

Generalizable Multi-Agent Planning from Signal Temporal Logic Specifications via Diffusion

Joe Eappen, Zikang Xiong, Shreyash S. Iyengar, and Suresh Jagannathan

Abstract—Multi-agent systems in the real-world (e.g., drone swarms, autonomous cars, warehouse robots) must satisfy rich, temporal tasks while avoiding collisions. Signal Temporal Logic (STL) elegantly encodes such objectives, but current STL planning methods face critical limitations. State-of-the-art optimization-based approaches can handle arbitrary STL specifications but struggle with scalability, becoming computationally impractical as the number of agents grows. Learning-based methods efficiently handle a large number of agents with rapid planning times but fare poorly when deployment-time objectives differ from those used during training, and do not support planning tasks that require different specifications to be ascribed to different agents (i.e., heterogeneity) or team-level specifications requiring coordination of multiple agents. This fundamental trade-off between *generalizability* and *scalability* presents a challenge for realizing multi-agent STL planning algorithms in practice. To overcome this challenge, we introduce a new diffusion method for multi-agent planning with STL specifications. Using a differentiable approximation of STL, we integrate the STL gradient in the denoising process, making our approach generalizable to novel formulas whose predicates are placed anywhere within the goal region covered during training, while achieving the same scalability as existing learning-based methods. Our method supports heterogeneous specifications, and by using diffusion models, naturally enhances plan *diversity*, thereby significantly reducing safety-related violations (e.g., collisions) among agents. A detailed evaluation study justifies the utility of STL-guided diffusion-based multi-agent planners for constructing generalizable, scalable, and diverse plans. Videos and code are available at <https://www.jeappen.com/diff-ma-stl/> and <https://github.com/jeappen/diff-ma-stl>.

Index Terms—Multi-Robot Systems, Hybrid Logical/Dynamical Planning and Verification, Generative Models

I. INTRODUCTION

Symbolic specification methods have recently experienced a resurgence through neuro-symbolic algorithms [1], which aim to combine the generalizability of neural methods with the interpretability and modifiability of symbolic systems by human users. Symbolic techniques have also emerged in robot motion planning, particularly with the use of Signal Temporal Logic (STL) to specify objectives for multi-robot systems, which can then be addressed by Mixed-Integer Linear Programming (MILP) solvers [2], graph-based algorithms [3], or sampling-based methods [4]. STL specifies complex temporal properties over continuous signals, making it suitable for complex

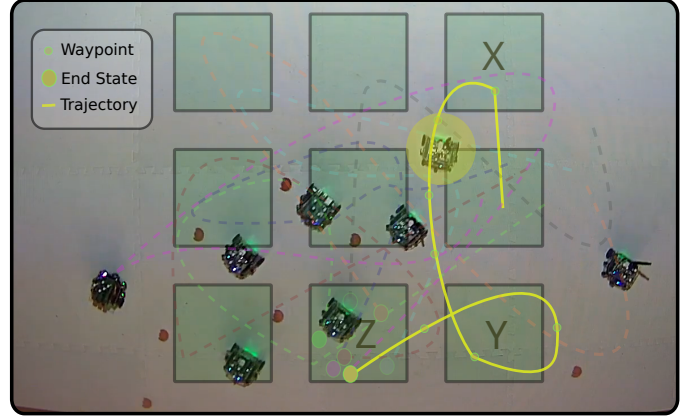


Fig. 1: Demo on $N = 10$ differential drive robots [10] following a (heterogeneous) *seq* specification (Sec. V-B) that requires agents to visit designated locations in a prescribed order (X, Y, Z in the figure for the yellow robot) at specified time intervals. A video is provided on our website: <https://www.jeappen.com/diff-ma-stl/>

TABLE I: A comparison of MA-STL-A planner choices. Our diffusion-based method excels in Diversity and Generalizability with capabilities beyond other planners. **Het. (Specs):** Heterogeneous Specifications i.e., different tasks among agents, **MA Aware:** Considers Multi-agent interactions while planning, **Cmplx. Specs:** Can handle complex specifications (with $N > 5$ agents as studied in prior work [9, Sec. 3]), **Scale:** Safely scale to large N (at least 32 agents), **Plan. Time:** Average planning time (s) to generate a single end-to-end plan for novel agent placements, **Div.:** Exhibits Plan Diversity (as in Sec V-C), **Gen.:** Allows Generalizability i.e., handling new specifications at runtime without additional training (in the order of several hours).

Method	Het. (Specs)	MA Aware	Cmplx. Specs	Scale	Div.	Gen.	Plan. Time
PWL MA-STL [2]	✓	✓	×	×	×	✓	> 10 s
STL PY-SA [11]	✓	×	✓	×	×	✓	> 10 s
GRADIENT [12]	✓	×	×	×	×	✓	> 10 s
GNN-ODE [9]	×	✓	✓	✓	×	×	< 0.1 s
DIFF-SA ([13])	✓	×	✓	✓	✓	✓	~ 1 s
DIFF-MA(Ours)	✓	✓	✓	✓	✓	✓	~ 1 s

temporal tasks in robotics [5] such as coverage, patrolling, and surveillance [6]. Importantly, collision avoidance remains a practical challenge for multi-robot systems operating in crowded spaces under complex logic constraints [7, 8]. At the same time, prominent existing methods [2] have been shown to scale poorly as the number of agents and the complexity of the specifications increase [9].

We consider Multi-Agent STL in the sense of MA-STL [2], where a specification $\Psi = \bigwedge_{i=1}^N \phi_i$ constrains each of N agents (e.g., $\phi_i = \Diamond_{[0,T]}(A_i) \wedge \Diamond_{[0,T]}(B_i) \wedge \Diamond_{[0,T]}(C_i)$ requires agent i to cover A_i, B_i, C_i within horizon T). A direct optimization approach that computes a piecewise-linear (PWL) trajectory with K segments per agent [2] does not scale

Manuscript received: April 21, 2026; Revised: July 15, 2026; Accepted: August 7, 2026.

This paper was recommended for publication by Editor M. Ani Hsieh upon evaluation of the Associate Editor and Reviewers' comments.

The authors are affiliated with the ECE and CS Departments of Purdue University. Contact: {jeappen, sjaganna}@purdue.edu

Digital Object Identifier (DOI): see top of this page.

with specification complexity and team size, especially once inter-agent interactions are included. In particular, pairwise collision-avoidance constraints introduce $\mathcal{O}\left(\binom{N}{2}K^2\right)$ additional decision variables, which quickly increases problem size. Recent work [9, Sec. 3] provides empirical evidence, reporting timeouts even for simple STL tasks with $N = 5$ single-integrator agents in a 2-D space. To address this, they adopt an *asynchronous* execution model combined with a GNN-based planner (GNN-ODE) and runtime safety filtering via control barrier functions [8], demonstrating results up to $N = 32$ agents in a nonlinear setting. This execution model differs from MA-STL, and we refer to the asynchronous variant as MA-STL-A (formalized in Sec. III).

An important caveat is that the GNN-ODE planner applies only to homogeneous tasks where all agents follow the same specification. This is restrictive, as real-world agents often have diverse objectives, such as covering different goals. Further, it does not allow zero-shot generalization [14] to new specifications since the specification is fixed at training time. While the approach handles different numbers and positions of agents during testing, a new specification ϕ' would require retraining times in the order of several hours.

