IEEE Control Systems Letters paper presented at
2023 American Control Conference (ACC)
San Diego, CA, USA. May 31 - June 2, 2023

Federated reinforcement learning for generalizable motion planning

Zhenyuan Yuan[1] Siyuan Xu[1] Minghui Zhu[1]

Abstract— **This paper considers the problem of learning a control policy that generalize well to novel environments given a set of sample environments. We develop a federated learning framework that enables collaborative learning of multiple learners and a centralized server without sharing their raw data. In each iteration, each learner uploads its local control policy and the corresponding estimated normalized arrival time to the server, which then computes the global optimum among the learners and broadcasts the optimal policy to the learners. Each learner then selects between its local control policy and that from the server for next iteration. By leveraging generalization error, our analysis shows that the proposed framework is able to provide generalization guarantees on arrival time and safety as well as consensus at global optimal value in the limiting case. Monte Carlo simulation is conducted for evaluation.**

I. Introduction

Motion planning is a fundamental problem in robotics, and it aims to generate a series of low-level specifications for a robot to move from one point to another [1]. In the real world, robots' operations are usually accompanied by uncertainties, e.g., from the environments they operate in and from the errors in the modeling of robots' dynamics. To deal with the uncertainties, a number of existing methods leverage techniques in robust control (e.g., [2] [3]) and stochastic control (e.g., [4] [5]). Recently, learning-based approaches have been developed to relax the need of prior explicit uncertainty models by directly learning the best mapping from sensory inputs to control inputs from repetitive trials. For example, paper [6] uses kernel methods to learn the control policy for a spider-like robot with 18 degrees of freedom using GPS inputs. Deep neural network is used in [7], [8] to plan a sequence of actions toward goals using camera inputs. A combination of convolutional neural network and recurrent neural network is employed in [9] to learn the best mapping from 2D LiDAR inputs to steering angles for a scaled autonomous car for navigation.

This paper focuses on the generalizability of learning-based approaches. Expected cost minimization is widely adopted to train machine learning models generalizable to unseen environments, which follow the same distribution as training environments [10] [11]. Since the distribution of the data is generally unknown, these methods instead solve an empirical mean minimization problem (possibly with regularization) given a finite amount of training data. Related methods can be categorized into two classes. The first one is modifying an expected cost function and solving the modified problem through empirical cost minimization [12]. The other class is incorporating regularizers into empirical mean minimization to improve the generalizability of the solution. A necessarily incomplete list of references includes [8], [13]. While most regularization methods are heuristic, paper [14] uses the sum of the empirical cost and the generalization error from PAC-Bayes theory as an upper bound of the expected cost and synthesizes a controller which can minimize the upper bound. Nevertheless, empirical mean minimization (with regularization) is an approximation to the expected cost minimization problem, and the optimality loss is unknown. In this paper, we aim to directly solve the expected cost minimization problem and analyze the optimality of the solution.

The papers aforementioned focus on centralized learning, where all the training data are possessed by a single learning agent. On the other hand, the advent of ubiquitous sensing and mobile storage renders some scenarios, in which training data are distributed across multiple entities, e.g., the driving data in different autonomous cars. It is well-known that learning models trained with more data have better performance [10]. However, directly using the raw data for collective learning can risk compromising the privacy of the data owners, e.g., exposing the living and working locations of the drivers. To tackle this challenge, distributed learning is usually leveraged, where multiple learning agents perform training collaboratively by exchanging their locally learned models. There are mainly two approaches: decentralized learning and federated learning. In decentralized learning, learning agents directly communicate their locally learned models with each other to facilitate the updates of the local models [15]–[18]. In federated learning, learning agents are orchestrated by a central server, i.e., the learning agents download shared models from the server, implement local updates based on local data and report the local models to the server for the updates of the shared models [19]. Decentralized learning is executed over peer-to-peer (P2P) networks, does not require a centralized entity and thus is robust to the failures of individual learning agents. With the support of a central server, federated learning has access to more resources in, e.g., computation, memory and power, and hence enables a much larger scale of learning processes.

Contribution Statement: In this paper, we propose a novel framework, federated learning for generalizable motion planning (FedGen), to tackle the challenge of generalizable motion planning in the presence of distributed data across multiple learning entities. Specifically, each learner updates

[1]Zhenyuan Yuan, Siyuan Xu and Minghui Zhu are with School of Electrical Engineering and Computer Science, Pennsylvania State University, University Park, PA 16802, USA (email: {zqy5086,spx5032,muz16}@psu.edu). This work was partially supported by NSF grants ECCS-1710859, CNS-1830390, ECCS-1846706 and the Penn State College of Engineering Multidisciplinary Research Seed Grant Program.

its local controller and sends its observation of the objective function to a central server for global minimization among the controllers of the learners. The global minimizer is then sent back to the learners for updates of the local controllers. The algorithm explicitly takes into account the generalization to unseen environments. This allows the framework to provide generalization guarantees in terms of the arrival time and the collision avoidance with obstacles for the robot navigating in unseen environments. Almost-sure convergence, almost consensus and Pareto improvement can also be achieved. In addition, the algorithm can be executed over P2P networks after a minor change. In summary, our contributions are: Monte Carlo simulations are conducted for evaluations.

Notations. We use superscript $(\cdot)^{[i]}$ to distinguish the local values of robot i and $\|\cdot\|$ to denote 2-norm.

