $Flag_i^h$ and $\boldsymbol{\delta^{ij}}$, obtain $Flag_j^h$ and $\boldsymbol{\delta^{ji}}$, $\forall j \neq i \in \{1, \ldots, n\}$;

  if $\exists \{\beta_{ij}, \beta_{ji}\} > 0$ **then** $\mathcal{J}_i\left(\mathbf{u}, \boldsymbol{\delta_1}, \ldots, \boldsymbol{\delta_{p_{total}}}\right) = \mathcal{J}_i\left(\mathbf{u}, \boldsymbol{\delta_1}, \ldots, \boldsymbol{\delta_{p_{total}}}\right) + \bar{\alpha}\boldsymbol{\gamma}^{ij}$;

end

return : State trajectory $\mathbf{x}_i$

V. Case Studies

We consider quadrotor agents for missions that require the inspection and monitoring of an indoor environment that is risky for human staff. For simplicity, 2D planar motion is considered with $\mathrm{x} = [x \ y]^T \in \mathbb{R}^2$ position states, and input $u = [u_x \ u_y]^T \in \mathbb{R}^2$. Maximum available input, i.e., speed, is restricted as $u_{x,y_{max}} = 0.2 \ m/s$ in all directions with $u_{x,y_{min}} = -u_{x,y_{max}}$. In order to guarantee the collision avoidance between discrete time steps, the limit on the velocities and collision distance r_{min} must be chosen with $u_{max}\Delta t < r_{min}$. The optimization problem of each agent in Eq. (17) is modeled with YALMIP [33] and solved in MATLAB R2019b with the Gurobi as the underlying solver [32]. A laptop computer with 1.8 GHz, Intel Core i5 processor is used to run the simulations. Resultant high level trajectories with single integrator dynamics, which satisfy STL specifications, are tracked by a low level Model Predictive Control (MPC) scheme with 6-DOF quadrotor dynamics linearized around hover.

All regions of interest such as servicing areas and static obstacles are defined as quadrangular shapes, mostly squares (boxes). Besides, collision avoidance and coherence circles are again approximated by outer and inner fit square boxes, respectively as shown in Fig. 2. In other words, more conservative constraints are applied regarding the relative motion among the quadrotors. The other agents are required to fly inside the shaded areas denoted as *feasible regions* in Fig. 2 which are centered on each agent. To illustrate the fundamentals of the algorithm, we first present a sample scenario with two agents. We define distinct STL specifications (Eq. (18)) for the agents with particular time bounds, obstacle, base, and goal placements such that they are able to satisfy the specifications just in time without flexibility.

$$\begin{aligned} \Phi_1 &= F_{[0,16]} G_{[0,4]} \left(Region\ 1\right) \wedge G_{[0,20]} \left(Obs_1 \wedge Obs_2\right) \wedge G_{[0,20]} \left(1.2 \geq \|\mathrm{x}_1 - \mathrm{x}_2\| \geq 0.2\right), \\ \Phi_2 &= F_{[0,16]} G_{[0,4]} \left(Region\ 2\right) \wedge G_{[0,20]} \left(Obs_1 \wedge Obs_2\right) \wedge G_{[0,20]} \left(1.2 \geq \|\mathrm{x}_2 - \mathrm{x}_1\| \geq 0.2\right). \end{aligned} \tag{18}$$

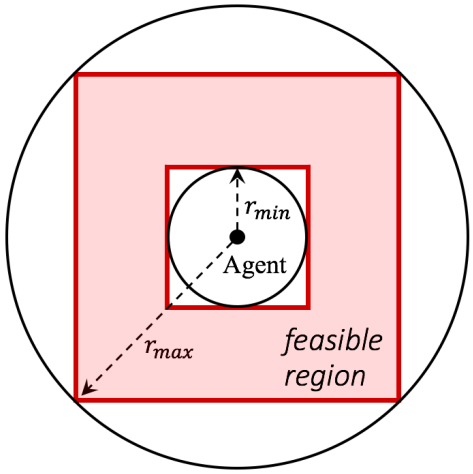


Fig. 2 Minimum and maximum distance bounds are approximated as inner and outer square boxes, respectively. While the inner fit box is related to the collision avoidance with a half side length of $a_{min} = r_{min}$, the outer fit box constitutes a coherence region with a half side length of $a_{max} = r_{max}/\sqrt{2}$ where both approximations yield safer limitations than required.