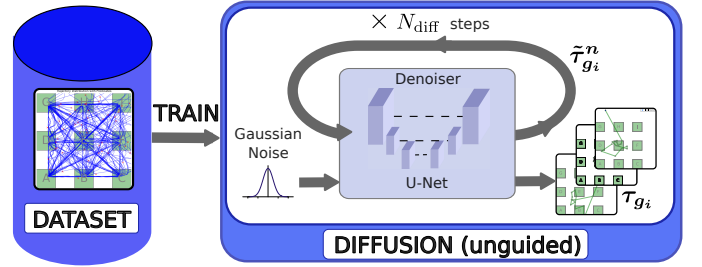
Moreover, the recurrent planner generates similar plans across agents, causing clustering-induced deadlocks and collisions in practice. Therefore, ideally, our final approach should possess three general characteristics:

- **Scalability:** Can the method efficiently accommodate large numbers of agents?
- **Generalizability:** Can the approach handle different goals among agents and new formulas over rectangular predicates placed within the trained goal region at runtime?
- **Diversity:** Are generated plans dispersed, thus reducing congestion?

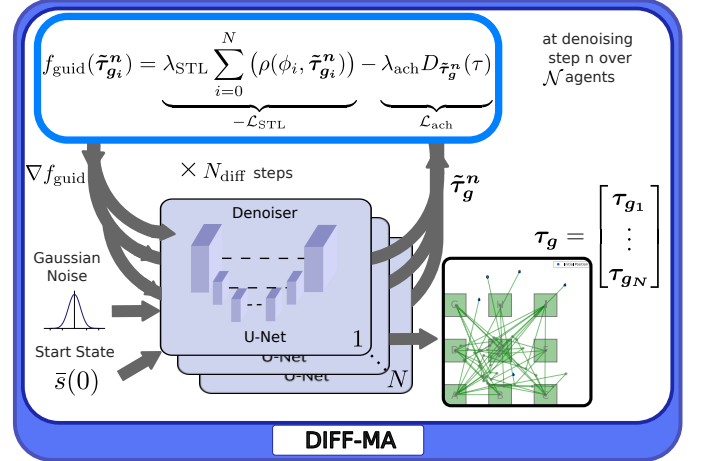
Table I summarizes MA-STL-A planner capabilities along these dimensions. Inspired by the success of diffusion models in robotics [15] to model complex objectives and dynamics, we propose a novel approach to *multi-agent planning* that addresses these challenges. Our contributions are as follows: 1) We propose DIFF-MA, a diffusion-based framework for multi-agent planning that generates diverse, jointly-optimized trajectories sampled from a trained single-agent diffusion model. 2) Our approach simultaneously satisfies MA-STL-A and team-level Capability Temporal Logic-based (CaTL+ [16]) specifications while incorporating inter-agent collision avoidance. 3) DIFF-MA enables test-time generalization to new specifications whose predicates are sampled within the trained goal region, without requiring retraining.

Related Work

Scalable control in high-dimensional environments often separates temporal logic planning from low-level tracking. High-level approaches include automata and optimization-based STL planners [17, 18], as well as multi-agent methods that coordinate or distribute specifications across agents [9]. Differentiable STL planners instead optimize trajectories through robustness gradients [19]. The resulting plans can be tracked using hierarchical RL [20] or controllers paired with classical planners [21]. Neural Control Barrier Function (CBF)



(a) Training process for the diffusion model on single agent trajectories (Sec. IV-A) without any STL-based guidance.



(b) Our approach (DIFF-MA) showing the use of a single-agent diffusion model for joint STL-guided multi-agent plan generation (Alg. 1 Lines 10-12) for the agents $\mathcal{N}$ with a given specification $\Psi = \bigwedge_{i=1}^N \phi_i$ (Sec. IV-B). $\tilde{\tau}_g^n$ represents denoised goal trajectories for all agents at diffusion step n .

Fig. 2: **Top:** Training overview, **Bottom:** MA STL-guided inference.

and Graph CBF (GCBF) methods filter actions during execution to preserve safety [7, 8], making them complementary to planners that synthesize trajectories satisfying temporal tasks.

A growing body of work uses diffusion and flow matching models as priors for robot policy and trajectory synthesis [15, 22, 23]. Temporal logic guidance has also been applied to generative planners, but direct STL guidance often achieves only moderate success in the single-agent setting [19, 24, 25]. Shaoul et al. [26] considers multi-robot planning through a Multi-Agent Path Finding formulation, but assumes perfect tracking to avoid collisions, which is difficult in practice. In contrast, our method adapts the EDM sampler [27] to guide a single-agent diffusion prior at inference time toward safe joint plans that satisfy STL for many agents, with an emphasis on scalable planning and reliable execution.

II. BACKGROUND

A. Multi Agent Systems with Partial Observability

A multi-agent system (MAS) with N agents can be denoted as $\mathcal{N} = \{1, 2, \dots, N\}$. Each agent i possesses a state $s_i(t) \in \mathcal{S}_i \subset \mathcal{R}^n$ and can execute an action $u_i(t) \in \mathcal{U}_i \subset \mathcal{R}^m$. The evolution of the agents' states is determined by the dynamics function $s_i(t+1) = f_i(s_i(t), u_i(t))$. For simplicity, we consider that all agents share the same dynamics function $f_i = f$, identical state space $\mathcal{S}_i = \mathcal{S}$, and action space $\mathcal{U}_i = \mathcal{U}$.

A system trajectory τ_s is defined as a sequence of states $\tau_s = (\bar{s}(0), \bar{s}(1), \dots, \bar{s}(T_h))$, where T_h represents the time horizon. The state trajectory for an agent i is denoted by $\tau_{s_i} = (s_i(0), s_i(1), \dots, s_i(T_h))$. Here, $\bar{s}(t) = (s_1(t), \dots, s_N(t))$. The system's state is partially observable, meaning each agent can only perceive its own state and the states of other agents within its sensing range R given as input to its controller. Each agent follows a policy π_i , which is a function mapping the agent's state to an action, expressed as $u_i = \pi_i(\bar{s}_{N_i})$ where N_i is the neighborhood of agent i within radius R (Sec. II-C).

B. Signal Temporal Logic

Signal Temporal Logic (STL) extends first-order logic with time-bounded variants of the temporal operators from Linear Temporal Logic (LTL). The Boolean connectives are $\wedge$ (and), $\neg$ (not), and $\vee$ (or), and the temporal operators are $\Diamond_{[a,b]}$ (eventually within the interval $[a, b]$), $\Box_{[a,b]}$ (always on $[a, b]$), and $\mathcal{U}_{[a,b]}$ (until within the interval from $[a, b]$). The syntax is as follows:

$$\phi := \mathcal{P} \mid \neg\phi \mid \phi \wedge \psi \mid \phi \vee \psi \mid \Diamond_{[a,b]}\phi \mid \Box_{[a,b]}\phi \mid \phi \mathcal{U}_{[a,b]}\psi$$

where $\mathcal{P}$ denotes a predicate that maps states to real values. The quantitative (robustness) semantics of STL [28] assign a real value $\rho(\phi, \tau_{s_i})$ indicating the degree to which a state trace τ_{s_i} satisfies or violates ϕ . This robustness can be differentiable [29], which enables direct gradient-based optimization of STL specifications using differentiable planners, including neural network-based methods.