II. Problem Formulation

In this section, we introduce the dynamic system of the robot, the problem of motion planning, the setting of federated learning, and the objective of this paper. The dynamic system of the robot is given as follows:

$$x_{t+1} = f(x_t, u_t) + d_E(x_t, u_t), \tag{1}$$

where $x_t \in \mathcal{X} \subseteq \mathbb{R}^{n_x}$ is the state of the robot, $u_t \in \mathcal{U} \subseteq \mathbb{R}^{n_u}$ is its control input, and d_E is the unknown external disturbance induced by environment $E \in \mathcal{E}$.

A. Environment-specific motion planning

Let environment E be composed of obstacle region $\mathcal{X}_{O,E} \subseteq \mathcal{X}$, free region $\mathcal{X}_{F,E} \triangleq \mathcal{X} \setminus \mathcal{X}_{O,E}$, and goal region $\mathcal{X}_{G,E} \subseteq \mathcal{X}_{F,E}$, The objective of the environment-specific motion planning problem is to synthesize a controller, which can drive system (1) to the goal region with obstacle collision avoidance. The arrival time function under controller $\pi : \mathcal{X} \times \mathcal{E} \to \mathcal{U}$ for system (1) starting from initial state x_{int} is given by

$$\begin{aligned} t_E(x_{int}; \pi) \triangleq \inf\{t > 0 \mid\ & x_t \in \mathcal{X}_{G,E}, x_0 = x_{int}, \\ & x_{\tau+1} = f(x_\tau, u_\tau) + d_E(x_\tau, u_\tau), \\ & u_\tau = \pi(x_\tau; E), x_\tau \in \mathcal{X}_{F,E}, \forall 0 \leqslant \tau \leqslant t\}. \end{aligned}$$

If the robot never reaches the goal, or hits the obstacles before arrival, then $t_E(x_{int}; \pi) = \infty$. We say safe arrival is achieved from initial state x_{int} under controller π if $t_E(x_{int}; \pi) < \infty$. Note that $t_E(x_{int}; \pi)$ is potentially infinite, and it can cause numerical issues. Therefore, we normalize the arrival time function by the adjusted Kruzkov transform $\Psi(r) \triangleq 1 - e^{-\alpha r}$, where $\alpha > 0$. Note that when α is small, $\Psi \circ t_E(x_{int}; \pi)$ is close to the indicator function that only returns zero when safe finite-time arrival happens. We obtain the standard Kruzkov transform when $\alpha = 1$. Then the normalized cost functional is given by $J_E(x_{int}; \pi) \triangleq \Psi \circ t_E(x_{int}; \pi)$.

B. Generalizable motion planning

In the problem of generalizable motion planning, we aim to synthesize a single controller that performs well in different environments. In statistical learning theory [10], this can be formulated as minimizing the expectation of the normalized arrival time over different environments. In particular, we assume the environments are sampled from an unknown distribution.

Assumption II.1. *(Stochastic environment).* There is an unknown distribution $\mathcal{P}_E$ over $\mathcal{E}$ from which environments are drawn from. ■

Further, we assume that the initial state is a random variable which is conditional on the environment.

Assumption II.2. *(Stochastic initialization).* There is an unknown conditional distribution $\mathcal{P}_{int|E}$ from which x_{int} is drawn conditional on environment $E \in \mathcal{E}$. ■

The objective of the generalizable motion planning problem is to synthesize a controller $\pi_* \in \Gamma \triangleq \{u(\cdot) : \mathcal{X} \times \mathcal{E} \to \mathcal{U}, \text{measurable}\}$, such that the expected normalized cost over all possible, including unseen, environments is minimized:

$$\pi_* = \arg\min_{\pi \in \Gamma} \mathbb{E}_{x_{int}, E}[J_E(x_{int}; \pi)]. \tag{2}$$

Since Γ is a function space, problem (2) is a functional optimization problem and hard to solve in general. In order to make the problem tractable, we approximate the space Γ using, e.g., deep neural networks and basis functions. Consider a class of controllers $\pi_\theta \in \Gamma$ parameterized by $\theta \in \mathbb{R}^{n_\theta}$, e.g., the weights of a deep neural network. Denote $\eta(\theta) \triangleq \mathbb{E}_{x_{int}, E}[J_E(x_{int}; \pi_\theta)]$. Then problem (2) becomes a parameter minimization problem:

$$\theta_* = \arg\min_{\theta \in \mathbb{R}^{n_\theta}} \eta(\theta). \tag{3}$$

Denote $\eta_* \triangleq \min_{\theta \in \mathbb{R}^{n_\theta}} \eta(\theta)$ and $\Theta_* \triangleq \{\theta \in \mathbb{R}^{n_\theta} \mid \eta(\theta) = \eta_*\}$. Problem (3) is a standard expected cost minimization problem. However, since the distribution of the environments is unknown, (3) cannot be solved directly. A typical practice is to approximate it by empirical cost minimization (with regularization), e.g., [8], [13], [14], where a controller is synthesized by minimizing the empirical cost (with regularization) over a finite number of training environments. In this paper, we aim to directly solve (3) and analyze the optimality of the solutions.

