Decentralized Safe Multi-agent Stochastic Optimal Control using Deep FBSDEs and ADMM

Marcus A. Pereira, Augustinos D. Saravanos, Oswin So, and Evangelos A. Theodorou
Autonomous Control and Decision Systems Lab
Georgia Institute of Technology, Atlanta, Georgia
{mpereira30, asaravanos3, oswinso, evangelos.theodorou}@gatech.edu

***Abstract*—In this work, we propose a novel safe and scalable decentralized solution for multi-agent control in the presence of stochastic disturbances. Safety is mathematically encoded using stochastic control barrier functions and safe controls are computed by solving quadratic programs. Decentralization is achieved by augmenting to each agent's optimization variables, copy variables, for its neighboring agents. This allows us to decouple the centralized multi-agent optimization problem. However, to ensure safety, neighboring agents must agree on *what is safe for both of us* and this creates a need for consensus. To enable safe consensus solutions, we incorporate an ADMM-based approach. Specifically, we propose a Merged CADMM-OSQP implicit neural network layer, that solves a mini-batch of both, local quadratic programs as well as the overall consensus problem, as a single optimization problem. This layer is embedded within a Deep FBSDEs network architecture at every time step, to facilitate end-to-end differentiable, safe and decentralized stochastic optimal control. The efficacy of the proposed approach is demonstrated on several challenging multi-robot tasks in simulation. By imposing requirements on safety specified by collision avoidance constraints, the safe operation of all agents is ensured during the entire training process. We also demonstrate superior scalability in terms of computational and memory savings as compared to a centralized approach.**

I. Introduction

A vast variety of robotics applications such as coverage control [15], flocking of UAVs [32], multi-robot navigation [2], etc., falls into the class of multi-agent control problems. Such settings usually include a team of autonomous agents which are required to cooperate in order to accomplish a common goal. Effective frameworks for addressing these problems should be able to control the agents in an *optimal* manner, while ensuring their *safety under uncertainty*. In addition, it is of paramount importance that such methods are *scalable* to large-scale systems in terms of computational efficiency, memory usage and communication requirements.

Deep reinforcement learning has enjoyed significant fame over the past few years [30, 41, 42, 28], although being restricted to simulation settings where safety is not a primary concern. Therefore, very recently, the focus has shifted to the area of safe reinforcement learning [47, 12], where safety is required during the entire training process.

Control Barrier Functions (CBFs) [3, 4], have been popularly used in the robotics community to design controllers that guarantee invariance of user-defined safe sets. These are generally combined with *off-the-shelf* Quadratic Programming (QP) solvers to deliver real-time safe control solutions. However, most work has focused primarily on deterministic systems. Very recent works employing, so called, Stochastic CBFs (SCBFs) [14, 37, 39, 51] aim to bridge the gap, but lack scalability to large scale systems.

The confluence of Deep Learning and traditional optimization methods for *intra-layer optimization* has been of special interest lately. In particular, recent works embed QPs [1, 5, 7], root-finding methods [8] and even non-convex solvers [6, 20, 21] into the forward-pass of deep neural network layers and are popularly referred to as *implicit* neural network layers (other examples are [10, 27]). In most of these methods, the backward-pass is efficiently computed by invoking the implicit function theorem rather backpropagating through the unrolled graph of the forward-pass.

A general approach for continuous-time Stochastic Optimal Control (SOC) relying on FBSDEs (Forward-Backward Stochastic Differential Equations) was recently combined with deep learning [23] to solve high-dimensional Hamilton-Jacobi-Bellman PDEs (HJB-PDEs). Most noteworthy being deep FBSDEs [33, 34, 48, 11], a scalable framework for SOC problems that, at its core, leverages the function approximation capabilities of deep recurrent neural networks (specifically LSTMs) to learn the gradient of the value-function, which can then be used to compute optimal control policies.

Distributed optimization-based frameworks have been gaining significant attention for tackling multi-agent control problems. A suitable method for deriving such algorithms is the Alternating Direction Method of Multipliers (ADMM) [9]. In particular, ADMM-based methods have been recently proposed [45, 49, 22, 13], yielding elegant decentralized solutions for multi-agent control. Furthermore, recent works employing ADMM in a stochastic setting, [38, 36, 26], have shown to be capable of successfully encompassing both the safety under uncertainty and scalability desired attributes.

There has been very little work trying to put together all these ingredients into one framework. The Safe Deep FBSDEs framework [35] is one recent approach that partly addresses this need, the missing component being decentralization. More specifically, it uses an instantiation of implicit layers to solve embedded SCBF-based QPs combined with deep learning. Additionally, it utilized a conservative SCBF that tries to

Under Review.

gaurantee safety with a probability of 1. The method was tested on low-dimensional systems in simulation and the resulting policies were too conservative. Inspired by Safe Deep FBSDEs with the aim to overcome its limitations in order to scale to large multi-agent SOC problems, we propose a decentralized approach to safety. To this end, the specific contributions of our work are,

1) A safe end-to-end differentiable framework that uses a novel SCBF formulation that is practically more meaningful and allows for safe, yet non-conservative policies,
2) A *fully decentralized* ADMM based algorithm for large scale QPs embedded into an implicit neural network layer with drastic improvements in memory efficiency and training time as compared to a centralized approach,
3) A safe reinforcement framework, i.e., one that ensures safety not only at test time, but also while training the policy, and,
4) Extensive testing on several multi-robot challenging tasks in simulation, demonstrating the capabilities of the proposed framework.

II. Notation

Here, we introduce the notation followed throughout this paper. Non-bold symbols are used for scalars $a \in \mathbb{R}$, and bold lowercase and uppercase symbols for vectors $\mathbf{a} \in \mathbb{R}^n$ and matrices $\mathbf{A} \in \mathbb{R}^{n\times m}$, respectively. With $\mathbf{a} = [\mathbf{a}_1; \ldots; \mathbf{a}_N]$, we denote the vertical concatenation of vectors $\mathbf{a}_1, \ldots, \mathbf{a}_N$. The ℓ_2-norm of a vector $\mathbf{a} \in \mathbb{R}^n$ is defined as $\|\mathbf{a}\|_2 = \sqrt{\sum_{i=1}^n a_i^2}$ where $\mathbf{a} = [a_1; \ldots; a_N]$. Moreover, the trace of a matrix $\mathbf{A} \in \mathbb{R}^{n\times n}$ is denoted with $\mathrm{tr}(\mathbf{A})$. We also define as $\mathrm{diag}(a_1, \ldots, a_N) \in \mathbb{R}^{N\times N}$ the diagonal matrix with diagonal elements scalars $a_1, \ldots, a_N$, and as $\mathrm{bdiag}(\mathbf{A}_1, \ldots, \mathbf{A}_N)$ the block diagonal matrix constructed by the matrices $\mathbf{A}_1, \ldots, \mathbf{A}_N$. Given a set $\mathcal{N}$, its cardinality is denoted by $|\mathcal{N}|$. With $[\![a, b]\!]$, we denote the integer interval $[a, b] \cap \mathbb{Z}$. Finally, given a convex set $\mathcal{C}$, $\Pi_{\mathcal{C}}(\mathbf{x})$ denotes the projection of a vector $\mathbf{x}$ onto the set and $\mathcal{I}_c(\mathbf{x})$ denotes the set indicator function such that $\mathcal{I}_c(\mathbf{x}) = 0$ if $\mathbf{x} \in \mathcal{C}$ and $\mathcal{I}_c(\mathbf{x}) = +\infty$ otherwise.

III. Problem Formulation

The framework developed in this paper can be applied to various problems, however, we focus on a multi-robot setting so as to establish a notion of distance as well as to provide context for constraints commonly encountered in this setting.

Consider a collection of N agents. The stochastic, nonlinear and control-affine dynamics for any agent i are given by,

$$\mathrm{d}\mathbf{x}_i = \big(\mathbf{f}_i(\mathbf{x}_i) + \mathbf{G}_i(\mathbf{x}_i)\mathbf{u}_i\big)\mathrm{d}t + \mathbf{\Sigma}_i(\mathbf{x}_i)\mathrm{d}\mathbf{w}_i \tag{1}$$

where $\mathbf{x}_i = \mathbf{x}_i(t) \in \mathbb{R}^{n_i}$, $\mathbf{u}_i = \mathbf{u}_i(t) \in \mathbb{R}^{m_i}$, $\mathbf{w}_i = \mathbf{w}_i(t) \in \mathbb{R}^{p_i}$ are the state, control and standard Brownian-motion vectors, respectively, and $\mathbf{f}_i(\cdot) : \mathbb{R}^{n_i} \to \mathbb{R}^{n_i}$, $\mathbf{G}_i(\cdot) : \mathbb{R}^{n_i} \to \mathbb{R}^{n_i\times m_i}$ and $\mathbf{\Sigma}_i(\cdot) : \mathbb{R}^{n_i} \to \mathbb{R}^{n_i\times p_i}$ denote the drift vector, actuation matrix and diffusion matrix respectively. In order to state the centralized problem, we construct the global state, control and Brownian-motion vectors, $\mathbf{x} = [\mathbf{x}_1; \mathbf{x}_2; \ldots; \mathbf{x}_N]$, $\mathbf{u} = [\mathbf{u}_1; \mathbf{u}_2; \ldots; \mathbf{u}_N]$, and $\mathbf{w} = [\mathbf{w}_1; \mathbf{w}_2; \ldots; \mathbf{w}_N]$ respectively, by concatenation. Thus, the global stochastic dynamics are given by,

$$\mathrm{d}\mathbf{x} = \big(\mathbf{f}(\mathbf{x}) + \mathbf{G}(\mathbf{x})\mathbf{u}\big)\mathrm{d}t + \mathbf{\Sigma}(\mathbf{x})\mathrm{d}\mathbf{w} \tag{2}$$

with $\mathbf{f} = [\mathbf{f}_1; \mathbf{f}_2; \ldots; \mathbf{f}_N]$, $\mathbf{G} = \mathrm{bdiag}(\mathbf{G}_1, \ldots, \mathbf{G}_N)$ and $\mathbf{\Sigma} = \mathrm{bdiag}(\mathbf{\Sigma}_1, \ldots, \mathbf{\Sigma}_N)$.