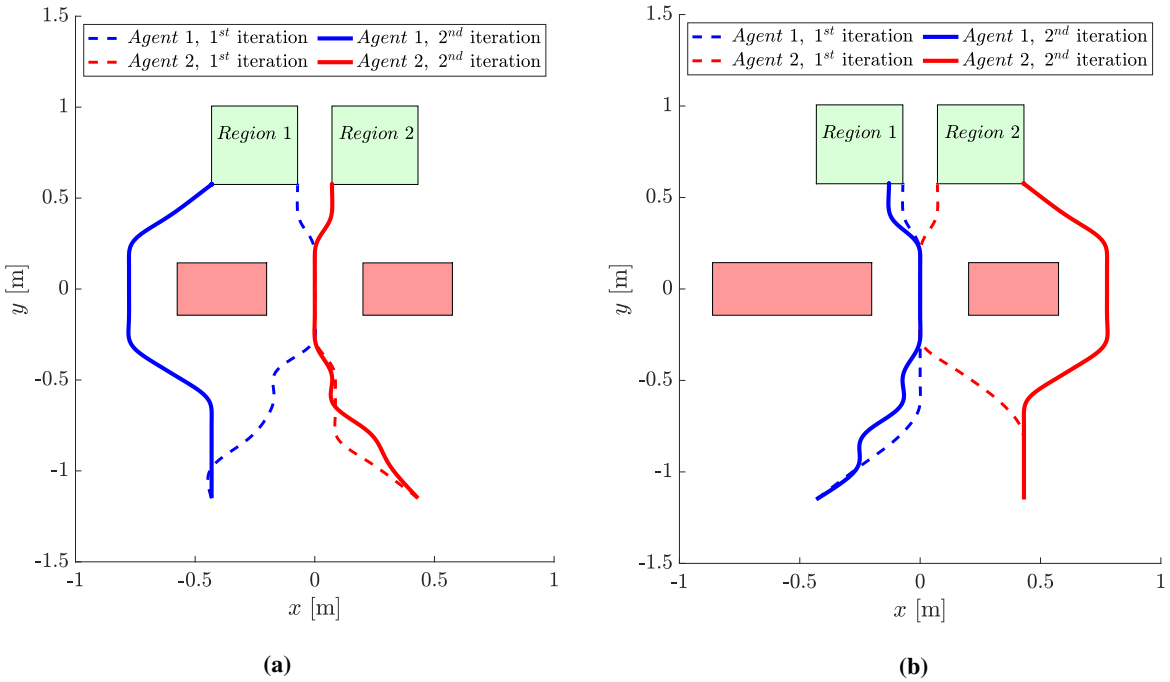


Fig. 3 Agents with tightly timed specifications that they can barely satisfy. First iterations, designated by dashed lines, include no coupling. (a) Agent 1 avoid the other agent in response to its available trajectory from the first iteration, i.e., the red dashed trajectory. (b) Agent 1 has no other option than being in conflict with the other agent (since collision with a static obstacle is more expensive). Consequently, Agent 2 avoids the Agent 1.

In English, agents need to reach and stay in the corresponding regions for four consecutive seconds within the mission horizon of twenty seconds without colliding with each other or separating much. We present the trajectories generated by Algorithm 1 in each iteration (two iterations in total) for each agent in Fig. 3. The static obstacles (in red) and goal regions (in green) are placed such that passing between the obstacles is slightly cheaper than going around. This is the reason for agents to pass through the middle in their first iterations in which they do not take into account

the coupled specifications. On the second iterations, Agent 1 reacts to the trajectory of the other agent by taking the alternative path (Fig. 3a). Whenever the Agent 1 is constrained more by extending the obstacle in front of it and making the only feasible way to achieve the mission to pass between the obstacles, this time Agent 2 needs to update its trajectory (Fig. 3b). Note that a possible scenario in which both of the agents must pass between the obstacles at the same time is infeasible and kept out of the scope of this paper.