C. Federated learning

We consider a network of learners $\mathcal{V} \triangleq \{1, \cdots, N\}$. Each learner $i \in \mathcal{V}$ observes function η by sampling a set of local environments $E_l^{[i]} \overset{i.i.d.}{\sim} \mathcal{P}_E$, $l = 1, \cdots, n_{\mathcal{E}}^{[i]}$, and a set of initial states $x_{int|E_l^{[i]}, l'}^{[i]} \sim \mathcal{P}_{int|E_l^{[i]}}$, $l' = 1, \cdots, n_{int|\mathcal{E}}^{[i]}$, for each $E_l^{[i]}$. Given a triple of $(\theta^{[i]}, E_l^{[i]}, x_{int|E_l^{[i]}, l'}^{[i]})$, learner i can measure $J_{E_l^{[i]}}(x_{int|E_l^{[i]}, l'}^{[i]}; \pi_{\theta^{[i]}})$ by running the robot under controller $\pi_{\theta^{[i]}}$ from initial state $x_{int|E_l^{[i]}, l'}^{[i]}$ in environment $E_l^{[i]}$, measuring the arrival time and taking the Kruzkov

transform. Then learner i can approximate the gradient $\nabla_{\theta^{[i]}} J_{E_l^{[i]}}(x^{[i]}_{int|E_l^{[i]},l'};\pi_{\theta^{[i]}})$ by, e.g., using natural evolution strategies [20]. The learners can communicate to a Cloud but cannot communicate with each other. The objective of the multi-learner network and the Cloud is to collaboratively solve problem (3).

III. Algorithm Statement

In this section, we propose a federated optimization algorithm and analyze the performances the algorithm renders.

Algorithm 1 FedGen

Input: Local sample sizes: $n_{\mathcal{E}}^{[i]}, n_{int|\mathcal{E}}^{[i]}$; Kruzkov transform constant: α; Initial step size: $r^{[i]}$; Initial estimate: $\theta_0^{[i]}$; Threshold for gradient: $q^{[i]}$; Local bias: $s^{[i]}$; Step exponent: $\rho \in (0,1)$.
Init: $\zeta_0^{[i]} \leftarrow 1$, $\mathsf{Converge}_0^{[i]} \leftarrow \mathsf{False}$, Collects $(y_0^{[i]}, z_0^{[i]})$, $\forall i \in \mathcal{V}$.
for $k = 1, 2, \cdots, K$ **do**
{Learner-based update}
  for $i \in \mathcal{V}$ **do**
    Sends $(\theta_{k-1}^{[i]}, y_{k-1}^{[i]})$ to the Cloud
    if $\|z_{k-1}^{[i]}\| \geqslant q^{[i]}$ and $\mathsf{Converge}_{k-1}^{[i]} == \mathsf{False}$ **then**
      $\hat{\theta}_k^{[i]} \leftarrow \theta_{k-1}^{[i]} - \frac{r^{[i]}}{k^\rho} z_{k-1}^{[i]}$
      $\mathsf{Converge}_k^{[i]} \leftarrow \mathsf{False}$
    else
      $\hat{\theta}_k^{[i]} \leftarrow \theta_{k-1}^{[i]}$
      $(y_k^{[i]}, z_k^{[i]}) \leftarrow (y_{k-1}^{[i]}, z_{k-1}^{[i]})$
      $\mathsf{Converge}_k^{[i]} \leftarrow \mathsf{True}$
    end if
  end for
{Cloud update}
  $(j, l) \leftarrow \arg\min_{i \in \mathcal{V}, l'=0,\cdots,k-1} y_{l'}^{[i]} + s^{[i]}$
  Sends $(\theta_l^{[j]}, y_l^{[j]}, s^{[j]})$ to all $i \in \mathcal{V}$
{Learner-based fusion}
  for $i \in \mathcal{V}$ **do**
    if $j \neq i$ and $y_l^{[j]} + s^{[j]} < \min\{y_{k-1}^{[i]} - s^{[i]}, \zeta_{k-1}^{[i]}\}$ and $\|z_{k-1}^{[i]}\| < q^{[i]}$ **then**
      $\theta_k^{[i]} \leftarrow \theta_l^{[j]}$
      $\zeta_k^{[i]} \leftarrow y_l^{[j]}$
      $\mathsf{Converge}_k^{[i]} \leftarrow \mathsf{False}$
    else
      $\theta_k^{[i]} \leftarrow \hat{\theta}_k^{[i]}$
      $\zeta_k^{[i]} \leftarrow \zeta_{k-1}^{[i]}$
    end if
    if $\mathsf{Converge}_k^{[i]} == \mathsf{False}$ **then**
      Collects $(y_k^{[i]}, z_k^{[i]})$
    end if
  end for
end for

A. Federated optimization

Denote $\theta_k^{[i]}$ the empirical estimate of the solution to problem (3) by learner i at iteration k, $y_k^{[i]} \triangleq \frac{1}{n_{\mathcal{E}}^{[i]} n_{int|\mathcal{E}}^{[i]}} \sum_{l=1}^{n_{\mathcal{E}}^{[i]}} \sum_{l'=1}^{n_{int|\mathcal{E}}^{[i]}} J_{E_l^{[i]}}(x^{[i]}_{int|E_l^{[i]},l'};\pi_{\theta_k^{[i]}})$ the empirical estimate of $\eta(\theta_k^{[i]})$, and $z_k^{[i]} \triangleq \frac{1}{n_{\mathcal{E}}^{[i]} n_{int|\mathcal{E}}^{[i]}} \sum_{l=1}^{n_{\mathcal{E}}^{[i]}} \sum_{l'=1}^{n_{int|\mathcal{E}}^{[i]}} \nabla J_{E_l^{[i]}}(x^{[i]}_{int|E_l^{[i]},l'};\pi_{\theta_k^{[i]}})$ the empirical estimate of $\nabla\eta(\theta_k^{[i]})$. We propose the FedGen algorithm stated in Algorithm 1.