In the context of multi-agent systems with N agents, the MA-STL-A specification Ψ is constructed from N separate STL specifications, represented as $\bigwedge_{i=1}^N \phi_i$, where ϕ_i corresponds to the STL specification for each individual agent indexed by i . The MA-STL-A specification Ψ is fulfilled when all individual STL specifications are met and there are no collisions among the agents. Similar to the variant of STL depicted in Sun et al. [2], MA-STL-A is a subset of all STL specifications defined on the joint state space, yet captures many intended multi-agent behaviors.

We also consider team-level specifications in the form of CaTL+ [16], whose atomic unit is a task $\mathcal{T} = \langle \phi, m \rangle^1$ satisfied when at least m agents meet an inner-logic formula ϕ on their individual trajectories. The outer logic composes tasks via Boolean connectives, e.g. $\mathcal{T}_1 \wedge \mathcal{T}_2$ requires both tasks to be satisfied while $\mathcal{T}_A \vee \mathcal{T}_B$ requires either task to be satisfied. Using MA-STL-A as the inner logic, this framework covers both per-agent independence ($m=1$) and inter-dependent team tasks ($m > 1$) where multiple agents must coordinate on a shared formula.

C. Graph-Based Modeling and Safety via Graph Control Barrier Functions

Our controller (GCBF+ [8]), uses a *Graph Control Barrier Function* (GCBF), and adopts a standard graph-based abstraction for multi-agent systems (MAS), where agents are nodes of a graph $G = (V, E)$. Edges encode local interactions with an

edge $(v_i, v_j) \in E$ being present when the Euclidean distance between nodes v_i and v_j does not exceed a sensing radius R . In the GCBF+ controller, a graph neural network (GNN) processes this representation and parameterizes a local control policy for each agent. GCBF+ supports both agent-agent and agent-obstacle interactions, but we focus on the former for our scalability analysis due to a known limitation of deadlocks when obstacles are present [30]. The safe set $\mathcal{S}_s \subset \mathcal{S}^N$ for an N -agent MAS (where $\mathcal{S}^N$ denotes the joint state space over N agents) is the collection of states $\bar{s}$ satisfying the inter-agent separation requirements in Definition 1:

$$\mathcal{S}_s := \left\{ \bar{s} \in \mathcal{S}^N \mid \min_{i,j \in N, i \neq j} \|p_i - p_j\| > 2r \right\} \quad (1)$$

where r is the radius of each agent, and p_i is the position of agent i . The complement $\mathcal{S}_u := \mathcal{S}^N \setminus \mathcal{S}_s$ is the unsafe set. A continuously differentiable function $h : \mathcal{S}^M \rightarrow \mathbb{R}$ is a GCBF if there exists an extended class- $\mathcal{K}_\infty$ function α and a local policy $\pi_i : \mathcal{S}^M \rightarrow \mathcal{U}$ for each agent $i \in V_a$ such that, for all $\bar{s} \in \mathcal{S}^N$ with $N \geq M$,

$$\dot{h}(\bar{s}_{N_i}) + \alpha(h(\bar{s}_{N_i})) \geq 0, \quad \forall i \in V_a, \quad (2)$$

$$\dot{h}(\bar{s}_{N_i}) = \sum_{j \in N_i} \frac{\partial h(\bar{s}_{N_i})}{\partial s_j} f^c(s_j, u_j). \quad (3)$$

Here f^c denotes the continuous-time agent dynamics, whose sampled-data discretization yields the discrete-time model in Sec. II-A, and the controller enforces (2) at each sampling instant. Under these conditions, the zero-superlevel set of h is forward invariant: if a policy π_i and a GCBF h satisfy (2) for all agents and all $\bar{s} \in \mathcal{S}_s$, then trajectories starting in the zero-superlevel set of h (contained in $\mathcal{S}_s$) do not enter $\mathcal{S}_u$ [8].

D. Diffusion Models as Trajectory Generators

Diffusion models [23, 27] are a class of generative models that have recently shown great promise in modeling complex data distributions, particularly in high-dimensional spaces. They are designed to model the way data can be transformed through a diffusion process, which progressively adds noise to data and then learns to reverse this process to generate new samples. In the context of planning, diffusion models can be used to generate state trajectories [31] that adhere to system dynamics and satisfy certain constraints or objectives. Our framework follows techniques in Karras et al. [27] (EDM) where, for a data point $\mathbf{x}$, each denoising step i of N_{diff} steps of an Ordinary Differential Equation (ODE) for a noise schedule $\sigma(i)$ applies $d\mathbf{x} = -\dot{\sigma}(i)\sigma(i)\nabla_{\mathbf{x}} \log p(\mathbf{x}; \sigma(i)) di$, where $\dot{\sigma} = \frac{d\sigma}{di}$ and $\nabla_{\mathbf{x}} \log p(\mathbf{x}; \sigma(i))$ is the score function. Following EDM, we estimate the score from a learned denoiser $D_\theta(\mathbf{x}; \sigma)$ with weights θ . As a convention, throughout this work we will highlight the noised (marked $\hat{\mathbf{x}}$) and denoised outputs (marked $\tilde{\mathbf{x}}$) of diffusion models in bold.

III. PROBLEM STATEMENT

Consider Ψ to be either an MA-STL-A specification ($\Psi = \bigwedge_{i=1}^N \phi_i$) or a CaTL+ team specification (Sec. II-B) for N agents $\mathcal{N} = \{1, \dots, N\}$ with positions $p_i(t) \in \mathbb{P} \subset \mathbb{R}^n$ ($n \in \{2, 3\}$). Each agent's state $s_i(t) \in \mathcal{S}$ maps to its

¹In this paper, we consider identical agents sharing a single capability [16], so the capability argument is omitted from the original task tuple.

position via the projection $\text{filter}_{p_i} : S \rightarrow \mathbb{P}$, i.e., the first n components of $s_i(t)$. Agents are modeled as balls of radius $r > 0$: when an agent is at $p \in \mathbb{P}$, it occupies $B_r(p)$. Given local controllers $(\pi_i)_{i \in \mathcal{N}}$, we aim to design a centralized planner π_g^ψ that produces per-agent goal sequences $\tau_{g_i} = (g_i(0), \dots, g_i(T))$ of length $T < T_h$ over a planning horizon T_h , while execution is decentralized under $(\pi_i)_{i \in \mathcal{N}}$.

