2026 IEEE International Conference on Robotics and Automation (ICRA 2026)
June 1-5, 2026. Vienna, Austria

CLOT: Multi-robot Motion Planning via Collaborative Optimal Transport under Signal Temporal Logic Tasks

Ying Zhang[1], Yunyi Zhang[1], An T. Le[2,3] and Meng Guo[1]

***Abstract*— Multi-robot systems often need to navigate cluttered environments while performing complex tasks. To ensure collision-free trajectories among the robots and with the obstacles is essential for the overall safety, along with additional requirements such as dynamic feasibility, relative formation, connectivity maintenance and temporal tasks. Existing work mostly focuses on the design of analytical controllers that encapsulate all these constraints, which often suffer from undesired local minima due to conflicting non-convex objectives. This work proposes a novel motion planning scheme for multi-robot systems under various safety and high-level tasks, specified as signal temporal logic (STL) formulas over collective states such as collision avoidance, relative formation and connectivity maintenance. A gradient-free method called collaborative optimal transport (CLOT) is proposed that optimizes batches of system-wide smooth trajectories over highly nonlinear costs handled through the zero-order Sinkhorn Step. Via parallel computation on GPUs, the method achieves a planning time of few seconds for small teams and maintains tractability for over 100 robots. Lastly, its applicability is extensively demonstrated both in simulation and hardware, over complex environments and high-level temporal tasks.**

I. Introduction

Multi-robot systems have been deployed in practice for numerous tasks, such as service, maintenance, and search and rescue. As a fundamental requirement, the system should be able to navigate safely within a given complex workspace [1], i.e., to drive the robots from an initial state to the target state while staying within the allowed workspace and avoiding collision with obstacles or other robots. In addition, the system is often subject to various constraints and high-level tasks, e.g., the dynamics can be nonlinear; to follow a relative formation [2], [3], [4]; to remain connected during motion for information exchange [5], [6]; and to fulfill task specifications in temporal logics [7], [8]. Addressing such objectives and constraints simultaneously can be challenging due to their non-convexity and often conflicting goals.

A. Related Work

Collaborative motion planning has been traditionally approached through analytical controllers or search-based methods. Control objectives include reference tracking, consensus, and formation control, often incorporating collision avoidance and connectivity constraints [2], [9], [4], [5]. In addition, search-based methods directly generate trajectories for navigation, formation, or information gathering [10], [11], but typically only consider simple and monotone tasks without temporal or logic constraints. Moreover, temporal logics such as linear temporal logic (LTL) and signal temporal logic (STL) [12], [8], [13], [14] have emerged as a formal tool for specifying rich multi-robot tasks. Centralized approaches encode these specifications into optimization [15], [16], [17], achieving tight coordination via mixed integer or nonlinear optimization, but have proven difficult to scale and to handle nonlinear, non-convex constraints. Decentralized approaches [7], [8], [18] improve scalability via distributed reasoning or consensus, and learning-based extensions enable online adaptation [19]. Nevertheless, most existing STL frameworks address only primitive navigation and overlook more complex spatial couplings such as relative formation or connectivity maintenance, which are typical for multi-robot collaborative motion.

Lastly, motion planning via optimal transport (MPOT) [20] has recently emerged as a powerful trajectory optimization tool. By leveraging entropy-regularized OT and the Sinkhorn algorithm [21], it enables efficient parallel optimization with

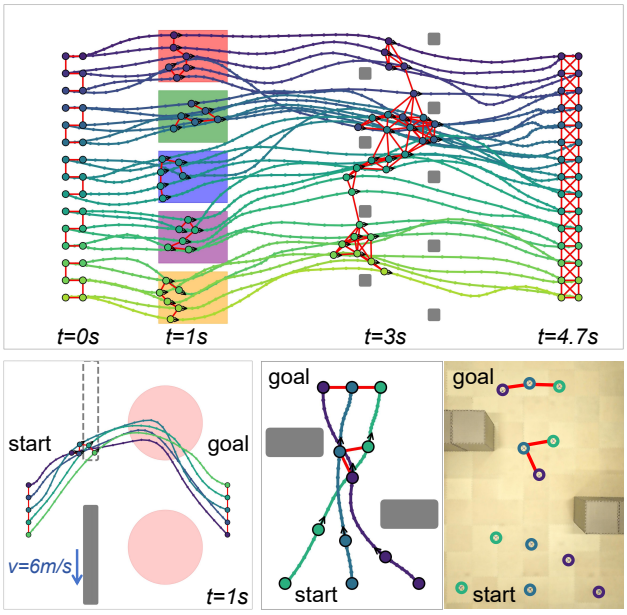

Fig. 1. **Top**: A split-and-merge mission for a group of 30 robots under the collaborative STL: $\phi = \bigwedge_{i=1}^{5} \left(\mathsf{G}_{[0,2.4]}\mu_B^{1,i}\right) \wedge \left(\mathsf{F}_{[0,2.4]}\left(\bigwedge_{i=1}^{5}\mu_A^i\right)\right) \wedge \left(\mathsf{G}_{[2.4,4.7]}\mu_B^2\right)$, where each team of 6 robots should maintain a linear formation and reach its corresponding designated region $\{\mathcal{W}_i\}_{i=1}^5$ during $[0, 2.4s]$, then all subteams merge to global connectivity; **Bottom-Left**: A fleet of 5 dynamically adapts its task region to the dynamic obstacles; **Bottom-Right**: Hardware experiments with 3 UAVs under the STL mission: $\phi = \mathsf{G}_{[8,15]}\mu_B$, where they converge from scattered states and maintain a linear formation while passing through a narrow corridor.

The authors are with: [1]School of Advanced Manufacturing and Robotics, Peking University, China; [2]College of Engineering and Computer Sciences, VinUniversity, Vietnam; [3]Intelligent Autonomous Systems, TU Darmstadt, Germany. This work was supported by the National Natural Science Foundation of China (NSFC) under grants U2241214, T2121002.
Corresponding author: Meng Guo, meng.guo@pku.edu.cn.

strong performance in smoothing, collision avoidance, and manipulation tasks. OT-based planners often outperform gradient methods in high-dimensional and highly non-convex settings [22], [23], [24], but remain focused on single-robot or centralized cases. Scalable collaborative OT for multi-robot planning is still largely unexplored.

B. Our Method

This work addresses multi-robot planning from a novel perspective, formulating it as a collaborative trajectory optimization task. From a planning-as-inference view, trajectories are sampled from Gaussian Process priors for smoothness, while dynamics and task constraints are incorporated as costs. A gradient-free method termed collaborative optimal transport (CLOT) is introduced, which optimizes batches of smooth trajectories under highly nonlinear costs. The Sinkhorn Step provides fast, zero-order, and parallelizable updates that collaboratively transport trajectory waypoints toward low-cost regions. To improve scalability, the scheme is extended with search over robot planning sequences and coupled local optimizations. The framework is applicable to complex workspaces with convex, non-convex, or dynamic obstacles, and high-level signal temporal logic (STL) tasks. Extensive numerical simulations and hardware experiments demonstrate its effectiveness.

Main contribution of this work is *threefold*: (I) it provides an alternative approach for multi-robot motion planning under complex constraints and objectives, applicable to a wide range of scenarios; (II) it demonstrates advantages compared to popular analytical planners and sampling-based motion planners; (III) it exemplifies the efficient usage of modern hardware such as GPUs for multi-robot planning.

II. Preliminary

A. Signal Temporal Logic and Space Robustness

Consider a discrete time signal $\mathbf{x} : \mathbb{N} \to \mathcal{X}$, where $\mathcal{X}$ is the value space. A signal predicate μ is a formula of the form $f(\mathbf{x}) > 0$ with $f : \mathcal{X} \to \mathbb{R}$. Afterwards, the STL task can be specified inductively via the following recursive syntax [25]: $\phi ::= \mu \mid \neg\phi \mid \phi_1 \wedge \phi_2 \mid \phi_1 \vee \phi_2 \mid \phi_1 \mathsf{U}_I \phi_2$, where μ is the predicate; $I \triangleq [t_1, t_2]$ is a time interval for $0 \leqslant t_1 \leqslant t_2$; the expressions ϕ, ϕ_1, and ϕ_2 are STL formulas; and $\neg$, $\wedge$ and $\vee$ are negation, conjunction and disjunction, respectively. The temporal operator U_I specifies that ϕ_1 holds until ϕ_2 holds in the time interval $[t_1, t_2]$. Further temporal operators such as F_I (eventually), G_I (globally) can be derived accordingly. In this work, only bounded-time specifications are considered with a finite I. Last but not least, the satisfaction of an STL formula ϕ by a system trajectory $\mathbf{x}(t)$ at time step t is denoted as $(\mathbf{x}, t) \models \phi$, of which the detailed semantics are omitted here.