Informally speaking, each learner i first updates its local controller based on its empirical estimates of η and sends the estimates to the central server, e.g., the Cloud. The Cloud finds the best local estimate by taking into account the estimation error between $y_k^{[i]}$ and $\eta(\theta_k^{[i]})$, and then sends it back to the learners. Each learner then chooses between its local controller and the controller from the Cloud to further reduce η. The latter two parts utilize the power of the Cloud to identify the controller that can potentially achieve better performance in expectation and allow the learners to escape from their local minima. Specifically, in each iteration k, the algorithm sequentially executes three components as follows.

1) Learner-based update: First, each learner i updates its estimate based on the measurement $(y_{k-1}^{[i]}, z_{k-1}^{[i]})$ and the estimate $\theta_k^{[i]}$ from last iteration $k-1$. If $\|z_{k-1}^{[i]}\|$ is greater than a local threshold $q^{[i]}$, the learner makes one gradient descent step and updates its local estimate to $\hat{\theta}_k^{[i]}$. The threshold $q^{[i]}$ indicates whether a local minimum of η is achieved. If $\|z_{k-1}^{[i]}\|$ is not greater than $q^{[i]}$, the learner is labeled as converged, skips updating its local controller and maintains the previous measurement. The previous measurement is also sent to the Cloud for global minimization.

2) Cloud update: Upon the receipt of local estimates of η, $(y_{k-1}^{[i]}, \theta_{k-1}^{[i]})$, from each $i \in \mathcal{V}$, the Cloud returns the global minimizer of $y_{l'}^{[j]} + s^{[j]}$ over all the local estimates $\theta_{l'}^{[j]}$, $j \in \mathcal{V}, l' = 0, \cdots, k-1$, and sends the global minimizer and minimum to the learners. The local bias $s^{[i]}$ bounds the estimation error between $y_k^{[i]}$ and $\eta(\theta_k^{[i]})$. Different from the regularizers used in the literature of empirical cost minimization, the local bias $s^{[i]}$ is a constant value and does not depend on the estimate $\theta_k^{[i]}$. This procedure can be implemented recursively by comparing the learner-wise global minimum in the previous iteration with the values obtained in the current iteration. If one wants to implement Algorithm 1 over P2P networks without the Cloud, this step can be executed using the minimum consensus algorithm [21] in a distributed manner.

3) Learner-based fusion: Learner i only chooses the global minimizer $\theta_l^{[j]}$ sent by the Cloud when two conditions are satisfied: (i) estimate $\theta_l^{[j]}$ achieves a smaller estimate of η, i.e., $y_l^{[j]} + s^{[j]}$ is less than the minimum between $y_{k-1}^{[i]} - s^{[i]}$, and $\zeta_{k-1}^{[i]}$, the previous global minimum adopted by learner i; and (ii) the local estimate converges, i.e., $z_{k-1}^{[i]}$ is small. That is, the global minimizer helps learner i escape from its local minimum. When the global minimizer is chosen, the

local estimate is labeled as not converged at the iteration. New measurements are collected on the current estimate if the learner is not labeled as converged.

B. Performance guarantees

In this section, we provide the theoretic guarantees of FedGen (Algorithm 1). Due to space limitation, all the proofs are omitted and can be found in the complete version [22].

Let $\gamma \in (0,1)$ and $b_\gamma^{[i]} \triangleq \sqrt{\frac{\log(2/\gamma)}{2n_{\mathcal{E}}^{[i]} n_{int|\mathcal{E}}^{[i]}}}$. Theorem III.1 below shows the generalized performance of the local controller estimates to unseen environments.

Theorem III.1. Suppose Assumptions II.1 and II.2 hold. The following properties are true for all $i \in \mathcal{V}$:

(T1, Generalization error). For each $k \geqslant 0$, it holds that $\eta(\theta_k^{[i]}) \leqslant y_k^{[i]} + b_\gamma^{[i]}$ with probability at least $1-\gamma$.

(T2, Generalized safety). For each $k \geqslant 0$, the policy $\pi_{\theta_k^{[i]}}$ achieves safe arrival with probability at least $1-\gamma-(1-\gamma)(y_k^{[i]}+b_\gamma^{[i]})$ for $E \sim \mathcal{P}_E$ and $x_{int} \sim \mathcal{P}_{int|E}$. ■

Next we investigate the limiting behavior of the algorithm. Similar to most analysis of stochastic gradient descent (please see [23] [24] and the references therein), we assume η is Lipschitz continuous and $L_{\nabla\eta}$-smooth.

Assumption III.2. *(Lipschitz continuity).* There exists positive constant L_η such that $|\eta(\theta)-\eta(\theta')| \leqslant L_\eta \|\theta-\theta'\|$ for all $\theta, \theta' \in \mathbb{R}^{n_\theta}$. ■

Assumption III.3. *($L_{\nabla\eta}$-smooth).* There exists positive constant $L_{\nabla\eta}$ such that $\|\nabla\eta(\theta)-\nabla\eta(\theta')\| \leqslant L_{\nabla\eta}\|\theta-\theta'\|$ for all $\theta, \theta' \in \mathbb{R}^{n_\theta}$. ■

Furthermore, we assume that the variance of the errors of gradient estimation is bounded. This is a standard assumption in the analysis of stochastic optimization [23] [24].

