games

MDPI

Article

Physics-Informed Graph Neural Operator for Mean Field Games on Graph: A Scalable Learning Approach

Xu Chen [1], Shuo Liu [2] and Xuan Di [1,*]

[1] Department of Civil Engineering and Engineering Mechanics, Columbia University, New York, NY 10027, USA; xc2412@columbia.edu
[2] Department of Computer Science, Columbia University, New York, NY 10027, USA; sl4921@columbia.edu
* Correspondence: sharon.di@columbia.edu

Abstract: Mean-field games (MFGs) are developed to model the decision-making processes of a large number of interacting agents in multi-agent systems. This paper studies mean-field games on graphs ($\mathcal{G}$-MFGs). The equilibria of $\mathcal{G}$-MFGs, namely, mean-field equilibria (MFE), are challenging to solve for their high-dimensional action space because each agent has to make decisions when they are at junction nodes or on edges. Furthermore, when the initial population state varies on graphs, we have to recompute MFE, which could be computationally challenging and memory-demanding. To improve the scalability and avoid repeatedly solving $\mathcal{G}$-MFGs every time their initial state changes, this paper proposes physics-informed graph neural operators (PIGNO). The PIGNO utilizes a graph neural operator to generate population dynamics, given initial population distributions. To better train the neural operator, it leverages physics knowledge to propagate population state transitions on graphs. A learning algorithm is developed, and its performance is evaluated on autonomous driving games on road networks. Our results demonstrate that the PIGNO is scalable and generalizable when tested under unseen initial conditions.

Keywords: mean-field game; scalable learning; physics-informed neural operator

Citation: Chen, X.; Liu, S.; Di, X. Physics-Informed Graph Neural Operator for Mean Field Games on Graph: A Scalable Learning Approach. *Games* **2024**, *15*, 12. https://doi.org/10.3390/g15020012

Academic Editors: Ulrich Berger and Michael Wooldridge

Received: 3 February 2024
Revised: 18 March 2024
Accepted: 26 March 2024
Published: 30 March 2024

1. Introduction

Multi-agent systems (MAS) are prevalent in engineering and robotics applications. With a large number of interacting agents in the MAS, solving agents' optimal control could be computationally intractable and not scalable. To solve this challenge, MFGs are [1,2] developed to model strategic interactions among many agents who make dynamically optimal decisions, while a population distribution is propagated to represent the state of interacting agents. Since its inception, MFGs have been widely applied to social networks [3], swarm robotics [4] and intelligent transportation [5,6].

MFGs are micro-macro games that bridge agent dynamics and population behaviors with two coupled processes: individuals' dynamics solved by optimal control (i.e., agent dynamic) and system evolution arising from individual choices (i.e., population behaviors).

In this work, we focus on a class of MFGs [7], namely, mean-field games on graphs ($\mathcal{G}$-MFG), where the state space of the agent population is a graph and agents select a sequence of nodal and edge transitions with a minimum individual cost. Solving these $\mathcal{G}$-MFGs, however, poses the following challenges: (1) With a graph-based state space, the action space expands significantly, encompassing both nodes and edges, resulting in a high-dimensional search space. More specifically, the decision-making of a representative agent in $\mathcal{G}$-MFG consists of not only en-route choices at nodes but also continuous velocity control on edges subject to congestion effects. (2) Existing work mainly assumes that the initial population distribution is fixed. The change in initial population states leads to the re-computation of mean-field equilibria (MFE), a task that requires computational and memory resources and hinders the practicality of deploying MFG solutions.

To address these challenges, this paper proposes a new learning tool for $\mathcal{G}$-MFGs, namely, a physics-informed graph neural operator (PIGNO). The key element is a graph neural operator (GNO), which can generate population dynamics given the initial population distribution. To enhance the training process, the GNO incorporates physics knowledge regarding how agent and population dynamics propagate over the spatiotemporal domain.