As the main case study, we now consider three quadrotors with a redundancy in the mission such that all agents are given the same mission specifications. This would also be useful when multiple data sets are desired to be collected for the sake of correct and accurate information. Three regions of interest are specified to be visited for the inspection of the environment. STL specifications considered in the simulations comprise several parts shaped by mission tasks such as

1) **Reach and Stay:** Quadrotors have to reach three specific regions during the mission, and upon arrival, they need to stay there for at least 2 consecutive seconds to collect data. For reaching or staying in a goal box, it is required to satisfy all four linear predicates representing the box edges. That is, predicates are connected by *conjunction* operators as in Eq. (4). Goal regions (shown with green in Figs. 5a, 6a, and 7a) to be serviced namely A, B, and C are defined by the corner positions of $\big([-0.575, 1.725] \times [-1.150, 1.725] \times [-1.150, 1.150] \times [-0.575, 1.150]\big)$, $\big([-1.438, 0.288] \times [-2.000, 0.288] \times [-2.000, -0.288] \times [-1.438, -0.288]\big)$, and $\big([-0.575, -1.150] \times [-1.150, -1.150] \times [-0.288.150, -1.725] \times [-0.575, -1.725]\big)$, respectively.

2) **Avoid Static Obstacles:** To avoid static obstacles that are again represented as quadrangular regions, it is enough to stay out of only one of its edges. Therefore, linear predicates representing these box edges are connected by *disjunction* operators as in Eq. (7). Two static obstacle regions Obs_1 and Obs_2 (shown with red in the Figs. 5a, 6a, and 7a) are defined with the corner positions of $\big([0, 1.150] \times [-0.288, 1.150] \times [-0.288, 0.216] \times [0, 0.216]\big)$ and $\big([0, -0.216] \times [-0.288, -0.216] \times [-0.288, -1.150] \times [0, -1.150]\big)$, respectively.

3) **Collision avoidance:** The distance between two agents in each particular direction must be more than a predefined distance, $r_{min} = 0.2\ m$. An outer fit box to the circle with radius r_{min} denotes a no-fly zone attached to each agent (Fig. 2). As in the static obstacle case, to avoid a collision with an agent, it is enough to stay out of only one of its edges. Therefore, linear predicates representing these box edges are connected by *disjunction* operators.

4) **Coherence:** Agents are supposed to stay each other close during the mission. A parameter $r_{max} = 1.20\ m$ is defined such that the distance between the agents is always less than r_{max}. An inner fit box to the circle with radius r_{max} designates a stay-in zone attached to each agent (Fig. 2). As in the goal region case, to get or stay inside this box, it is required to satisfy all four linear predicates representing box edges. That is, predicates are connected by *conjunction* operators.

For simplicity and redundancy, the same general STL specification for all three agents is applied to represent all previously mentioned mission requirements as

$$\begin{aligned}\Phi_{i=\{1,2,3\}} = {} & F_{[20,38]} G_{[0,2]} \big(Region\ A\big) \wedge F_{[20,38]} G_{[0,2]} \big(Region\ B\big) \wedge F_{[20,38]} G_{[0,2]} \big(Region\ C\big) \\ & \wedge G_{[0,40]} \left(\neg Obs_1 \wedge \neg Obs_2\right) \\ & \wedge G_{[0,40]} \Big(\bigwedge_{j=1 \neq i}^{3} \|\mathrm{x_i} - \mathrm{x_j}\| \geq 0.2\Big) \\ & \wedge G_{[0,40]} \Big(\bigwedge_{j=1 \neq i}^{3} \|\mathrm{x_i} - \mathrm{x_j}\| \leq 1.2\Big),\end{aligned} \tag{19}$$

which implies "within the second half of the mission with a horizon of 40 seconds, eventually visit each regions A, B, and C for 2 consecutive seconds, never collide with the static obstacles, always keep a minimum distance with other agents about $0.2\ m$, and always keep a maximum distance with other agents about $1.20\ m$."