Assumption III.4. *(Bounded variance).* It holds that $\mathbb{E}[\|z_k^{[i]} - \nabla\eta(\theta_k^{[i]})\|^2] \leqslant (\sigma^{[i]})^2$ for some $\sigma^{[i]} > 0$. ■

The lemma below shows that $z_k^{[i]}$ is an unbiased estimate of $\nabla\eta(\theta_k^{[i]})$.

Lemma III.5. *(Unbiased estimator).* Suppose Assumptions II.1, II.2 and III.2 hold. Then it holds that $\mathbb{E}[z_k^{[i]} - \nabla\eta(\theta_k^{[i]})] = 0$ for all $k \geqslant 1$. ■

Since $z_k^{[i]}$ is an unbiased estimate of $\nabla\eta(\theta_k^{[i]})$, by the law of large numbers (Proposition 6.3 in [25]), $(\sigma^{[i]})^2$ diminishes as $n_{\mathcal{E}}^{[i]}$ and $n_{int|\mathcal{E}}^{[i]}$ increase. The following theorem summarizes the properties of almost-sure convergence, almost consensus and Pareto improvement of the algorithm.

Theorem III.6. Suppose Assumptions II.1, II.2, III.2 III.3 and III.4 hold. For all $i \in \mathcal{V}$, if $r^{[i]} \leqslant \frac{1}{2L_{\nabla\eta}}$ and $q^{[i]} = 2\beta^{[i]}\sigma^{[i]}$ for some $\beta^{[i]} > 1$, then the followings hold:

(T3, Almost-sure convergence). There exists $\theta_*^{[i]} \in \mathbb{R}^{n_\theta}$ such that $\theta_k^{[i]} \to \theta_*^{[i]}$ almost surely.

(T4, Almost consensus). It holds that $\mathbb{E}_{\theta_*^{[i]}, i\in\mathcal{V}}[\eta(\theta_*^{[i]}) - \min_{j\in\mathcal{V}} \eta(\theta_*^{[j]})] \leqslant 2\max_{j\in\mathcal{V}} b_\gamma^{[j]}$.

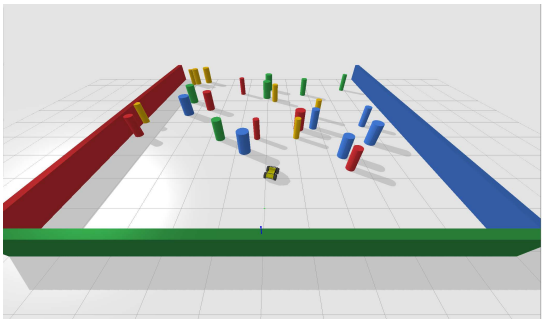


Fig. 1: A sample environment in PyBullet

Denote $k^{[i]} \triangleq \min\{k \geqslant 0 \mid \|z_k^{[i]}\| < q^{[i]}\}$ the first time learner i's estimate converges. Then we further have

(T5, Pareto improvement). It holds that $\mathbb{E}_{\theta_*^{[i]}, \theta_{k^{[i]}}^{[i]}}[\eta(\theta_*^{[i]}) - \eta(\theta_{k^{[i]}}^{[i]}) \mid \theta_*^{[i]} \neq \theta_{k^{[i]}}^{[i]}] \leqslant -2\min_{j\in\mathcal{V}} b_\gamma^{[j]}$. ■

IV. Simulation

In this section, we conduct a set of Monte Carlo simulations to evaluate the performance of the FedGen algorithm in the PyBullet simulator [26].

Environment configuration. The evaluation is conducted using Zermelo's navigation problem [27] in a 2D space, where the environments are randomly generated. A sample of the environments is shown in Figure 1. Each environment E consists of n_{obs} cylinder obstacles and three walls as the boundary of the 2D environment with $p_x \in [-5,5]$ and $p_y \in [0,10]$, where p_x is the horizontal coordinate and p_y is the vertical coordinate. The environments are generated by sampling the obstacle number n_{obs} uniformly between 15 and 30, and then independently sampling the centers of the cylinders from a uniform distribution over the ranges $[-5,5] \times [2,10]$. The radius of each obstacle is sampled independently from a uniform distribution over $[0.1, 0.25]$. The goal is to reach the open end of the environment while avoiding collision with the walls and the obstacles.

Robot dynamics. We consider a four-wheel robot with simple car dynamics [1] and constant speed $v = 2.5$ and length $L = 0.08$ subject to unknown environment-specific disturbances d_E. The dynamics of the robot with state $x = [p_x, p_y, \varphi]$ is given by $\dot{p}_x = v\cos(\varphi) + d_E(p_x, p_y)$, $\dot{p}_y = v\sin(\varphi)$, $\dot{\varphi} = \tan(u)/L$, where φ is the heading of the robot, control $u \in [-0.25\pi, 0.25\pi]$, and d_E is generated using the Von Karman power spectral density function as described in [28] representing the road texture disturbance (e.g., bumps and slippery surface) in environment E.