Definition 1 (Motion planning problem under MA-STL-A/CaTL+): Given Ψ , agents $\mathcal{N}$, and policies $(\pi_i)_{i \in \mathcal{N}}$ as above, a solution is a planner π_g^ψ such that the resulting closed-loop trajectories over $[0, T_h]$ satisfy:

- **Agent safety.** For all $t \in [0, T_h]$, and for all $i, j \in \mathcal{N}$ where $i \neq j$, $\|p_i(t) - p_j(t)\| > 2r$
- **Asynchronous STL satisfaction.** For each agent $i \in \mathcal{N}$ there exists $t_{i,0} < t_{i,1} < \dots < t_{i,T}$ such that $t_{i,k} \in (0, \dots, T_h)$ and the trajectories $\tau_i = (s_i(t_{i,0}), \dots, s_i(t_{i,T}))$ jointly satisfy Ψ under the robustness semantics of Sec. II-B, i.e. $\rho(\Psi, \{\tau_i\}_{i \in \mathcal{N}}) \geq 0$. For MA-STL-A this reduces to $\rho(\phi_i, \tau_i) \geq 0$ for all i .
- **Achievability of goals.** For all $i \in \mathcal{N}$, given the goal trajectory τ_{g_i} of length T from $\pi_g^{\phi_i}$, the tracking deviation $D_{\tau_i}(\tau_{g_i}) = \sum_{t'=0}^T \|\text{filter}_{p_i}(s_i(t_{i,t'})) - \text{filter}_{p_i}(g_i(t'))\|_2 < \epsilon$ for a small $\epsilon \in \mathbb{R}^+$.

IV. APPROACH

We implement a diffusion guidance approach which utilizes the diffusion process (EDM) in Karras et al. [27] to guide the plans generated by the diffusion model towards satisfying the STL objectives while respecting multi-agent safety constraints. Our solution, DIFF-MA (shown in Fig. 2b), is a two-stage process that first trains a diffusion model on single-agent trajectories (Sec. IV-A, Fig. 2a) covering a set of STL predicates $\mathcal{P}$, and then uses the trained model (Sec. IV-B) to jointly generate multi-agent trajectories $(\tilde{\tau}_g)$ for a system of agents (Sec. IV-C) guided by a new STL specification at test time. Training remains single-agent because the multi-agent coupling is injected at test time through the STL and achievability guidance rather than learned in the prior. We embed GCBF+ (Sec. II-C) in guidance (via closed-loop rollouts to calculate the achievability loss $\mathcal{L}_{\text{ach}}$) and execution, promoting compliance with safety requirements.

This process allows for safe, test-time generalization to new STL objectives involving arbitrary compositions of rectangular predicates placed within the goal region covered during training, following rules described in Sec. II-B.

A. Training Diffusion Models for Downstream STL Tasks

The first stage involves fixing a set of Predicates $\mathcal{P}$ which are used in our STL objectives. For our work, we assume rectangular predicate regions as shown in Fig. 1. With this set of predicates, we sample trajectories for a single agent achieving various randomly sampled sequences of goals following our same controller (GCBF+) to get a dataset covering the state space or a set of trajectories τ_i for a single agent i with a time horizon T_h . Since our objective is to obtain a planner π_g^ψ to generate τ_g for the multi-agent system, we fix the planning length T and sample the states $s_i(t)$ and goals $g_i(t)$

Algorithm 1 Plan sampling via STL-guided diffusion [27].

```

1: Parameters: Noise levels  $\sigma^n$ , guidance levels  $\lambda^n$ , noise factor  $\gamma^n$ ,
   noise scale  $S_{\text{noise}}$ , diffusion steps  $N_{\text{diff}}$ , small constant  $\epsilon_g$ 
2: Required: Denoiser model  $D_\theta$ , guidance function  $f_{\text{guid}}$ 
3: sample  $\tau^0 \sim \mathcal{N}(0, \sigma_0^2 \mathbf{I})$  // Sample random noise trajectory
4: for  $n = 0$  to  $N_{\text{diff}} - 1$  do
5:   sample  $\epsilon^n \sim \mathcal{N}(0, S_{\text{noise}}^2 \mathbf{I})$  ▷ Temporarily raise noise
6:    $\hat{\sigma}^n \leftarrow \sigma^n + \gamma^n \sigma^n$ 
7:    $\hat{\tau}^n \leftarrow \tau^n + \sqrt{(\hat{\sigma}^n)^2 - (\sigma^n)^2} \epsilon^n$ 
8:    $\tilde{\tau}^n \leftarrow [D_\theta(\hat{\tau}_i^n; \hat{\sigma}^n)]_{i=1}^N$  ▷ Estimate denoised trajectory
9:    $\mathbf{d}^n \leftarrow (\hat{\tau}^n - \tilde{\tau}^n) / \hat{\sigma}^n$  ▷ Evaluate  $\frac{\partial \tau}{\partial \sigma}$  at  $\hat{\sigma}^n$ 
10:   $\mathbf{g}^n \leftarrow \nabla_{\tilde{\tau}_g^n} f_{\text{guid}}(\tilde{\tau}_g^n)$  ▷ Denoised plan gradient
11:   $\|\mathbf{g}^n\|_{2, \text{agent}} \triangleq [\|\mathbf{g}_1^n\|_2, \dots, \|\mathbf{g}_N^n\|_2]^\top$  ▷ Agent-wise norm
12:   $\tilde{\tau}_g^n \leftarrow \tilde{\tau}^n + \lambda^n (\mathbf{g}^n \odot (\|\mathbf{g}^n\|_{2, \text{agent}} + \epsilon_g))$  ▷ MA-STL-A guidance ( $\odot$ : agent-axis division,  $\epsilon_g > 0$  for stability)
13:   $\tau^{n+1} \leftarrow \hat{\tau}^n + (\sigma^{n+1} - \hat{\sigma}^n) \mathbf{d}^n$  ▷ Apply Euler step
14:  if  $\sigma^{n+1} \neq 0$  then
15:     $\mathbf{d}'^n \leftarrow (\tau^{n+1} - [D_\theta(\tau_i^{n+1}; \sigma^{n+1})]_{i=1}^N) / \sigma^{n+1}$  ▷ 2nd order correction over  $N$  agents
16:     $\tau^{n+1} \leftarrow \hat{\tau}^n + (\sigma^{n+1} - \hat{\sigma}^n) (\frac{1}{2} \mathbf{d}^n + \frac{1}{2} \mathbf{d}'^n)$ 
17:  end if
18: end for
19: return  $\tau_g^{N_{\text{diff}}}$ 

```

for each agent i at each time step $t = \frac{T_h}{T} j$ for $j = 0, 1, \dots, T$. Given this dataset, we run a supervised training process where we train a diffusion model (Fig. 2a) to generate state-goal trajectories that represent $\tau_i = (\tau_{s_i}, \tau_{g_i})$ for an agent i .

B. Guiding Diffusion Models with STL Objectives

The DIFF-MA diffusion framework, presented in Algorithm 1, is based on the idea of using the denoised trajectory at a given diffusion step² n to estimate the gradient of the trajectory with respect to the STL robustness score ρ (section II-B). This gradient is then used to guide the trajectory towards satisfying the STL objectives. The initial condition is imposed by inpainting, overwriting the first plan step with the agent's current state after each denoising step [31].