The agents take-off from a base region represented with black in Figs. 5a, 6a, and 7a. Simulations are implemented for three different cases in terms of couplings. First, no relative agent motion is considered, i.e., $\Phi_i = \phi_{service} \wedge \phi_{obstacle}$ for all i, and results are revealed in Fig. 5. Although the agents are able to achieve the given servicing requirements and avoid the static obstacles, since they do not care about each other, 1) the minimum relative distance bound is crossed; therefore, collisions take place, and 2) agents spread out by violating the maximum distance bound. Pairwise Euclidean distances and the violated relative distance constraints can be seen in Fig. 5b. Second, collision avoidance is added

to the STL specification as $\Phi_i = \phi_{service} \wedge \phi_{obstacle} \wedge \phi_{coll.avoid.}$. Resultant trajectories (Fig. 6a) show that while servicing goal regions, agents also avoid each other (lower relative distance bound is never crossed in Fig. 6b) as well as the static obstacles. However, agents still spread out over the environment by violating the maximum relative distance constraints. Finally, in the third case, all specifications are applied including the coherence as in Eq. (19). This time, in addition to reach, stay, and avoid tasks, agents can be seen to perform together in a coordination as shown in Fig. 7a without violating any relative distance constraint (Fig. 7b). Results of the trials including the optimized cost and solver time are presented in Table 1 together with the results of the same problem (for full coupling case with the collision avoidance and coherence constraints) solved in a centralized manner. Although centralized solution have optimality guarantee, it is not scalable with the high number of agents and complex (e.g., longer) mission specifications, especially for the specifications with couplings as in this study. Accordingly, although the centralized solution results in less cost compared to our approach, the solution process of it takes much longer.

Table 1 Comparison of different coupling settings of the proposed approach with the centralized solution.

Scenario/Solution	No Coupling	Only Coll. Avoid. Coupling	Full Coupling	Centralized
Total Cost	71.8	73.9	87.9	74.8
Total Solution Time $[s]$	9.5	26.8	59.4	341.83

To see the response of the algorithm with different initial conditions, the same scenario with full coupling is run by switching the agent initial conditions. While the mean of solver time is found to be 66.9 s, the average resultant cost was 79.9. Furthermore, these trials revealed some cases in which relaxation-free trajectories can be found with a higher number of iterations while the total run-time of the algorithms is still shorter compared to the cases with less number of iterations. As a result, the number of iterations does not appear to be a correct indicator of the time complexity.

The trajectories generated by the proposed method are also verified experimentally on a team of three Crazyflie 2.0 nano quadrotor platforms. The experiment is conducted in a testing environment of $4.025m \times 3.450m$ motion-capture space identical to the simulation environment with full coupling mission specifications as shown in Fig. 7, using a `VICON` camera system with 8 cameras. Robot Operating System (ROS) [34] is used to conduct the experiment. An instance from the experiment is shown in Fig. 4, and the video of the experiment can be found at `https://youtu.be/FJSupiqWZAo`.