Sensor model. The robot is equipped with a depth sensor (e.g., LiDAR). The readings of the depth sensor depend on the environment E, the location and the heading of the robot. Specifically, the output of the sensor has 20 entries, where each entry ϕ corresponds to the distance measurement at angle $\varphi - \pi/3 + (\phi-1)\pi/60$ with $\phi = 1, \cdots, 20$. The measurement $h_\phi(x; E)$ provides the shortest distance between the obstacles, if there is any, at the angle of entry ϕ of the robot and the robot at location (p_x, p_y). The sensing

range is 5, i.e., $h_\phi(x;E) \in [0,5]$. Denote the output of sensor measurement as $h(x;E) \triangleq [h_\phi(x;E)]_{\phi=1,\cdots,20}$.

A. Training

We consider a deep neural network-based controller $u_t = \pi_\theta(x_t;E) = \pi_\theta(x_t, h(x_t;E))$, that is parameterized by θ, the weights of the neural network. Note that the controller is periodic in φ. Thus, the input φ is replaced by two inputs $\sin(\varphi)$ and $\cos(\varphi)$. During training, especially during the early phase, the original cost functional $J_E(x_{int}, \pi_\theta)$ may have zero gradient for some initial state x_{int} since collisions with obstacles dominate most of the trial runs. Therefore, to facilitate training, we consider the surrogate $\hat{J}_E(x_{int},\theta) \triangleq 0.1(\rho_E(x_{int},\pi_\theta) + J_E(x_{int},\pi_\theta))$, where $\rho_E(x_{int},\pi_\theta) \triangleq \min_{x_G \in \mathcal{X}_{G,E}} \|x(t_{end}(x_{int},\pi_\theta;E)) - x_G\|$, $t_{end}(x_{int},\pi_\theta;E) \triangleq \min\{t_{col}(x_{int},\pi_\theta;E), \bar{t}\}$ and $t_{col}(x_{int},\pi_\theta;E)$ is the first time the robot collides with an obstacle in environment E starting from x_{int} under policy π_θ, and $\bar{t}$ is the maximum allowable traveling time of the robot. The cost $\rho_E(x_{int},\pi_\theta)$ is to drive the robot approaching the goal without collision, and the cost $J_E(x_{int},\pi_\theta)$ is to minimize the arrival time when the robot is able to safely reach the goal.

Since it is challenging to derive the analytical expression of $\nabla \hat{J}_E(x_{int},\theta)$, we approximate it by natural evolution strategies [20]. In particular, we suppose θ follows a multivariate Gaussian distribution such that $\theta \sim \mathcal{N}(\mu,\Sigma)$ with a mean $\mu \in \mathbb{R}^{n_\theta}$ and a diagonal covariance $\Sigma \in \mathbb{R}^{n_\theta \times n_\theta}$. Let $\sigma \in \mathbb{R}^{n_\theta}$ be a vector aggregating the square-root of the diagonal elements of Σ. The gradients of $\mathbb{E}_\theta\left[\hat{J}_E(x_{int},\pi_\theta)\right]$ with respect to μ and σ are

$$\nabla_\mu \mathop{\mathbb{E}}_{\theta\sim P}\left[\hat{J}_E(x_{int},\pi_\theta)\right] = \mathop{\mathbb{E}}_{\epsilon\sim\mathcal{N}(0,I)}\left[\hat{J}_E(x_{int},\pi_{\mu+\sigma\odot\epsilon})\epsilon\right] \oslash \sigma,$$

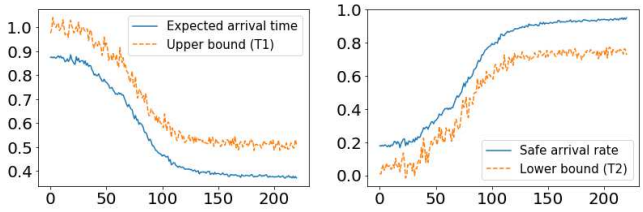
$$\nabla_\sigma \mathop{\mathbb{E}}_{\theta\sim P}\left[\hat{J}_E(x_{int},\pi_\theta)\right] = \mathop{\mathbb{E}}_{\epsilon\sim\mathcal{N}(0,I)}\left[\hat{J}_E(x_{int},\pi_{\mu+\sigma\odot\epsilon})(\epsilon\odot\epsilon - \mathbf{1})\right] \oslash \sigma,$$

where $\oslash$ is the element-wise division, $\odot$ is the elementwise product, and $\mathbf{1}$ is a vector of 1's with dimension n_θ. We approximate the expectation by sampling m times $\epsilon \sim \mathcal{N}(0,I)$ and taking the average.

Selection of hyperparameters. The neural network controller consists of an input layer of size 24, followed by 3 hidden layers of size 20 with ReLU nonlinearities and an output layer of size 1. We sample $m = 30$ to approximate the gradients. We pick $n_{\mathcal{E}}^{[i]} = 10$, $n_{int|\mathcal{E}}^{[i]} = 1$, $\gamma = 0.01$, $r = 0.01$, $L_\eta = 0.1$, $\alpha = 0.1$ $q^{[i]} = 0.04$ and $s^{[i]} = \sqrt{\frac{\log(2/\gamma)}{2n_{\mathcal{E}}^{[i]} n_{int|\mathcal{E}}^{[i]}}}$. We use 8 learners, i.e., $N = 8$, for the experiments. Note that the generalized performance in unseen environments is defined as an expectation over all possible environments, which cannot be obtained exactly. Therefore, we estimate the generalized performances using 10^4 sample environments.