Stochastic Optimal Control (SOC) aims to minimize an expected cost subject to (2) given by,

$$\begin{aligned}\mathcal{J}(\mathbf{x},\, \mathbf{u},\, t_0) &= \sum_{i=1}^N J_i(\mathbf{x}_i,\, \mathbf{u}_i,\, t_0) \\ &= \sum_{i=1}^N \mathbb{E}\bigg[\phi_i\big(\mathbf{x}_i(T)\big) + \int_{t_0}^T \Big(q_i\big(\mathbf{x}_i\big) + \frac{1}{2}\mathbf{u}_i^{\mathrm{T}}\mathbf{R}_i\mathbf{u}_i\Big)\,\mathrm{d}t\bigg]\end{aligned} \tag{3}$$

where for any agent i, $\phi_i(\cdot)$ is the terminal state-cost function and the running cost comprises of a purely state-dependent term, $q_i(\cdot)$, and a quadratic control-cost term with weighting coefficients given by the matrix $\mathbf{R}_i \in \mathbb{R}^{m_i\times m_i}$.

To solve the SOC problem using dynamic programming, we define the value function as, $V(\mathbf{x},\, t) = \inf_{\mathbf{u}} \mathcal{J}(\mathbf{x},\, \mathbf{u},\, t)$, i.e., the optimal *cost-to-go* from state $\mathbf{x}$ at time step t. Next, using Ito's formula [40, Chapter 4], one can derive the Hamilton-Jacobi-Bellman Partial Differential Equation (HJB-PDE),

$$\begin{aligned}&\frac{\partial V}{\partial t} + \inf_{\mathbf{u}} \mathcal{H} = 0,\, V(\mathbf{x},\, T) = \sum_{i=1}^N \phi_i\big(\mathbf{x}_i(T)\big) \\ &\text{where, } \mathcal{H} = \frac{1}{2}\mathrm{tr}\bigg(\frac{\partial^2 V}{\partial \mathbf{x}^2}\mathbf{\Sigma}\mathbf{\Sigma}^{\mathrm{T}}\bigg) + \frac{\partial V}{\partial \mathbf{x}}^{\mathrm{T}}\Big(\mathbf{f} + \mathbf{G}\mathbf{u}\Big) \\ &\qquad\qquad + \sum_{i=1}^N \Big(q_i + \frac{1}{2}\mathbf{u}_i^{\mathrm{T}}\mathbf{R}_i\mathbf{u}_i\Big)\end{aligned} \tag{4}$$

which is a backward semilinear PDE and $\mathcal{H}$ is referred to as the Hamiltonian. Owing to the construction of the global state and control by concatenation, the Hamiltonian minimization can be split into a sum of decoupled minimizations of the Hamiltonians associated with each agent i $\big($i.e., $\inf_{\mathbf{u}} \mathcal{H} = \sum_{i=1}^N \inf_{\mathbf{u}_i} \mathcal{H}_i\big)$ in the absence of inter-agent safety constraints. However, when the latter are considered, this decoupling is no longer valid. In this work, we adopt a probabilistic approach to safety by imposing such constraints using Stochastic Control Barrier Functions (SCBFs). We assume that a safe set defined by $\mathcal{C} = \{\mathbf{x} : h(\mathbf{x}) \geq 0\}$ is known, where $h(\mathbf{x})$ is task-dependent and usually hand-designed and the goal is to stay within the safe set (with a high probability) for the entire time horizon. In the context of multi-robot systems, these SCBFs can encode obstacle and inter-agent collision avoidance constraints.

There are two types of SCBFs currently in literature – *almost-sure* (type-I) and those which allow for violations (type-II). The type-I SCBFs ensure that the system remains inside the safe set with a probability of 1 [14] for all time $t \in [0,\, T]$. These were successfully utilized within a recent deep-learning based SOC framework [35], but simulation results clearly indicate that type-I SCBFs lead to conservative

policies preventing agents from getting close to each other, thus, reducing flexibility of the trained policy. This restriction can prohibit application to systems with a large number of agents. On the other hand, type-II SCBFs allow tuning the probability of failure based on one's risk appetite and therefore have a higher practical appeal. These have been employed in recent works [37, 51] and are based on derivation of Lyapunov functions for finite time stability of stochastic systems [25, Chapter 3]. Restating the result from [37, Proposition 1] here for convenience of the reader: *suppose there exists a twice differentiable function $B(\mathbf{x})$, that satisfies the following inequalities,*

$$B(\mathbf{x}) \geq 0 \quad \forall\, \mathbf{x} \in \mathbb{R}^{Nn_i} \tag{5}$$

$$B(\mathbf{x}) \geq 1 \quad \forall\, \mathbf{x} \in (\mathbb{R}^{Nn_i} \backslash \mathcal{C}) \tag{6}$$

$$\frac{\partial B}{\partial \mathbf{x}}^{\mathrm{T}} \Big(\mathbf{f} + \mathbf{G}\mathbf{u}\Big) + \frac{1}{2} tr\Big(\frac{\partial^2 B}{\partial \mathbf{x}^2} \boldsymbol{\Sigma}\boldsymbol{\Sigma}^{\mathrm{T}}\Big) \leq -\alpha B(\mathbf{x}) + \beta, \tag{7}$$

where (7) is satisfied $\forall\, t \in [0,\, T]$ and $\forall\, \mathbf{x} \in \mathbb{R}^{Nn_i}$ for some $\alpha \geq 0$ and $\beta \geq 0$. Based on the chosen values of α and β, one can compute bounds on the probability of failure. We refer the reader to the supplementary material for additional details.

Inequalities (5) and (6) can be satisfied by choosing $B(\mathbf{x}) = e^{-\gamma h(\mathbf{x})}$ so that when the system exits the safe set $\mathcal{C}$, then $B(\mathbf{x}) > 1$ because $h(\mathbf{x}) < 0$. To satisfy (7), we impose it as a hard constraint on the Hamiltonian minimization in (4). Since the objective $\mathcal{H}(\mathbf{u})$ is quadratic and the constraint (7) is linear in $\mathbf{u}$, then a safe optimal control can be obtained by solving the resulting Quadratic Program (QP) at every time step. Similar to the work in [35], we have the following HJB-PDE,

$$\frac{\partial V}{\partial t} + \inf_{\mathbf{u} \in \mathbf{u}_{\text{safe}}} \left\{ \sum_{i=1}^{N} \left(\frac{1}{2} \mathbf{u}_i^{\mathrm{T}} \mathbf{R}_i \mathbf{u}_i + \frac{\partial V}{\partial \mathbf{x}_i}^{\mathrm{T}} \mathbf{G}_i \mathbf{u}_i \right) \right\} + \sum_{i=1}^{N} \left(q_i + \frac{\partial V}{\partial \mathbf{x}_i}^{\mathrm{T}} \mathbf{f}_i + \frac{1}{2} \mathrm{tr}\Big(\frac{\partial^2 V}{\partial \mathbf{x}_i^2} \boldsymbol{\Sigma}_i \boldsymbol{\Sigma}_i^{\mathrm{T}} \Big) \right) = 0 \tag{8}$$

$$\text{where, } \mathbf{u}_{\text{safe}} = \left\{ \mathbf{u} \middle| \frac{\partial B_k}{\partial \bar{\mathbf{x}}_k}^{\mathrm{T}} \left(\bar{\mathbf{f}}_k + \bar{\mathbf{G}}_k \bar{\mathbf{u}}_k \right) + \frac{1}{2} \mathrm{tr}\left(\frac{\partial^2 B_k}{\partial \bar{\mathbf{x}}_k^2} \bar{\boldsymbol{\Sigma}}_k \bar{\boldsymbol{\Sigma}}_k^{\mathrm{T}} \right) \leq -\alpha B_k(\bar{\mathbf{x}}_k) + \beta,\, \forall\, k \in [\![1, N_{\text{ineq}}]\!] \right\}$$

where each B_k is either a pairwise safety constraint between 2 agents or between an agent and an obstacle, ($\bar{\mathbf{x}}_k$ and $\bar{\mathbf{u}}_k$) are vectors obtained by concatenating states and controls of the 2 agents corresponding to B_k for inter-agent constraints or just the state and the control vectors of the ego agent for agent-obstacle constraints, ($\bar{\mathbf{f}}_k$, $\bar{\mathbf{G}}_k$ and $\bar{\boldsymbol{\Sigma}}_k$) are constructed similarly and $N_{\text{ineq}} = \binom{N}{2} + NN_o$, is the total number of inequality constraints. The $\binom{N}{2}$ constraints account for all possible agent pairs for collision avoidance similar to work in [35, Section 4.3.2] and N_o is the number of obstacles. However, this clearly would not scale for large values of N. Due to these constraints, the minimization of individual $\mathcal{H}_i$ can no longer be decoupled. Thus, the safe optimal control $\mathbf{U}^*$, has to be solved in a *centralized* manner as one large QP.

IV. Centralized Solution Using Deep FBSDEs

Before we propose our decentralized solution to address scalability, we summarize the deep learning based solution to safe SOC problems adapted from recent work [35].

The unique solution of (8) is linked to that of a system of FBSDEs via the Nonlinear Feynman-Kac lemma (see supplementary material for derivation). Assuming that a solution $\mathbf{u}^*$ exists, the FBSDE system that solves (8) is given by,

$$\textbf{(FSDE)} \quad \mathrm{d}\mathbf{x} = (\mathbf{f} + \mathbf{G}\mathbf{u}^*)\mathrm{d}t + \boldsymbol{\Sigma}\mathrm{d}\mathbf{w},\, \mathbf{x}(0) = \mathbf{x}_0 \tag{9}$$

$$\textbf{(BSDE)} \begin{cases} \mathrm{d}V = -\left[\sum_{i=1}^{N} \left(q_i + \frac{1}{2} \mathbf{u}_i^{*\mathrm{T}} \mathbf{R}_i \mathbf{u}_i^* \right) \right] \mathrm{d}t + \frac{\partial V}{\partial \mathbf{x}}^{\mathrm{T}} \boldsymbol{\Sigma} \mathrm{d}\mathbf{w} \\ V\big(\mathbf{x}(T)\big) = \sum_{i=1}^{N} \phi_i\big(\mathbf{x}_i(T)\big) \end{cases} \tag{10}$$

where, $\mathbf{u}^* = \underset{\mathbf{u} \in \mathbf{u}_{\text{safe}}}{\arg\min}\, \mathcal{H}$.