We use τ^n to denote the state-goal trajectory at diffusion step n and τ_s^n and τ_g^n to denote the state and goal components of the trajectory, respectively. Our guidance function (f_{guid}) combines both the STL robustness score and a notion of achievability to steer the trajectory towards satisfying the STL objectives while ensuring that the generated trajectory is achievable by the agents in the system while following the GCBF+ controller. To this end we define the guidance function as follows, where ρ_ϕ is the STL robustness score and $\mathcal{L}_{\text{ach}}$ is the achievability loss function $f_{\text{guid}} = -\lambda_{\text{STL}} \mathcal{L}_{\text{STL}} - \lambda_{\text{ach}} \mathcal{L}_{\text{ach}}$ where λ_{STL} and λ_{ach} are positive weights for the STL loss and achievability loss, respectively. The STL loss $\mathcal{L}_{\text{STL}_i}$ for agent i is defined as the negative of the STL robustness score $\rho(\phi_i, \tilde{\tau}_{g_i}^n)$, with the total $\mathcal{L}_{\text{STL}} = \sum_{i=1}^N \mathcal{L}_{\text{STL}_i}$ for all agents $\mathcal{N}$. For CaTL+ specifications, whose robustness is not agent-decomposable [16], we instead set $\mathcal{L}_{\text{STL}} = -\rho(\Psi, \tau_g^n)$ using the joint multi-agent trajectory. The achievability loss function $\mathcal{L}_{\text{ach}}$ follows ideas from prior work [9] and is the tracking error between the denoised trajectory $\tilde{\tau}_g^n$ and a sampled trajectory using the GCBF+ controller. We overload

²We use superscripts on variables to show diffusion timesteps and subscripts to denote agent indices.

the notation of trajectory distance for the system of agents $\mathcal{L}_{\text{ach}} = D_{\tau^n}(\tilde{\tau}_g^n) = \sum_{i=1}^N D_{\tau_{s_i}^n}(\tilde{\tau}_{g_i}^n)$. By minimizing this loss, we ensure that the generated trajectory is cognizant of the system dynamics and the controller’s capabilities in the presence of other agents. Since calculating D_{τ^n} is expensive, we find it sufficient to include this in the last fraction (k_{ach} , e.g. 20%) of diffusion steps avoiding wasted computation on the initial near-Gaussian samples. After N_{diff} steps the final plan $\tau_g^{N_{\text{diff}}}$ is generated guided by both the MA-STL objective and our multi-agent achievability objective.

C. Applying Diffusion Planning to Multi-Agent Systems

Diffusion inference is inherently slow [32], and multi-agent joint sampling compounds this cost. We use a JAX-based [33] JIT-compiled vectorization of Algorithm 1 for fast inference. However, diffusion samples may still violate STL constraints. Related work [13, 19, 24] reports only 50–80% success, which worsens in multi-agent settings. Our sampler therefore *resamples* each agent i ’s plan before execution, up to N_{sample} times, until $\rho(\phi_i, \tau_{g_i}) > \epsilon_{\text{sample}} > 0$ (included in reported planning time). Each iteration draws $B = 8$ candidate joint plans in one reverse pass and keeps the one with highest exact robustness (joint team robustness for CaTL+), never mixing agents across candidates. Accepted plans are frozen and excluded from further updates, so the accepted set is monotonically non-decreasing (for CaTL+, acceptance uses the joint test $\rho(\Psi, \tau_g) > \epsilon_{\text{sample}}$ without per-agent freezing). A resampled agent keeps its previous plan unless a new draw scores strictly higher exact robustness, so kept robustness is monotone non-decreasing, and at the cap (N_{sample}) the loop returns the highest-robustness plan found. Acceptance certifies STL satisfaction at the plan level only. The achievability condition in Definition 1 is promoted through the soft guidance term $\mathcal{L}_{\text{ach}}$ rather than formally guaranteed, and is assessed empirically through the reported success rates.

V. EXPERIMENT SETUP

All experiments are analyzed with respect to the three characteristics introduced in Section I:

C1: Scalability **C2:** Generalizability **C3:** Diversity.

A. Performance at Scale (C1)

We validate our method on the DubinsCar benchmark, a widely used non-linear testbed for multi-agent navigation [8]. Our framework is constructed using JAX [33], building on the diffusion model framework provided by Jackson et al. [34] and GCBF+ [8] (see Sec. II-C), with *all* comparisons employing this underlying collision avoidance controller. We also include a video demonstration on a real differential-drive robot task run via the Robotarium test-bench [10] and results on other linear environments on our website. All diffusion training and inference procedures were run on an AWS g6.2xlarge instance or equivalent with 8 Intel Xeon-based CPU Cores and 32 GB of RAM with an Nvidia L4 GPU. We use 256 diffusion steps (N_{diff}) and up to 40 plan resamples (N_{sample}) with the noise levels ($\sigma_{\text{max}}, \sigma_{\text{min}}$) following the default settings of Jackson et al. [34].

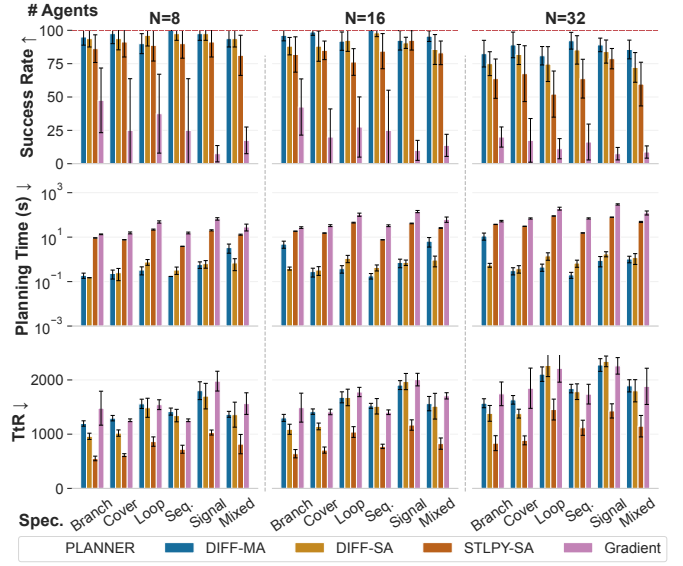


Fig. 3: Results over different number of agents and specifications. **Top:** Success Rate (%). **Middle:** Planning Time (s). **Bottom:** Time-to-Reach (TiR, steps). Standard deviations are shown around the mean. Our method (DIFF-MA) on average exceeds STLPY-SA by 20% with a notable relative success rate improvement of 36% for the crowded map ($N = 32$), while being 55x faster with zero retraining.

We vary the number of agents and benchmark against the strongest available baselines on various tasks. Since no existing multi-agent STL planner supports this setting at the scales considered, the fairest comparison is to execute state-of-the-art single-agent planners independently for each agent: a MILP-based planner (STLPY-SA) [11], a counterexample-guided gradient method (GRADIENT) [12], and a single-agent ablation of our approach (DIFF-SA), which can be viewed as a multi-agent extension of Feng et al. [13] with STL-score-based resampling. These baselines do not model inter-agent interactions during planning (Table I) and therefore rely on GCBF+ for collision avoidance (Sec. II-C) at execution time. Separately, to analyze plan diversity (Table II) in the *homogeneous* setting (where all agents share the same specification), we include the GNN-ODE planner [9], which does not support heterogeneous tasks. Collision-aware MILP approaches are known not to scale past five agents ($N > 5$) [9, Sec. 3], making this the most sensible comparison.