B. Results

Generalization. Figure 2 compares the upper bound on the expected normalized arrival time (T1) and the lower bound on the safe arrival rate (T2) in Theorem III.1 respectively with the actual expected normalized arrival time and the actual safe arrival rate of learner 1. Other learners have similar behaviors. As the figure illustrates, the upper bound and the lower bound derived in the theorem are valid. This shows that the controller trained can generalize well to the 10^4 unseen environments.



(a) Normalized arrival time (b) Safe arrival rate

Fig. 2: Generalized performances on unseen environments

Near consensus and Pareto improvement. In Table I below, we show the performances of the learners' estimates in terms of the expected cost $\hat{J}_E$, the expected normalized arrival time J_E, and the expected safe arrival rate. We compare with the controllers at initialization ($\theta_0^{[i]}$), the controller obtained without communication ($\check{\theta}_*^{[i]}$), i.e., the controller obtained by running FedGen using $\mathcal{V} = \{i\}$, and the final convergence ($\theta_*^{[i]}$) under FedGen. We can observe that all the expected costs, expected normalized arrival times and expected safe arrival rates at $\theta_*^{[i]}$ are roughly equal. This is aligned with the near consensus in Theorem III.6. Furthermore, we can observe that all the expected costs and the expected normalized arrival times at $\theta_*^{[i]}$ are no larger than those of $\theta_0^{[i]}$ and $\check{\theta}_*^{[i]}$, while the expected safe arrival rates at $\theta_*^{[i]}$ are no smaller than those at $\theta_0^{[i]}$ and $\check{\theta}_*^{[i]}$. This shows that FedGen brings Pareto improvement for each learner through communication.

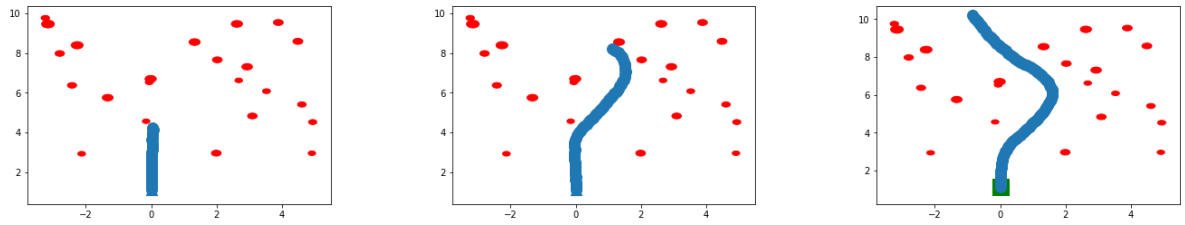
Figure 3 respectively shows the trajectories of the robot in a sample unseen environment using learner 1's initial policy $\theta_0^{[1]}$, locally converged policy $\check{\theta}_*^{[1]}$ and finally converged policy $\theta_*^{[1]}$. The red disks represent the obstacles. Both the initial controller (Figure 3a) and the locally converged controller (Figure 3b) cannot bring the robot to the open end, despite the locally converged controller is able to drive the robot closer to the open end. Nevertheless, the path generated by the final controller $\theta_*^{[1]}$ is able to drive the robot to the open end. This illustrates that FedGen helps the learners escape from their local minimas and achieve better generalizability.

V. Conclusion

We propose FedGen, a federated learning algorithm which allows a group of learners to collaboratively learn a single controller for motion planning in unseen environments. The problem is formulated as an expected cost minimization problem and solved in a federated manner. The proposed algorithm is able to provide generalization guarantees on the performances of the local controllers in unseen environments

Learner ID (i)		1	2	3	4	5	6	7	8
Cost ($\hat{J}$)	Init	0.5198	0.5170	0.5208	0.5210	0.5148	0.5231	0.5237	0.5167
	Local	0.0436	0.0396	0.0331	0.4810	0.4105	0.0341	0.3992	0.4989
	Final	0.0374	0.0396	0.0331	0.0335	0.0353	0.0341	0.0363	0.0335
Normalized arrival time (J)	Init	0.8743	0.8761	0.8744	0.8782	0.8692	0.8748	0.8797	0.8732
	Local	0.3759	0.3763	0.3701	0.8385	0.7815	0.3700	0.7622	0.8569
	Final	0.3748	0.3763	0.3701	0.3679	0.3711	0.3700	0.3716	0.3704
Safe arrival rate	Init	0.1802	0.1776	0.1800	0.1746	0.1876	0.1794	0.1724	0.1818
	Local	0.9320	0.9408	0.9522	0.2314	0.3172	0.9450	0.3426	0.2054
	Final	0.9386	0.9408	0.9522	0.9452	0.9432	0.9450	0.9428	0.9468

TABLE I: The expected costs, normalized arrival times, safe arrival rates of the estimates at initialization, local convergence and final convergence.



(a) Trajectory produced by $\theta_0^{[1]}$ (b) Trajectory produced by $\check{\theta}_*^{[1]}$ (c) Trajectory produced by $\theta_*^{[1]}$

Fig. 3: Comparison between initial policy, locally converged policy and globally converged policy

as well as consensus at global optimal value in the limiting case. Monte Carlo simulations for conducted for evaluation.

REFERENCES

[1] S. M. LaValle, *Planning algorithms*. Cambridge university press, 2006.

[2] A. Majumdar and R. Tedrake, "Funnel libraries for real-time robust feedback motion planning," *The International Journal of Robotics Research*, vol. 36, no. 8, pp. 947–982, 2017.