B. Robustness Measure of Satisfaction

More importantly, the notion of robustness from [8], [25] or the so-called quantitative semantics [26] is adopted here to measure the degree of satisfaction for the finite system trajectory $\mathbf{x}$ at time $t > 0$ regarding the STL formula ϕ, i.e., $\rho^{\phi}(\mathbf{x}, t) \in \mathbb{R}$. The space robustness is formally defined as:

$$\rho^{\mu \geqslant 0}(\mathbf{x}, t) \triangleq f(\mathbf{x}(t)),\ \rho^{\neg\phi}(\mathbf{x}, t) \triangleq -\rho^{\phi}(\mathbf{x}, t),$$
$$\rho^{\phi_1 \wedge \phi_2}(\mathbf{x}, t) \triangleq \mathbf{min}\Big(\rho^{\phi_1}(\mathbf{x}, t), \rho^{\phi_2}(\mathbf{x}, t)\Big),$$
$$\rho^{\phi_1 \vee \phi_2}(\mathbf{x}, t) \triangleq \mathbf{max}\Big(\rho^{\phi_1}(\mathbf{x}, t), \rho^{\phi_2}(\mathbf{x}, t)\Big),$$
$$\rho^{\mathsf{G}_I \phi}(\mathbf{x}, t) \triangleq \min_{t' \in t \oplus I} \rho^{\phi}(\mathbf{x}, t'),\ \rho^{\mathsf{F}_I \phi}(\mathbf{x}, t) \triangleq \max_{t' \in t \oplus I} \rho^{\phi}(\mathbf{x}, t'),$$
$$\rho^{\phi_1 \mathsf{U}_I \phi_2}(\mathbf{x}, t) \triangleq \max_{t' \in t \oplus I} \mathbf{min}\Big(\rho^{\phi_2}(\mathbf{x}, t'), \min_{t'' \in (t, t')} \rho^{\phi_1}(\mathbf{x}, t'')\Big);$$

where $t \oplus I$ is the translation of set I by t. It is proven in [8], [26] that $\rho^{\phi}(\mathbf{x}, t)$ is positive only if $\mathbf{x} \models \phi$.

III. Problem Description

A. Multi-Robot System

Consider a multi-robot system consisting of N robots, where each robot $i \in \mathcal{N} \triangleq \{1, \cdots, N\}$ follows discrete-time first-order dynamics in the workspace $\mathcal{W} \subset \mathbb{R}^D$ for $D \geqslant 2$. Specifically, the position update is given by:

$$\mathbf{p}_i(t+1) = \mathbf{p}_i(t) + \mathbf{v}_i(t)\Delta t, \tag{1}$$

where $\mathbf{p}_i(t), \mathbf{v}_i(t) \in \mathbb{R}^D$ represent the position and velocity of robot i at $t \in \{0, 1, \cdots, T\}$, and Δt is the discrete time step size. Each robot $i \in \mathcal{N}$ occupies an area $\mathcal{A}_i(t)$ at time $t \geqslant 0$, e.g., a disk around $\mathbf{p}_i(t)$. Let $\mathbf{q}_i(t) \triangleq (\mathbf{p}_i(t), \mathbf{v}_i(t))$ denote the full state. Moreover, the stacked positions and control inputs are denoted as $\mathbf{x}(t) \triangleq [\mathbf{p}_1(t), \cdots, \mathbf{p}_N(t)]$ and $\mathbf{u}(t) \triangleq [\mathbf{v}_1(t), \cdots, \mathbf{v}_N(t)]$, while the full state is given by $\mathbf{z}(t) \triangleq (\mathbf{x}(t), \mathbf{u}(t))$. The full trajectory over the planning horizon T is defined as $\mathbf{Z}(T) \triangleq [\mathbf{z}(0), \cdots, \mathbf{z}(T)]^{\intercal}$, with $\mathbf{Z}_i(T)$ being the full trajectory of robot $i \in \mathcal{N}$. Finally, there are static obstacles $\mathcal{O} \subset \mathcal{W}$ within the workspace.

B. Constraints and Costs

The control effort and trajectory smoothness for each robot $i \in \mathcal{N}$ are evaluated through a cost function $h_i : \mathbb{R}^{2D} \times \mathbb{R}^{2D} \to \mathbb{R}^{+}$, capturing the dynamic feasibility and smoothness from $\mathbf{q}_i(t)$ to $\mathbf{q}_i(t+1)$. For brevity, the summed control cost is denoted by:

$$c_{\texttt{ctr}}\big(\mathbf{Z}_i(T)\big) \triangleq \sum_{t=0}^{T-1} h_i\big(\mathbf{q}_i(t), \mathbf{q}_i(t+1)\big), \tag{2}$$

where $h_i(\cdot, \cdot)$ may vary across robots due to heterogeneity.

On the other hand, the state-induced cost consists of three parts: (I) obstacle avoidance, with $g_{\texttt{obs}}\big(\mathbf{p}_i(t)\big) \triangleq \mathbf{max}\big\{0, (r_{\circ} - \texttt{MSD}(\mathcal{A}_i(t), \mathcal{O}))\big\}$, where $r_{\circ} > 0$ is a safety margin and $\texttt{MSD}(\cdot, \cdot)$ denotes the minimum signed distance between two manifolds [27]. The cost decreases as robot $i \in \mathcal{N}$ moves farther from the obstacles; (II) inter-robot collision avoidance, with $g_{\texttt{int}}\big(\mathbf{p}_i(t), \mathbf{p}_j(t)\big) \triangleq \mathbf{max}\big\{0, (r_{\texttt{s}} - \texttt{MSD}\big(\mathcal{A}_i(t), \mathcal{A}_j(t)\big))\big\}$, decreasing as robots $i, j \in \mathcal{N}$ move apart; (III) task-related cost, detailed in Sec. III-C.

C. Complex Collaborative Tasks as STL Formulas

To begin with, consider two primary classes of atomic propositions as predicates here: (I) μ_A^k, if all robots remain inside a static, pre-defined region $\mathcal{W}_k \subset \mathcal{W}$ at time $t \geqslant 0$, i.e., $\mu_A^k \triangleq -\mathbf{max}_{i=1,\cdots,N}\big(\texttt{MSD}(\mathcal{A}_i(t), \mathcal{W}_k) + 2r\big) \geqslant 0$, where r is the radius of the robot; (II) μ_B^k, if the system state $\mathbf{z}(t)$ satisfies the inter-robot collaborative constraints at time $t \geqslant 0$, i.e., $\mu_B^k \triangleq \mathcal{F}(\mathbf{z}(t)) \geqslant 0$. *Two* types of constraints considered in the work are:

- *Relative formation*, with function $\mathcal{F}_{\texttt{form}}\big(\mathbf{z}(t)\big) \triangleq \epsilon_{\texttt{f}} - \sum_{(i,j)\in\mathcal{E}} f_{ij}(\mathbf{q}_i, \mathbf{q}_j)$, where $\epsilon_{\texttt{f}} > 0$ denotes the maximum allowable formation error. The term $f_{ij} : \mathbb{R}^{2D} \times \mathbb{R}^{2D} \to \mathbb{R}^+$ quantifies the formation deviation between neighboring robots $(i,j) \in \mathcal{E}$, such as $f_{ij} \triangleq \big|\|\mathbf{p}_i - \mathbf{p}_j\| - d_{ij}\big|$ for distance-based formation [6] and $f_{ij} \triangleq \|(\mathbf{p}_i - \mathbf{p}_j) - \overline{\mathbf{p}}_{ij}\|$ for the relative-pose formation [9];
- *Connectivity maintenance*, with function $\mathcal{F}_{\texttt{con}}\big(\mathbf{z}(t)\big) \triangleq \lambda_2\big(L(t)\big)$, where $L(t) \in \mathbb{R}^{N\times N}$ is the Laplacian matrix associated with the graph $\mathcal{G}_{\texttt{c}}(t)$ at time $t \geqslant 0$. Thus, the underlying graph $\mathcal{G}_{\texttt{c}}(t)$ is connected *if and only if* the second smallest eigenvalue $\lambda_2(L(t))$ is positive [5], which is a system-wise property (not pairwise).

