Distributed Sampling-Based Model Predictive Control via Belief Propagation for Multi-Robot Formation Navigation

Chao Jiang, *Member, IEEE*

***Abstract*—Sampling-based stochastic optimal control has become an appealing robotic control framework due to its ability to handle complex and general forms of dynamics models and task specifications. Although sampling-based methods have been shown successful in a variety of single-robot control tasks, studies on their extension to multi-robot problems are limited. In this letter, we propose a distributed framework for sampling-based optimal control. The framework formulates multi-robot optimal control as probabilistic inference over graphical models and leverages belief propagation to achieve inference via distributed computation. We developed a distributed sampling-based model predictive control (MPC) algorithm based on the proposed framework, which obtains optimal controls via variational inference. The algorithm was validated in a multi-robot formation navigation problem. The simulation results show the efficacy of our proposed method with improved control performance over a gradient-based distributed MPC algorithm.**

***Index Terms*—Distributed robot systems, model predictive control, variational inference, optimization and optimal control.**

I. Introduction

COORDINATED control of multi-robot systems has been extensively studied for decades as one of the key algorithmic enablers for a broad range of multi-robot applications such as collaborative transportation, precision agriculture, and environment monitoring. Traditionally, multi-robot control problems were approached via heuristic-based behaviors, artificial potentials, and decentralized feedback control. Although such methods were successful in producing coordinated behaviors of multi-robot systems, they fall short of meeting various performance optimality and operational constraints in real-world deployment. As a result, optimization-based approaches have gained considerable attention due to their ability to exploit rigorous mathematical optimization framework to achieve performance and constraint specifications.

There have been a myriad of works on optimization-based multi-robot control, where control problems are framed as constrained optimization programs which are solved by, e.g., dynamic programming [1], sequential convex programming [2], or model predictive control (MPC) [3]. MPC is a noteworthy optimal control scheme where long time-horizon optimization problems are divided into short time-horizon sub-problems and individual sub-problems are solved in a receding horizon manner. Owing to its predictive and online re-planning abilities, MPC has become one of the most appealing robotic control approaches. The aforementioned optimization-based approaches are gradient-based, which leverage well established mathematical optimization tools to obtain solutions. On the downside, the dependence on gradients limits the applicability of those approaches in complex scenarios: as gradient-based approaches require computation of derivatives, they are not well-suited for non-smooth dynamics or cost functions. While theories have been developed for non-smooth problems, there is still a lack of practical numerical methods for different types of non-smoothness [4]. Moreover, as the complexity (e.g., non-convexity, local minima) of the functions increases, those approaches could become inefficient or even fail to find feasible solutions. Existing MPC methods using gradient-based optimization were mostly successful in problems that can be described with a smooth optimization landscape.

In recent years, sampling-based approaches have been proposed to address stochastic optimal control with complex models of system dynamics and uncertainties. Such approaches use random trajectory samples produced by forward prediction or simulation to synthesize optimal controls. Sampling-based approaches do not rely on computation of exact gradients of the models, and therefore accepts more general forms of dynamics and task specifications. Moreover, the stochastic forward search in sampling-based approaches provides a principled way to reason about uncertainty and leads to exploration that makes the algorithm less prone to undesirable local minima. Recent sampling-based methods for robotic control include path integral control [5], cross-entropy method [6], information-theoretic MPC [7], and variational inference MPC [8], [9], [10].

However, most of the prior works on sampling-based optimal control are concerned with single-robot problems. There is limited work on extending sampling-based approaches and providing a distributed framework to multi-robot systems. In [11], multi-robot optimal control was framed as graphical model inference and the solutions were obtained by a path integral method. Nonetheless, the method therein relied on the assumption that the global state of all robots is observable to each robot, which only stands valid in few scenarios. In [12], a multi-robot system was factorized into multiple sub-systems, each of which consisted of

Manuscript received 7 November 2023; accepted 19 February 2024. Date of publication 22 February 2024; date of current version 1 March 2024. This letter was recommended for publication by Associate Editor K. Leahy and Editor L. Pallottino upon evaluation of the reviewers' comments. The work of Chao Jiang was supported by the US National Science Foundation under Grant IIS-2024813.

The author is with the Department of Electrical Engineering and Computer Science, University of Wyoming, Laramie, WY 82071 USA (e-mail: cjiang1@uwyo.edu).

This letter has supplementary downloadable material available at https://doi.org/10.1109/LRA.2024.3368794, provided by the author.

Digital Object Identifier 10.1109/LRA.2024.3368794

a central robot and its neighbors. The central robot obtained the joint state and computed the joint control of the sub-system using a single-agent path integral algorithm. However, the method could not ensure consensus between sub-systems due to local observability and thus tended to settle on sub-optimal solutions for coordination. The work in [13] developed a decentralized path integral method where robots iteratively exchange local information to reach consensus. More recently, a stein variational belief propagation algorithm was introduced in [14], which solves distributed multi-robot control using particle belief propagation.

In this letter, we propose a distributed framework for sampling-based optimal control, which formulates multi-robot optimal control as probabilistic inference over graphical models and leverages belief propagation to achieve inference via distributed computation. The framework could lead to various distributed optimal control algorithms grounded on sampling-based stochastic optimization. We present a distributed sampling-based MPC algorithm based on the proposed framework and demonstrate the efficacy of our method in a multi-robot formation navigation problem which has a broad range of applications such as collaborative transportation, search and rescue, and environment exploration.

The main contribution of this letter is the distributed framework that extends the sampling-based optimization methods [6], [7], [8], [9], [10] to multi-robot problems. Unlike the work described in [11], our framework is fully distributed and only local information is needed by each robot. Compared to the path integral-based algorithms [12], [13] and recent works using belief propagation [14], [15], our framework offers the flexibility to choose the class of control/sampling policy and sample evaluation criterion, which are two major design components in sampling-based optimal control algorithms. This flexibility allows a variety of algorithms to be designed for different control tasks.

The remainder of this letter is organized as follows. Section II describes the multi-robot formation navigation problem. Section III presents the distributed framework for sampling-based optimal control based on belief propagation. Simulation and results are presented in Section IV. Finally, our work is concluded in Section V.

II. Problem Definition

A. Robot Motion Model

Consider a multi-robot team comprising N non-holonomic ground robots with two differential-drive wheels. The robot state $\boldsymbol{x}_i \triangleq [x_i, y_i, \theta_i]^T \in \mathbb{R}^3, \forall i \in \{1, \ldots, N\}$, where $\boldsymbol{p}_i \triangleq [x_i, y_i]^T \in \mathbb{R}^2$ is position and $\theta_i \in \mathbb{R}$ is orientation. Robot motion is controlled by the velocity of left and right wheels, denoted as $\boldsymbol{u}_i \triangleq [v_{il}, v_{ir}]^T \in \mathbb{R}^2$. Control magnitude is limited to $[-v_{\max}, v_{\max}]$ with $v_{\max}$ being the maximum speed. The discrete-time robot motion is then modeled by

$$\begin{aligned} x_{i,t+1} &= x_{i,t} + 0.5\Delta t(v_{ir,t} + v_{il,t}) \cos\theta_{i,t} \\ y_{i,t+1} &= y_{i,t} + 0.5\Delta t(v_{ir,t} + v_{il,t}) \sin\theta_{i,t} \\ \theta_{i,t+1} &= \theta_{i,t} + 0.5\Delta t(v_{ir,t} - v_{il,t})/l \end{aligned} \tag{1}$$

with t denoting time step and Δt denoting updating interval. l is the distance between left and right wheels. The motion model (1) represents a large class of wheeled robots.