We report the mean *Planning Time* (in seconds), the *Success Rate* (percentage of runs where the STL specification was satisfied and no collisions occurred), and the *TiR* (time-to-reach, the steps successful runs take to complete the task) over 30 seeds (see Fig. 3). The effect of the achievable-loss coefficient λ_{ach} is studied in an ablation with $\lambda_{\text{ach}}=10^{-3}$ (LA) vs. $\lambda_{\text{ach}}=1$ (full) at fixed $\lambda_{\text{STL}}=1$ (Table III). We further compare (Fig. 6) the per-agent execution delay of DIFF-MA due to runtime maneuvering against the time an isolated single agent (planned with STLPY-SA) takes to satisfy its task on the *Mixed* specification. Lastly, we evaluate DIFF-MA on a larger map ($2\times$ the grid size) with up to $N = 128$ agents for the *Mixed* specification to study scalability at high N (Fig. 4).

B. Generalizing to Different Goals (C2)

To get a dataset for training our diffusion model, we first define a set of predicates $\mathcal{P}$ that we want to cover in our dataset. Our predicates are defined as rectangular regions in the 2D space, and placed evenly in a grid of 4.0×4.0 units. We use 9 predicates in total, which are shown in green in Fig. 1. To create a dataset to train our diffusion model that covers our predicate space $\mathcal{P}$, we sampled 10000 trajectories using the STLPY-SA planner on a *seq* specification with 3 predicates randomly sampled from $\mathcal{P}$. During inference (for the MA-STL-A results in Fig. 3, 6), we sample arbitrary predicate centers from the 4.0×4.0 unit space, with a size sampled uniformly from $[0.5, 1.0]$ units, to demonstrate that our diffusion model generalizes to unseen predicates (within the training space). For the CaTL+ (Fig. 5) and diversity experiments (Table II), we use fixed predicates from $\mathcal{P}$ on all settings for ease of visualization and comparison.

Unlike the homogeneous setting (Table I, [9]), the heterogeneous setting allows for different specifications among agents. We evaluate five STL task templates that capture common multi-agent missions – *sequence (seq)*: $\bigwedge_{i=1}^3 \Diamond_{[\tau_{i-1}, \tau_i]}(p_i)$ with $p = (X, Y, Z)$ and $\tau = (0, \frac{T}{3}, \frac{2T}{3}, T)$, *coverage (cover)*: $\bigwedge_{p \in \{X, Y, Z\}} \Diamond_{[0, T]}(p)$, *looping over goals (loop)*: $\Box_{[0, T/2]}(\bigwedge_{p \in \{X, Y, Z\}} \Diamond_{[0, T/2]}(p))$, *branching (branch)*: $\bigvee_{S \in \{\{X, Y\}, \{Z, W\}\}} \bigwedge_{p \in S} \Diamond_{[0, T]}(p)$, *loop until twice (signal)*: *loop* $\mathcal{U}_{[T_\ell, T]} W$ (loop over $\{X, Y, Z\}$ twice, then reach a new goal W in the final steps $[T_\ell, T]$, where T_ℓ is the loop horizon), and a *mixed* structure (*Mixed*): $\phi_i \sim \text{Uniform}(\text{seq}, \text{cover}, \text{loop}, \text{branch}, \text{signal})$, where $T = 15$ is the planning horizon, $k = 20$ the goal-sampling interval and unique predicates X, Y, Z, W are uniformly sampled for each agent i from the set $\mathcal{P}$. These encode ordered waypoint visits, unordered area coverage, periodic patrols, spatially partitioned goals, and uniformly sampled heterogeneous missions among agents. The above specifications assign one formula per agent ($m = 1$).

We also evaluate two CaTL+ tasks (Sec. II-B) with redundancy ($m > 1$) and inter-task avoidance (agents in one task avoid the goals of others): *Choice-Seq*: $(\mathcal{T}_0^A \wedge \mathcal{T}_1^A) \vee (\mathcal{T}_0^B \wedge \mathcal{T}_1^B)$, a disjunction of two branches, each pairing a 3-stage *seq* ($m = 3N/4$: gathering point then two waypoints along one edge) with a support task ($m = N/4$: two sequential goals). *Redundant*: $\bigwedge_{q=1}^4 \mathcal{T}_q$, i.e. 4 goals from $\mathcal{P}$ each reached by $m = N/4$ agents where $\mathcal{T}_q = \langle \Diamond_{[0, T]}(p_q), m \rangle$. With m agents converging on the same region, independent planning cannot anticipate collision-avoidance maneuvers. We additionally report the *Task Rate*, the fraction of episodes where the team-level specification is satisfied (Fig. 5). Task Rate ignores collisions, whereas Success Rate requires both per-agent task satisfaction and no collisions. Our baselines here are STLPY-GLOBAL, a global MILP jointly solving allocation (tractable only at $N = 8$, timing out > 10 min otherwise), and per-agent STLPY-SA under oracle task-to-agent allocation (best of closest first-goal, random, and Hungarian travel-distance matching). DIFF-MA requires no allocation scheme.

TABLE II: Comparison of plan diversity using the average number of agents per cluster and path overlap (%) for different planners (D-MA: DIFF-MA, GO: GNN-ODE, STLPY: STLPY-SA) in the Dubins car environment for Homogeneous specifications (same across agents). The best score (w.r.t. diversity) is highlighted in bold.

Spec.	N	Agents per Cluster ↓			Path Overlap (%) ↓		
		D-MA	GO	STLPY	D-MA	GO	STLPY
<i>Branch</i>	16	1.09	3.81	2.35	0.00	0.08	0.00
<i>Cover</i>	16	1.08	6.40	3.02	0.00	0.18	0.00
<i>Loop</i>	16	1.04	4.40	3.40	0.00	8.72	0.27
<i>Seq.</i>	16	1.08	7.74	2.89	0.00	0.25	12.42

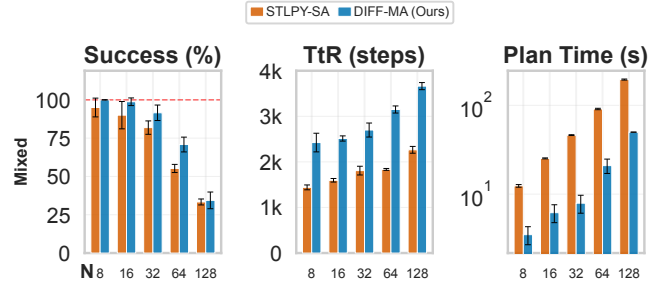


Fig. 4: We evaluate DIFF-MA on a larger map with up to $N = 128$ agents for the *Mixed* spec. and note degradation in success at high N (matching STLPY-SA at $N = 128$) with modest increases in planning time.

C. Diversity in Generated Plans (C3)

We evaluate how much the agents “share” their trajectories by comparing against the GNN-ODE and STLPY-SA in the homogeneous setting (Table II), where each agent had the same specification, using two complementary measures: (i) *Path Overlap* uses a 2D occupancy grid of the environment to compute the fraction of cells visited by all agents divided by the cells visited by any agent. A value of 0 means the plans are completely disjoint, while a value of 1 means all agents have the same trajectory. (ii) *Agents per Cluster*: from the pair-wise discrete Fréchet distance matrix [35] we connect two paths when their distance is at most $d_\tau = 1$ and count the average number of agents per connected component.