Given these predicates, a complex collaborative task can be specified inductively as STL formulas ϕ over $\mu \triangleq \{\mu_A^k, \mu_B^k\}$, following the syntax in Sec. II-A. Moreover, the robustness in Sec. II-B is adopted to measure the degree of satisfaction for a finite system trajectory $\mathbf{Z}$ at time $t > 0$ regarding the STL formula ϕ, i.e., $\rho^\phi(\mathbf{Z}, t) \in \mathbb{R}$. The robustness $\rho^\phi(\mathbf{Z}, t)$ is positive if $\mathbf{Z}(T) \models \phi$. Consequently, the collective cost over the system trajectory is given by:

$$c_{\texttt{tsk}}\big(\mathbf{Z}(T)\big) \triangleq -\rho^\phi(\mathbf{Z}, 0) + \sum_{t=0}^{T}\Big(\sum_{i\in\mathcal{N}} g_{\texttt{obs}}\big(\mathbf{p}_i(t)\big) + \sum_{i<j\in\mathcal{N}} g_{\texttt{int}}\big(\mathbf{p}_i(t), \mathbf{p}_j(t)\big)\Big), \tag{3}$$

which summarizes three types of state-induced costs, and should be minimized by the collaborative motion.

D. Collaborative Motion Planning Problem

Problem 1. Given the multi-robot system with initial and target states $\mathbf{x}_{\texttt{s}}, \mathbf{x}_{\texttt{g}} \in \mathbb{R}^{DN}$ and the STL task ϕ, the collaborative motion planning problem is formulated below:

$$\begin{aligned} \min_{\mathbf{Z}(T)} \quad & \Big(c_{\texttt{tsk}}\big(\mathbf{Z}(T)\big) + \sum_{i\in\mathcal{N}} c_{\texttt{ctr}}\big(\mathbf{Z}_i(T)\big)\Big) \\ \textbf{s.t.} \quad & \mathbf{x}(0) = \mathbf{x}_{\texttt{s}},\ \mathbf{x}(T) = \mathbf{x}_{\texttt{g}}; \\ & (1) - (3),\ \forall i \in \mathcal{N},\ \forall t \in [0, T]; \end{aligned} \tag{4}$$

where the optimization variables are the full trajectory of the system $\mathbf{Z}(t)$ over the given task duration T. ■

Remark 1. STL enforcement induces time-indexed disjunctions and conjunctions, requiring mixed-integer encodings [25]. Standard gradient-based NLP cannot be applied directly, and MILP/MICP formulations suffer exponential and combinatorial complexity in formula length, horizon, and number of robots [8], [16], [17]. ■

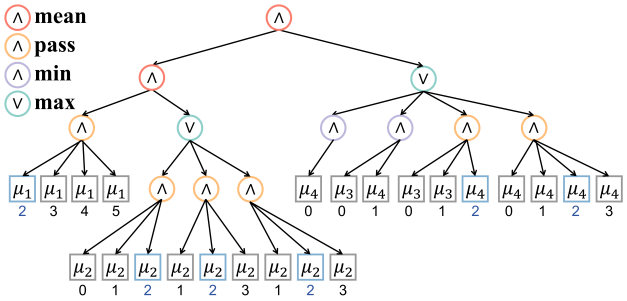

Fig. 2. The STL tree and perturbed robustness evaluation for formula $\phi = (\mathsf{G}_{[2,5]}\mu_1) \wedge (\mathsf{F}_{[0,2]}\mathsf{G}_{[0,2]}\mu_2) \wedge (\mu_3\mathsf{U}_{[0,3]}\mu_4)$, with the 2nd waypoint perturbed.

IV. Proposed Solution

This section introduces a unified framework that combines batch STL evaluation, local optimal transport, and hybrid search. First, batch robustness computation in Sec. IV-A uses STL parse-trees for efficient assessment of pointwise perturbations. This guides the sequential local optimal transport in Sec. IV-B, which plans agent trajectories in a dependency-aware order under task constraints. To enhance feasibility and optimality, a hybrid search scheme in Sec. IV-C jointly optimizes both planning sequences and trajectories. Finally, the execution scheme, complexity analyses, and generalizations are discussed in Sec. IV-D.

A. Batch Computation of STL Perturbed Robustness

To enable batch optimization of trajectories, this section describes how to efficiently evaluate the change in robustness when a single waypoint is perturbed in multiple directions.

1) Tree Structure of STL: An STL formula ϕ can be recursively decomposed into subformulas and operators [28], which can be organized as a tree [17]. The STL tree is $\mathcal{T}^\phi \triangleq (\widehat{\mathcal{T}}, \mathcal{I}, \circ)$, where $\widehat{\mathcal{T}} = [\mathcal{T}^{\phi_1}, \cdots, \mathcal{T}^{\phi_K}]$ is the list of $K \geqslant 1$ subtrees, $\mathcal{I} = [I^{\phi_1}, \cdots, I^{\phi_K}]$ are the time intervals, and $\circ \in \{\wedge, \vee\}$ are Boolean operators. Temporal operators expand as: F_I to disjunction $\vee$, G_I to conjunction $\wedge$, and U_I into a recursive combination. Thus, $\mathcal{T}^\phi$ is recursive, leaves correspond to predicates at time instants, and parent nodes represent operators. An example is shown in Fig. 2.

2) Parallel Evaluation of Perturbed Robustness: As in Sec. II-B, robustness of a trajectory $\mathbf{Z}(T)$ is computed recursively on $\mathcal{T}^\phi$ by evaluating leaves and applying **min** for conjunctions and **max** for disjunctions [25], [17], [28]. However, this may suppress perturbations, since a perturbed leaf dominated by **min** is discarded. To retain sensitivity, modified rules are used when only the $t^\star$-th waypoint is perturbed, denoted $\widehat{\mathbf{Z}}_{t^\star}(T)$: (I) leaf nodes calculate pointwise robustness $f_i^k(\widehat{\mathbf{z}}(t))$; (II) immediate conjunction parents pass $f_i^k(\widehat{\mathbf{z}}(t^\star))$ upward if $t^\star \in I^{\phi_i}$; (III) higher-level conjunctions aggregate via **mean**; and (IV) disjunctions use standard **max**. The perturbed robustness is

$$\widehat{\rho}^\phi\big(\widehat{\mathbf{Z}}_{t^\star}(T)\big) \triangleq \texttt{Eval}_{\mathcal{T}^\phi}\Big(\{f_i^k(\widehat{\mathbf{z}}(t))\}, \{\mathbf{max}, \mathbf{min}, \mathbf{mean}\}\Big),$$

where $\texttt{Eval}_{\mathcal{T}^\phi}(\cdot)$ denotes the recursive evaluation with modified operators. This amplifies the perturbation effects while approximating STL semantics, enabling efficient batch evaluation across all perturbations for each robot.

B. Sequential Local Optimal Transport

Given the batch evaluation of STL robustness, this section formalizes the sequential planning framework based on local optimal transport. The high-dimensional multi-robot planning problem is decomposed into a sequence of lower-dimensional subproblems, each optimizing one robot's trajectory in a prescribed order, with previously planned trajectories incorporated as dynamic constraints.