B. Proximity Graph

The multi-robot team is modeled as an undirected graph $\mathcal{G} \triangleq (V, E_t)$, where $V \triangleq \{1, \ldots, N\}$ is the set of vertices representing robots and E_t is the set of edges representing communication connectivity between robots at time step t. Robot i and robot j are considered connected if the Euclidean distance between them is equal to or smaller than a given communication range limit $d_{\max}$. Then, the set of edges E_t at each time step is defined as

$$E_t \triangleq \{(i,j) \mid \|\boldsymbol{p}_{i,t} - \boldsymbol{p}_{j,t}\| \leq d_{\max}, \forall i, j \in V, i \neq j\}. \tag{2}$$

Note that E_t may vary over time as robots move around. At each time step, robot i and robot j are neighbors if they are connected by a valid edge, i.e., $(i,j) \in E_t$. We define the set of neighbors of robot i as $\mathcal{N}_{i,t} \triangleq \{j \mid (i,j) \in E_t\}$.

C. Collision and Obstacle Avoidance

Collision avoidance between two robots i and j is ensured by the following inequality at every time step t:

$$\|\boldsymbol{p}_{i,t} - \boldsymbol{p}_{j,t}\| \geq d_{\min}, \forall i \in V, \forall j \in \mathcal{N}_{i,t}, \tag{3}$$

where $d_{\min}$ is a minimum distance for safety. Likewise, each robot's position can not coincide with the space occupied by obstacles at all times, i.e., $\boldsymbol{p}_{i,t} \notin \mathcal{X}_o, \forall i \in V$, where $\mathcal{X}_o \in \mathbb{R}^2$ is the position space occupied by obstacles and is extended by a small margin to account for the body size of robots.

D. Navigation and Formation Preservation

The robots are tasked to navigate to a goal region in a desired formation while avoiding collisions with neighboring robots and obstacles. To guide navigation, a sequence of reference states over the task duration, $\{\boldsymbol{x}_{r,t}\}, \forall t \in [0, T]$, is planned before the navigation task. Each robot tracks the reference by minimizing $\|\boldsymbol{x}_{i,t} - \boldsymbol{x}_{r,t}\|, \forall t \in [0, T]$. Given an environment map, a reference trajectory can be obtained by various global motion planners such as differential dynamic programming used in [1].

The desired formation is specified by the relative position $\Delta\boldsymbol{p}_{ij}$ between robot i and its neighbor j. The following equality holds when the desired formation is preserved:

$$g_{ij}(\boldsymbol{p}_{i,t}, \boldsymbol{p}_{j,t}) \triangleq \boldsymbol{p}_{i,t} - \boldsymbol{p}_{j,t} - \Delta\boldsymbol{p}_{ij} = 0. \tag{4}$$

E. Model Predictive Formation Control

The multi-robot formation navigation is formulated as finite-horizon (of length M) trajectory optimization with the following cost function:

$$\min_{\{\boldsymbol{\tau}_i\}_{i=1}^N} J \triangleq \sum_{i=1}^{N} \left(\sum_{k=1}^{M-1} w_s \|\boldsymbol{x}_{i,t+k} - \boldsymbol{x}_{r,t+k}\|^2 + w_u \|\boldsymbol{u}_{i,t+k-1}\|^2 \right) \tag{5}$$

where w_s and w_u are the weights for reference tracking error and control effort cost, respectively. The optimization (5) is subject to collision avoidance constraint (3), obstacle avoidance constraint, i.e., $\boldsymbol{p}_{i,t} \notin \mathcal{X}_o$, formation configuration (4), and control constraint $\boldsymbol{u}_{i,t} \in [-v_{\max}, v_{\max}]$.

At each time step t, MPC uses the motion model (1) and the current robot state $\boldsymbol{x}_{i,t}$ to solve problem (5). The solution is optimized trajectories $\{\boldsymbol{\tau}_i\}_{i=1}^N$ of all robots. A trajectory $\boldsymbol{\tau}_i$ is defined as a state-control sequence over a prediction horizon M, i.e., $\boldsymbol{\tau}_i \triangleq \{\boldsymbol{x}_{i,t+k}, \boldsymbol{u}_{i,t+k}\}_{k=0}^{M-1}$, starting at time step t. Each

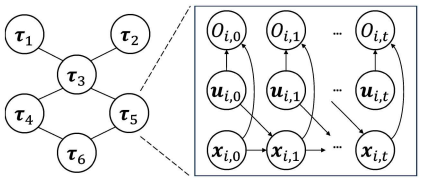

Fig. 1. Graphical model of multi-robot control. Left: A multi-robot team modeled as a Markov random field. Right: A single robot node factorized into a graph comprising a sequence of state, control, and optimality variables.

robot executes the control of the first step, i.e., $\boldsymbol{u}_{i,t}$, to move one step forward and starts a new MPC cycle. MPC is performed repeatedly in a receding horizon manner over the task duration $[0, T]$. In the next section, we present a distributed framework for sampling-based optimal control, which solves (5) via distributed probabilistic inference.

III. Method

We view the multi-robot predictive control problem (5) as an instance of distributed probabilistic inference and approach the problem using the control as inference framework [16]. The goal of the inference problem is to find the probability distribution over optimal trajectory and thus optimal control for each robot. We first model the multi-robot team as a probabilistic graphical model named Markov random field (MRF) [17] and then employ belief propagation for distributed inference via local message-passing. Lastly, a sampling-based MPC algorithm is developed to obtain optimal control of each robot.

A. Markov Random Field and Belief Propagation

1) Markov Random Field: A multi-robot MRF, as shown on the left side of Fig. 1, factorizes the joint trajectories of all robots into an undirected connected graph $\mathcal{G} \triangleq (V, E)$ with each node $i \in V$ representing a robot's trajectory $\boldsymbol{\tau}_i$ as a hidden random variable and each edge in E representing the dependence between two neighboring nodes. The definition of edges here is the same as that described in (2). Each node also includes an observable variable $\boldsymbol{O}_i$ (omitted in the MRF graph) representing the robot's local observation. Let $\mathcal{T} \triangleq \{\boldsymbol{\tau}_i \mid i \in V\}$ denotes the set of all hidden trajectory variables of the multi-robot MRF and $\mathcal{O} \triangleq \{\boldsymbol{O}_i \mid i \in V\}$ be the set of local observations obtained by all robots. The joint probability of the trajectories in $\mathcal{T}$ is expressed as the following factorization:

$$Pr(\mathcal{T}) \propto \prod_{(i,j)\in E} \psi_{j,i}(\boldsymbol{\tau}_j, \boldsymbol{\tau}_i) \prod_{i\in V} \phi_i(\boldsymbol{\tau}_i, \boldsymbol{O}_i) \tag{6}$$

where $\psi_{j,i}(\boldsymbol{\tau}_j, \boldsymbol{\tau}_i)$ is a pairwise potential, or compatibility, between each pair of neighboring robots i and j, which describes how likely the robots could take their respective trajectories $\boldsymbol{\tau}_i$ and $\boldsymbol{\tau}_j$; $\phi_i(\boldsymbol{\tau}_i, \boldsymbol{O}_i)$ is a unary potential describing the likelihood of a robot's own trajectory $\boldsymbol{\tau}_i$ given its local observation $\boldsymbol{O}_i$. The design of pairwise and unary potentials is problem-specific and depends on the objectives of a given control problem. The design details for our control problem are presented in Section III-C.