Related Work

Researchers have explored various machine learning methods, such as reinforcement learning (RL) [8–11], and physics-informed neural networks (PINN) [12–14]. However, it can be time-consuming and memory-demanding for these learning tools to adapt to changes in initial population density. Specifically, each unique initial condition may require the assignment and retraining of a dedicated neural network to obtain the corresponding MFE. To enhance the scalability of the learning framework for MFGs, Chen et al. [15] introduced a physics-informed neural operator (PINO) framework. This framework utilizes a Fourier neural operator (FNO) to establish a functional mapping between mean-field equilibrium and boundary conditions. However, the FNO fails to solve $\mathcal{G}$-MFGs because it cannot directly project information over a graph into a high-dimensional space and generate population dynamics in the graph state space. Therefore, in this paper, we propose a graph neural operator (GNO) that learns mappings between graph-based function spaces to solve $\mathcal{G}$-MFGs. The GNO leverages message-passing neural networks (MPNNs) to handle state space and propagate state information efficiently by aggregating the neighborhood messages.

Our contributions include: (1) We propose a scalable learning framework leveraging PIGNO to solve $\mathcal{G}$-MFGs with various initial population states; (2) We develop a learning algorithm and apply it to autonomous driving games on road networks to evaluate the algorithm performance.

The rest of this paper is organized as follows: Section 2 introduces preliminaries about $\mathcal{G}$-MFGs. Section 3 presents the details of our scalable learning framework for $\mathcal{G}$-MFGs. Section 4 presents the solution approach. Section 5 demonstrates numerical experiments. Section 6 concludes.

2. Background

2.1. Mean-Field Games on Graphs ($\mathcal{G}$-MFG)

Mean-field games on graphs ($\mathcal{G}$-MFG) model population dynamics and a generic agent's optimal control on both nodes and edges. A $\mathcal{G}$-MFG consists of a forward FPK and multiple backward HJB equations, which are defined on a graph $\mathcal{G} = \{\mathcal{N}, \mathcal{L}\}$ as follows:

Definition 1. *A $\mathcal{G}$-MFG with discrete time graph states [16] is:*

$$
\begin{aligned}
&[\mathcal{G}\text{-}MFG]: \\
&\quad (FPK)\ \boldsymbol{\rho}^{\tau+1} = [P^{\tau}]^{T}\boldsymbol{\rho}^{\tau},\ \boldsymbol{\rho}^{0} \equiv \tilde{\boldsymbol{\rho}} && \text{(1a)}\\
&\quad (HJB)\ \boldsymbol{V}^{\tau} = \min_{\boldsymbol{u},\boldsymbol{\beta}} P^{\tau}\boldsymbol{V}^{\tau+1} + \boldsymbol{r}^{\tau},\ \boldsymbol{V}^{T} \equiv \tilde{\boldsymbol{V}} && \text{(1b)}
\end{aligned}
$$

$\boldsymbol{\rho}^{\tau} = [\rho_{ij}^{\tau}]^{T}$ is the population density on each edge $(i, j) \in \mathcal{L}$ at time step τ. $\tilde{\boldsymbol{\rho}}$ denotes the initial population density over the graph. The Fokker–Planck (FPK) equation captures the evolution of the population state on the graph. The Hamilton–Jacobi–Bellman (HJB) equation captures the optimal control of a generic agent, including the velocity control on edges and route choice on nodes. $\boldsymbol{V}^{\tau} = [V_{ij}^{\tau}]^{T}$ is the value function at each edge. $\tilde{\boldsymbol{V}}$ denotes the terminal cost. $\boldsymbol{u}^{\tau} = [u_{ij}^{\tau}]^{T}$ denotes the exit rate at each edge, which represents the agent's velocity control. $\boldsymbol{\beta}^{\tau} = [\beta_{ij}^{\tau}]^{T}$ is the probability of choosing node j as the next-go-to node at node i, i.e., route choice. $\boldsymbol{r}^{\tau} = [r_{ij}^{\tau}]^{T}$ is the cost incurred by the agent at time step τ. The transition matrix P^{τ} is determined by $\boldsymbol{u}^{\tau}$ and $\boldsymbol{\beta}^{\tau}$. The MFE is denoted by $SOL([\mathcal{G}\text{-MFG}]) = \{\boldsymbol{\rho}^{*}, \boldsymbol{V}^{*}, \boldsymbol{u}^{*}, \boldsymbol{\beta}^{*}\}$, satisfying Equation (1). The mathematical details of

$\mathcal{G}$-MFG can be found in Appendix A.1. We provide a toy example in Appendix A.2 to help readers better understand it.