[3] S. Singh, A. Majumdar, J.-J. Slotine, and M. Pavone, "Robust online motion planning via contraction theory and convex optimization," in *Proc. Int. Conf. Robotics and Automation (ICRA)*, 2017, pp. 5883–5890.

[4] M. Ono, M. Pavone, Y. Kuwata, and J. Balaram, "Chance-constrained dynamic programming with application to risk-aware robotic space exploration," *Autonomous Robots*, vol. 39, no. 4, pp. 555–571, 2015.

[5] M. Castillo-Lopez, P. Ludivig, S. A. Sajadi-Alamdari, J. L. Sanchez-Lopez, M. A. Olivares-Mendez, and H. Voos, "A real-time approach for chance-constrained motion planning with dynamic obstacles," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 3620–3625, 2020.

[6] N. Virani, D. K. Jha, Z. Yuan, I. Shekhawat, and A. Ray, "Imitation of demonstrations using bayesian filtering with nonparametric data-driven models," *Journal of Dynamic Systems, Measurement, and Control*, vol. 140, no. 3, 2018.

[7] S. Gupta, J. Davidson, S. Levine, R. Sukthankar, and J. Malik, "Cognitive mapping and planning for visual navigation," in *Proc. IEEE Conf. Computer Vision and Pattern Recognition (CVPR)*, 2017, pp. 2616–2625.

[8] S. Levine, C. Finn, T. Darrell, and P. Abbeel, "End-to-end training of deep visuomotor policies," *The Journal of Machine Learning Research*, vol. 17, no. 1, pp. 1334–1373, 2016.

[9] Y. Fu, D. K. Jha, Z. Zhang, Z. Yuan, and A. Ray, "Neural network-based learning from demonstration of an autonomous ground robot," *Machines*, vol. 7, no. 2, p. 24, 2019.

[10] V. Vapnik, *The Nature of Statistical Learning Theory*. Springer science & business media, 2013.

[11] I. Goodfellow, Y. Bengio, and A. Courville, *Deep learning*. MIT press, 2016.

[12] J. Garcıa and F. Fernández, "A comprehensive survey on safe reinforcement learning," *Journal of Machine Learning Research*, vol. 16, no. 1, pp. 1437–1480, 2015.

[13] I. Lenz, H. Lee, and A. Saxena, "Deep learning for detecting robotic grasps," *The International Journal of Robotics Research*, vol. 34, pp. 705–724, 2015.

[14] A. Majumdar and M. Goldstein, "PAC-Bayes control: Synthesizing controllers that provably generalize to novel environments," in *Conf. Robot Learning (CoRL)*, 2018, pp. 293–305.

[15] K. Zhang, Z. Yang, H. Liu, T. Zhang, and T. Basar, "Fully decentralized multi-agent reinforcement learning with networked agents," in *Proc. Int. Conf. Machine Learning (ICML)*, 2018, pp. 5872–5881.

[16] Z. Yuan and M. Zhu, "Communication-aware distributed Gaussian process regression algorithms for real-time machine learning," in *Proc. American Control Conf. (ACC)*, July 2020, pp. 2197–2202.

[17] A. Nedić, A. Olshevsky, and C. A. Uribe, "A tutorial on distributed (non-bayesian) learning: Problem, algorithms and results," in *Proc. IEEE Conf. Decision and Control (CDC)*, 2016, pp. 6795–6801.

[18] J. Liu, Y. Liu, A. Nedic, and T. Başar, "An approach to distributed parametric learning with streaming data," in *Proc. IEEE Conf. Decision and Control (CDC)*, 2017, pp. 3206–3211.

[19] Q. Yang, Y. Liu, Y. Cheng, Y. Kang, T. Chen, and H. Yu, *Federated learning*. Morgan & Claypool, 2020.

[20] D. Wierstra, T. Schaul, T. Glasmachers, Y. Sun, J. Peters, and J. Schmidhuber, "Natural evolution strategies," *The Journal of Machine Learning Research*, vol. 15, no. 1, pp. 949–980, 2014.

[21] R. Olfati-Saber and R. M. Murray, "Distributed cooperative control of multiple vehicle formations using structural potential functions," *IFAC Proceedings Volumes*, vol. 35, no. 1, pp. 495–500, 2002.

[22] Z. Yuan, S. Xu, and M. Zhu, "Federated reinforcement learning for generalizable motion planning," *https://www.dropbox.com/s/rrkvovzcee1zkpj/FedGen.pdf?dl=0*, 2022.

[23] B. Fehrman, B. Gess, and A. Jentzen, "Convergence rates for the stochastic gradient descent method for non-convex objective functions," *Journal of Machine Learning Research*, vol. 21, 2020.

[24] S. Ghadimi and G. Lan, "Stochastic first-and zeroth-order methods for nonconvex stochastic programming," *SIAM Journal on Optimization*, vol. 23, no. 4, pp. 2341–2368, 2013.

[25] R. Bhattacharya, L. Lin, and V. Patrangenaru, *A course in mathematical statistics and large sample theory*. Springer, 2016.

[26] "Pybullet," *https://pybullet.org/wordpress/*.

[27] S. Zlobec, *Zermelo's Navigation Problems*. Boston, MA: Springer US, 2001.

[28] K. Cole and A. Wickenheiser, "Impact of wind disturbances on vehicle station keeping and trajectory following," in *AIAA Guidance, Navigation, and Control Conference*, 2013, p. 4865.