2) Belief Propagation: The factored structure of the multi-robot MRF described in (6) allows the global inference problem to be solved in a distributed way using BP method [18]. Specifically, each node computes its own marginal posterior distribution, denoted as $p(\boldsymbol{\tau}_i|\mathcal{O})$, given the robot's own observation and information obtained via local communication with neighbors. The marginal posterior distribution, or belief, of robot trajectory at each node is estimated by

$$p(\boldsymbol{\tau}_i|\mathcal{O}) \propto \phi_i(\boldsymbol{\tau}_i, \boldsymbol{O}_i) \prod_{j\in\mathcal{N}_i} m_{j,i}(\boldsymbol{\tau}_i) \tag{7}$$

where $m_{j,i}(\boldsymbol{\tau}_i)$ is a message sent from robot j to robot i and is defined as

$$m_{j,i}(\boldsymbol{\tau}_i) = \int_{\boldsymbol{\tau}_j} \phi_j(\boldsymbol{\tau}_j, \boldsymbol{O}_j)\psi_{j,i}(\boldsymbol{\tau}_j, \boldsymbol{\tau}_i) \prod_{k\in\mathcal{N}_j\setminus i} m_{k,j}(\boldsymbol{\tau}_j)d\boldsymbol{\tau}_j. \tag{8}$$

Message $m_{j,i}(\boldsymbol{\tau}_i)$ carries robot j's belief about the likelihood of robot i's trajectory $\boldsymbol{\tau}_i$ as a product of robot j's unary potential, the pairwise potential between robots i and j, and the messages received by robot j from its neighbors excluding robot i, with robot j's trajectory $\boldsymbol{\tau}_j$ marginalized out by integrating over all values of $\boldsymbol{\tau}_j$.

In BP algorithms, messages computed by (8) are passed between neighboring robots via local communication and each robot updates its belief $p(\boldsymbol{\tau}_i|\mathcal{O})$ based on (7) continuously until the algorithm converges to the true marginal posterior distributions. The original BP algorithm guarantees convergence to exact solutions on tree graphs [18]. In practice, however, multi-robot team can form graphs that contain cycles under the definition of proximity graph in Section II-B. Therefore, the loopy BP algorithm [19] is used, which iterates between (8) and (7) to find an approximation of $p(\boldsymbol{\tau}_i|\mathcal{O})$. Loopy BP has been shown effective for graphs with cycles in empirical studies [19], [20].

B. Robot Control as Inference

In this subsection, we formulate the multi-robot optimal control as a distributed probabilistic inference problem that is solved by the BP framework presented in Section III-A.

1) Graphical Representation of Predictive Control: Under the control as inference framework [16], predictive control of each robot is formulated as a directed graphical model as shown on the right side of Fig. 1. The sequence of state and control variables, $\boldsymbol{x}_{i,t}$ and $\boldsymbol{u}_{i,t}$, are laid out as nodes. The nodes are connected by directed edges pointing from a parent node to a child node that is dependent on the parent node. The dependence between state and control variables is described by state transition probability $p(\boldsymbol{x}_{i,t+1}|\boldsymbol{x}_{i,t}, \boldsymbol{u}_{i,t})$, which is a probabilistic generalization of the robot motion model (1). The graphical model also includes an observable variable $O_{i,t}$ serving as a notion of optimality for state-control pair $(\boldsymbol{x}_{i,t}, \boldsymbol{u}_{i,t})$. Specifically, $O_{i,t}$ is defined as a binary random variable with $O_{i,t} = 1$ representing that the state-control pair is optimal and $O_{i,t} = 0$ representing that it is not optimal. It should be noted that the optimality $O_{i,t}$, as a robot's local observation, is defined with regard to each robot's private control objectives (i.e., reference tracking and obstacle avoidance) and does not account for inter-robot control objectives (i.e., desired formation and collision avoidance). We use $p(O_{i,t} = 1|\boldsymbol{x}_{i,t}, \boldsymbol{u}_{i,t})$ to denote the likelihood of $(\boldsymbol{x}_{i,t}, \boldsymbol{u}_{i,t})$ being optimal with regard to private objectives (hereafter referred to as private optimality likelihood and $O_{i,t} = 1$ is simplified as $O_{i,t}$ for notational brevity). The

design of the optimality likelihood essentially influences the behavior of inference algorithm to search for the distribution over optimal control. More details are presented in Section III-C.

2) Distributions of Optimal Trajectory and Control: With the graphical representation introduced in Section III-B1, we can factorize the unary potential $\phi_i(\boldsymbol{\tau}_i, \boldsymbol{O}_i)$ in (6) as

$$\phi_i(\boldsymbol{\tau}_i, \boldsymbol{O}_i) = p(\boldsymbol{O}_i|\boldsymbol{\tau}_i)p(\boldsymbol{\tau}_i) = p(\boldsymbol{O}_i|\boldsymbol{\tau}_i)p(\boldsymbol{X}_i|\boldsymbol{U}_i)p(\boldsymbol{U}_i) \tag{9}$$

where $p(\boldsymbol{O}_i|\boldsymbol{\tau}_i) = \prod_{t=0}^{M-1} p(O_{i,t}|\boldsymbol{x}_{i,t}, \boldsymbol{u}_{i,t})$ with $\boldsymbol{O}_i \triangleq \{O_{i,0}, \ldots, O_{i,M-1}\}$ defined as a sequence of private optimality variables of trajectory $\boldsymbol{\tau}_i$. The trajectory distribution $p(\boldsymbol{\tau}_i)$ is factorized as $p(\boldsymbol{X}_i|\boldsymbol{U}_i)p(\boldsymbol{U}_i)$ with $p(\boldsymbol{X}_i|\boldsymbol{U}_i) = p(\boldsymbol{x}_0)\prod_{t=0}^{M-1} p(\boldsymbol{x}_{i,t+1}|\boldsymbol{x}_{i,t}, \boldsymbol{u}_{i,t})$ and $p(\boldsymbol{U}_i) = \prod_{t=0}^{M-1} p(\boldsymbol{u}_{i,t})$ being the distribution over control sequence $\boldsymbol{U}_i \triangleq \{\boldsymbol{u}_{i,0}, \ldots, \boldsymbol{u}_{i,M-1}\}$. The factor $p(\boldsymbol{u}_{i,t})$ is a prior distribution over control for each time step (e.g., a uniform distribution if no prior knowledge is available or a Gaussian as in this letter). (9) implies that given a prior $p(\boldsymbol{U}_i)$, the probability for an observed trajectory is proportional to the product of state transition probability $p(\boldsymbol{x}_{i,t+1}|\boldsymbol{x}_{i,t}, \boldsymbol{u}_{i,t})$, private optimality likelihood $p(O_{i,t}|\boldsymbol{x}_{i,t}, \boldsymbol{u}_{i,t})$, and probability of initial state $p(\boldsymbol{x}_0)$.

Substituting (9) into (7), we specify the marginal posterior $p(\boldsymbol{\tau}_i|\mathcal{O})$ that describes the distribution over optimal trajectories of each robot

$$p(\boldsymbol{\tau}_i|\mathcal{O}) \propto p(\boldsymbol{X}_i|\boldsymbol{U}_i)p(\boldsymbol{O}_i|\boldsymbol{\tau}_i)p(\boldsymbol{U}_i)\prod_{j\in\mathcal{N}_i} m_{j,i}(\boldsymbol{\tau}_i) \tag{10}$$

The marginal posterior (10) is considered a target trajectory distribution that we aim to reproduce by using an optimal control policy. To this end, we first describe the trajectory distribution given by an arbitrary control policy as

$$\begin{aligned} q(\boldsymbol{\tau}_i) &= p(\boldsymbol{x}_0)\prod_{t=0}^{M-1} p(\boldsymbol{x}_{i,t+1}|\boldsymbol{x}_{i,t}, \boldsymbol{u}_{i,t})q(\boldsymbol{u}_{i,t}) \\ &= p(\boldsymbol{X}_i|\boldsymbol{U}_i)q(\boldsymbol{U}_i) \end{aligned} \tag{11}$$

with $q(\boldsymbol{U}_i) = \prod_{t=0}^{M-1} q(\boldsymbol{u}_{i,t})$ denoting the distribution of control sequence induced by the control policy. Then the optimal control problem amounts to finding an optimal distribution $q(\boldsymbol{U}_i)$ so that the resulting trajectory distribution (11) best matches the marginal posterior distribution (10). Finding $q(\boldsymbol{U}_i)$ can be approached by probabilistic inference. We employ variational inference (VI) methods to solve the inference problem in a computationally efficient way.