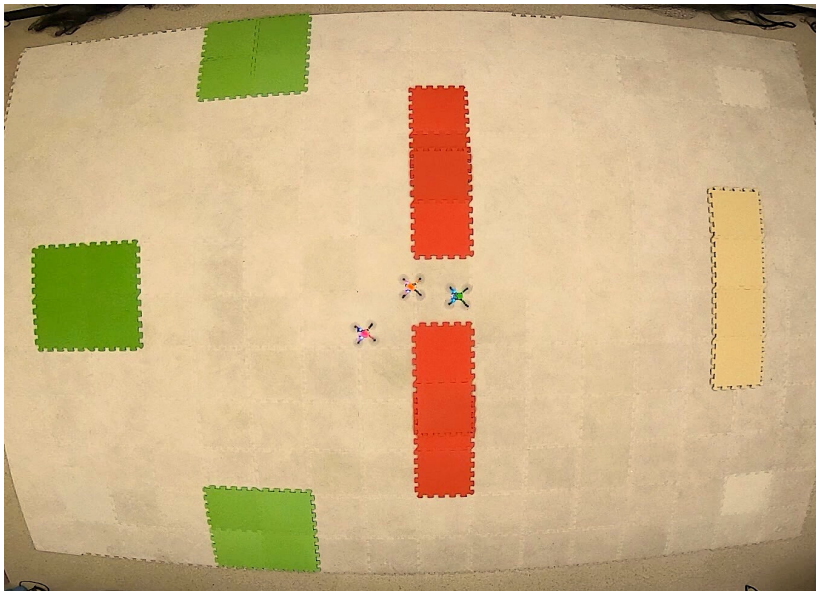
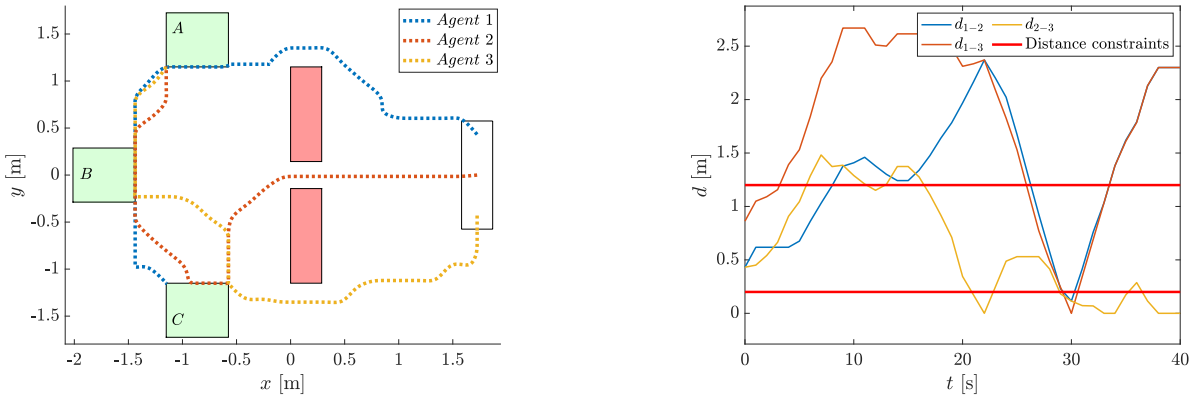
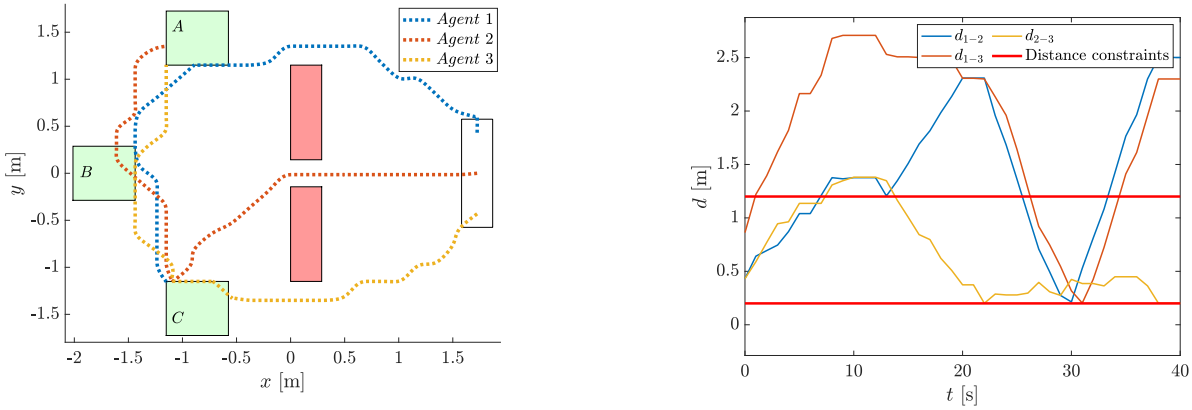


Fig. 4 Agents move in a shared experiment environment divided into two parts by static obstacles (no-fly zones in red) that allow agents only to pass through any of the doors at the top, middle, or bottom. Due to coherence requirement, all agents use the door in the middle to reach the part including the regions of interest, i.e., green zones.