1) Design of Planning Sequence: To support sequential optimization under inter-robot constraints, an undirected dependency graph is defined as $\overline{\mathcal{G}} \triangleq (\mathcal{N}, \overline{\mathcal{E}})$, where $\overline{\mathcal{E}}$ encodes pairwise constraints. Global constraints such as connectivity maintenance are modeled by the initial communication graph. The neighboring set of robot i is $\mathcal{N}_i^{\overline{\mathcal{E}}} \triangleq \{j \in \mathcal{N} \mid (i,j) \in \overline{\mathcal{E}}, i \neq j\}$. The planning sequence of all robots is denoted by:

$$\kappa \triangleq [i_1, i_2, \cdots, i_N], \tag{5}$$

where $i_\ell \in \mathcal{N}$ for all ℓ. A sequence is *valid* if each robot i_ℓ is connected to at least one preceding robot in κ, i.e., $i_\ell \in \bigcup_{j \in \mathcal{N}_\ell^-} \mathcal{N}_j^{\overline{\mathcal{E}}}$, where $\mathcal{N}_\ell^- = \{i_1, \cdots, i_{\ell-1}\}$ and $\ell > 1$. For time-varying constraints $\{\mathcal{G}_s\}_{s=1}^S$, the condition strengthens to $i_\ell \in \bigcap_{s=1}^S (\bigcup_{j \in \mathcal{N}_\ell^-} \mathcal{N}_j^{\overline{\mathcal{E}}_s})$, ensuring that i_ℓ remains constrained with respect to $\mathcal{N}_\ell^-$ across all time steps.

Remark 2. These conditions ensure that each trajectory is feasible w.r.t. constraints imposed by preceding robots, while random orderings typically lead to infeasible solutions. ■

2) Constrained Trajectory Optimization for Each Robot: Given the sequence κ, each step is a single-robot optimization conditioned on previously planned robots. For the first robot i_1, the task cost reduces to obstacle avoidance $g_{\texttt{obs}}(\cdot)$ and robustness $-\rho^\phi(\cdot)$, yielding $\mathbf{Z}_{i_1}^\star$ which initializes $\Gamma_\kappa = \{\mathbf{Z}_{i_1}^\star\}$. For robot i_ℓ such that $2 \leqslant \ell \leqslant N$ with the preceding set Γ_ℓ^-, the task cost is given by:

$$\begin{aligned} c_{\texttt{tsk}}^{i_\ell}\big(\mathbf{Z}_{i_\ell}(T)\big) \triangleq \sum_{t=0}^{T} \Big(\sum_{i_j \in \mathcal{N}_\ell^-} g_{\texttt{int}}\big(\mathbf{p}_{i_\ell}(t), \mathbf{p}_{i_j}(t)\big) \\ + g_{\texttt{obs}}(\mathbf{p}_{i_\ell}(t)) \Big) - \widehat{\rho}^\phi(\Gamma_\ell), \end{aligned} \tag{6}$$

where $\widehat{\rho}^\phi(\cdot)$ are defined in Sec. IV-A.2, and $\Gamma_\ell = \Gamma_\ell^- \cup \{\mathbf{Z}_{i_\ell}(T)\}$ is the collective trajectory. With the control cost $c_{\texttt{ctr}}(\mathbf{Z}_{i_\ell}(T))$, the constrained trajectory optimization of each robot i_ℓ is defined similarly to Problem 1.

3) Local Optimal Transport via Sinkhorn Step: This part introduces the core method of CLOT as the Sinkhorn Step, which formulates local trajectory optimization as an entropy-regularized optimal transport problem [20]. It iteratively transports waypoints toward low-cost regions, yielding scalability, parallelization, and convergence guarantees.

As summarized in Alg. 1, the procedure has four steps. (I) *Initialization*. The trajectory of robot $i_\ell \in \kappa$ is denoted $\mathbf{Z}_\ell^k \in \mathbb{R}^{T \times 2D}$, sampled from GP priors; (II) *Probing*. Each waypoint is shifted along M directions $\boldsymbol{D}_\mathcal{P}$ constructed from a $2D$-dimensional polytope $\mathcal{P}$, and costs are evaluated to form $\mathbf{C} \in \mathbb{R}^{T \times M}$ using cost function defined in Sec. IV-B.2;

Algorithm 1: `SinkhornStep`$(\cdot)$

Input: Cost functions $c_{\texttt{tsk}}(\cdot)$, $c_{\texttt{ctr}}(\cdot)$.
Output: Optimized robot trajectory $\mathbf{Z}_\ell^\star$.

Initialize α_0, λ, and $\mathbf{Z}_\ell^0$;
Set histograms $\boldsymbol{n} = \mathbf{1}_T/T$, $\boldsymbol{m} = \mathbf{1}_M/M$;
while *($k \leqslant K$) and ($\|\mathbf{Z}_\ell^{k+1} - \mathbf{Z}_\ell^k\| > \epsilon$)* **do**
  Build $\boldsymbol{D}_\mathcal{P}$ with rotation;
  Compute $\boldsymbol{C}$ given $(\mathbf{Z}_\ell^k, \boldsymbol{D}_\mathcal{P})$;
  Compute $\boldsymbol{W}_\lambda^\star$ by (7) ;
  Update $\mathbf{Z}_\ell^{k+1}$ by (8);
  Update α_k and $k \leftarrow (k+1)$;
return $\mathbf{Z}_\ell^k$;

(III) *Weight optimization*. Transfer weights $\boldsymbol{W} \in \mathbb{R}^{T \times M}$ are optimized by solving the OT problem as:

$$OT_\lambda(\boldsymbol{n}, \boldsymbol{m}) \triangleq \min_{\boldsymbol{W} \in U(\boldsymbol{n}, \boldsymbol{m})} \big\{ \langle \boldsymbol{W}, \boldsymbol{C} \rangle - \lambda H(\boldsymbol{W}) \big\}, \tag{7}$$

where $\boldsymbol{n} \in \Sigma_T$ and $\boldsymbol{m} \in \Sigma_M$ are the histograms, each set to a uniform distribution, $U(\boldsymbol{n}, \boldsymbol{m})$ ensures that the marginals are matched, $\langle \boldsymbol{W}, \boldsymbol{C} \rangle$ is the inner product, $H(\boldsymbol{W})$ is the entropy regularization, and $\lambda > 0$ controls the regularization strength; (IV) *Trajectory update*. The updated trajectory is:

$$\mathbf{Z}_\ell^{k+1} = \mathbf{Z}_\ell^k + \alpha_k \left(\texttt{diag}(\boldsymbol{n})^{-1}\right) \boldsymbol{W}_\lambda^\star \boldsymbol{D}_\mathcal{P}, \tag{8}$$

where $\alpha_k > 0$ is the step size; and $\boldsymbol{W}_\lambda^\star$ is the convergent solution of (7). The resulting optimal trajectory $\mathbf{Z}_\ell^\star$ is appended to Γ_κ. The exact derivations are omitted here due to limited space [20], [21], [29].

Consequently, the next robot $i_{\ell+1} \in \kappa$ is planned given $\Gamma_{\ell+1}^-$, yielding $\mathbf{Z}_{\ell+1}^\star$. This process iterates until all robots are planned, resulting in the collaborative trajectory $\Gamma_\kappa = \{\mathbf{Z}_{i_1}^\star, \cdots, \mathbf{Z}_{i_N}^\star\}$. While efficient and scalable, this greedy approach depends on the planning sequence and ignores downstream effects, motivating the hybrid search introduced next. An example is shown in Fig. 3 with 5 robots under different choices of κ.

C. Hybrid Search for Sequence and Collective Trajectory

To overcome the limitations of sequential optimization, a hybrid search framework is proposed that integrates discrete search over planning sequences and trajectory candidates with continuous optimization of collective trajectories, as illustrated in Fig. 3. The key insight is that sequence selection and trajectory optimization are tightly coupled, and efficiency can be improved by bounding strategies, heuristic cost estimation, and parallel batch updates.

1) Hybrid Search Framework: The search is organized as a tree $\mathfrak{T} \triangleq (\mathcal{V}, \rightarrow)$, where each node $\nu \triangleq (\mathcal{N}_\nu, \Gamma_\nu)$ represents the set of robots already planned $\mathcal{N}_\nu \subset \mathcal{N}$ and their joint trajectories $\Gamma_\nu \triangleq \{\mathbf{Z}_{i_1}, \cdots, \mathbf{Z}_{i_\ell}\}$. The root node is $\nu_0 \triangleq (\varnothing, \varnothing)$, and child nodes are created by extending the sequence with new robots and solving the associated local optimization problems. As summarized in Alg. 2, the overall algorithm proceeds in four stages.