2.2. Graph Neural Operator (GNO)

Graph neural operators (GNOs) are generalized neural networks that can learn functional mappings between high-dimensional spaces [17]. GNO utilizes an MPNN to update space representation according to messages from the neighborhood. In this paper, we adopt a GNO to establish mappings between initial population state $\boldsymbol{\rho}^0$ and population $\boldsymbol{\rho}^*$ at MFE. We leverage the physics knowledge (i.e., FPK and HJB equations) to train the GNO for solving MFE with various initial population densities, eliminating the need to recompute MFE.

3. Scalable Learning Framework

In this section, we propose a physics-informed graph neural operator (PIGNO) to learn $\mathcal{G}$-MFGs. Figure 1 illustrates the workflow of two couple modules: FPK for population behaviors and HJB for agent dynamics. The FPK and the HJB modules internally depend on each other. In the FPK module, we estimate population density $\boldsymbol{\rho}^{0:T}$ over the graph and update the GNO using a residual defined by the physical rule that captures population dynamics triggered by the transition matrix defined in the FPK equation. In the HJB module, the transition matrix $P^{0:T}$ is obtained given the population density $\boldsymbol{\rho}^{0:T}$. We adopt another GNO to solve the HJB equation since the dynamics of the agents and the cost functions are known in the MFG system. We now delve into the details of the proposed PIGNO.

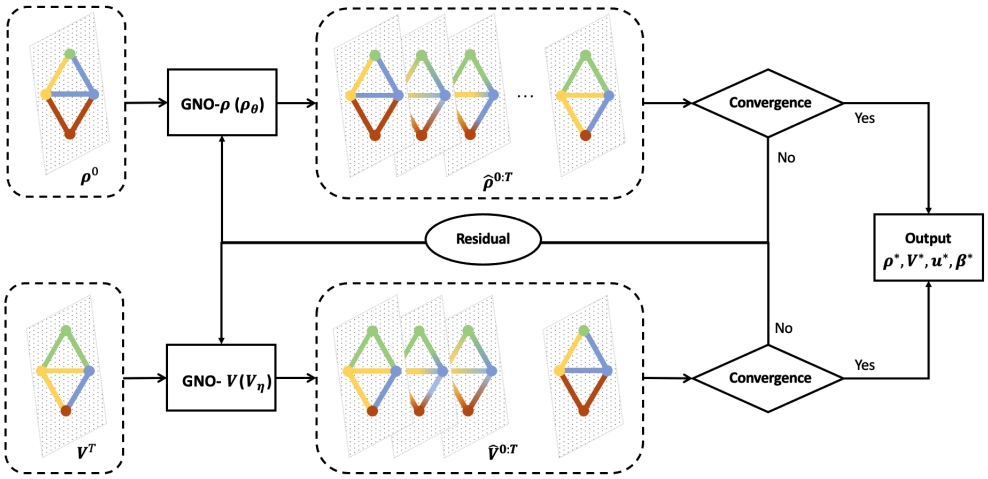


Figure 1. PIGNO for $\mathcal{G}$-MFGs: We leverage graph neural networks to establish a functional mapping between the initial population density along with terminal cost and mean field equilibrium over the entire spatial temporal domain. The population density and terminal cost over the space domain at each time step is denoted by color bars over the graph. The PIGNO allows us to obtain MFEs corresponding to each initial population distribution and terminal cost without recomputing them.

3.1. PIGNO for Population Behaviors

The GNO-ρ maps the initial population distribution and the population distribution from time 0 to T. The input of GNO-ρ is the initial population density $\boldsymbol{\rho}^0$ along with the transition information to propagate population dynamics. The output of GNO-ρ is the population dynamics over the spatiotemporal domain, denoted by $\hat{\boldsymbol{\rho}} \equiv \hat{\boldsymbol{\rho}}^{0:T}$. The PIGNO is instantiated as the following MPNN: $\forall (i,j) \in \mathcal{L}, \tau = 0, 1, \ldots, T$

$$\hat{\rho}_{ij}^{\tau} = \rho_\theta(\rho_{ij}^0, \sum_{\forall (k,l) \in \mathcal{L}_{ij}^{\tau}} \kappa_\phi(\rho_{ij}^0, \rho_{kl}^0, e_{ij,kl}^{\tau})) \tag{2}$$