3) Variational Inference: Variational inference uses a candidate distribution to estimate a target distribution by minimizing the Kullback-Leibler (KL) divergence between the candidate and target distributions, thus transforming the inference problem to an optimization problem. Given a candidate distribution $q(\boldsymbol{U}_i)$ and the resulting trajectory distribution (11), the optimization is formulated as

$$q^*(\boldsymbol{\tau}_i) = \arg\min_{q(\boldsymbol{\tau}_i)} D_{KL}\left(q(\boldsymbol{\tau}_i)\|p(\boldsymbol{\tau}_i|\mathcal{O})\right) \tag{12}$$

Using the definition of KL divergence and substituting (10) and (11), the KL divergence is further written as

$$\begin{aligned} D_{KL}\left(q(\boldsymbol{\tau}_i)\|p(\boldsymbol{\tau}_i|\mathcal{O})\right) &= \int q(\boldsymbol{\tau}_i)\log\frac{q(\boldsymbol{\tau}_i)}{p(\boldsymbol{\tau}_i|\mathcal{O})}d\boldsymbol{\tau}_i \\ &= \int q(\boldsymbol{\tau}_i)\log\frac{p(\boldsymbol{X}_i|\boldsymbol{U}_i)q(\boldsymbol{U}_i)}{p(\boldsymbol{X}_i|\boldsymbol{U}_i)p(\boldsymbol{O}_i|\boldsymbol{\tau}_i)p(\boldsymbol{U}_i)\prod_{j\in\mathcal{N}_i} m_{j,i}(\boldsymbol{\tau}_i)}d\boldsymbol{\tau}_i \\ &= \int q(\boldsymbol{\tau}_i)\log\frac{q(\boldsymbol{U}_i)}{p(\boldsymbol{O}_i|\boldsymbol{\tau}_i)p(\boldsymbol{U}_i)\prod_{j\in\mathcal{N}_i} m_{j,i}(\boldsymbol{\tau}_i)}d\boldsymbol{\tau}_i \\ &= \mathbb{E}_{q(\boldsymbol{\tau}_i)}\left[\log\frac{q(\boldsymbol{U}_i)}{p(\boldsymbol{U}_i)} - \log p(\boldsymbol{O}_i|\boldsymbol{\tau}_i) - \log\prod_{j\in\mathcal{N}_i} m_{j,i}(\boldsymbol{\tau}_i)\right] \\ &= -\mathbb{E}_{q(\boldsymbol{\tau}_i)}\left[\log p(\boldsymbol{O}_i|\boldsymbol{\tau}_i) + \sum_{j\in\mathcal{N}_i}\log m_{j,i}(\boldsymbol{\tau}_i)\right] + D_{KL}\left(q\|p\right) \end{aligned} \tag{13}$$

The optimization of (13) aims to maximize in expectation the log-likelihood of private optimality with the first term and the log-likelihood of trajectories given all neighbors' believes with the second term. The two terms in the expectation can be viewed as the overall (both private and inter-robot) control objective for each robot. The third term is a regularization that penalizes for the difference between the target $q(\boldsymbol{U}_i)$ and the prior $p(\boldsymbol{U}_i)$. It's worth mentioning that (13) takes the form of the variational lower bound, or variational objective, which is commonly seen in single-agent stochastic optimal control [8], [9], [10], while our formulation (13) includes an additional term that encapsulates the BP messages received from neighbors to achieve inter-robot coordination.

The objective (13) represents a general and novel formulation of stochastic optimization for distributed dynamical systems. In this letter, the distributed control for multi-robot formation navigation is solved as an instance of (13) with specific design of the private optimality likelihood $p(\boldsymbol{O}_i|\boldsymbol{\tau}_i)$, the BP message $m_{j,i}(\boldsymbol{\tau}_i)$, and the distribution family for $q(\boldsymbol{U}_i)$. We propose a distributed VI-MPC algorithm that employs particle belief propagation [21] to approximately compute the message in (13) and a sampling-based optimization algorithm to solve (13) for $q(\boldsymbol{U}_i)$.

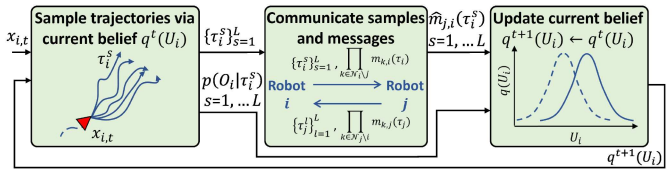

Fig. 2. Overview of the distributed VI-MPC algorithm deployed on robot i.

C. Distributed VI-MPC via Particle BP

The idea of distributed VI-MPC algorithm is to use importance sampling to solve (13) iteratively. An overview of the algorithm running on each robot is shown in Fig. 2. In each MPC cycle, the algorithm first samples L trajectories $\{\boldsymbol{\tau}_i^s\}_{s=1}^L$ using current belief $q^t(\boldsymbol{U}_i)$ and evaluates each sample by $p(\boldsymbol{O}_i|\boldsymbol{\tau}_i^s)$, then communicates trajectory samples with neighbors and updates BP messages $\hat{m}_{j,i}(\boldsymbol{\tau}_i^s)$ for each sample s, and finally updates current belief as $q^{t+1}(\boldsymbol{U}_i)$. Next, we explain the details of the algorithm.

1) Design of Optimality Likelihood: The private optimality likelihood at each time step is defined as an exponential family

distribution, i.e.,

$$p(O_{i,t}|\boldsymbol{x}_{i,t},\boldsymbol{u}_{i,t}) = \exp\left(-\frac{1}{\lambda_u}(C_p(\boldsymbol{x}_{i,t},\boldsymbol{u}_{i,t}) + C_o(\boldsymbol{x}_{i,t}))\right) \tag{14}$$

where $C_p(\boldsymbol{x}_{i,t},\boldsymbol{u}_{i,t})$ is the cost related to reference tracking and control effort and is given by $C_p(\boldsymbol{x}_{i,t},\boldsymbol{u}_{i,t}) = \frac{1}{2}(w_s\|\boldsymbol{x}_{i,t} - \boldsymbol{x}_{r,t}\|^2 + w_u\|\boldsymbol{u}_{i,t}\|^2)$ with w_s and w_u being the weights. $C_o(\boldsymbol{x}_{i,t})$ is the obstacle collision cost which evaluates to a positive value (e.g., 10 was used in the numerical example presented in Section IV) if $\boldsymbol{p}_{i,t} \in \mathcal{X}_o$ and 0 otherwise. The cost value should be chosen to prioritize collision avoidance over reference tracking and formation objectives. Note that $C_o(\boldsymbol{x}_{i,t})$ is non-differentiable as sampling-based optimization methods do not require differentiability of cost functions (and dynamics functions) as opposed to gradient-based methods. One can exploit this advantage to allow more flexible representations of environment and control objective specifications. The parameter λ_u in (14) is known as the inverse temperature of the exponential family distribution, which determines the extend to which optimality is sought by an optimization algorithm.