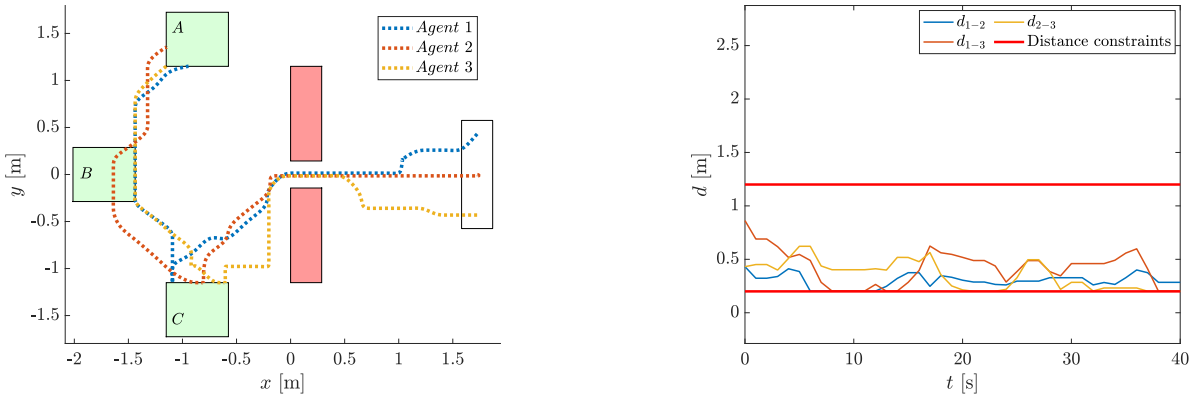
(a) (b)

Fig. 5 (a) Trajectories of the agents that consider servicing tasks and static obstacles only, (b) Pairwise relative distances between agents, note that this case does not include any specification with relative distance constraints, i.e., both lower and upper bounds may be violated.



(a) (b)

Fig. 6 (a) Trajectories of the agents that consider servicing tasks, static obstacles, and avoidance of other agents, (b) Pairwise relative distances between agents, note that this case has no specifications regarding the coherence, i.e., upper bound in relative distance may be violated.



(a) (b)

Fig. 7 (a) Trajectories of the agents that consider servicing tasks, static obstacles along with avoidance of and coherence with other agents, (b) Pairwise relative distances between agents where both bounds are enforced.

VI. Conclusions and Future Work

In this study, we propose a distributed algorithm to solve multi-agent trajectories that satisfy complex spatio-temporal specifications. The specifications constraining agent movement relative to each to other, i.e., including coupled tasks, are defined as Signal Temporal Logic (STL) constraints in addition to servicing and static obstacle avoidance requirements. The sequential trajectory planning algorithm we propose (Sec. IV.C) runs much faster compared to the centralized solution by compromising some control effort in return. We deduce that under the assumption of the existence of feasible trajectories that satisfy the temporal logic specifications, our algorithm are able provide those by allocating computation burden to agents. To avoid the computational expense, we use single integrator dynamics in high level planning, and low level control is implemented considering 6-DOF quadrotor dynamics via MPC method. We present simulation and experiment results which verify the trajectories generated by the proposed method. As a future direction, we will investigate relaxation of infeasible specifications. Note that although we allow relaxation of STL specifications in the intermediate iterations, Alg. 1 currently runs until the specification is satisfied assuming that it is feasible by nature. In the case of an infeasible STL specification, we are currently able to diagnose which part of the specification yields a conflict via relaxations made for particular predicates. This information can be used in minimal relaxations of the specifications both spatially and temporally. A human operator can also be placed in the loop when a decision needed what to compromise in case of the violation of the original specification.

VII. Acknowledgements

This work was partially supported by Honeywell Aerospace and MnDRIVE, University of Minnesota.

References

[1] Tanner, H., Pappas, G., and Kumar, V., "Leader-to-formation stability," *IEEE Transactions on robotics and automation*, Vol. 20, No. 3, 2004, pp. 443–455. doi:10.1109/TRA.2004.825275.

[2] Ren, W., and Beard, R., "Consensus seeking in multiagent systems under dynamically changing interaction topologies," *IEEE Transactions on automatic control*, Vol. 50, No. 5, 2005, pp. 655–661. doi:10.1109/TAC.2005.846556.

[3] Bhat, R., Yazıcıoğlu, Y., and Aksaray, D., "Distributed Path Planning for Executing Cooperative Tasks with Time Windows," *IFAC-PapersOnLine*, Vol. 52, No. 20, 2019, pp. 187–192. doi:10.1016/j.ifacol.2019.12.156.