Deep FBSDEs use deep networks to learn $\frac{\partial V}{\partial \mathbf{x}}(\mathbf{x},\, t;\, \theta)$, which can be used to compute optimal control policies $\mathbf{u}^*(\mathbf{x})$. Traditional methods [16, 19, 18, 17] to solve FBSDEs relied on back-propagating and approximating the conditional expectation of the value function, $\mathbb{E}\big[V(\mathbf{x},\, t)\big]$, using least squares. This approach is prone to numerical ill-conditioning issues depending on which area of the state-space the system visits, requires hand-picking basis functions and suffers from compounding least squares errors over time as $\mathbb{E}\big[V(\mathbf{x},\, t)\big]$ is back-propagated from $t = T$ to $t = t_0$. Deep FBSDEs circumvent the need to back-propagate $\mathbb{E}\big[V(\mathbf{x},\, t)\big]$ by instead parameterizing $\hat{V}\big(\mathbf{x}(0),\, 0\big)$ with trainable weights. Using this approximation of the initial condition, $V(\mathbf{x},\, t)$ can then be forward propagated similar to a forward SDE using (10). At the end of the time horizon, the predicted terminal value $\hat{V}\big(\mathbf{x}(T),\, T\big)$ that relies on the predictions of $\frac{\partial V}{\partial \mathbf{x}}(\mathbf{x},\, t;\, \theta)$ provided by a deep LSTM network, is compared to the true terminal value $V\big(\mathbf{x}(T),\, T\big)$ to construct a loss function. $V\big(\mathbf{x}(T),\, T\big)$ is evaluated using the given $\phi_i\big(\mathbf{x}_i(T)\big)$ and terminal states $\mathbf{x}(T)$ obtained by forward propagation of (9). This is then used to train the deep LSTM network using optimizers such as Adam [24]. Thus, deep FBSDEs is a self-supervised learning framework. Over iterations, as the loss is minimized, the network improves its predictions of $\hat{V}\big(\mathbf{x}(0),\, 0\big)$ and $\frac{\partial V}{\partial \mathbf{x}}(\mathbf{x},\, t;\, \theta)$.

For unconstrained problems, $\mathbf{u}_i^* = -\mathbf{R}_i^{-1} \mathbf{G}_i^{\mathrm{T}} \frac{\partial V}{\partial \mathbf{x}_i}$. However, when combined with SCBFs, $\mathbf{u}_i^*$ can only be computed numerically. Similar to work in [35], safe optimal controls $\mathbf{u}_i^*$ can be computed in an end-to-end differentiable manner using an OptNet-like [5] implicit layer to solve the constrained QP (8) and to ensure efficient backpropagation for DNN training using the implicit function theorem.

V. Decentralized Approach

In this section, we propose a decentralized approach for addressing the Hamiltonian minimization in (8). To achieve this, we reformulate the centralized problem in a distributed form. Subsequently, we propose a decentralized ADMM-based method combining elements from Consensus ADMM [9]

and OSQP [43] for solving large-scale QPs, which is then employed for solving our problem.

A. Decentralized Problem

The key restriction in problem (8) which prevents directly solving it in a distributed manner is the coupling induced by the inter-agent constraints. To overcome this issue, let us introduce the *neighborhood* sets $\mathcal{N}_i,\ i \in [\![1, N]\!]$, which contain the neighboring agents of each agent i. We also define the sets $\mathcal{P}_i = \{j : i \in \mathcal{N}_j\},\ i \in [\![1, N]\!]$, where each set $\mathcal{P}_i$ contains all the agents that have agent i as a neighbor.

Assumption 1. *Each agent $i \in [\![1, N]\!]$ is able to communicate with all agents $j \in \mathcal{N}_i \cup \mathcal{P}_i$, and vice versa.*

Next, we consider for each agent i, the copy control and state variables $\{\mathbf{u}_j^{(i)}\}_{j\in\mathcal{N}_i}$ and $\{\mathbf{x}_j^{(i)}\}_{j\in\mathcal{N}_i}$, respectively. Essentially, a copy variable $\mathbf{u}_j^{(i)}$ (or $\mathbf{x}_j^{(i)}$) can be interpreted as *agent i deciding what is safe for its neighbor j from its own perspective*. Let us also define the augmented local variables containing the states and controls of all agents within agent i's neighborhood:

$$\tilde{\mathbf{u}}_i = \big[\mathbf{u}_i; \{\mathbf{u}_j^{(i)}\}_{j\in\mathcal{N}_i}\big],\ \tilde{\mathbf{x}}_i = \big[\mathbf{x}_i; \{\mathbf{x}_j^{(i)}\}_{j\in\mathcal{N}_i}\big],\ i \in [\![1, N]\!].$$

Nevertheless, the inclusion of the copy variables creates a requirement for enforcing a consensus between variables that correspond to the same agents. For this reason, we also introduce the global control variable $\mathbf{g} = [\mathbf{g}_1, \dots, \mathbf{g}_N]$ whose components $\mathbf{g}_i,\ i \in [\![1, N]\!]$ should be equal to all local variables that refer to agent i. Therefore, we impose the constraints,

$$\mathbf{u}_j^{(i)} = \mathbf{g}_j,\ \forall j \in \mathcal{N}_i \cup \{i\},\ \forall i \in [\![1, N]\!]. \tag{11}$$

We can now formulate a *decentralized* form of the Hamiltonian minimization problem as,

$$\min \sum_{i=1}^{N} \mathcal{H}_i(\tilde{\mathbf{u}}_i) \tag{12a}$$

$$\text{s.t.}\quad \frac{\partial B^{i,k}}{\partial \tilde{\mathbf{x}}_{i,k}}\big(\tilde{\mathbf{f}}_{i,k} + \tilde{\mathbf{G}}_{i,k}\tilde{\mathbf{u}}_{i,k}\big) + \frac{1}{2}\mathrm{tr}\bigg(\frac{\partial^2 B^{i,k}}{\partial \tilde{\mathbf{x}}_{i,k}^2}\tilde{\mathbf{\Sigma}}_{i,k}\tilde{\mathbf{\Sigma}}_{i,k}^{\mathrm{T}}\bigg)$$
$$\leq -\alpha B^{i,k} + \beta,\ \forall k \in [\![1, N_{\text{ineq},i}]\!],\ \forall i \in [\![1, N]\!] \tag{12b}$$
$$\tilde{\mathbf{u}}_i = \tilde{\mathbf{g}}_i,\ \forall i \in [\![1, N]\!] \tag{12c}$$

where the vectors $\tilde{\mathbf{x}}_{i,k}$, $\tilde{\mathbf{u}}_{i,k}$ are defined as $\tilde{\mathbf{x}}_{i,k} = [\mathbf{x}_i; \mathbf{x}_j^{(i)}]$, $\tilde{\mathbf{u}}_{i,k} = [\mathbf{u}_i; \mathbf{u}_j^{(i)}]$, if (12b) is an inter-agent constraint involving a specific neighboring agent $j \in \mathcal{N}_i$ and as $\tilde{\mathbf{x}}_{i,k} = \mathbf{x}_i$, $\tilde{\mathbf{u}}_{i,k} = \mathbf{u}_i$, if (12b) is an obstacle avoidance constraint. The functions $\tilde{\mathbf{f}}_{i,k}$, $\tilde{\mathbf{G}}_{i,k}$, $\tilde{\mathbf{\Sigma}}_{i,k}$ are defined accordingly in each case. Finally, $\tilde{\mathbf{g}}_i$ is defined as $\tilde{\mathbf{g}}_i = \big[\mathbf{g}_i; \{\mathbf{g}_j\}_{j\in\mathcal{N}_i}\big]$ and $N_{\text{ineq},i} = |\mathcal{N}_i| + N_o$.

Subsequently, by denoting the linear inequality constraints (12b) as $\mathbf{A}_i\tilde{\mathbf{u}}_i \leq \mathbf{d}_i$, we can rewrite problem (12) as,

$$\min \sum_{i=1}^{N} \mathcal{H}_i(\tilde{\mathbf{u}}_i) + \mathcal{I}_{\mathbf{A}_i\tilde{\mathbf{u}}_i\leq\mathbf{d}_i}(\mathbf{A}_i\tilde{\mathbf{u}}_i)$$
$$\text{s.t.}\quad \tilde{\mathbf{u}}_i = \tilde{\mathbf{g}}_i,\ \forall i \in [\![1, N]\!] \tag{13}$$

Problem (13) is now in a form where CADMM can be applied. This yields a bilevel distributed optimization algorithm where at every ADMM iteration, each agent first locally solves a QP, and then the local solutions are used to perform the global and dual updates [9, Chapter 7]. These local QPs can be solved by well-known solvers such as OSQP [44], interior-point methods [31, Chapter 16] [29, 5], etc.

B. Merged CADMM-OSQP Method

In this work, we exploit the fact that the inner QP program in CADMM can itself be solved using ADMM via OSQP, flattening the bilevel optimization problem and propose the Merged CADMM-OSQP method for solving QPs in a decentralized manner. First, let us define $\tilde{\mathbf{R}}_i = \mathrm{blkdiag}(\mathbf{R}_i, \{\mathbf{0}\}_{j\in[\![1,|\mathcal{N}_i|]\!]})$ and $\tilde{\mathbf{p}}_i = [\mathbf{p}_i; \{\mathbf{0}\}_{j\in[\![1,|\mathcal{N}_i|]\!]}]$. We then reformulate (13) as,

$$\min \sum_{i=1}^{N} \frac{1}{2}\tilde{\mathbf{u}}_i^{\mathrm{T}}\tilde{\mathbf{R}}_i\tilde{\mathbf{u}}_i + \tilde{\mathbf{p}}_i^{\mathrm{T}}\tilde{\mathbf{u}}_i + \mathcal{I}_{\tilde{\mathbf{z}}_i\leq\mathbf{d}_i}(\tilde{\mathbf{z}}_i)$$
$$\text{s.t.}\quad \mathbf{A}_i\tilde{\mathbf{u}}_i = \tilde{\mathbf{z}}_i,\ \tilde{\mathbf{u}}_i = \tilde{\mathbf{g}}_i,\ \forall i \in [\![1, N]\!] \tag{14}$$

where each $\mathcal{H}_i(\tilde{\mathbf{u}}_i)$ is written in a form $\mathcal{H}_i(\tilde{\mathbf{u}}_i) = \frac{1}{2}\tilde{\mathbf{u}}_i^{\mathrm{T}}\tilde{\mathbf{R}}_i\tilde{\mathbf{u}}_i + \tilde{\mathbf{p}}_i^{\mathrm{T}}\tilde{\mathbf{u}}_i$. Next, we introduce the auxiliary variables $\hat{\mathbf{z}}_i,\ i \in [\![1, N]\!]$ in a similar manner as in the OSQP derivation [44, Section 3] and transform (14) to,

$$\min \sum_{i=1}^{N} \frac{1}{2}\tilde{\mathbf{u}}_i^{\mathrm{T}}\tilde{\mathbf{R}}_i\tilde{\mathbf{u}}_i + \tilde{\mathbf{p}}_i^{\mathrm{T}}\tilde{\mathbf{u}}_i + \mathcal{I}_{\mathbf{A}_i\tilde{\mathbf{u}}_i=\tilde{\mathbf{z}}_i}(\tilde{\mathbf{u}}_i, \tilde{\mathbf{z}}_i) + \mathcal{I}_{\tilde{\mathbf{z}}_i\leq\mathbf{d}_i}(\hat{\mathbf{z}}_i)$$
$$\text{s.t.}\quad \tilde{\mathbf{z}}_i = \hat{\mathbf{z}}_i,\ \tilde{\mathbf{u}}_i = \tilde{\mathbf{g}}_i,\ \forall i \in [\![1, N]\!] \tag{15}$$