2) Design of BP Potentials: The unary potential defined in (9) describes the likelihood of a robot's trajectory given its observation of private optimality. With the private optimality likelihood given in (14), the unary potential $\phi_i(\boldsymbol{\tau}_i, \boldsymbol{O}_i)$ is specified by substituting (14) into (9). For deterministic dynamics as considered in this letter, $p(\boldsymbol{x}_{i,t+1}|\boldsymbol{x}_{i,t},\boldsymbol{u}_{i,t})$ and $p(\boldsymbol{x}_0)$ in (9) evaluate to one, and it follows that

$$\begin{aligned}\phi_i(\boldsymbol{\tau}_i,\boldsymbol{O}_i) &= \prod_{t=0}^{M-1} p(O_{i,t}|\boldsymbol{x}_{i,t},\boldsymbol{u}_{i,t})p(\boldsymbol{u}_{i,t}) \\ &= \exp\left(-\frac{1}{\lambda_u}\sum_{t=0}^{M-1}(C_p(\boldsymbol{x}_{i,t}) + C_o(\boldsymbol{x}_{i,t}))\right)p(\boldsymbol{U}_i). \end{aligned} \tag{15}$$

The pairwise potential describes the joint likelihood of two neighboring robots' trajectories. Intuitively, two trajectories achieving inter-robot control objectives (i.e., desired formation and collision avoidance) should be given highest probability. Therefore, we define the pairwise potential as an exponential family distribution, i.e.,

$$\psi_{j,i}(\boldsymbol{\tau}_j,\boldsymbol{\tau}_i) = \exp\left(-\frac{1}{\lambda_p}\sum_{t=0}^{M-1}(C_l(\boldsymbol{x}_{j,t},\boldsymbol{x}_{i,t}) + C_f(\boldsymbol{x}_{j,t},\boldsymbol{x}_{i,t}))\right) \tag{16}$$

where $C_l(\boldsymbol{x}_{j,t},\boldsymbol{x}_{i,t})$ is the cost related to inter-robot collision and is given by ([14]) $C_l = w_l(1 - (\|\boldsymbol{p}_{i,t} - \boldsymbol{p}_{j,t}\|/d_{\min})^\beta)$ if $\|\boldsymbol{p}_{i,t} - \boldsymbol{p}_{j,t}\| \le d_{\min}$ and $C_l = 0$ otherwise. w_l is the weight. This function imposes penalty when the distance between two robots is smaller than $d_{\min}$ and the cost increases at a rate β. $C_f(\boldsymbol{x}_{j,t},\boldsymbol{x}_{i,t}) = \frac{1}{2}w_f\|\boldsymbol{p}_{i,t} - \boldsymbol{p}_{j,t} - \Delta\boldsymbol{p}_{ij}\|^2$ is the formation-dependent cost, where w_f is the weight. The parameter λ_p in (16) is the inverse temperature.

3) Particle Belief Propagation: The computation of BP message (8) is intractable due to the integral over continuous domain of trajectory variable. We thus resort to particle belief propagation [21] which uses Monte-Carlo samples to approximately compute messages. Suppose that robot $j, \forall j \in \mathcal{N}_i$, uses a proposal distribution $W(\boldsymbol{\tau}_j)$ to sample L trajectories, then (8) equals an importance-adjusted expectation:

$$m_{j,i}(\boldsymbol{\tau}_i) = \mathbb{E}_{\boldsymbol{\tau}_j\sim W}\left[\frac{\phi_j(\boldsymbol{\tau}_j,\boldsymbol{O}_j)}{W(\boldsymbol{\tau}_j)}\psi_{j,i}(\boldsymbol{\tau}_j,\boldsymbol{\tau}_i)\prod_{k\in\mathcal{N}_j\setminus i} m_{k,j}(\boldsymbol{\tau}_j)\right]. \tag{17}$$

The expectation can be estimated empirically as the mean of L trajectory samples. Specifically, the message defined on the s-th trajectory sample of robot i is estimated using importance-sampling, i.e.,

$$\hat{m}_{j,i}(\boldsymbol{\tau}_i^s) = \frac{1}{L}\sum_{l=1}^{L}\frac{\phi_j(\boldsymbol{\tau}_j^l,\boldsymbol{O}_j^l)}{W(\boldsymbol{\tau}_j^l)}\psi_{j,i}(\boldsymbol{\tau}_j^l,\boldsymbol{\tau}_i^s)\prod_{k\in\mathcal{N}_j\setminus i} m_{k,j}(\boldsymbol{\tau}_j^l). \tag{18}$$

Since the continuous trajectory space is represented by a finite set of samples, the message (8) is now defined on each trajectory rather than the entire trajectory space.

The estimated message $\hat{m}_{j,i}(\boldsymbol{\tau}_i^s)$ is then used to solve (13) to find the best estimation of the marginal posterior distribution $p(\boldsymbol{\tau}_i|\mathcal{O})$. As mentioned in Section III-A, since the graph formed by multi-robot team likely contains cycles, we apply loopy BP algorithm to solve for $p(\boldsymbol{\tau}_i|\mathcal{O})$. Loopy BP algorithm [19] is an iterative procedure that alternately updates message $m_{j,i}(\boldsymbol{\tau}_i)$ via (18) and marginal posterior distribution $p(\boldsymbol{\tau}_i|\mathcal{O})$ by optimizing (13). When updating messages via (18) at each iteration, the robot's current belief $q(\boldsymbol{\tau}_i)$ (i.e., the solution to (13)) is used as the proposal distribution $W(\boldsymbol{\tau}_j)$ to generate trajectory samples. The current belief $q(\boldsymbol{\tau}_i)$ also serves as the prior $p(\boldsymbol{\tau}_i)$ for computing the unary potential $\phi_j(\boldsymbol{\tau}_j^l,\boldsymbol{O}_j^l)$ in (18). Note that the samples used for message computation are the same as the ones used to update the current belief $q(\boldsymbol{\tau}_i)$ through the sampling-based stochastic optimization presented in Section III-C4. This enables a natural integration of particle belief propagation and sampling-based stochastic optimization.

4) Stochastic Optimization via Sampling: Given the private optimality likelihood (14) and BP message (18), the minimization of (13) has a closed-form solution (known as the Gibbs distribution) as shown in prior works [7], [8], [9], [10]:

$$q^*(\boldsymbol{\tau}_i) = q^*(\boldsymbol{U}_i) = \frac{\exp\left(-C(\boldsymbol{\tau}_i)\right)p(\boldsymbol{U}_i)}{\mathbb{E}_{\boldsymbol{U}_i\sim p(\boldsymbol{U}_i)}\left[\exp\left(-C(\boldsymbol{\tau}_i)\right)\right]} \tag{19}$$

where $C(\boldsymbol{\tau}_i) = -\log p(\boldsymbol{O}_i|\boldsymbol{\tau}_i) - \sum_{j\in\mathcal{N}_i}\log\hat{m}_{j,i}(\boldsymbol{\tau}_i)$ represents the trajectory cost combining both private and inter-robot control objectives; $p(\boldsymbol{U}_i)$ is the prior as in the factorization (9). The first equality in (19) holds for deterministic dynamics since the factor $p(\boldsymbol{X}_i|\boldsymbol{U}_i)$ evaluates to one. Since the computation of the closed-form solution (19) is intractable, an iterative importance sampling procedure is applied to search for $q^*(\boldsymbol{U}_i)$ based on Monte-Carlo estimate.