[4] Yazıcıoğlu, A., Egerstedt, M., and Shamma, J., "Communication-free distributed coverage for networked systems," *IEEE Transactions on Control of Network Systems*, Vol. 4, No. 3, 2016, pp. 499–510. doi:10.1109/TCNS.2016.2518083.

[5] Seyedi, S., Yazıcıoğlu, Y., and Aksaray, D., "Persistent Surveillance With Energy-Constrained UAVs and Mobile Charging Stations," *IFAC-PapersOnLine*, Vol. 52, No. 20, 2019, pp. 193–198.

[6] Baier, C., and Katoen, J., *Principles of model checking*, MIT press, 2008. doi:10.1093/comjnl/bxp025.

[7] Lindemann, L., Nowak, J., Schönbächler, L., Guo, M., Tumova, J., and Dimarogonas, D., "Coupled Multi-Robot Systems Under Linear Temporal Logic and Signal Temporal Logic Tasks," *IEEE Transactions on Control Systems Technology*, 2019. doi:10.1109/TCST.2019.2955628.

[8] Chen, Y., Ding, X., and Belta, C., "Synthesis of distributed control and communication schemes from global LTL specifications," *2011 50th IEEE Conference on Decision and Control and European Control Conference*, IEEE, 2011, pp. 2718–2723. doi:10.1109/CDC.2011.6160740.

[9] Aksaray, D., Leahy, K., and Belta, C., "Distributed multi-agent persistent surveillance under temporal logic constraints," *IFAC-PapersOnLine*, Vol. 48, No. 22, 2015, pp. 174–179. doi:10.1016/j.ifacol.2015.10.326.

[10] Guo, M., Tumova, J., and Dimarogonas, D., "Communication-free multi-agent control under local temporal tasks and relative-distance constraints," *IEEE Transactions on Automatic Control*, Vol. 61, No. 12, 2016, pp. 3948–3962. doi:10.1109/TAC.2016.2527731.

[11] Fang, J., Zhang, Z., and Cowlagi, R. V., "Decentralized Route-Planning to Satisfy Global Linear Temporal Logic Specifications on Multiple Aircraft," *2018 AIAA Guidance, Navigation, and Control Conference*, 2018, p. 1862. doi:10.2514/6.2018-1862.

[12] Karaman, S., and Frazzoli, E., "Vehicle routing with linear temporal logic specifications: Applications to Multi-UAV Mission Planning," *AIAA guidance, navigation and control conference and exhibit*, 2008, p. 6838. doi:10.2514/6.2008-6838.

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

[14] Koymans, R., "Specifying real-time properties with metric temporal logic," *Real-time systems*, Vol. 2, No. 4, 1990, pp. 255–299.

[15] Maler, O., and Nickovic, D., "Monitoring temporal properties of continuous signals," *Formal Techniques, Modelling and Analysis of Timed and Fault-Tolerant Systems*, Springer, 2004, pp. 152–166. doi:10.1007/978-3-540-30206-3_12.

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

[17] Raman, V., Donzé, A., Maasoumy, M., Murray, R., Sangiovanni-Vincentelli, A., and Seshia, S., "Model predictive control with signal temporal logic specifications," *53rd IEEE Conference on Decision and Control*, IEEE, 2014, pp. 81–87. doi:10.1109/CDC.2014.7039363.

[18] Aksaray, D., Yazıcıoğlu, Y., Feron, E., and Mavris, D. N., "Message-passing strategy for decentralized connectivity maintenance in multiagent surveillance," *Journal of Guidance, Control, and Dynamics*, Vol. 39, No. 3, 2016, pp. 542–555.

[19] Marchidan, A., and Bakolas, E., "Collision Avoidance for an Unmanned Aerial Vehicle in the Presence of Static and Moving Obstacles," *Journal of Guidance, Control, and Dynamics*, Vol. 43, No. 1, 2020, pp. 96–110. doi:10.2514/1.G004446.

[20] Ames, A. D., Coogan, S., Egerstedt, M., Notomista, G., Sreenath, K., and Tabuada, P., "Control barrier functions: Theory and applications," *2019 18th European Control Conference (ECC)*, IEEE, 2019, pp. 3420–3431. doi:10.23919/ECC.2019.8796030.