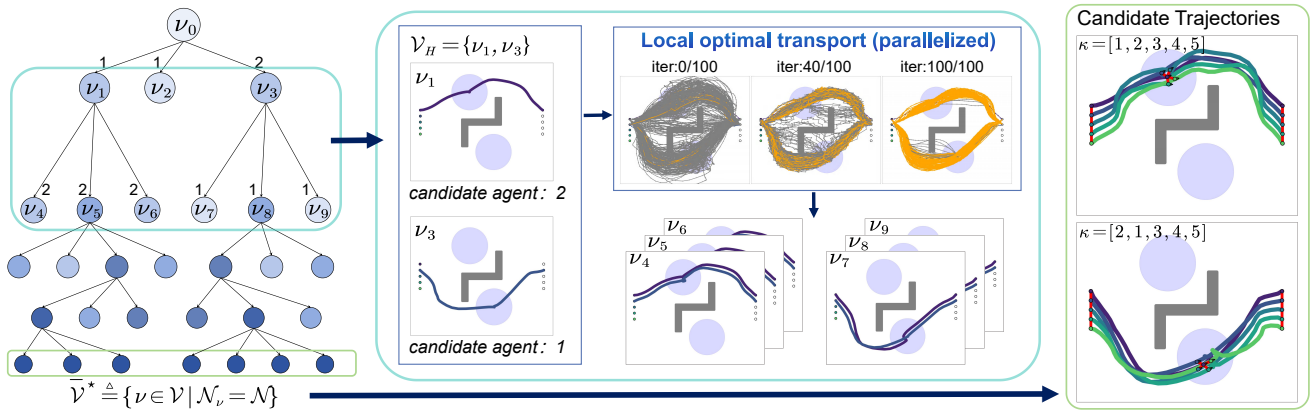

Fig. 3. The proposed hybrid search framework over the optimal planning sequence and collective trajectories (**Left**), with the interleaved sequential optimal transport (**Middle**). The final candidate solutions are ranked (**Right**).

(I) Selection. A subset of H candidate nodes is chosen for parallel expansion: $\mathcal{V}_H \triangleq \{\nu_1^\star, \cdots, \nu_H^\star\}$. Each node maximizes a value function given by:

$$\chi(\nu) \triangleq \frac{|\mathcal{N}_\nu|}{N} - \eta\, \xi(\Gamma_\nu) + \psi(\nu), \tag{9}$$

where the first term measures planning progress, the second term approximates accumulated cost, and the third term $\psi(\nu)$ propagates feedback from explored nodes. Here $\eta > 0$ is a weight, $\xi(\cdot)$ estimates trajectory cost similar to Problem 1, and $\psi(\cdot)$ is initialized as zero and updated during search.

(II) Expansion. Each $\nu_h^\star$ is expanded by choosing one unplanned robot $i^\star \in \mathcal{N}^{\text{ext}}_{\nu_h^\star}$, where $\mathcal{N}^{\text{ext}}_{\nu_h^\star} \triangleq \left(\bigcup_{i\in\mathcal{N}_{\nu_h^\star}} \mathcal{N}_i^{\overline{\mathcal{E}}}\right)\backslash\mathcal{N}_{\nu_h^\star}$. Local optimization for $i^\star$ produces candidate trajectories $\mathbf{Z}_{i^\star} \triangleq \{\boldsymbol{\tau}_{i^\star}^1, \cdots, \boldsymbol{\tau}_{i^\star}^K\}$ following the Sinkhorn Step in Alg. 1. Each candidate defines a child node by:

$$\nu_k^+ \triangleq (\mathcal{N}_{\nu_h^\star} \cup \{i^\star\},\ \Gamma_{\nu_h^\star} \cup \{\boldsymbol{\tau}_{i^\star}^k\}), \tag{10}$$

and edges $(\nu_h^\star, \nu_k^+)$ are added to $\rightarrow$. Note that if $\mathbf{Z}_{i^\star}$ is empty, expansion fails at this node.

(III) Back-propagation. The score $\psi(\cdot)$ in (9) is updated to reflect outcomes. For each ancestor ν of a new node ν^+,

$$\psi(\nu) \leftarrow \psi(\nu) + \psi(\nu^+)\cdot\gamma^{d(\nu,\nu^+)},\ \forall \nu \in \mathcal{V}^-_{\nu^+}, \tag{11}$$

where $\psi(\nu^+)$ is initialized positive; $\mathcal{V}^-_{\nu^+} \subset \mathcal{V}$ is the set of ancestors of ν^+; $d(\nu, \nu^+)$ is the path length in the tree; and $\gamma \in (0, 1)$ is a decay factor. Thus, feasible expansions reinforce the potential of their ancestors.

(IV) Termination. The cycle of selection, expansion, and back-propagation repeats until the time budget is exhausted. Leaf nodes where all robots are planned form $\overline{\mathcal{V}}^\star \triangleq \{\nu \in \mathcal{V} \mid \mathcal{N}_\nu = \mathcal{N}\}$. The final solution is $\overline{\nu}^\star \triangleq \textbf{argmax}_{\nu\in\overline{\mathcal{V}}^\star}\{\chi(\nu)\}$, and the collective trajectories are $\Gamma_{\overline{\nu}^\star}$.

2) Parallelization and Practical Improvements: Expansion and back-propagation are highly parallelizable. Batch processing optimizes trajectories for all $\nu_h^\star \in \mathcal{V}_H$, child nodes are created simultaneously, and updates in (11) are aggregated once, significantly reducing runtime. Adaptive control of the number of expanded nodes and trajectory samples balances search breadth and depth, enabling wide exploration initially and refined search at later stages. When expansion fails at node ν, a similarity-based pruning removes redundant nodes $\widetilde{\mathcal{V}}_\nu \triangleq \{\nu' \in \mathcal{V} \mid \|\Gamma_\nu - \Gamma_{\nu'}\| < \delta,\ \mathcal{N}_\nu = \mathcal{N}_{\nu'}\}$, preventing repeated search in infeasible regions. These strategies enhance scalability, reduce cascading failures, and accelerate convergence, as demonstrated in Sec. V.

D. Overall Analyses

1) Verification and Execution: After each local optimization, candidate trajectories are obtained by verifying all hard constraints, including collision avoidance, inter-robot constraints, and positive robustness under standard STL semantics. Thus, only trajectories that satisfy all feasibility and safety requirements are added to the search tree. Moreover, to improve execution quality, the trajectories $\{\mathbf{Z}_i^\star\}$ are further smoothed and refined into denser waypoints with a smaller step size, which are then directly tracked by onboard low-level controllers or by model predictive control (MPC).

2) Complexity Analysis: The dominant computation cost of Alg. 2 is the Sinkhorn Step during each expansion, with complexity $\widetilde{O}\left(\frac{2D}{\epsilon^2}\cdot\frac{(HN_pT)^2}{\varepsilon^3}\right)$, where D is the spatial dimension, ϵ the convergence precision, H the number of parallel expansions, N_p the number of trajectory samples, T the trajectory length, and ε the threshold of the Sinkhorn-Knopp scaling [20], [21], [29]. Constraint evaluation adds $O(M|\mathcal{T}^\phi|)$ per robot, where M and $|\mathcal{T}^\phi|$ denote the number of probing directions and STL tree nodes, respectively. Moreover, with N robots, the search tree depth is at most N. In practice, pruning and heuristics reduce the search to about CN expansions where $C \in [1, 3]$ empirically.

3) Generalization: The framework also generalizes to more complex scenarios: (I) *Split-and-merge for large fleets*. The fleet can be partitioned into subgroups $\{\mathcal{N}_1, \cdots, \mathcal{N}_G\}$ with individual formation constraints. A global STL formula coordinates these groups, enabling them to split, navigate independently, and later merge seamlessly; (II) *Dynamic obstacles*. When obstacles have known or predictable trajectories, they can be incorporated into the task evaluation by (3), ensuring collision avoidance and STL robustness in dynamic environments; for uncertain motions, conservative reachable sets can be adopted to maintain robustness.