The Augmented Lagrangian (AL) of (15) yields,

$$\mathcal{L} = \sum_{i=1}^{N} \frac{1}{2}\tilde{\mathbf{u}}_i^{\mathrm{T}}\tilde{\mathbf{R}}_i\tilde{\mathbf{u}}_i + \tilde{\mathbf{p}}_i^{\mathrm{T}}\tilde{\mathbf{u}}_i + \mathcal{I}_{\mathbf{A}_i\tilde{\mathbf{u}}_i=\tilde{\mathbf{z}}_i}(\tilde{\mathbf{u}}_i, \tilde{\mathbf{z}}_i) + \mathcal{I}_{\tilde{\mathbf{z}}_i\leq\mathbf{d}_i}(\hat{\mathbf{z}}_i)$$
$$+ \frac{\rho}{2}\Big\|\tilde{\mathbf{z}}_i - \hat{\mathbf{z}}_i + \frac{\mathbf{y}_i}{\rho}\Big\|_2^2 + \frac{\mu}{2}\Big\|\tilde{\mathbf{u}}_i - \tilde{\mathbf{g}}_i + \frac{\boldsymbol{\xi}_i}{\mu}\Big\|_2^2 \tag{16}$$

where $\mathbf{y}_i$, $\boldsymbol{\xi}_i$ are the dual variables for the corresponding equality constraints and $\rho, \mu > 0$ are penalty parameters.

Therefore, it is possible to use ADMM in a manner such that the consensus and OSQP updates take place within the same ADMM cycle of updates. The first block of updates on $\tilde{\mathbf{u}}_i$ and $\tilde{\mathbf{z}}_i$ will be,

$$\{\tilde{\mathbf{u}}_i, \tilde{\mathbf{z}}_i\}^{l+1} = \underset{\tilde{\mathbf{u}}_i, \tilde{\mathbf{z}}_i}{\arg\min}\ \frac{1}{2}\tilde{\mathbf{u}}_i^{\mathrm{T}}\tilde{\mathbf{R}}_i\tilde{\mathbf{u}}_i + \tilde{\mathbf{p}}_i^{\mathrm{T}}\tilde{\mathbf{u}}_i$$
$$+ \frac{\rho}{2}\Big\|\tilde{\mathbf{z}}_i - \hat{\mathbf{z}}_i^l + \frac{\mathbf{y}_i^l}{\rho}\Big\|_2^2 + \frac{\mu}{2}\Big\|\tilde{\mathbf{u}}_i - \tilde{\mathbf{g}}_i^l + \frac{\boldsymbol{\xi}_i^l}{\mu}\Big\|_2^2$$
$$\text{s.t. } \mathbf{A}_i\tilde{\mathbf{u}}_i = \tilde{\mathbf{z}}_i. \tag{17}$$

The second block where $\hat{\mathbf{z}}_i$ and $\mathbf{g}$ are updated will consist of,

$$\hat{\mathbf{z}}_i^{l+1} = \Pi_{\mathcal{C}_i}(\tilde{\mathbf{z}}_i^{l+1} + \frac{1}{\rho}\mathbf{y}_i^l) \tag{18a}$$

$$\mathbf{g}_i^{l+1} = \frac{1}{|\mathcal{P}_i| + 1}\sum_{j\in\mathcal{P}_i\cup\{i\}}\bigg(\mathbf{u}_i^{(j),l+1} + \frac{1}{\mu}\boldsymbol{\xi}_i^{(j),l}\bigg) \tag{18b}$$

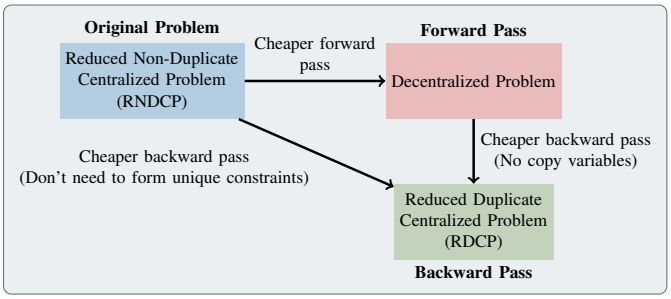


Fig. 1: Relationship between RNDCP, RDCP (22) and the Decentralized Problem (Section V).

where $\boldsymbol{\xi}_i^{(j)}$ is the part of the dual variable $\boldsymbol{\xi}_i$ corresponding to the constraint $\mathbf{u}_j^{(i)} = \mathbf{g}_j$. Finally, the dual variables updates will be,

$$\mathbf{y}_i^{l+1} = \mathbf{y}_i^l + \rho(\tilde{\mathbf{z}}_i^{l+1} - \hat{\mathbf{z}}_i^{l+1}) \tag{19a}$$

$$\boldsymbol{\xi}_i^{l+1} = \boldsymbol{\xi}_i^l + \mu(\tilde{\mathbf{u}}_i^{l+1} - \tilde{\mathbf{g}}_i^{l+1}). \tag{19b}$$

Since each variable with subscript i being updated only requires variables that have the same subscript and hence also belong to agent i, the algorithm composed by the updates (17), (18), (19) can be executed in a *fully decentralized* manner since all updates can be performed in parallel by each agent. During every ADMM iteration, two communication steps are required. The first takes place after the updates (17) have been completed by all agents, where every agent $j \in \mathcal{P}_i$ must send its variables $\tilde{\mathbf{u}}_j^{l+1}$ and $\boldsymbol{\xi}_j^l$ to each agent i. After the updates (18) take place, every agent $j \in \mathcal{N}_i$ must send $\mathbf{g}_j^{l+1}$ to agent i, so that the latter can construct the vector $\tilde{\mathbf{g}}_i^{l+1}$.

It should be highlighted that the $N_{\text{ineq},i} = r + N_o$ constraints involved in each of the local subproblems, in parallel solvable, will be drastically fewer than the $N_{\text{ineq}} = \binom{N}{2} + NN_o$ constraints of the centralized problem for $r \ll N$. The algorithm terminates when the norms of the primal and dual residuals get below their corresponding tolerance levels,

$$r_{pri,a} \leq \epsilon_{pri,a}, \quad r_{dual,a} \leq \epsilon_{dual,a}, \quad \forall a = 1, 2.$$

Detailed expressions for the residual norms $r_{pri,a}$, $r_{dual,a}$ and the tolerances $\epsilon_{pri,a}$, $\epsilon_{dual,a}$ are provided in the Supplementary Material.

VI. Implementation Details

We make the following assumption for our implementation,

Assumption 2. *All neighborhood sets within a single time step t, $\mathcal{N}_i(t)$, are of equal size r.*

This assumption is required to simulate a batch of trajectories in parallel, thereby allowing training on GPUs. Note that this is not a very restrictive assumption as we still allow the individual $\mathcal{N}_i(t)$ to change across time.

A. Time Discretization

In order to use deep learning, we consider a Euler-Maruyama time discretization of (9) and (10) so that back-propagation through time can be performed on a finite number of time steps to train the deep network. This is similar to past deep FBSDE works [33, 34, 35] wherein using a finite time interval of Δt, the time horizon is divided into equal intervals of length $\frac{T}{\Delta t}$. The time-discretized equations are,

$$\mathbf{x}[\tau+1] = \mathbf{x}[\tau] + (\mathbf{f} + \mathbf{G}\mathbf{u}^*)\Delta t + \boldsymbol{\Sigma}\sqrt{\Delta t}\,\boldsymbol{\epsilon} \tag{20}$$

$$V[\tau+1] = V[\tau] - \left[\sum_{i=1}^{N}\left(q_i + \frac{1}{2}\mathbf{u}_i^{*\mathrm{T}}\mathbf{R}_i\mathbf{u}_i^*\right)\right]\Delta t + \frac{\partial V}{\partial \mathbf{x}}^{\mathrm{T}}\boldsymbol{\Sigma}\sqrt{\Delta t}\,\boldsymbol{\epsilon} \tag{21}$$

where $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ and $\tau \in [\![1, \frac{T}{\Delta t}]\!]$.

B. Forward Pass

Similar to past work [33, 35], the forward pass involves propagating the discretized FBSDE and BSDE forward in time using (20) and (21). The primary distinction between [35] and our approach is a new implicit safe layer based on CADMM. This difference is clear by comparing the unrolled compute graph shown in figure 2 with that of [35, Figure 1]. The additional difference from past works is the inclusion of extra fully-connected layers FC_c, FC_h and FC_V to allow for training from random initial conditions. These extra networks serve to initialize the initial cell-state and initial hidden-state of the LSTM layers and the initial value-function respectively.

C. Backward Pass

Our proposed decentralized Merged CADMM-OSQP solver is an instantiation of an implicit neural network layer. Hence, we utilize the implicit function theorem to compute the necessary gradients for the backward pass.