At each iteration m of the importance sampling, the prior $p(\boldsymbol{U}_i)$ in (19) is set as the current distribution $q^m(\boldsymbol{U}_i) = \prod_{t=0}^{M-1} q^m(\boldsymbol{u}_{i,t})$ from which samples of control sequence $\boldsymbol{U}_i$ are drawn. Then, the resulting trajectory samples (obtained using the known motion model (1)) are evaluated with respect to the trajectory cost $C(\boldsymbol{\tau}_i)$ to update the current sampling distribution $q^m(\boldsymbol{U}_i)$. The update law is given by

$$q^{m+1}(\boldsymbol{U}_i) = \frac{\exp\left(-C(\boldsymbol{\tau}_i)\right)q^m(\boldsymbol{U}_i)}{\mathbb{E}_{\boldsymbol{U}_i\sim q^m(\boldsymbol{U}_i)}\left[\exp\left(-C(\boldsymbol{\tau}_i)\right)\right]}. \tag{20}$$

We set the sampling distribution $q^m(\boldsymbol{u}_{i,t})$ as a Gaussian distribution and initialize it with a zero-mean and variance $\Sigma = \sigma^2 \boldsymbol{I}_{2\times 2}$ at $m = 0$ with σ being a fixed standard deviation. By taking the expectation of both sides of (20), we obtain the update law for the mean of the Gaussian:

$$\overline{\boldsymbol{U}}_i^{m+1} = \frac{\mathbb{E}_{\boldsymbol{U}_i\sim q^m(\boldsymbol{U}_i)}\left[\exp\left(-C(\boldsymbol{\tau}_i)\right)\boldsymbol{U}_i\right]}{\mathbb{E}_{\boldsymbol{U}_i\sim q^m(\boldsymbol{U}_i)}\left[\exp\left(-C(\boldsymbol{\tau}_i)\right)\right]}$$
$$= \sum_{s=1}^{L} \frac{\exp\left(-C(\boldsymbol{\tau}_i^s)\right)}{\sum_{s'=1}^{L}\exp\left(-C(\boldsymbol{\tau}_i^{s'})\right)}\boldsymbol{U}_i^s = \sum_{s=1}^{L}\mathcal{W}(\boldsymbol{\tau}_i^s)\boldsymbol{U}_i^s \quad (21)$$

where L is the number of samples drawn from the current sampling distribution $q^m(\boldsymbol{U}_i)$ and $\mathcal{W}(\boldsymbol{\tau}_i^s)$ denotes the normalized importance weight of each sample s based on the trajectory cost. The update law (21) suggests that the mean is updated as the sum of the current control samples weighted by their importance $\mathcal{W}(\boldsymbol{\tau}_i^s)$. The importance sampling is repeated to improve the Monte-Carlo estimate of $q^*(\boldsymbol{U}_i)$ by using a more accurate sampling distribution. The mean $\overline{\boldsymbol{U}}_i^m \triangleq \{\overline{\boldsymbol{u}}_{i,0}^m, \ldots, \overline{\boldsymbol{u}}_{i,M-1}^m\}$ obtained by (21) serves as the solution of robot control at current iteration m.

5) The Overall Algorithm: We now frame the particle BP algorithm (Section III-C3) with the sampling-based stochastic optimization (Section III-C4) into the MPC scheme introduced in Section II-E, resulting in our proposed distributed VI-MPC algorithm. In our MPC scheme, iterations of BP algorithm and importance sampling for belief update are split into time steps over the task duration $[0, T]$. That is, at each time step, the BP message update (18) and the belief update (21) are performed once. The updated message and belief are carried over to the next time step to perform a new iteration.

The algorithm deployed on each robot is summarized in Algorithm 1. At $t = 0$, each robot starts with an initial state $\boldsymbol{x}_{i,0}$ and its belief $q^0(\boldsymbol{U}_i)$ is initialized as a Gaussian with a zero-mean and variance Σ. All messages $m_{j,i}(\cdot)$ from neighbors are initialized to one. At each time step t, each robot obtains its current state $\boldsymbol{x}_{i,t}$ (line 2) and samples L trajectory $\{\boldsymbol{\tau}_i^s\}_{s=1}^L$ from the current belief $q^t(\boldsymbol{U}_i)$ (lines 3-6). Each robot exchanges trajectory samples and messages with its neighbors (lines 8–9), estimates message $\hat{m}_{j,i}(\boldsymbol{\tau}_i^s)$ using (18) for each sample $\boldsymbol{\tau}_i^s$ and saves the results (lines 10-13). The cost and importance weight are computed for each sample $\boldsymbol{\tau}_i^s$ (lines 15–17) and the mean $\overline{\boldsymbol{U}}_{i,t}$ of the current belief $q^t(\boldsymbol{U}_i)$ is updated using the update law (21) (line 18). Each robot executes the first control action $\overline{\boldsymbol{u}}_{i,t}$ from the updated mean $\overline{\boldsymbol{U}}_{i,t}$ (line 19). The unexecuted control actions in $\overline{\boldsymbol{U}}_{i,t}$ are used to construct the mean of the belief $q^{t+1}(\boldsymbol{U}_i)$ for the next time step (line 20).

6) Connections and Comparisons to Related Works: The update law (21) resembles the MPPI algorithm in [7], but our framework allows for different choices for $p(\boldsymbol{O}_i|\boldsymbol{\tau}_i)$ and $q(\boldsymbol{U}_i)$ (e.g., if we represent $q(\boldsymbol{U}_i)$ by a set of particles, an algorithm similar to [14] could be developed). In contrast, the related works [12], [13] are based on the path integral control that synthesizes optimal controls via forward sampling of an uncontrolled diffusion process driven by Brownian noise. Thus, the sampling distribution is limited to the noise distribution. Also, the theory relies on an exponential transformation of the value function of an optimal control problem, limiting the evaluation of sampled trajectories (similar to the notion of $p(\boldsymbol{O}_i|\boldsymbol{\tau}_i)$ in our method) to the exponential family. The method in [15] uses a Gaussian graphical model and requires an analytical dynamics model. Our method could incorporate a general dynamics model (e.g., neural networks) for trajectory sampling.

Algorithm 1: Distributed VI-MPC via Particle BP.

Input : Initial state $\boldsymbol{x}_{i,0}$, initial sampling distribution $q^0(\boldsymbol{U}_i)$ with a zero-mean and variance Σ, and initial messages $m_{j,i}(\cdot), \forall j \in \mathcal{N}_i$ set to one;
Output: Optimized robot trajectory $\{\boldsymbol{x}_{i,t}, \boldsymbol{u}_{i,t}\}_{t=0}^T$;
1 **for** *time step* $t = 0$ *to* T **do**
2 Get current state $\boldsymbol{x}_{i,t}$;
3 **for** *each sample* $s \in \{1, ..., L\}$ **do**
4 Sample control $\boldsymbol{U}_i^s = \{\boldsymbol{u}_{i,t}^s, ..., \boldsymbol{u}_{i,t+M-1}^s\}$ from current distribution $q^t(\boldsymbol{U}_i)$;
5 Compute sample trajectories $\boldsymbol{\tau}_i^s$ using robot motion model (1);
6 **end**
7 **for** *each neighbor* $j \in \mathcal{N}_i$ **do**
8 Send L trajectory samples $\{\boldsymbol{\tau}_i^s\}_{s=1}^L$ and products $\prod_{k\in\mathcal{N}_i\setminus j} m_{k,i}(\boldsymbol{\tau}_i^s)$, $\forall s \in \{1, ..., L\}$ to robot j;
9 Receive L trajectory samples $\{\boldsymbol{\tau}_j^l\}_{l=1}^L$ and products $\prod_{k\in\mathcal{N}_j\setminus i} m_{k,j}(\boldsymbol{\tau}_j^l)$, $\forall l \in \{1, ..., L\}$ from robot j;
10 **for** *each sample* $s \in \{1, ..., L\}$ **do**
11 Estimate message $\hat{m}_{j,i}(\boldsymbol{\tau}_i^s)$ using (18);
12 Save sample $\boldsymbol{\tau}_i^s$ and updated message $m_{j,i}(\boldsymbol{\tau}_i^s) \leftarrow \hat{m}_{j,i}(\boldsymbol{\tau}_i^s)$;
13 **end**
14 **end**
15 **for** *each sample* $s \in \{1, ..., L\}$ **do**
16 Evaluate trajectory cost $C(\boldsymbol{\tau}_i^s)$ and compute importance weight $\mathcal{W}(\boldsymbol{\tau}_i^s)$;
17 **end**
18 Update mean $\overline{\boldsymbol{U}}_{i,t} = \{\overline{\boldsymbol{u}}_{i,t}, ..., \overline{\boldsymbol{u}}_{i,t+M-1}\}$ of current belief $q^t(\boldsymbol{U}_i)$ using (21);
19 Execute the first control action $\overline{\boldsymbol{u}}_{i,t}$;
20 Initialize $\overline{\boldsymbol{u}}_{i,t+M} \leftarrow \overline{\boldsymbol{u}}_{i,t+M-1}$ and $q^{t+1}(\boldsymbol{U}_i)$ with mean $\overline{\boldsymbol{U}}_{i,t+1} = \{\overline{\boldsymbol{u}}_{i,t+1}, ..., \overline{\boldsymbol{u}}_{i,t+M}\}$;
21 **end**