V. Numerical Experiments

For numerical validations, the proposed method is implemented in `Python3` using PyTorch for trajectory vectoriza-

Algorithm 2: Collab. Optimal Transport (CLOT)

Input: Initial state $\mathbf{x}_{\mathrm{s}}$, goal state $\mathbf{x}_{\mathrm{g}}$, cost functions $c_{\mathtt{ctr}}(\cdot)$ and $c_{\mathtt{tsk}}(\cdot)$.
Output: Feasible trajectory $\Gamma_{\overline{\nu}^\star}$.

Initialize $\mathfrak{T} = (\{\nu_0\}, \varnothing)$;
while *not terminated (*In Parallel*)* **do**
  /* **Selection** */
  Select $\mathcal{V}_H$ via (9);
  /* **Expansion** */
  foreach $\nu_h^\star \in \mathcal{V}_H$ **do**
    $\mathcal{N}_{\nu_h^\star}^{\mathrm{ext}} \leftarrow \left(\bigcup_{i\in\mathcal{N}_{\nu_h^\star}} \mathcal{N}_i^{\overline{\mathcal{E}}}\right) \setminus \mathcal{N}_{\nu_h^\star}$;
    if $\mathcal{N}_{\nu_h^\star}^{ext} \neq \varnothing$ **then**
      Choose $i^\star \in \mathcal{N}_{\nu_h^\star}^{\mathrm{ext}}$;
      $\mathbf{Z}_{i^\star} \leftarrow \mathtt{SinkhornStep}(\cdot)$ by Alg. 1;
      foreach $\boldsymbol{\tau}_{i^\star}^k \in \mathbf{Z}_{i^\star}$ **do**
        Generate ν_k^+ by (10);
        Update $\mathcal{V}$ and $\rightarrow$;
  /* **Back-propagation** */
  Update $\psi(\nu)$ by (11);
  $\overline{\mathcal{V}}^\star \leftarrow \{\nu \in \mathcal{V} \,|\, \mathcal{N}_\nu = \mathcal{N}\}$;
/* **Termination** */
$\overline{\nu}^\star \leftarrow \mathbf{argmax}_{\nu\in\overline{\mathcal{V}}^\star} \chi(\nu)$;
return $\Gamma_{\overline{\nu}^\star}$;

tion and optimization. The experiments are conducted on a computer with an Intel Core i9-14900KF CPU and NVIDIA RTX 4080 GPU. Simulation and experiment videos can be found in the supplementary files.

A. Numerical Simulations

1) System Description: As illustrated in Fig. 3, 4, and 5, three distinct scenarios are considered, characterized by different numbers of robots, STL task specifications, and workspace configurations. Each robot has a radius of $0.1\mathrm{m}$. The planning horizon is set to $T = 48$ with discrete time step $dt = 0.1\mathrm{s}$. The inter-robot formation distance is set to $r_{\mathrm{c}} = 0.8\mathrm{m}$. For each local optimal transport iteration, 800, 800, and 1000 trajectories are optimized in parallel for the three scenarios, respectively.

(I) *Scenario 1*: As shown in Fig. 3, a team of 5 robots operates in a $12\mathrm{m} \times 10\mathrm{m}$ workspace. Robots must reach one of two designated blue regions $\mathcal{W}_1, \mathcal{W}_2$ and remain there for at least $0.5s$ with a star formation. Outside these regions, they maintain a linear formation, i.e., $\phi_1 \triangleq \left(\mathsf{F}_{[0,4.7]}\mathsf{G}_{[0,0.5]}(\mu_A^1 \vee \mu_A^2)\right) \wedge \left(\mathsf{G}_{[0,4.7]}(\mu_B^1 \vee (\mu_A^1 \vee \mu_A^2))\right) \wedge \left(\mathsf{G}_{[0,4.7]}(\mu_B^2 \vee (\neg\mu_A^1 \wedge \neg\mu_A^2))\right)$, where μ_A^1, μ_A^2 are the region predicates and μ_B^1, μ_B^2 as the formation predicates.

(II) *Scenario 2*: As shown in Fig. 4, a fleet of 12 robots operates in a $20\mathrm{m} \times 7\mathrm{m}$ workspace. During the first half of the mission interval, robots must preserve connectivity while traversing a red region $\mathcal{W}_1$. In the second half, they remain connected and avoid a blue region $\mathcal{W}_2$ until transitioning into the linear formation, which they maintain thereafter to pass the narrow corridor, i.e., $\phi_2 \triangleq \left(\mathsf{G}_{[0,2.4]}\mu_B^1\right) \wedge \left(\mathsf{F}_{[0,2.4]}\mu_A^1\right) \wedge \left((\neg\mu_A^2 \wedge \mu_B^1)\mathsf{U}_{[2.4,4.7]}(\mu_B^2 \wedge \mathsf{G}_{[0,4.7]}\mu_B^2)\right)$, where μ_A^1, μ_A^2 correspond to region predicates and μ_B^1, μ_B^2 encode connectivity and formation constraints.

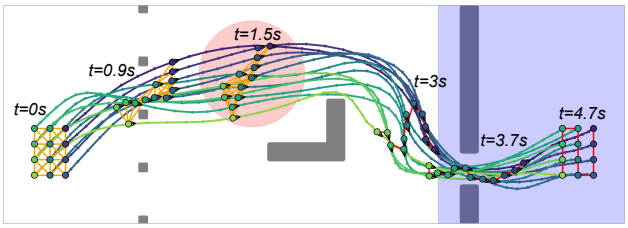

Fig. 4. A fleet of 12 robots under the STL mission ϕ_2, where the global connectivity is ensured during $[0, 2.4s]$ and a line formation is required before entering the blue region by $2.4s$.

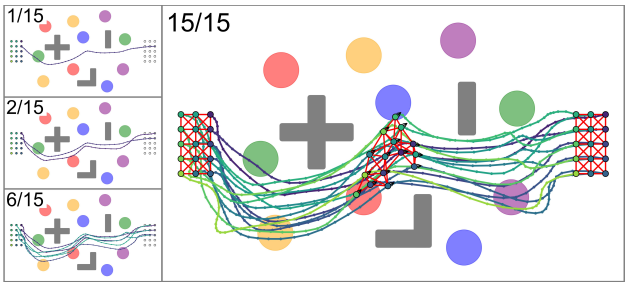

Fig. 5. The fleet of 15 robots should cross one of 5 types of regions while remaining connected, as specified as ϕ_3.

(III) *Scenario 3*: As shown in Fig. 5, a fleet of 15 robots operates in a $16\mathrm{m} \times 10\mathrm{m}$ workspace. Robots must remain connected at all times and visit five types of regions $\{\mathcal{W}_i\}_{i=1}^5$, each containing two candidates. At least one region from each group must be visited, i.e., $\phi_3 \triangleq \bigwedge_{i=1}^5 \left(\bigvee_{j=1}^2 \left(\mathsf{F}_{[0,4.7]}\mu_{A^-}^{i,j}\right)\right) \wedge \left(\mathsf{G}_{[0,4.7]}\mu_B^1\right)$, where $\mu_{A^-}^{i,j}$ represent each region and μ_B^1 for connectivity. For each scenario, 20 independent trials are conducted with distinct random seeds for initialization and optimization.