VI. RESULTS

Performance (C1, C2) Unlike GNN-ODE, which cannot accommodate heterogeneous specifications, DIFF-MA does so and, on average, attains a 20% higher success rate than STLPY-SA while planning $3 \times -200 \times$ faster (Fig. 3). At the largest evaluated scale ($N = 32$), DIFF-MA achieves a 36% relative increase in success over STLPY-SA. As with GNN-ODE, DIFF-MA deliberately trades time-to-reach (TtR) for safety: its paths are $\approx 48\%$ longer than those of STLPY-SA, whose average success at this scale is only 64%, to maintain safety in dense multi-agent settings. By contrast, GRADIENT underperforms across most specifications (never exceeding 20% success on the 32-agent tasks) and shows only marginal improvements on *seq/cover*, while DIFF-SA performs well on smaller instances but falls below 79% average success at this scale. Overall, DIFF-MA remains consistently strong, achieving at least 84% success even with 32 agents. We

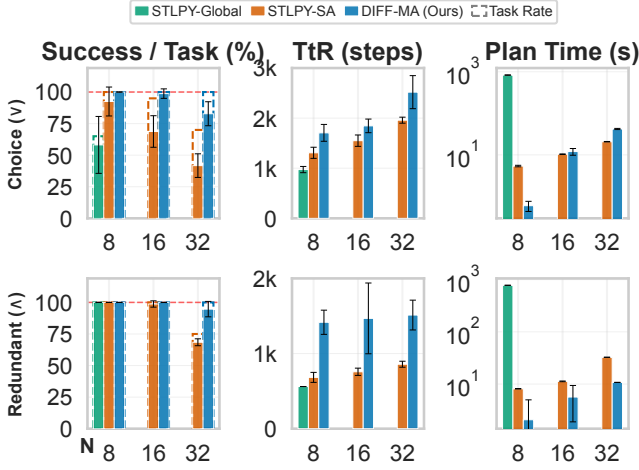


Fig. 5: Team-level spec. results on CaTL+ tasks with redundancy ($m > 1$) and inter-task avoidance constraints: *Choice-Seq* (two branches, each a pair of sequential tasks) and *Redundant* (shared waypoint sequences). Baselines: STLPY-GLOBAL (global MILP over all agents, feasible only at $N=8$) and STLPY-SA with oracle allocation (best of greedy and Hungarian matching by travel distance). DIFF-MA requires no task-to-agent allocation and maintains high success at scale ($N=32$) where STLPY-SA degrades sharply.

attribute the weaker performance of GRADIENT, STLPY-SA, and DIFF-SA to their design choices: they optimize individual-agent objectives without modeling inter-agent collisions during planning, instead relying on runtime collision avoidance. A failure-mode breakdown on our website supports this, as the remaining failures at $N=32$ are dominated by collisions, with the collision share of DIFF-MA (10.9%) staying $2.5\text{--}3\times$ below STLPY-SA (34.4%) and DIFF-SA (27.5%) while planner-side STL failures remain minimal.

On increasingly dense multi-agent scenarios (Fig. 4), the GCBF+ controller’s limited collision avoidance leads to degraded success at high N (matching STLPY-SA at $N=128$), noting that Zhang et al. [8] use a much larger map (roughly $3\times$ our size) for a reasonable safety rate at $N=128$. This is reflected by the measured safety rate of 42% at $N=128$ for DIFF-MA (vs. 95% at $N=32$). At that map scale, the diffusion model needed further optimization to cover the larger predicate space, which we leave for future studies.

Increasing the weight of the achievable loss $\mathcal{L}_{\text{ach}}$, which samples the environment to capture multi-agent interactions and GCBF+ dynamics during planning, lowers TtR by 17.37% and increases success by 2.37% in dense settings (32 agents, Table III). At this scale, safety-aware STL planning with DIFF-MA incurs an average per-agent delay (Fig. 6) of $1.3\text{--}1.9\times$ the single-agent *best-case* time reported by STLPY-SA.

Team Specifications (C1, C2) On team tasks (Fig. 5), STLPY-GLOBAL solves allocation near-optimally for *Redundant* at $N=8$ but fails to scale (timing out at $N > 8$), and per-agent STLPY-SA under oracle allocation (explicit) still suffers safety violations that lower success. DIFF-MA, with no allocation scheme (implicit), holds near 90% average success rate and beats the best STLPY-SA baseline by over 30% for $N=32$. Thus explicit allocation is sufficient at low N , but implicit allocation scales far better as agent safety interactions

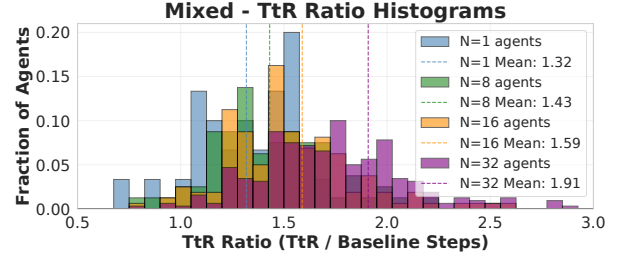


Fig. 6: Per-agent delay (from TtR) with DIFF-MA relative to the average time a single agent ($N=1$) using STLPY-SA takes to satisfy its specification. Histograms summarize 30 runs of the *Mixed* task with Baseline Steps = 990.

TABLE III: Achievable-loss coefficient ablation. D-MA (LA): low coefficient $\lambda_{\text{ach}}=10^{-3}$ (vs. $\lambda_{\text{ach}}=1$ for full DIFF-MA, $\lambda_{\text{STL}}=1$ throughout). $\Delta\%$ gives change relative to full DIFF-MA.

Spec	Success Rate (%) $\uparrow$	$\Delta\%$	Planning Time (s) $\downarrow$	$\Delta\%$	TtR (steps) $\downarrow$	$\Delta\%$
	D-MA (LA)		D-MA (LA)		D-MA (LA)	
<i>Mixed</i> 8	95.00	-4.60%	0.68	-38.75%	1868.17	+0.98%
16	91.25	-5.81%	0.67	-56.53%	1820.72	+0.46%
32	90.00	-2.37%	1.20	-61.96%	2038.92	+17.37%

intensify.

Diversity (C3) The Agents per Cluster and Path Overlap metrics in Table II show that existing planners (GNN-ODE and STLPY-SA) tend to generate similar trajectories for all agents, leading to a high overlap in their paths (notably GNN-ODE has over 7 of 16 agents following close trajectories in the *seq* task and a consistently non-zero Path Overlap across tasks). Our DIFF-MA planner, on the other hand, generates diverse trajectories for the agents, as indicated by the low values (i.e. agents having disjoint paths).

VII. CONCLUSION

We propose a novel approach for multi-agent planning that uses STL specification-guided diffusion models to generate joint plans for multiple agents while satisfying safety constraints. Our approach is capable of test-time generalization to new specifications with predicates placed anywhere within the trained goal region and is trained on a single agent trajectory dataset, allowing for efficient training and execution on multiple agents. Extensive experiments in simulation and real-world environments show that it matches the scalability of existing learning-based models (viz. GNN-ODE) while outperforming comparable methods in performance and plan diversity.