To compute the gradients, we solve a KKT system resulting from the following QP,

$$\begin{aligned} &\min_{\mathbf{u}} \ \mathcal{H}(\mathbf{u}) \\ &\text{s.t. } \mathbf{C}\mathbf{u} \leq \mathbf{d} \end{aligned} \tag{22}$$

where $\mathbf{d} = [\mathbf{d}_1; \mathbf{d}_2; \ldots; \mathbf{d}_{Nr}]$ and $\mathbf{C}$ can be constructed from the neighborhood constraint matrices $\mathbf{A}_i$ (please see supplementary material). As a result, $\mathbf{C}$ will contain a subset of the original N_{ineq} constraints. Additionally, if two agents i, j are mutual neighbors, i.e., $j \in \mathcal{N}_i$ and $i \in \mathcal{N}_j$, then the constraint involving i and j will appear twice in $\mathbf{C}$ (i.e., as duplicate row entries). We therefore refer to (22) as the *Reduced Duplicate Centralized Problem*. There are two reasons we choose this problem for solving the backward pass over the original *Reduced Non-Duplicate Centralized Problem* (RNDCP):

1) Construction of the non-duplicate constraint matrix $\bar{\mathbf{C}}$ requires checking if mutual neighbors exist for every agent which does not scale for large N
2) $\mathbf{C}$ can be easily constructed using the matrices $\mathbf{A}_i$ used in the forward pass of Merged CADMM-OSQP

To justify the usage of (22), we make the following assumption about the corresponding RNDCP:

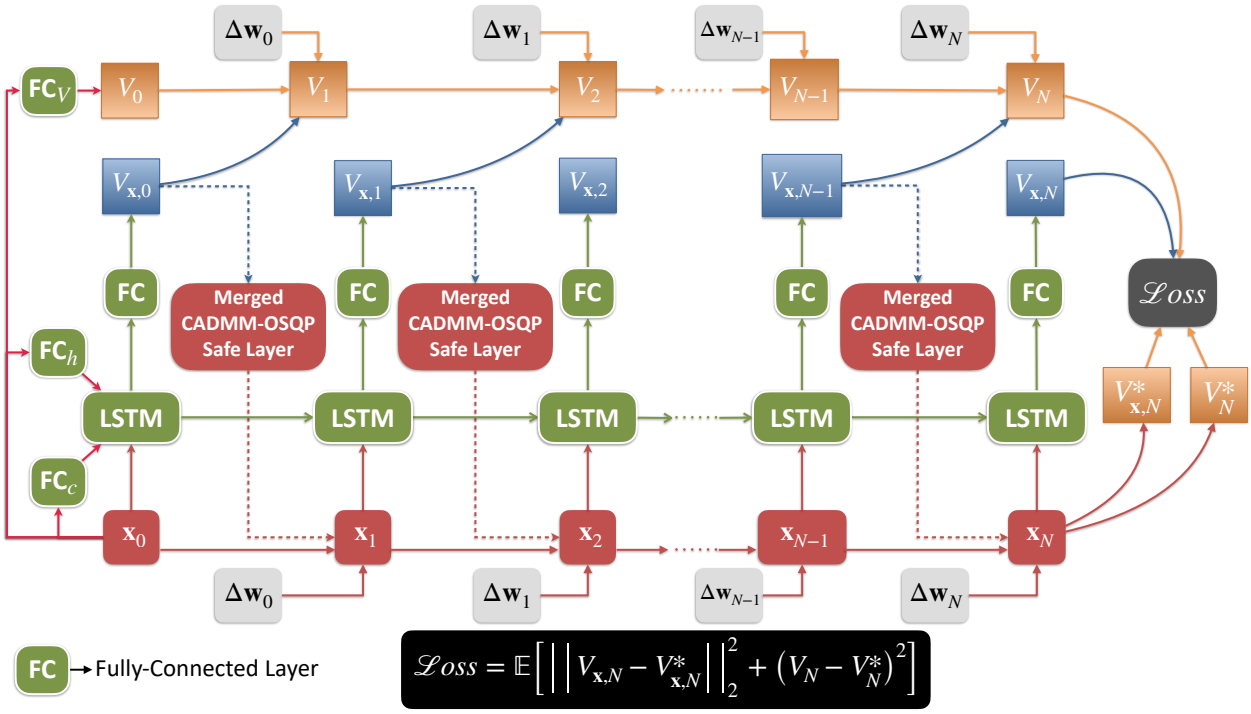


Fig. 2: Unrolled Deep FBSDE compute graph using the proposed Merged CADMM-OSQP implicit safe layer.

Assumption 3. *The QP of the non-duplicate problem satisfies the LICQ (Linear Independence Constraint Qualification), i.e., the active constraints of the matrix $\bar{\mathbf{C}}$ are linearly independent.*

Constraint qualifications are necessary for optimal solutions to constrained optimization problems to satisfy the KKT conditions. LICQ is one of the most frequently used constraint qualification in optimization literature [31]. In our case, we additionally rely on LICQ to ensure that the Lagrange multipliers satisfying the KKT conditions are unique [46, Section 3]. The connection between the duplicate problem (22) and the corresponding RNDCP is shown in the following lemma.

Lemma 1. *Let $\mathcal{D} := \{\mathcal{D}_i\}_{i=1}^{m}$ with $\mathcal{D}_i := \{\mathbf{c}_{d_{i,j}}\}_{j=1}^{n_i}$ denote the set of equivalence classes induced by the equality equivalence relation*

$$\mathbf{c}_{d_{i,a}} \sim \mathbf{c}_{d_{i,b}} \iff \mathbf{c}_{d_{i,a}} = \mathbf{c}_{d_{i,b}}, \quad \forall a, b \in [\![1, n_i]\!] \, \forall i \tag{23}$$

where $d_{i,j}$ denotes the row corresponding to the jth instance of the ith unique constraint. In other words, all constraint rows in $\mathcal{D}_i$ for a particular i refer to the same constraint and n_i denotes the number of duplicates of constraint $\bar{C}_i$. Suppose that $\mathbf{R}$ is positive definite and LICQ holds for the non-duplicate problem. Then,

$$\lambda_{nondup,i} = \sum_{j=1}^{n_i} \lambda_{dup,d_{i,j}} \tag{24}$$

$$\lambda_{nondup,i} \, \mathrm{d}\lambda_{nondup,i} = \sum_{j=1}^{n_i} \lambda_{dup,d_{i,j}} \, \mathrm{d}\lambda_{dup,d_{i,j}} \tag{25}$$

where $\boldsymbol{\lambda}_{dup}, \boldsymbol{\lambda}_{non\text{-}dup}$ denotes the Lagrange multipliers and $\mathrm{d}\boldsymbol{\lambda}_{dup}, \mathrm{d}\boldsymbol{\lambda}_{non\text{-}dup}$ denotes the variables of the KKT system for the duplicate and non-duplicate problems respectively.

The above lemma then allows to prove the following theorem.

Theorem 1. *Let $\mathbf{M}$ denote the matrix describing the relationship between the duplicate and non-duplicate constraints:*

$$\mathbf{C} = \mathbf{M}\bar{\mathbf{C}}, \quad \mathbf{d} = \mathbf{M}\bar{\mathbf{d}} \tag{26}$$

Then, for loss ℓ, the gradients $\nabla_{\mathbf{R}}\ell, \nabla_{\mathbf{q}}\ell, \nabla_{\bar{\mathbf{C}}}\ell, \nabla_{\bar{\mathbf{d}}}\ell$ coincide for the duplicate and non-duplicate problems and are unique.

Proof Sketch: We first show that the gradients $\nabla_{\mathbf{R}}\ell, \nabla_{\mathbf{q}}\ell, \nabla_{\bar{\mathbf{C}}}\ell, \nabla_{\bar{\mathbf{d}}}\ell$ depend only on $\mathrm{d}\mathbf{u}$ and the sums of $\boldsymbol{\lambda}$ and $\mathrm{d}\boldsymbol{\lambda}$ associated to each unique constraint of matrix $\bar{\mathbf{C}}$. Applying Lemma 1 then shows that the theorem holds. ■

We refer the reader to our supplementary for a proof of Lemma 1 and Theorem 1. In practice, there may be situations at certain time steps when Assumption-3 is violated, in which case the computed gradient serves as a noisy version of the true gradient.

D. Penalty Parameters Adaptation

The selection of the penalty parameters ρ and μ is important since the former encourages the satisfaction of the local constraints of each agent subproblem, while the latter encourages achieving consensus. Low values of these parameters could result to slow convergence of the algorithm or unsafe solutions. On the other hand, if their values are too high then

their corresponding terms in (17) will dominate the objective function. To accommodate for this, we adopt the following adaptation schemes for the penalty parameters from OSQP [44]:

$$\rho^{l+1} = \rho^l \sqrt{\frac{r^l_{pri,1}/\kappa^l_{pri,1}}{r^l_{dual,1}/\kappa^l_{dual,1}}}, \quad \mu^{l+1} = \mu^l \sqrt{\frac{r^l_{pri,2}/\kappa^l_{pri,2}}{r^l_{dual,2}/\kappa^l_{dual,2}}}.$$

E. Additional Constraints for Local Subproblems

In the algorithm proposed in Section V, achieving consensus—and thus ensuring the safety of the agents—fully relies on ADMM. To facilitate reaching a consensus, we suggest including in every subproblem of agent i: i) the obstacle avoidance constraints of its neighbors, and/or ii) the inter-agent constraints between its neighbors. The number of these additional constraints will be rN_o and $\binom{r}{2}$, respectively. Even after incorporating these constraints, we emphasize that, for $r \ll N$, the new $N_{\text{ineq},i} = r + N_o + rN_o + \binom{r}{2}$ constraints in each agent's local QP will still be substantially smaller than the $N_{\text{ineq}} = \binom{N}{2} + NN_o$ of the centralized problem.