2) Results: As shown in Fig. 3, 4, and 5, the method generates feasible smooth trajectories that satisfy all STL specifications. In scenario-1, only two feasible sequences exist, $\kappa = [1, 2, 3, 4, 5]$ and $\kappa = [2, 1, 3, 4, 5]$, where planning robot 1 first leads it to the upper region, while prioritizing robot 2 selects the lower region. These yield distinct yet valid joint trajectories as shown in Fig. 3. Optimization completes in the 5-th iteration with planning time $20.60s$, achieving 100% success and an average planning time of $34.89s$ over 20 trials. In scenario-2, multiple sequences are feasible, but the confined free space results in similar trajectories. An example is shown in Fig. 4, where robots reach $\mathcal{W}_1$ at $t = 1.5s$, form a line by $t = 3s$, then traverse a narrow corridor. The process terminates in the 12-th iteration with planning time at $75.64s$, with 100% success and an average planning time of $90.98s$ over all successful trials. In scenario-3, dispersed targets require robots to act as relays until the sixth robot ensures full coverage (Fig. 5). Optimization ends in the 15-th iteration with a planning time of $111.98s$, achieving 95% success and an average planning time of $151.91s$. Across the three scenarios, the average numbers of local iterations are 6.85, 13.95, and 17.37, only 1.37, 1.16, and 1.16 times the minimum iterations required, demonstrating efficiency of the proposed hybrid search scheme. The average computation

TABLE I

COMPARISON OF MULTI-ROBOT STL TASKS WITH $N = 3, 5, 12$ ROBOTS

Method	Success Rate			Planning Time[s]			Path Length			Smoothness		
	N=3	N=5	N=12	N=3	N=5	N=12	N=3	N=5	N=12	N=3	N=5	N=12
CLOT(ours)	**1.00**	**1.00**	**1.00**	4.19	6.93	**20.3**	14.5	14.0	13.5	0.51	0.46	0.43
MPOT	**1.00**	**1.00**	0.10	3.14	5.55	26.5	14.6	14.3	13.7	0.54	0.49	0.46
FSOT	**1.00**	**1.00**	0.70	4.00	6.63	19.8	13.4	12.8	12.0	0.45	0.40	0.36
MICP	0.00	0.00	0.00	28.5	-	-	-	-	-	-	-	-
NLP	0.00	0.00	0.00	16.0	-	-	-	-	-	-	-	-
CONS	**1.00**	**1.00**	0.00	**1.41**	**3.00**	-	12.3	12.1	-	0.56	0.54	-
CBS	**1.00**	0.00	0.00	4.64	6.73	-	16.0	15.0	-	-	-	-

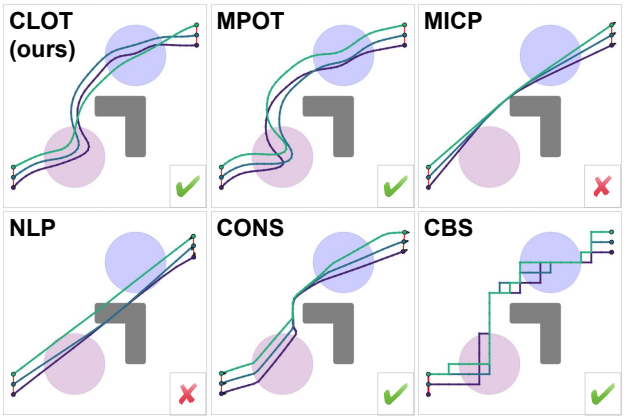

Fig. 6. Comparison of different baseline methods for $N = 3$ fleet with the STL task $\phi = (\mathsf{F}_{[0,4.7]}\mu_A^1) \wedge (\mathsf{F}_{[0,4.7]}\mu_A^2) \wedge (\mathsf{G}_{[0,4.7]}\mu_B^1)$.

times per iteration are $5.09s$, $6.52s$, and $8.74s$, with 9.40, 44.70, and 19.05 final valid joint trajectories generated on average, providing diverse candidate solutions.

3) Comparisons: **CLOT** is benchmarked against six baselines: (I) **MPOT**, an OT-based planner directly extended to multi-robot [20]; (II) **FSOT**, an ablation of CLOT using fixed predefined sequences; (III) **MICP**, a mixed-integer convex program encoding STL tasks with `stlpy` [17]; (IV) **NLP**, a gradient-based nonlinear program [17]; (V) **CONS**, a distributed consensus-based controller [2]; (VI) **CBS**, a discrete MAPF solver based on conflict-based search [11]. Initial experiments on three scenarios $\phi_{1,2,3}$ show that *all baselines fail* to produce feasible solutions. Thus, a simpler evaluation is conducted instead shown in Fig. 6, where robots must traverse $\mathcal{W}_1$, $\mathcal{W}_2$, and maintain connectivity under $\phi_4 = (\mathsf{F}_{[0,4.7]}\mu_A^1) \wedge (\mathsf{F}_{[0,4.7]}\mu_A^2) \wedge (\mathsf{G}_{[0,4.7]}\mu_B^1)$. Metrics include success rate, planning time, path length, and trajectory smoothness for $N = 3, 5, 12$.

As reported in Table I, CLOT achieves 100% success across all team sizes, whereas baseline performance degrades as N grows. MPOT drops to 0.10 at $N = 12$, FSOT to 0.70. Even for small teams, MICP and NLP fail, and CBS yields only discrete trajectories unsuitable for continuous execution. In runtime, CONS is the fastest for small cases at $1.41s$ for $N = 3$, but does not scale, while CLOT maintains tractable runtime at $20.3s$ for $N = 12$, outperforming MPOT and comparable to FSOT. CLOT yields slightly longer paths than FSOT and CONS (13.5 vs. 12.0–12.3) but remain competitive, and the smoothness closely matches other baselines.

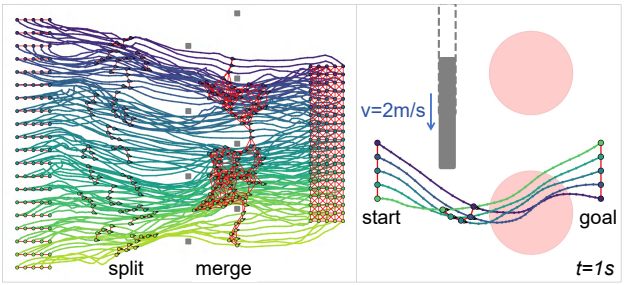

Fig. 7. **Left**: A split-and-merge mission for a group of 100 robots; **Right**: Fleet of 5 adopts lower task region with slow obstacles.

Overall, CLOT exhibits superior scalability, robustness, and reliability, particularly in large-scale multi-robot settings.

4) Generalization: As discussed in Sec.IV-D.3 and shown in Fig. 1 and 7, several generalizations of the proposed method are evaluated. (I) **Split-and-merge for large fleets**: a fleet of $N = 30$ robots in an $18\text{m}\times10\text{m}$ workspace is divided into five subteams of six. Each subteam first maintains a linear formation and reaches its designated region, then all reconnect to maintain global connectivity, i.e., $\phi = \bigwedge_{i=1}^{5}(\mathsf{G}_{[0,2.4]}\mu_B^{1,i}) \wedge (\mathsf{F}_{[0,2.4]}(\bigwedge_{i=1}^{5}\mu_A^i)) \wedge (\mathsf{G}_{[2.4,4.7]}\mu_B^2)$. Optimization converges in 30 iterations (238.9s), achieving 95% success with an average runtime of 242.4s over 20 trials. For 100 robots on similar tasks, planning completes in 971.5s with 100% success. (II) **Dynamic obstacles**: 5 robots in a $10\text{m} \times 10\text{m}$ workspace avoid a moving obstacle while reaching one of two regions and maintaining formation, i.e., $\phi = (\mathsf{F}_{[0,3.1]}(\mu_A^1 \vee \mu_A^2)) \wedge (\mathsf{G}_{[0,3.1]}\mu_B^1)$. Robots adapt paths depending on obstacle velocity, passing below it at 2m/s in Fig. 7, or detouring above at 6m/s in Fig. 1. In both cases, the optimization terminates within $9s$ with 100% success.