Discussion and Outlook

a) Model-based assumptions: Our method currently assumes access to accurate system dynamics (as in [7, 8]) to deploy the GCBF+ controller, providing per-step safety via control barrier functions, and to differentiate through $\mathcal{L}_{\text{ach}}$ (Sec. IV-B), which quantifies plan achievability with multi-agent interactions. These assumptions could be relaxed via model-free techniques [30].

b) Interaction with obstacles: We demonstrate strong scalability primarily in obstacle-free settings (with additional experiments in cluttered environments on our website), showing that the approach has modest hits to success in cluttered environments. Namely, in cluttered maps, coupling GCBF+ with

rich temporal objectives can yield deadlocks under decentralized execution because the barrier function prioritizes instantaneous safety over trajectory-level guarantees [30]. This highlights an opportunity to integrate controllers that reason about global or cumulative safety.

c) *Scalability*: Effective coordination beyond 32 agents on fixed maps/predicate sizes likely requires communication or distributed planning. To curb planning-time growth with agent count, faster diffusion inference [32] or conditional-flow models [24] are promising drop-in replacements, with tunable trade-offs between speed and plan diversity.

d) *Task complexity*: Our focus is on MA-STL-A and CaTL+ specifications (per-agent or team-level) with asynchronous timing, aligning with scenarios where users prefer brief delays over collisions (Sec. III). Extending expressivity to joint-state specifications and synchronized temporal relations (MA-STL [2]), or explicitly modeling delays [36], presents natural directions for broader applicability.

REFERENCES

- [1] S. Chaudhuri *et al.*, “Neurosymbolic programming,” *Found. Trends Program. Lang.*, 2021.
- [2] D. Sun, J. Chen, S. Mitra, and C. Fan, “Multi-agent motion planning from signal temporal logic specifications,” *IEEE RA-L*, 2022.
- [3] A. T. Buyukkocak, D. Aksaray, and Y. Yazıcıoğlu, “Planning of heterogeneous multi-agent systems under signal temporal logic specifications with integral predicates,” *IEEE RA-L*, 2021.
- [4] C. I. Vasile, X. Li, and C. Belta, “Reactive sampling-based path planning with temporal logic specifications,” *IJRR*, 2020.
- [5] A. Puranic, J. Deshmukh, and S. Nikolaidis, “Learning from demonstrations using signal temporal logic,” in *CoRL*, 2021.
- [6] S. Wang, X. Yin, S. Li, and X. Yin, “Tractable reinforcement learning for signal temporal logic tasks with counterfactual experience replay,” *IEEE L-CSS*, 2024.
- [7] Z. Qin *et al.*, “Learning safe multi-agent control with decentralized neural barrier certificates,” in *ICLR*, 2021.
- [8] S. Zhang, O. So, K. Garg, and C. Fan, “GCBF+: A Neural Graph Control Barrier Function Framework for Distributed Safe Multi-Agent Control,” 2024.
- [9] J. Eappen, Z. Xiong, D. Patel, A. Bera, and S. Jagannathan, “Scaling safe multi-agent control for signal temporal logic specifications,” in *CoRL*, 2024.
- [10] S. Wilson *et al.*, “The robotarium: Globally impactful opportunities, challenges, and lessons learned in remote-access, distributed control of multirobot systems,” *IEEE CSM*, 2020.
- [11] V. Kurtz and H. Lin, “Mixed-integer programming for signal temporal logic with fewer binary variables,” *IEEE L-CSS*, 2022.
- [12] C. Dawson and C. Fan, “Robust Counterexample-guided Optimization for Planning from Differentiable Temporal Logic,” in *IEEE/RSJ IROS*, 2022.
- [13] Z. Feng, H. Luan, P. Goyal, and H. Soh, “LTLDog: Satisfying temporally-extended symbolic constraints for safe diffusion-based planning,” *IEEE RA-L*, 2024.
- [14] P. Vaezipoor *et al.*, “LTL2Action: Generalizing LTL instructions for multi-task rl,” in *ICML*, 2021.
- [15] J. Carvalho *et al.*, “Motion planning diffusion: Learning and planning of robot motions with diffusion models,” in *IEEE/RSJ IROS*, 2023.
- [16] W. Liu, K. Leahy, Z. Serlin, and C. Belta, “Robust multi-agent coordination from CaTL+ specifications,” in *ACC*, 2023.
- [17] R. T. Icarte *et al.*, “Reward machines: Exploiting reward function structure in reinforcement learning,” *JAIR*, 2022.
- [18] W. Liu, N. Majcherczyk, and F. Pecora, “Scalable multi-robot task allocation and coordination under signal temporal logic specifications,” in *IEEE ICRA*, 2025.
- [19] P. Kapoor *et al.*, “STLCG++: A Masking Approach for Differentiable Signal Temporal Logic Specification,” 2025.
- [20] O. Nachum *et al.*, “Data-efficient hierarchical reinforcement learning,” in *NeurIPS*, 2018.
- [21] Z. Xiong, J. Eappen, A. H. Qureshi, and S. Jagannathan, “Model-free neural lyapunov control for safe robot navigation,” in *IEEE/RSJ IROS*, 2022.
- [22] J. Urain *et al.*, “Deep generative models in robotics: A survey on learning from multimodal demonstrations,” 2024.
- [23] J. Ho, A. Jain, and P. Abbeel, “Denoising diffusion probabilistic models,” in *NeurIPS*, 2020.
- [24] Y. Meng and C. Fan, “Telograf: Temporal logic planning via graph-encoded flow matching,” *ICML*, 2025.
- [25] R. Liu *et al.*, “Zero-shot trajectory planning for signal temporal logic tasks,” in *NeurIPS*, 2025.
- [26] Y. Shaoul *et al.*, “Multi-robot motion planning with diffusion models,” in *ICLR*, 2025.
- [27] T. Karras, M. Aittala, T. Aila, and S. Laine, “Elucidating the design space of diffusion-based generative models,” *NeurIPS*, 2022.
- [28] O. Maler and D. Nickovic, “Monitoring temporal properties of continuous signals,” in *FORMATS*, 2004.
- [29] K. Leung, N. Aréchiga, and M. Pavone, “Backpropagation through STL specifications: Infusing logical structure into gradient-based methods,” *IJRR*, 2022.
- [30] S. Zhang, O. So, M. Black, and C. Fan, “Discrete GCBF proximal policy optimization for multi-agent safe optimal control,” in *ICLR*, 2025.
- [31] M. Janner, Y. Du, J. Tenenbaum, and S. Levine, “Planning with diffusion for flexible behavior synthesis,” in *ICML*, 2022.
- [32] S. Li *et al.*, “Faster diffusion: Rethinking the role of the encoder for diffusion model inference,” *NeurIPS*, 2024.
- [33] J. Bradbury *et al.*, “JAX: composable transformations of Python+NumPy programs,” 2018.
- [34] M. T. Jackson *et al.*, “Policy-guided diffusion,” *RLJ*, 2024.
- [35] T. Eiter and H. Mannila, “Computing discrete Fréchet distance,” 1994.
- [36] X. Yu, X. Yin, and L. Lindemann, “Efficient STL control synthesis under asynchronous temporal robustness constraints,” in *CDC*, 2023.