where, $\hat{\rho}_{ij}^{\tau}$ is the population density of edge (i,j) at time τ, $\mathcal{L}_{ij}^{\tau}$ is the set of neighborhood edges of edge (i,j) at time τ, κ_ϕ is the graph kernel function for ρ_θ, and $e_{ij,kl}^{\tau}$ denotes the cumulative message used to propagate population dynamics from time 0 to time τ. $e_{ij,kl}^{\tau}$ indicates the ratio of the population entering from edge (k,l) to edge (i,j) till time τ, which is determined by the ratio of population exiting the edge (k,l) (i.e., the velocity control u) and the ratio of the population choosing the edge (i,j) as the next-go-to edge (i.e., the route choice β). The MPNN utilizes the initial population distribution and the message to propagate the population dynamics in the $\mathcal{G}$-MFG system. The neighborhood message is transformed by a kernel function κ_ϕ and aggregated as an additional feature to estimate population density.

The GNO-ρ adopts a physics-informed training scheme, which combines both model-driven and data-driven methods. The training of GNO-ρ is guided by the residual determined by physical rules of population dynamics. Mathematically, the residual r_θ is:

$$r_\theta = \frac{1}{|\boldsymbol{\rho}^{\mathcal{D}}|} \sum_{\boldsymbol{\rho}^0 \in \boldsymbol{\rho}^{\mathcal{D}}} L_{\boldsymbol{\rho}^0}, \tag{3}$$

where, the set $\boldsymbol{\rho}^{\mathcal{D}}$ contains various initial densities over the graph. $L_{\boldsymbol{\rho}^0}$ is calculated as:

$$L_{\boldsymbol{\rho}^0} = \alpha_0 \cdot \frac{1}{T} \sum_{\tau=0}^{T-1} ||\hat{\boldsymbol{\rho}}^{\tau+1} - [P^{\tau}]^T \hat{\boldsymbol{\rho}}^{\tau}|| + \alpha_1 \cdot ||\hat{\boldsymbol{\rho}}^0 - \boldsymbol{\rho}^0||^2, \tag{4}$$

where, the first term in $L_{\boldsymbol{\rho}^0}$ evaluates the physical discrepancy based on Equation (1a). It integrates the residual of the FPK equation, ensuring that the model adheres to established laws of motion. When predicted $\boldsymbol{\rho}$ becomes closer to $\boldsymbol{\rho}^*$ satisfying the FPK equation, the residual gets closer to 0. The second term quantifies the discrepancy between the estimations and the ground truth of the initial density. The observed data comes from the initial distribution of population $\boldsymbol{\rho}^0$. The training of ρ_θ based on observed data follows the traditional supervised learning scheme. α_0 and α_1 are the weight coefficients.

3.2. PIGNO for Optimal Control

Similar to GNO-ρ, GNO-V learns a reverse mapping from the terminal costs to the value functions over the graph from time T to 0. The input of GNO-V is the terminal costs $\boldsymbol{V}^T$ and the transition information. The output of GNO-V is the value function over the spatiotemporal domain, denoted by $\hat{\boldsymbol{V}} \equiv \hat{\boldsymbol{V}}^{0:T}$. The GNO-$V$ also follows the MPNN formulation: $\forall (i,j) \in \mathcal{L}, \tau = 0, 1, \ldots, T$

$$\hat{V}_{ij}^{\tau} = V_\eta(V_{ij}^T, \sum_{\forall (k,l) \in \mathcal{L}_{ij}^{\tau}} \kappa_\psi(V_{ij}^T, V_{kl}^T, e_{ij,kl}^{\tau})) \tag{5}$$

where $\hat{V}_{ij}^{\tau}$ is the value function of edge (i,j) at time τ, $\mathcal{L}_{ij}^{\tau}$ is the set of neighborhood edges of edge (i,j) at time τ, κ_ψ is the graph kernel function for V_η, and $e_{ij,kl}^{\tau}$ denotes the cumulative message used to propagate population dynamics from time 0 to time τ, which comes from the transpose of the cumulative transition matrix in GNO-ρ. The MPNN utilizes the terminal costs and the message to propagate system value functions. The neighborhood message is transformed by a kernel function κ_ψ and aggregated as an additional feature to estimate value functions.