B. Hardware Experiment

1) System Description: Experiments are conducted in a $3.8m\times5.2m$ lab using Bitcraze Crazyflie 2.1 nano quadrotors of size $0.1m \times 0.1m$. An `OptiTrack` motion capture system operating at $120Hz$ tracks the quadrotor states. The captured data are sent to a central computer, which computes trajectories for all quadrotors. The trajectory is fitted to a seventh-order polynomial and tracked by a feedback controller based on [30]. The mission objective involves a fleet of three initially scattered quadrotors that must converge into a linear formation to navigate a narrow corridor. The target inter-agent distance is defined as $r_c = 0.6\,\text{m}$ with an

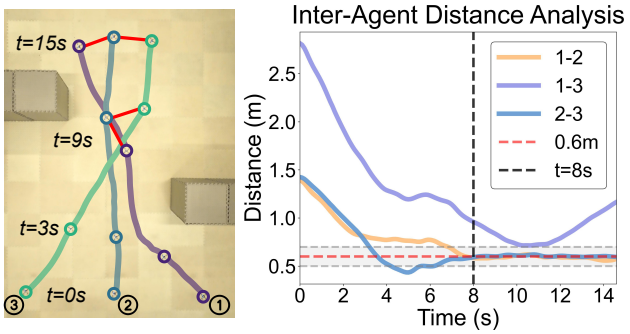

Fig. 8. Real-world quadcopter experiment: recorded trajectories (**Left**) and inter-agent distance analysis (**Right**)

allowable tolerance of $\pm 0.1\,\mathrm{m}$. The corresponding formula is $\phi = \mathsf{G}_{[8,15]}\mu_B$, which requires that the linear formation be maintained throughout the interval $t \in [8, 15]\,\mathrm{s}$.

2) Results: As shown in Fig. 1, feasible trajectories are planned in $2.91s$ and executed smoothly. The experimental trajectories obtained from the motion capture system are displayed on the left of Fig. 8, while the corresponding inter-agent distances along the trajectory are shown on the right. It can be observed that the quadrotors gradually converge from their dispersed initial positions. By $t = 8\,\mathrm{s}$, the fleet successfully establishes the desired linear formation and maintains this configuration thereafter. Notably, the agents coordinate their velocities and spatial offsets to deconflict their paths, effectively preventing potential collisions at the trajectory junction. The planned and executed paths align closely, which validates smoothness and dynamic feasibility.

VI. Conclusion

This work presented **CLOT**, a collaborative optimal transport framework for multi-robot motion planning under STL tasks. The method integrates sequential local optimization with hybrid search, leveraging parallelization and heuristics for efficiency. Future work will explore human interaction and additional motion constraints.

References

[1] S. M. LaValle, *Planning Algorithms*. Cambridge univ. press, 2006.

[2] R. Olfati-Saber, J. A. Fax, and R. M. Murray, "Consensus and cooperation in networked multi-agent systems," *Proceedings of the IEEE*, vol. 95, no. 1, pp. 215–233, 2007.

[3] M. Guo, J. Tumova, and D. V. Dimarogonas, "Communication-free multi-agent control under local temporal tasks and relative-distance constraints," *IEEE Transactions on Automatic Control*, vol. 61, no. 12, pp. 3948–3962, 2016.

[4] R. Herguedas, M. Aranda, G. López-Nicolás, C. Sagüés, and Y. Mezouar, "Multirobot control with double-integrator dynamics and control barrier functions for deformable object transport," in *IEEE International Conference on Robotics and Automation*, 2022, pp. 1485–1491.

[5] L. Sabattini, C. Secchi, N. Chopra, and A. Gasparri, "Distributed control of multirobot systems with global connectivity maintenance," *IEEE Transactions on Robotics*, vol. 29, no. 5, pp. 1326–1332, 2013.

[6] G. Antonelli, F. Arrichiello, F. Caccavale, and A. Marino, "Decentralized time-varying formation control for multi-robot systems," *The International Journal of Robotics Research*, vol. 33, no. 7, pp. 1029–1043, 2014.

[7] Y. Chen, S. Mitra, and S. Saha, "Decentralized task allocation in multi-robot systems with stl constraints," *IEEE Transactions on Robotics*, vol. 37, no. 4, pp. 1121–1136, 2021.

[8] L. Lindemann and D. V. Dimarogonas, *Formal Methods for Multi-Agent Feedback Control Systems*. MIT Press, 2025.

[9] J. Alonso-Mora, S. Baker, and D. Rus, "Multi-robot formation control and object transport in dynamic environments via constrained optimization," *The International Journal of Robotics Research*, vol. 36, no. 9, pp. 1000–1021, 2017.

[10] W. Hönig, S. Kiesel, A. Tinka, J. W. Durham, and N. Ayanian, "Persistent and robust execution of mapf schedules in warehouses," *IEEE Robotics and Automation Letters*, vol. 4, no. 2, pp. 1125–1131, 2019.

[11] G. Sharon, R. Stern, A. Felner, and N. R. Sturtevant, "Conflict-based search for optimal multi-agent pathfinding," *Artificial intelligence*, vol. 219, pp. 40–66, 2015.

[12] M. Guo and D. V. Dimarogonas, "Multi-agent plan reconfiguration under local ltl specifications," *The International Journal of Robotics Research*, vol. 34, no. 2, pp. 218–235, 2015.

[13] K. Leahy, Z. Serlin, C.-I. Vasile, A. Schoer, A. M. Jones, R. Tron, and C. Belta, "Scalable and robust algorithms for task-based coordination from high-level specifications (scratches)," *IEEE Transactions on Robotics*, vol. 38, no. 4, pp. 2516–2535, 2021.

[14] L. Lindemann, J. Nowak, L. Schönbächler, M. Guo, J. Tumova, and D. V. Dimarogonas, "Coupled multi-robot systems under linear temporal logic and signal temporal logic tasks," *IEEE Transactions on Control Systems Technology*, vol. 29, no. 2, pp. 858–865, 2019.

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

[16] Y. Liu, S. Sadraddini, and C. Belta, "Stlq: A quantitative signal temporal logic motion planning framework for multi-robot systems," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2022, pp. 7646–7653.

[17] V. Kurtz and H. Lin, "Mixed-integer programming for signal temporal logic with fewer binary variables," *IEEE Control Systems Letters*, vol. 6, pp. 2635–2640, 2022.

[18] K. Leung, V. Raman, and N. Ozay, "Distributed synthesis of multi-agent systems with temporal logic specifications," *IEEE Transactions on Control of Network Systems*, vol. 7, no. 1, pp. 490–500, 2020.

[19] D. Aksaray, Y. Wang, and C. Belta, "Online control synthesis for signal temporal logic tasks using learning-based methods," *Autonomous Robots*, vol. 46, pp. 243–260, 2022.

[20] A. T. Le, G. Chalvatzaki, A. Biess, and J. R. Peters, "Accelerating motion planning via optimal transport," *Advances in Neural Information Processing Systems*, vol. 36, pp. 78 453–78 482, 2023.

[21] M. Cuturi, "Sinkhorn distances: Lightspeed computation of optimal transport," *Advances in Neural Information Processing Systems*, vol. 26, pp. 2292–2300, 2013.

[22] N. Ratliff, J. A. Bagnell, and M. Zinkevich, "Learning to search: Functional gradient techniques for imitation learning," *Autonomous Robots*, vol. 27, no. 1, pp. 25–53, 2009.

[23] C. Chen, Y. Yang, M. Xu, Q. Liu, and et al., "Optimal transport based distributional soft actor critic," in *International Conference on Learning Representations*, 2020.

[24] A. T. Le, K. Hansel, J. Peters, and G. Chalvatzaki, "Hierarchical policy blending as optimal transport," in *Learning for Dynamics and Control Conference*. PMLR, 2023, pp. 797–812.

[25] G. E. Fainekos and G. J. Pappas, "Robustness of temporal logic specifications for continuous-time signals," *Theoretical Computer Science*, vol. 410, no. 42, pp. 4262–4291, 2009.

[26] J. V. Deshmukh, A. Donzé, S. Ghosh, X. Jin, G. Juniwal, and S. A. Seshia, "Robust online monitoring of signal temporal logic," *Formal Methods in System Design*, vol. 51, pp. 5–30, 2017.

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

[28] K. Leung, N. Aréchiga, and M. Pavone, "Backpropagation through signal temporal logic specifications: Infusing logical structure into gradient-based methods," *The International Journal of Robotics Research*, vol. 42, no. 6, pp. 356–370, 2023.

[29] G. Peyré, M. Cuturi *et al.*, "Computational optimal transport: With applications to data science," *Foundations and Trends® in Machine Learning*, vol. 11, no. 5-6, pp. 355–607, 2019.

[30] D. Mellinger and V. Kumar, "Minimum snap trajectory generation and control for quadrotors," in *IEEE International Conference on Robotics and Automation*, 2011, pp. 2520–2525.