[21] Schulman, J., Duan, Y., Ho, J., Lee, A., Awwal, I., Bradlow, H., Pan, J., Patil, S., Goldberg, K., and Abbeel, P., "Motion planning with sequential convex optimization and convex collision checking," *The International Journal of Robotics Research*, Vol. 33, No. 9, 2014, pp. 1251–1270. doi:10.1177/0278364914528132.

[22] Lindemann, L., Verginis, C. K., and Dimarogonas, D. V., "Prescribed performance control for signal temporal logic specifications," *2017 IEEE 56th Annual Conference on Decision and Control (CDC)*, IEEE, 2017, pp. 2997–3002. doi:10.1109/CDC.2017.8264095.

[23] Pant, Y. V., Abbas, H., Quaye, R. A., and Mangharam, R., "Fly-by-logic: control of multi-drone fleets with temporal logic objectives," *2018 ACM/IEEE 9th International Conference on Cyber-Physical Systems (ICCPS)*, IEEE, 2018, pp. 186–197. doi:10.1109/ICCPS.2018.00026.

[24] Luis, C. E., and Schoellig, A. P., "Trajectory generation for multiagent point-to-point transitions via distributed model predictive control," *IEEE Robotics and Automation Letters*, Vol. 4, No. 2, 2019, pp. 375–382. doi:10.1109/LRA.2018.2890572.

[25] Fainekos, G. E., "Revising temporal logic specifications for motion planning," *2011 IEEE International Conference on Robotics and Automation*, IEEE, 2011, pp. 40–45. doi:10.1109/ICRA.2011.5979895.

[26] Ahlberg, S., and Dimarogonas, D. V., "Human-in-the-loop control synthesis for multi-agent systems under hard and soft metric interval temporal logic specifications," *2019 IEEE 15th International Conference on Automation Science and Engineering (CASE)*, IEEE, 2019, pp. 788–793. doi:10.1109/COASE.2019.8842954.

[27] Ghosh, S., Sadigh, D., Nuzzo, P., Raman, V., Donzé, A., Sangiovanni-Vincentelli, A. L., Sastry, S. S., and Seshia, S. A., "Diagnosis and repair for synthesis from signal temporal logic specifications," *Proceedings of the 19th International Conference on Hybrid Systems: Computation and Control*, 2016, pp. 31–40.

[28] Donzé, A., and Maler, O., "Robust satisfaction of temporal logic over real-valued signals," *International Conference on Formal Modeling and Analysis of Timed Systems*, Springer, 2010, pp. 92–106. doi:10.1007/978-3-642-15297-9_9.

[29] Fainekos, G., and Pappas, G., "Robustness of temporal logic specifications for continuous-time signals," *Theoretical Computer Science*, Vol. 410, No. 42, 2009, pp. 4262–4291. doi:10.1016/j.tcs.2009.06.021.

[30] Dokhanchi, A., Hoxha, B., and Fainekos, G., "On-line monitoring for temporal logic robustness," *International Conference on Runtime Verification*, Springer, 2014, pp. 231–246. doi:10.1007/978-3-319-11164-3_19.

[31] Karaman, S., Sanfelice, R., and Frazzoli, E., "Optimal control of mixed logical dynamical systems with linear temporal logic specifications," *2008 47th IEEE Conference on Decision and Control*, IEEE, 2008, pp. 2117–2122. doi:10.1109/CDC.2008.4739370.

[32] Gurobi Optimization, L., "Gurobi Optimizer Reference Manual," , 2020.

[33] Löfberg, J., "YALMIP : A Toolbox for Modeling and Optimization in MATLAB," *In Proceedings of the CACSD Conference*, Taipei, Taiwan, 2004. doi:10.1109/CACSD.2004.1393890.

[34] Quigley, M., Conley, K., Gerkey, B., Faust, J., Foote, T., Leibs, J., Wheeler, R., and Ng, A., "ROS: an open-source Robot Operating System," *ICRA workshop on open source software*, Vol. 3, Kobe, Japan, 2009, p. 5.