IV. Simulation and Results

A. Simulation Setup and Parameters

The proposed method was validated in Matlab simulations, where the algorithm was implemented using the following parameters. The MPC prediction horizon $M = 10$. The cost function weights $w_s = 5$, $w_u = 0.5$, $w_l = 10$ and $w_f = 2$. The inverse temperature parameters $\lambda_u = 0.3$ and $\lambda_p = 0.1$. The Gaussian sampling distribution was initialized with a zero mean and a standard deviation $\sigma = 0.3$ such that $\Sigma = 0.3^2\boldsymbol{I}_{2\times 2}$. The sample size $L = 200$. All simulations were conducted on a desktop computer with an Intel Core i7-12700 K 3.6 GHz CPU with 12 cores and 16 GB of RAM.

A 20-by-20-m square environment with static obstacles was created, as shown in Fig. 3. The number of robots $N = 7$, with initial positions uniformly randomized in a 5-by-5-m area at the lower left corner of the environment, and initial orientations sampled from $[0, 2\pi)$. The formation objective was such that 6 robots position themselves into a regular hexagon and 1 robot is at the center of the formation. The distance from each robot to its closest neighbors was 1.2 m. The maximum communication range $d_{\max} = 1.5$ m and the minimum inter-robot distance $d_{\min} = 0.6$ m. The maximum control $v_{\max} = 1.2$ m/s. The robot

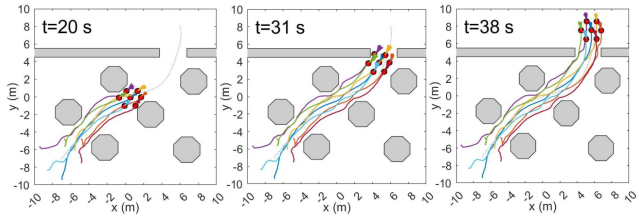

Fig. 3. Snapshots of the navigation in formation simulation with 7 robots at 20 s, 31 s, and 38 s.

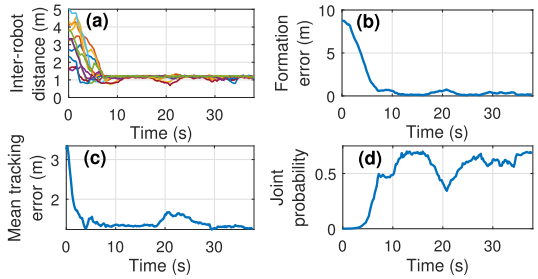

Fig. 4. Simulation results: (a) Inter-robot distance; (b) formation error; (c) tracking error averaged over all robots; (d) joint probability of robot trajectories over time steps.

kinematics parameter $l = 0.4$ m. The task duration $T = 190$ with $\Delta t = 0.2$ s.

B. Simulation Results

The snapshots of a simulation run at time $t = 20$ s, $t = 31$ s, and $t = 38$ s are shown in Fig. 3. The solid lines represent robot trajectories and the dotted line represents the reference trajectory. The short solid lines originating from each robot's position are predicted trajectories over the horizon M, sampled by current belief $q^t(\boldsymbol{U}_i)$. The gray areas are obstacles. One can see that the robots successfully navigate through the environment in the desired formation while avoiding obstacles ($t = 20$ s) and moving through a narrow passage ($t = 31$ s). Fig. 4(a) shows the distances between neighboring robots, i.e., $\|\boldsymbol{p}_{i,t} - \boldsymbol{p}_{j,t}\|, \forall (i,j) \in E$. The distances converged to 1.2 m as the robots achieved the desired formation. The perturbations were due to the robots compromising formation preservation to avoid obstacles. Fig. 4(b) shows the formation error $\|\boldsymbol{p}_{i,t} - \boldsymbol{p}_{j,t} - \Delta \boldsymbol{p}_{ij}\|$ averaged over the number of edges $|E|$, which converged nearly to 0 but increases while the robots avoided obstacles. Fig. 4(c) shows the mean reference tracking error of all robots, which eventually converged around 1.3 m. Note that the least mean tracking error is about 1.03 m when the desired formation is preserved. To show the convergence of BP in solving the global inference (6) on the multi-robot MRF graph, we factorized robot trajectories into a sequence of state-control pairs $\{\boldsymbol{x}_{i,t}, \boldsymbol{u}_{i,t}\}_{i=1}^{N}$ over $[0, T]$ and computed the joint probability (6) at each time step without normalization. Fig. 4(d) shows that the joint probability increased and plateaued around 0.68. More simulation results with $N = 3, 5, 7, 9$ can be found in the video file included with this article.

TABLE I
PERFORMANCE COMPARISON BETWEEN OUR METHOD AND TWO BASELINE METHODS FOR $N = 7$

	Centralized	ADMM	Our method
Success rate (%)	100	82	98
Formation error (m)	0.08 $\pm$0.01	0.20 $\pm$0.12	0.11 $\pm$0.05
Tracking error (m)	1.12 $\pm$0.05	1.43 $\pm$0.10	1.31 $\pm$0.07

C. Comparison With Baseline Methods

We compared the performance of our method with two gradient-based baseline methods: a centralized MPC and an ADMM-based distributed MPC similar to the one in [22] expect that we did not use the spline-based robot trajectories. The cost function used by the two baseline methods is based on (5) with an additional term $w_f \|\boldsymbol{p}_{i,t} - \boldsymbol{p}_{j,t} - \Delta \boldsymbol{p}_{ij}\|^2$ added as a soft formation constraint. The weights were set the same as our method. The collision and obstacle avoidance were formulated as inequality constraints using the method in [22]. The penalty parameter of the ADMM-based MPC was set to 2 and the algorithm executed only one ADMM iteration and used the solution of the previous iteration to warm start the optimization per MPC cycle. Both baseline methods used the interior-point algorithm from the Matlab Nonlinear Optimization toolbox (fmincon) to solve the constrained nonlinear optimization problems. We conducted 100 simulation runs for each methods with different initial states. The performance was evaluated with respect to 1) success rate (a test run is considered successful if robots reach the goal without collision), 2) formation error $\frac{1}{|E_t|} \sum_{(i,j) \in E_t} \|\boldsymbol{p}_{i,t+k} - \boldsymbol{p}_{j,t+k} - \Delta \boldsymbol{p}_{ij}\|$, and 3) tracking error $\frac{1}{N} \sum_{i \in V} \|\boldsymbol{x}_{i,t} - \boldsymbol{x}_{r,t}\|$. Both formation and tracking errors were averaged over time steps after the algorithms converged.

Table I shows the success rate, the mean and standard deviation of formation and tracking error of all successful runs. The centralized MPC successfully completed the task in all 100 runs and achieved the best performance over the two distributed MPC methods. Our method achieved a success rate of 98%, while the ADMM-based distributed MPC only achieved a success rate of 82%. Our method could obtain better performance than the ADMM-based distributed MPC. The success rate of our method could be improved by choosing a larger sample size L and adaptively updating σ of the sampling distribution. The ADMM-based MPC failed when the algorithm could not converge to a feasible solution. This is partly because the algorithm only performs one ADMM iteration per time step. Moreover, as gradient updates are deterministic, the algorithm is sensitive to initial conditions and thus prone to undesirable local minima. Our sampling-based method benefits from the probabilistic description of optimality which allows the algorithm to explore and seek optimal solutions via stochastic forward search.