The training of GNO-V takes the HJB residual r_η into consideration, which is

$$r_\eta = \frac{1}{|\boldsymbol{V}^{\mathcal{D}}|} \sum_{\boldsymbol{V}^T \in \boldsymbol{V}^{\mathcal{D}}} L_{\boldsymbol{V}^T}, \tag{6}$$

where, the set $\boldsymbol{V}^{\mathcal{D}}$ contains various terminal costs over the graph. $L_{\boldsymbol{V}^T}$ is calculated as:

$$L_{\boldsymbol{V}^T} = \alpha_0 \cdot \frac{1}{T} \sum_{\tau=0}^{T-1} ||\hat{\boldsymbol{V}}^{\tau+1} - P^{\tau}\hat{\boldsymbol{V}}^{\tau}|| + \alpha_1 \cdot ||\hat{\boldsymbol{V}}^{T} - \boldsymbol{V}^{T}||^2, \tag{7}$$

where, the first term in $L_{\boldsymbol{V}^T}$ evaluates the physical discrepancy based on Equation (1b). When predicted $\boldsymbol{V}$ becomes closer to the optimal $\boldsymbol{V}^*$, the residual gets closer to 0. The second term calibrates the predictions to the ground truth of the terminal costs $\boldsymbol{V}^T$ by supervised learning. Similarly, α_0 and α_1 are the weight coefficients.

Note that a meta assumption of this model is discrete time. There are significant limitations of adopting a continuous-time model: (1) A continuum formulation of forward process over a graph space cannot be easily captured by several coupled partial differential equation systems. One way is to simplify the decision making over the graph as discretized route choice at each node, rendering the game as a continuous time markov decision process, which fails to capture the real-time velocity control on each edge. The other way is to formulate the dynamic process as a graph ODE, which requires further investigation on scalability. (2) A continuum formulation of backward process can be solved by continuous-time reinforcement learning. We leave the scalability of this continuous time reinforcement learning scheme to future work.

4. Solution Approach

In this section, we present our learning algorithm (Algorithm 1). We first initialize the GNO-ρ ρ_θ parameterized by θ and GNO-V V_η parameterized by η. During the ith iteration of the training process, we first sample a batch of initial population densities $\boldsymbol{\rho}^0$ and terminal costs $\boldsymbol{V}^T$. The terminal cost $\boldsymbol{V}^T$ denotes the delay for agents who haven't arrived at their destinations at the terminal time step. In this work, we assume the time delay at the terminal step is proportional to the travel distance between the agent's location and her destination. $\boldsymbol{\rho}^{\mathcal{D}}$ represents the set of initial population density. In this work, we interpret initial population density as the travel demands (i.e., the number of agents entering a graph at time 0). Agents can enter the graph from each node. We assume at time 0, the number of agents at each node (i.e., travel demand) follow a uniform distribution. We use each pair of $\boldsymbol{\rho}^0$ and $\boldsymbol{V}^T$ to generate the population density $\hat{\boldsymbol{\rho}}^{0:T}$ and $\hat{\boldsymbol{V}}^{0:T}$ over the entire domain. Given $\hat{\boldsymbol{\rho}}^{0:T}$ and $\hat{\boldsymbol{V}}^{0:T}$, we obtain the spatiotemporal transition P for all nodes. We then update the parameter θ of the neural operator according to the residual. At the end of each iteration, we check the convergence according to:

Algorithm 1 PIGNO-MFG

1: Initialize: GNO-ρ ρ_θ parameterized by θ, GNO-V V_η parameterized by η;
2: **for** $i \leftarrow 0$ to I **do**
3: Sample a batch of initial population densities $\boldsymbol{\rho}^0$ from the set $\boldsymbol{\rho}^{\mathcal{D}}$ and terminal costs $\boldsymbol{V}^T$ from the set $\boldsymbol{V}^{\mathcal{D}}$;
4: Predict $\hat{\boldsymbol{\rho}}^{0:T(i)}$ and $\hat{\boldsymbol{V}}^{0:T(i)}$ by ρ_θ and V_η corresponding to each pair of $\boldsymbol{\rho}^0$ and $\boldsymbol{V}^T$ in the batch;
5: **for** each $\hat{\boldsymbol{\rho}}^{0:T(i)}$ and $\hat{\boldsymbol{V}}^{0:T(i)}$ **do**
6: Obtain $P^{0:T-1(i)}$ by solving the HJB.
7: **end for**
8: Obtain residual $r_{\theta^{(i)}}$ and $r_{\eta^{(i)}}$ according to Equations (3) and (6);
9: Update ρ_θ and V_η;
10: Check convergence (Equation (8)).
11: **end for**
12: Output MFE