VII. Simulation Results

We test the proposed approach on a system consisting of multiple agents with unicycle dynamics (similar to [50, Section V.B]) for any agent i given by,

$$\dot{x}_i = v_i \cos(\theta_i), \quad \dot{y}_i = v_i \sin(\theta_i), \quad \dot{\theta}_i = v_i u^\theta_i, \quad \dot{v}_i = u^v_i$$

Constructing a state vector $\mathbf{x}_i = [x_i;\, y_i;\, \theta_i;\, v_i]$ and a control vector $\mathbf{u}_i = [u^\theta_i;\, u^v_i]$ and assuming that noise only enters the acceleration channels, the stochastic dynamics for agent i can be written as,

$$\mathrm{d}\mathbf{x}_i = \mathbf{f}_i \mathrm{d}t + \mathbf{G}_i \mathbf{u}_i \mathrm{d}t + \mathbf{\Sigma}_i \mathrm{d}\mathbf{w}_i$$

where $\mathbf{f}_i$, $\mathbf{G}_i$, and, $\mathbf{\Sigma}_i$ are given by,

$$\mathbf{f}_i = \begin{bmatrix} v_i \cos(\theta_i) \\ v_i \sin(\theta_i) \\ 0 \\ 0 \end{bmatrix}, \mathbf{G}_i = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ v_i & 0 \\ 0 & 1 \end{bmatrix}, \mathbf{\Sigma}_i = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & \sigma & 0 \\ 0 & 0 & 0 & \sigma \end{bmatrix}$$

For our simulations we consider obstacle avoidance and collision avoidance safety constraints in four different types of tasks. These constraints are stated as functions of the positions of the agents and are therefore relative degree 2 (i.e., one needs to differentiate twice before the control shows up). However, the aforementioned SCBF in (7) assumes that the relative degree of h is 1. This is required to ensure that $\frac{\partial B}{\partial \mathbf{x}}^{\mathrm{T}} \mathbf{G}$ is not zero. Therefore, the original position constraint, which we hereon refer to as $h_{\text{pos}}(\mathbf{x})$, must be modified to ensure that the modified function has relative degree 1. In [35], this modification takes the following form,

$$h(\mathbf{x}) = h_{\text{pos}}(\mathbf{x}) - \mu\, v^2 \tag{27}$$

The parameter μ controls how fast the system can safely move inside the safe set and thereby, the implication of adding the $-\mu\, v^2$ term introduces constraints on the velocity in addition to the original position constraint. Intuitively, when $h_{\text{pos}} = 0$, for the system to stay safe (i.e., $h \geq 0$), the only allowable safe velocity is $v = 0$.

The main drawbacks of (27) are that it does not take into account the heading of the agents and that it penalizes positive and negative velocities equally. To overcome these we propose the following two types of barrier functions,

1) **Type-A**: This is a pairwise safety constraint concerning two agents and is constructed as follows:

$$h^A(\mathbf{x}) = h^A_{\text{pos}}(\mathbf{x}) - \mu\,(v_i\, \text{IP}_i + v_j\, \text{IP}_j) \tag{28}$$
$$\text{where, } h^A_{\text{pos}}(\mathbf{x}) = \frac{1}{2}\Big((x_i - x_j)^2 + (y_i - y_j)^2 - 4r^2\Big),$$
$$\text{IP}_i = \bar{p}^{\mathrm{T}}_{ij}\bar{\theta}_i, \text{ and, } \text{IP}_j = \bar{p}^{\mathrm{T}}_{ji}\bar{\theta}_j$$

   for any two agents i and j, each with a radius of r. The inner-product (IP) terms depend on vectors $\bar{p}_{ij}$ and $\bar{p}_{ji}$ which denote relative position vectors from i to j and j to i respectively and on the vectors $\bar{\theta}_i = [\cos\theta_i;\, \sin\theta_i]$ and $\bar{\theta}_j = [\cos\theta_j;\, \sin\theta_j]$ which denote unit vectors along the headings of agents i and j respectively.

2) **Type-B**: This type of safety constraint concerns an agent i and an obstacle o. It is constructed as follows,

$$h^B(\mathbf{x}) = h^B_{\text{pos}}(\mathbf{x}) - \mu\, v_i\, \text{IP} \tag{29}$$
$$\text{where, } h^B_{\text{pos}}(\mathbf{x}) = \frac{1}{2}\Big((x_i - x_o)^2 + (y_i - y_o)^2 - (r_i + r_o)^2\Big),$$
$$\text{and IP} = \bar{p}^{\mathrm{T}}_{io}\theta_i \tag{30}$$

   where, (x_o, y_o) is the position of the obstacle's center and r_o is the obstacle's radius. The vectors $\bar{p}_{io}$ and θ_i are defined similar to the type-A constraint above.

Similar to work in [35] our simulations are also conducted in a safe reinforcement learning setting where it is required to be safe not only during inference but also during the entire training process. The reader is encouraged to refer to the video included in the supplementary material to support this claim. The video shows the gradual emergence of optimal behavior as iterations progress, for each of the tasks described below, while staying safe during the entire training process. Additionally, the video depicts the behavior of a single batch instance on the left and distributions over entire batches on the right wherein agents are depicted as particles. The video on the right also demonstrates successful performance on average as justified by our choice of the mean cost function (3).

A. Swapping Task

We first consider the swapping cars task presented in [35]. We show that our proposed approach is not only scalable to larger teams of cars but can handle added complexity of obstacle avoidance. Each agent's goal is to swap positions with the diametrically opposite one while avoiding collisions. In Fig. 4a, the distributions of the positions show that the agents avoid collisions with a high probability. On closer inspection, one can argue that the problem is highly symmetrical as no matter where you stand on the initial circle, each agent effectively solves a similar problem. Therefore, one could be lead to believe that this is not a difficult problem for a

deep network to solve. To prove that our approach can handle more complex scenarios, we added extra obstacles to create an asymmetrical version of the same problem. The final policy is depicted in Fig. 4b. As seen in the figure, the cars on the top right quadrant encounter obstacles much sooner along with encountering their neighbors. Observing the second plot of Fig. 4b, we see that the final policy now leads to the agents circling around the new highly non-convex obstacle shape in order to complete the task.

B. Bottleneck Task

For this task the agents are required to pass through the bottleneck in the center (created by multiple circular obstacles) and achieve a desired formation on the other side of the bottleneck as seen in Fig. 5a. To train this policy we designed the running cost $q_i(\mathbf{x})$, and terminal cost $\phi_i(\mathbf{x})$, such that the task is divided into two objectives - (i.) pass through the bottleneck, and, (ii.) achieve the desired formation. This was done to *encourage* the agents to prioritize passing through the bottleneck, before attempting to achieve the desired formation on the other side. Without the first objective, many agents end up getting stuck in the highly non-convex regions between the circular obstacles while others pass through the bottleneck and move to their respective targets. Thus, the first objective helps avoid these undesirable local minima. The first objective was achieved by setting a target for the agents in the column close to the bottleneck at a point on the x-axis further away from the bottleneck and setting a target for the agents in the second column at a point on the x-axis close to the bottleneck. This *intermediate target* was set for 80% of the time horizon. For the remaining 20%, the targets were set so as to achieve the desired formation on the other side. For this task, we chose $r = 3$ with each neighborhood containing r type-A constraints between ego agent and neighbors and 6 type-B constraints between the ego agent and the obstacles.

C. Moving-obstacle (or uncooperative agent) Task

The goal of this task is the same as the swapping task with the added complexity of a moving obstacle. The moving obstacle can be interpreted as an uncooperative agent (i.e., one that cannot be controlled) and moves from bottom to top along the y-axis as seen in Fig. 5b. To train this policy, we first trained a policy containing the 8 agents without any obstacle on the swapping task. This pre-trained policy was then used as an initialization to train the policy to swap while avoiding an oncoming moving obstacle. The reason for this two step process being that a completely random initial policy is unable to safely explore in the presence of the moving obstacle. A common scenario that arises when one attempts to train the policy from scratch is that the agents try to directly move to their diametrically opposite targets but then come to a halt and congregate around the center because they encounter other agents. However, as the moving obstacle approaches, the closest agent to the obstacle is inevitably *run-over* by the moving obstacle, as it has no where to escape. Using a pre-trained policy embeds the agents with the ability to *move away from the moving obstacle's path* as they have pre-learned a behavior to initiate a coordinated turn in order to avoid colliding with other agents. For this task, we used a neighborhood size of $r = 3$ with r type-A constraints between the ego-agent and its neighbors and $r + 1$ type-B constraints for the ego-agent and its neighbors.

D. Large-Scale Formation Task

Here, we demonstrate the scalability of our framework by considering a task where a large-scale team of 32 agents must reach a to desired rectangular formation. Each agent has $r = 6$ neighbors. The obstacle avoidance constraints for the neighbors are also taken into consideration in every local subproblem to facilitate reaching a consensus. As shown in Fig. 5c, the distributions of the positions of agents successfully reach close to the desired targets.

VIII. Discussion

A. Constraints Reduction and Increased Memory Efficiency

In Table I, we compare the number of constraints considered by the centralized and decentralized approaches for each task presented in Section VII. Clearly, there is a substantial reduction on the number of constraints when using the proposed approach, implying that it is more scalable to large-scale systems than the equivalent centralized one. Table II also demonstrates the reduced required memory usage and training iteration time of our approach.

B. Low Position Constraint Violation

In Figure 3, the top and bottom plots show the fraction of batch instances where $h < 0$ and $h_{\text{pos}} < 0$, respectively, against the number of training iterations. For all tasks, $h_{\text{pos}} < 0$ occurs much less frequently as compared to $h < 0$. This indicates that although there are many instances of h violations (i.e., agents moving with *unsafe* velocities in the vicinity of other agents or other obstacles), the number of physical collisions between agents or between agents and obstacles are much lower or even zero for some tasks. Thus, the new type-A and type-B barrier formulations allow for more aggressive behavior as compared to (27) which instead encourages conservative behaviors and therefore cannot scale for large numbers of agents. Another interesting observation is the sharp decline in h_{pos} violations after around 40% of the total number of iterations. An intuitive explanation of this is that position violations only occur during the initial exploration phase.

IX. Conclusion

In this work, we introduced a novel and scalable deep-learning-based framework that extends the existing deep FBSDE framework to decentralized multi-agent safe stochastic optimal control problems. To achieve this, we proposed a new stochastic CBF formulation which encourages aggressive motion of moving agents, while still guaranteeing their safe operation with a high probability. Furthermore, we reformulated the multi-agent per-time-step Hamiltonian minimization problem into a decentralized version, which we solve by

Task	Number of Agents	Decentralized	Centralized
Swapping	16	5	136
Bottleneck	8	10	76
Moving obstacle	8	9	36
Large-scale formation	32	16	560

TABLE I: Comparison of number of constraints between the centralized problem and each local subproblem of the proposed decentralized approach.

	Max Memory Allocated (MiB)			Time per Training Iteration (s)		
Batch Size	Decentralized	Centralized	Percent Reduction	Decentralized	Centralized	Percent Reduction
32	1586	17879	-91.13%	14.75	1306	-98.87%
64	3124	N/A	N/A	17.92	N/A	N/A
384	18510	N/A	N/A	38.30	N/A	N/A

TABLE II: Comparison of memory usage (maximum of 10 iterations) and time per iteration (average over 10 iterations) between the decentralized and centralized formulations. The N/A values for batch sizes 64 and 384 indicate that the computer ran out of memory and was unable to run a single iteration.