D. Scaling and Computational Time

We evaluated the scalability of our method by testing it for robot team sizes $N = 3, 5, 7, 9$, and each team size was simulated 100 times. Fig. 5(a) shows the success rate, mean and standard deviation of formation and tracking errors of all successful runs. The tracking error for each N is the result after subtracting the least possible error (which varies with N) from the original error for fair comparison. One can see the success

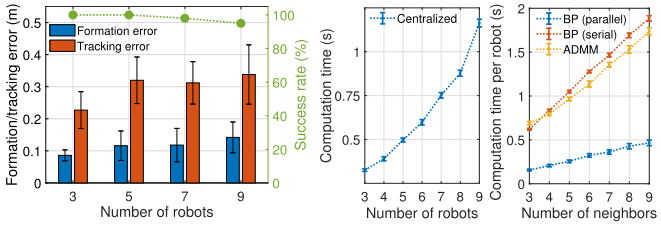

Fig. 5. Scalability of our method: (a) Success rate and formation/tracking error; (b) comparison of computation time.

rate was 100% for $N = 3$ and 5, and 98% and 95% for $N = 7$ and 9, respectively. The mean formation errors were between 0.09 m and 0.14 m and the mean tracking errors were between 0.22 m and 0.34 m for different team sizes. The results show that our method could achieve similar performance as N varies from 3 to 9.

We also compared the computational time per each MPC cycle with the centralized MPC and the ADMM-based distributed MPC. For the centralized MPC, we recorded the time used to solve the centralized optimization in each MPC cycle when robot team size $N = 3, 4, \ldots, 9$. For the ADMM-based distributed MPC (with one ADMM iteration per MPC cycle) and our method, we recorded the time used on each robot locally when the number of neighbors $|\mathcal{N}_i| = 3, 4, \ldots, 9$. We present two computing implementations of our method: 1) parallel computing of L trajectory samples on CPU using the Matlab Parallel Computing toolbox and 2) serial computing. Fig. 5(b) shows the mean and standard deviation of computational time for each method. While the centralized MPC grows super-linearly as N increases, all distributed MPC methods scales linearly with $|\mathcal{N}_i|$. It is notable that the computation time of our method can be significantly reduced by parallel computing. The mean computation time with parallelization is about 150 ms when $|\mathcal{N}_i| = 3$ and 460 ms when $|\mathcal{N}_i| = 9$. The computation of sampling-based methods are easily parallelizable and could be further accelerated using GPUs.

V. Conclusion

In this letter, we proposed a distributed framework for sampling-based optimal control, which leverages belief propagation and variational inference to solve multi-robot optimal control problems as probabilistic inference over graphical models in a distributed manner. We developed a distributed sampling-based MPC algorithm based on the proposed framework and validated our method in a multi-robot formation navigation problem. Our method showed effective control and outperformed an ADMM-based distributed MPC that uses gradient-based optimization. Our future work will consider stochastic robot and environmental dynamics, and other multi-robot coordinated control problems such as collective information gathering or active perception. We will also evaluate our method with real robot experiments.

References

[1] C. Jiang and Y. Guo, "Multi-robot guided policy search for learning decentralized swarm control," *IEEE Control Syst. Lett.*, vol. 5, no. 3, pp. 743–748, Jul. 2021.
[2] F. Augugliaro, A. P. Schoellig, and R. D'Andrea, "Generation of collision-free trajectories for a quadrocopter fleet: A sequential convex programming approach," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2012, pp. 1917–1922.
[3] C. Jiang and Y. Guo, "Incorporating control barrier functions in distributed model predictive control for multi-robot coordinated control," *IEEE Trans. Control Netw. Syst.*, early access, Jun. 28, 2023, doi: 10.1109/TCNS.2023.3290430.
[4] A. Nurkanović, T. Sartor, S. Albrecht, and M. Diehl, "A time-freezing approach for numerical optimal control of nonsmooth differential equations with state jumps," *IEEE Control Syst. Lett.*, vol. 5, no. 2, pp. 439–444, Apr. 2021.
[5] I. Abraham, A. Handa, N. Ratliff, K. Lowrey, T. D. Murphey, and D. Fox, "Model-based generalization under parameter uncertainty using path integral control," *IEEE Robot. Automat. Lett.*, vol. 5, no. 2, pp. 2864–2871, Apr. 2020.
[6] M. Kobilarov, "Cross-entropy motion planning," *Int. J. Robot. Res.*, vol. 31, no. 7, pp. 855–871, 2012.
[7] G. Williams, P. Drews, B. Goldfain, J. M. Rehg, and E. A. Theodorou, "Information-theoretic model predictive control: Theory and applications to autonomous driving," *IEEE Trans. Robot.*, vol. 34, no. 6, pp. 1603–1622, Dec. 2018.
[8] M. Okada and T. Taniguchi, "Variational inference MPC for Bayesian model-based reinforcement learning," in *Proc. Conf. Robot Learn.*, 2020, pp. 258–272.
[9] A. Lambert, F. Ramos, B. Boots, D. Fox, and A. Fishman, "Stein variational model predictive control," in *Proc. Conf. Robot Learn.*, 2021, pp. 1278–1297.
[10] T. Power and D. Berenson, "Variational inference MPC using normalizing flows and out-of-distribution projection," in *Proc. Robot.: Sci. Syst.*, 2022.
[11] B. V. D. Broek, W. Wiegerinck, and B. Kappen, "Graphical model inference in optimal control of stochastic multi-agent systems," *J. Artif. Intell. Res.*, vol. 32, pp. 95–122, 2008.
[12] N. Wan, A. Gahlawat, N. Hovakimyan, E. A. Theodorou, and P. G. Voulgaris, "Cooperative path integral control for stochastic multi-agent systems," in *Proc. Amer. Control Conf.*, 2021, pp. 1262–1267.
[13] P. Varnai and D. V. Dimarogonas, "Multi-agent stochastic control using path integral policy improvement," in *Proc. Amer. Control Conf.*, 2022, pp. 3406–3411.
[14] J. Pavlasek, J. Mah, O. Jenkins, and F. Ramos, "Stein variational belief propagation for decentralized multi-robot control," in *Proc. Workshop Distrib. Graph Algorithms Robot.*, 2023.
[15] A. Patwardhan, R. Murai, and A. J. Davison, "Distributing collaborative multi-robot planning with Gaussian belief propagation," *IEEE Robot. Automat. Lett.*, vol. 8, no. 2, pp. 552–559, Feb. 2023.
[16] H. J. Kappen, V. Gómez, and M. Opper, "Optimal control as a graphical model inference problem," *Mach. Learn.*, vol. 87, pp. 159–182, 2012.
[17] J. Butterfield, O. C. Jenkins, D. M. Sobel, and J. Schwertfeger, "Modeling aspects of theory of mind with Markov random fields," *Int. J. Social Robot.*, vol. 1, pp. 41–51, 2009.
[18] J. Pearl, *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. San Mateo, CA, USA: Morgan kaufmann, 1988.
[19] K. Murphy, Y. Weiss, and M. Jordan, "Loopy belief propagation for approximate inference: An empirical study," in *Proc. Uncertainty Artif. Intell.*, 1999, pp. 467–475.
[20] Y. Weiss and W. Freeman, "Correctness of belief propagation in Gaussian graphical models of arbitrary topology," in *Proc. Adv. Neural Inf. Process. Syst.*, 1999, pp. 673–679.
[21] A. Ihler and D. McAllester, "Particle belief propagation," in *Proc. Artif. Intell. Statist.*, 2009, pp. 256–263.
[22] R. V. Parys and G. Pipeleers, "Distributed MPC for multi-vehicle systems moving in formation," *Robot. Auton. Syst.*, vol. 97, pp. 144–152, 2017.