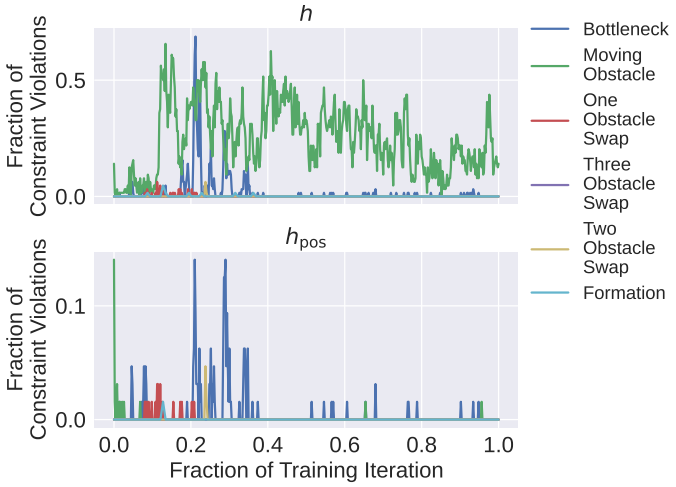


Fig. 3: Constraint Violations

proposing a novel distributed ADMM-based method for large-scale quadratic optimization problems. The framework was successfully tested in simulation on several challenging multi-vehicle tasks with higher complexity and more agents as compared to previous work using a centralized approach and simpler tasks. The results demonstrate that the agents maintain their safe operation during training, thus the framework can also be appreciated from a safe reinforcement learning perspective. Finally, the scalability of the approach in terms of memory usage and training time is also validated.

References

[1] Akshay Agrawal, Brandon Amos, Shane Barratt, Stephen Boyd, Steven Diamond, and Zico Kolter. Differentiable convex optimization layers. *arXiv preprint arXiv:1910.12430*, 2019.

[2] Javier Alonso-Mora, Eduardo Montijano, Mac Schwager, and Daniela Rus. Distributed multi-robot formation control among obstacles: A geometric and optimization approach with consensus. In *2016 IEEE international conference on robotics and automation (ICRA)*, pages 5356–5363. IEEE, 2016.

[3] Aaron D Ames, Xiangru Xu, Jessy W Grizzle, and Paulo Tabuada. Control barrier function based quadratic programs for safety critical systems. *IEEE Transactions on Automatic Control*, 62(8):3861–3876, 2016.

[4] Aaron D Ames, Samuel Coogan, Magnus Egerstedt, Gennaro Notomista, Koushil Sreenath, and Paulo Tabuada. Control barrier functions: Theory and applications. In *2019 18th European Control Conference (ECC)*, pages 3420–3431. IEEE, 2019.

[5] Brandon Amos and J Zico Kolter. Optnet: Differentiable optimization as a layer in neural networks. In *International Conference on Machine Learning*, pages 136–145. PMLR, 2017.

[6] Brandon Amos and Denis Yarats. The differentiable cross-entropy method. In *International Conference on Machine Learning*, pages 291–302. PMLR, 2020.

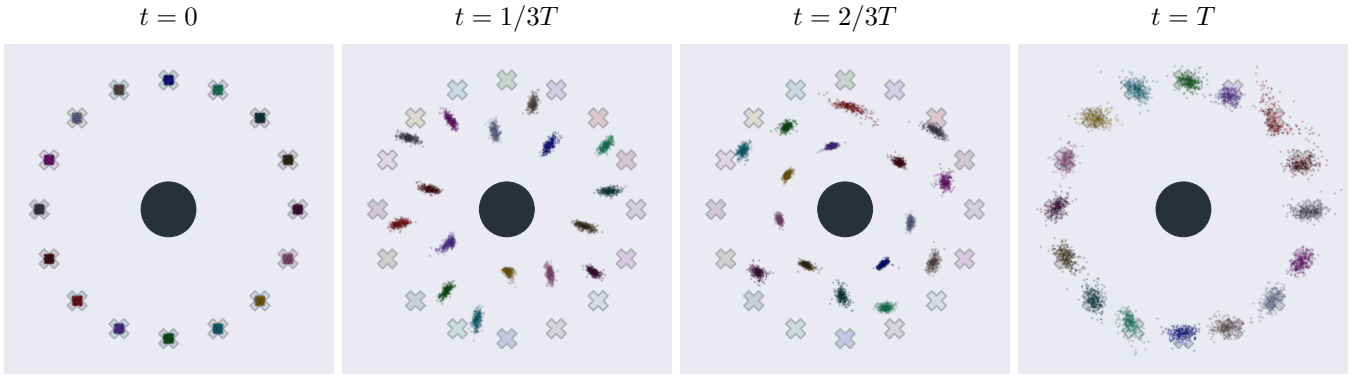
[7] Brandon Amos, Ivan Dario Jimenez Rodriguez, Jacob Sacks, Byron Boots, and J Zico Kolter. Differentiable mpc for end-to-end planning and control. *arXiv preprint arXiv:1810.13400*, 2018.

[8] Shaojie Bai, J Zico Kolter, and Vladlen Koltun. Deep equilibrium models. *arXiv preprint arXiv:1909.01377*, 2019.

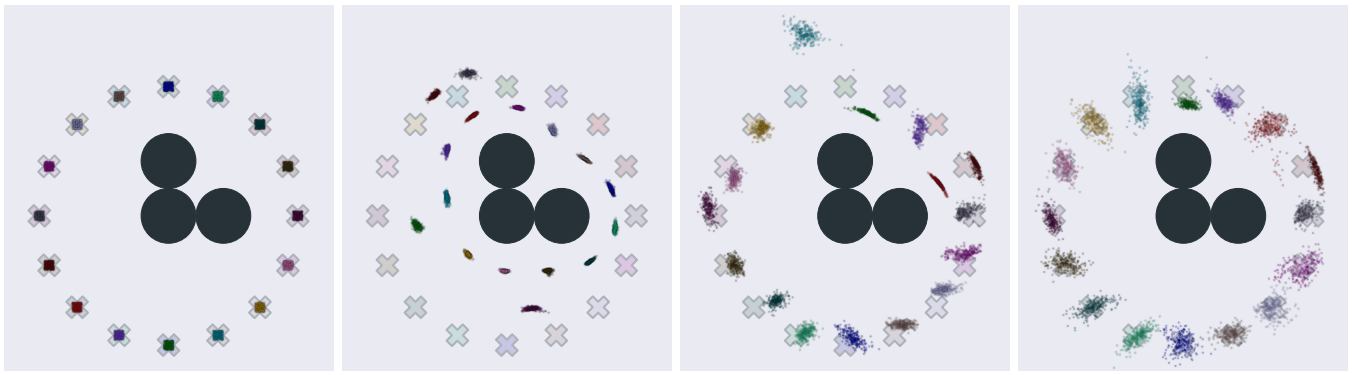
[9] Stephen Boyd, Neal Parikh, and Eric Chu. *Distributed optimization and statistical learning via the alternating direction method of multipliers*. Now Publishers Inc, 2011.

[10] Ricky TQ Chen, Yulia Rubanova, Jesse Bettencourt, and David Duvenaud. Neural ordinary differential equations. *arXiv preprint arXiv:1806.07366*, 2018.

[11] Tianrong Chen, Ziyi O Wang, Ioannis Exarchos, and Evangelos Theodorou. Large-scale multi-agent deep fbsdes. In *International Conference on Machine Learning*, pages 1740–1748. PMLR, 2021.



(a) Symmetrical Swapping Task with One Obstacle and 16 Agents.



(b) Unsymmetrical Swapping Task with Three Obstacles and 16 Agents.

Fig. 4: In all figures, the black circles indicate obstacles and the **X** markers indicate target positions for each agent with the same color as the marker. The snapshots demonstrate the positions of each agent for each batch at time instants $t = 0, 1/3T, 2/3T, T$ with the fully trained policy. The proposed approach can not only handle symmertical problems similar to [35], but can also successfully solve unsymmertical problems.

[12] Richard Cheng, Gábor Orosz, Richard M Murray, and Joel W Burdick. End-to-end safe reinforcement learning through barrier functions for safety-critical continuous control tasks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 3387–3395, 2019.

[13] Zilong Cheng, Jun Ma, Xiaoxue Zhang, Clarence W de Silva, and Tong Heng Lee. Admm-based parallel optimization for multi-agent collision-free model predictive control. *arXiv preprint arXiv:2101.09894*, 2021.

[14] Andrew Clark. Control barrier functions for stochastic systems. *Automatica*, 130:109688, 2021.

[15] Jorge Cortes, Sonia Martinez, Timur Karatas, and Francesco Bullo. Coverage control for mobile sensing networks. *IEEE Transactions on robotics and Automation*, 20(2):243–255, 2004.

[16] Ioannis Exarchos and Evangelos A Theodorou. Stochastic optimal control via forward and backward stochastic differential equations and importance sampling. *Automatica*, 87:159–165, 2018.

[17] Ioannis Exarchos, Evangelos A Theodorou, and Panagiotis Tsiotras. Game-theoretic and risk-sensitive stochastic optimal control via forward and backward stochastic differential equations. In *2016 IEEE 55th Conference on Decision and Control (CDC)*, pages 6154–6160. IEEE, 2016.

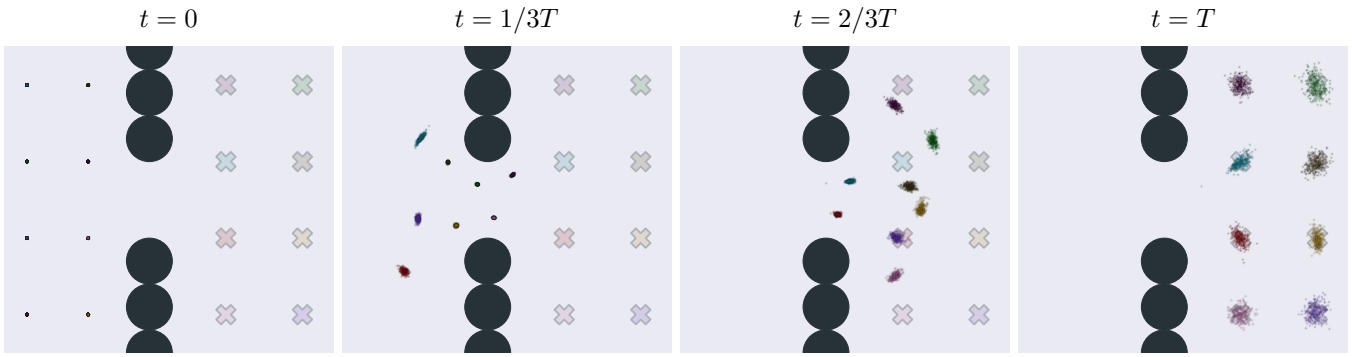
[18] Ioannis Exarchos, Evangelos A Theodorou, and Panagiotis Tsiotras. Stochastic l1-optimal control via forward and backward sampling. *Systems & Control Letters*, 118: 101–108, 2018.

[19] Ioannis Exarchos, Evangelos Theodorou, and Panagiotis Tsiotras. Stochastic differential games: A sampling approach via fbsdes. *Dynamic Games and Applications*, 9(2):486–505, 2019.

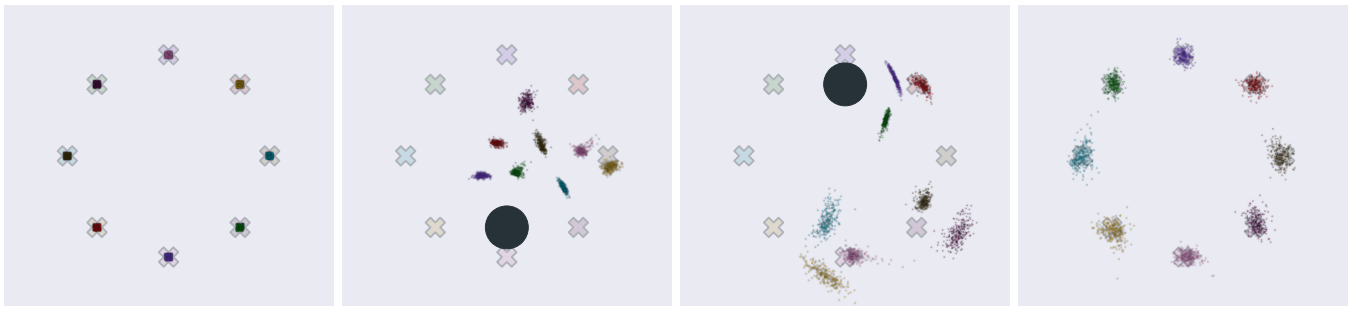
[20] Ioannis Exarchos, Marcus A Pereira, Ziyi Wang, and Evangelos A Theodorou. Novas: Non-convex optimization via adaptive stochastic search for end-to-end learning and control. *arXiv preprint arXiv:2006.11992*, 2020.

[21] Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *International Conference on Machine Learning*, pages 1126–1135. PMLR, 2017.

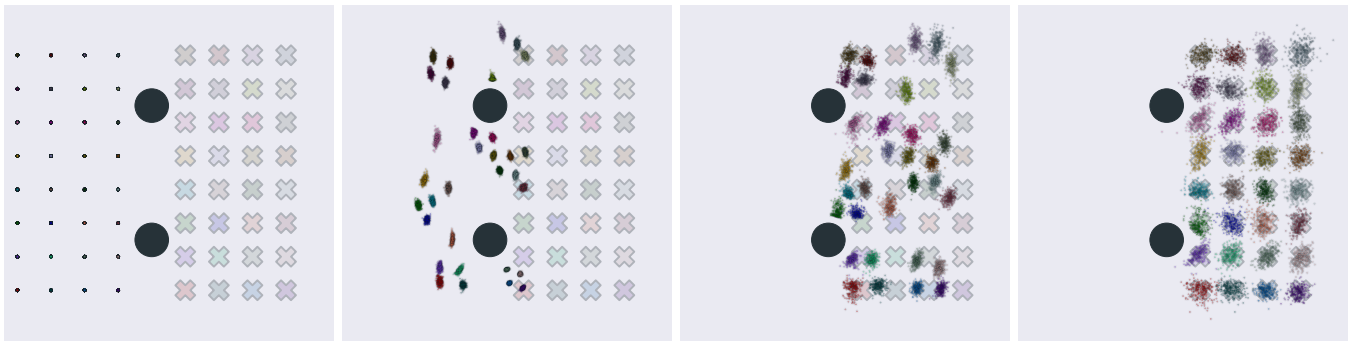
[22] Trevor Halsted, Ola Shorinwa, Javier Yu, and Mac Schwager. A survey of distributed optimization methods for multi-robot systems. *arXiv preprint arXiv:2103.12840*, 2021.



(a) Bottleneck task with 8 agents.



(b) Swapping task with 8 agents and a moving obstacle (uncooperative agent).



(c) Large-scale formation task with 32 agents.

Fig. 5: Similar to Fig. 4, black circles indicate obstacles and the **X** markers indicate target positions. The top and bottom rows involve static obstacles, while the center row considers a moving obstacle.

[23] Jiequn Han, Arnulf Jentzen, and E Weinan. Solving high-dimensional partial differential equations using deep learning. *Proceedings of the National Academy of Sciences*, 115(34):8505–8510, 2018.

[24] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[25] Harold J Kushner. Stochastic stability and control. Technical report, Brown Univ Providence RI, 1967.

[26] Viet-Anh Le and Truong X Nghiem. Gaussian process based distributed model predictive control for multi-agent systems using sequential convex programming and ADMM. In *2020 IEEE Conference on Control Technology and Applications (CCTA)*, pages 31–36. IEEE, 2020.

[27] Xuechen Li, Ting-Kam Leonard Wong, Ricky TQ Chen, and David Duvenaud. Scalable gradients for stochastic differential equations. In *International Conference on Artificial Intelligence and Statistics*, pages 3870–3882. PMLR, 2020.

[28] Timothy P Lillicrap, Jonathan J Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*, 2015.

[29] Jacob Mattingley and Stephen Boyd. Cvxgen: A code generator for embedded convex optimization. *Optimization and Engineering*, 13(1):1–27, 2012.

[30] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*, 2013.

[31] Jorge Nocedal and Stephen Wright. *Numerical optimization*. Springer Science & Business Media, 2006.

[32] Reza Olfati-Saber. Flocking for multi-agent dynamic systems: Algorithms and theory. *IEEE Transactions on automatic control*, 51(3):401–420, 2006.

[33] Marcus Pereira, Ziyi Wang, Ioannis Exarchos, and Evangelos A Theodorou. Learning deep stochastic optimal control policies using forward-backward sdes. *arXiv preprint arXiv:1902.03986*, 2019.

[34] Marcus Pereira, Ziyi Wang, Tianrong Chen, Emily Reed, and Evangelos Theodorou. Feynman-kac neural network architectures for stochastic control using second-order fbsde theory. In *Learning for Dynamics and Control*, pages 728–738. PMLR, 2020.

[35] Marcus Aloysius Pereira, Ziyi Wang, Ioannis Exarchos, and Evangelos A Theodorou. Safe optimal control using stochastic barrier functions and deep forward-backward sdes. *arXiv preprint arXiv:2009.01196*, 2020.

[36] V. Rostampour and T. Keviczky. Distributed stochastic model predictive control for large-scale linear systems with private and common uncertainty sources, 2019.

[37] Cesar Santoyo, Maxence Dutreix, and Samuel Coogan. A barrier function approach to finite-time stochastic system verification and control. *Automatica*, 125:109439, 2021.

[38] Augustinos D Saravanos, Alexandros Tsolovikos, Efstathios Bakolas, and Evangelos Theodorou. Distributed Covariance Steering with Consensus ADMM for Stochastic Multi-Agent Systems. In *Proceedings of Robotics: Science and Systems*, Virtual, July 2021. doi: 10.15607/RSS.2021.XVII.075.

[39] Meenakshi Sarkar, Debasish Ghose, and Evangelos A Theodorou. High-relative degree stochastic control lyapunov and barrier functions. *arXiv preprint arXiv:2004.03856*, 2020.

[40] Steven E Shreve. *Stochastic calculus for finance II: Continuous-time models*, volume 11. Springer Science & Business Media, 2004.

[41] David Silver, Thomas Hubert, Julian Schrittwieser, Ioannis Antonoglou, Matthew Lai, Arthur Guez, Marc Lanctot, Laurent Sifre, Dharshan Kumaran, Thore Graepel, et al. Mastering chess and shogi by self-play with a general reinforcement learning algorithm. *arXiv preprint arXiv:1712.01815*, 2017.

[42] David Silver, Thomas Hubert, Julian Schrittwieser, Ioannis Antonoglou, Matthew Lai, Arthur Guez, Marc Lanctot, Laurent Sifre, Dharshan Kumaran, Thore Graepel, et al. A general reinforcement learning algorithm that masters chess, shogi, and go through self-play. *Science*, 362(6419):1140–1144, 2018.

[43] B. Stellato, G. Banjac, P. Goulart, A. Bemporad, and S. Boyd. OSQP: an operator splitting solver for quadratic programs. *Mathematical Programming Computation*, 12 (4):637–672, 2020. doi: 10.1007/s12532-020-00179-2. URL https://doi.org/10.1007/s12532-020-00179-2.

[44] Bartolomeo Stellato, Goran Banjac, Paul Goulart, Alberto Bemporad, and Stephen Boyd. Osqp: An operator splitting solver for quadratic programs. *Mathematical Programming Computation*, 12(4):637–672, 2020.

[45] Wentao Tang and Prodromos Daoutidis. Fast and stable nonconvex constrained distributed optimization: the ellada algorithm. *Optimization and Engineering*, pages 1–43, 2021.

[46] Gerd Wachsmuth. On licq and the uniqueness of lagrange multipliers. *Operations Research Letters*, 41(1):78–80, 2013.

[47] Li Wang, Evangelos A Theodorou, and Magnus Egerstedt. Safe learning of quadrotor dynamics using barrier certificates. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2460–2465. IEEE, 2018.

[48] Ziyi Wang, Keuntaek Lee, Marcus A Pereira, Ioannis Exarchos, and Evangelos A Theodorou. Deep forward-backward sdes for min-max control. In *2019 IEEE 58th Conference on Decision and Control (CDC)*, pages 6807–6814. IEEE, 2019.

[49] Wei Xiao and Christos G Cassandras. Decentralized optimal merging control for connected and automated vehicles. In *2019 American Control Conference (ACC)*, pages 3315–3320. IEEE, 2019.

[50] Zhaoming Xie, C Karen Liu, and Kris Hauser. Differential dynamic programming with nonlinear constraints. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pages 695–702. IEEE, 2017.

[51] Shakiba Yaghoubi, Keyvan Majd, Georgios Fainekos, Tomoya Yamaguchi, Danil Prokhorov, and Bardh Hoxha. Risk-bounded control using stochastic barrier functions. *IEEE Control Systems Letters*, 5(5):1831–1836, 2020.