$$\frac{\sum_{\boldsymbol{\rho}^0 \in \boldsymbol{\rho}^D} |\hat{\boldsymbol{\rho}}^{0:T(i)} - \hat{\boldsymbol{\rho}}^{0:T(i-1)}|}{|\boldsymbol{\rho}^D|} + \frac{\sum_{\boldsymbol{V}^0 \in \boldsymbol{V}^D} |\hat{\boldsymbol{V}}^{0:T(i)} - \hat{\boldsymbol{V}}^{0:T(i-1)}|}{|\boldsymbol{V}^D|} < \epsilon \tag{8}$$

5. Numerical Experiments

In this section, we employ our algorithm to facilitate autonomous driving navigation in traffic networks. As illustrated in Figure 2, a substantial number of autonomous vehicles (AVs) move to destination node 4, with the objective of minimizing total costs subject to the congestion effect. We use a representative agent as an example to elaborate on the speed control and density dynamics of the population in this scenario. At node 1, the representative agent first selects edge l_{12}. The agent then drives along edge l_{12} and selects continuous-time-space driving velocities on the edge. The agent selects her next-to-go edge at node 2, following this pattern until she reaches her destination at node 4. These choices regarding her route and speed will actively influence the evolution of population density across the network. The mathematical formulation of this autonomous driving game can be found in [16].

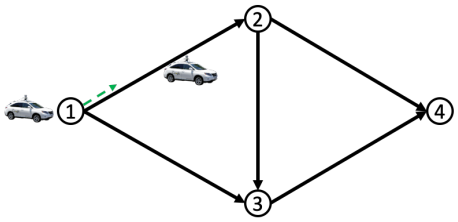
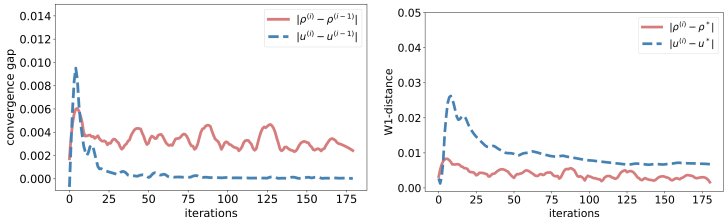


Figure 2. Autonomous driving game on the road network.

We construct the initial population state over the network as follows: We assume that at time 0, the traffic network is empty. Vehicles enter the road network at origin nodes $1, 2, 3$ and move toward the destination 4. Travel demands at each origin satisfy $d_i \sim \text{Uniform}[0, 1], i = 1, 2, 3$. Therefore, each initial population distribution over the network consists of travel demands at origins (i.e., $[d_1, d_2, d_3]$), which are sampled from three independent uniform distributions.

Figure 3 demonstrates the convergence performance of the algorithm in solving $\mathcal{G}$-MFG. The x-axis represents the iteration index during training, the y-axis displays the convergence gap, and the 1-Wasserstein distance measures the closeness between our results and the MFE obtained by numerical methods [16]. The results demonstrate that our algorithm can converge stably after 50 iterations.



(**a**) (**b**)

Figure 3. Algorithm performance (**a**) Convergence gap (**b**) W1-distance.

Figure 4 demonstrates the population density solved by our proposed method along three paths on the road network, i.e., $(1 \to 2 \to 3 \to 4)$, $(1 \to 2 \to 4)$, and $(1 \to 3 \to 4)$. The x-axis is the spatial position on the path, and the y-axis represents the time. The z-axis represents the population density $\boldsymbol{\rho}$. The running cost functional form follows a non-separable cost structure with a crossing term of the agent action and the population density. We visualize the population density in $\mathcal{G}$-MFG with three initial population states, which are constructed by travel demands $[d_1, d_2, d_3]$: $[0.6, 0.4, 0.2]$ (See Figure 4a–c), $[0.4, 0.4, 0.4]$

(See Figure 4d–f) and $[0.2, 0.4, 0.6]$ (See Figure 4g–i). The plots show the population density evolution along each path on the graph. At the node, the density flow can be split when making route choice. For example, at node 2, vehicles can choose node 3 or 4 as their next node. It means the vehicle flow on the edge (1,2) is split into flows on edges (2,3) and (2,4).

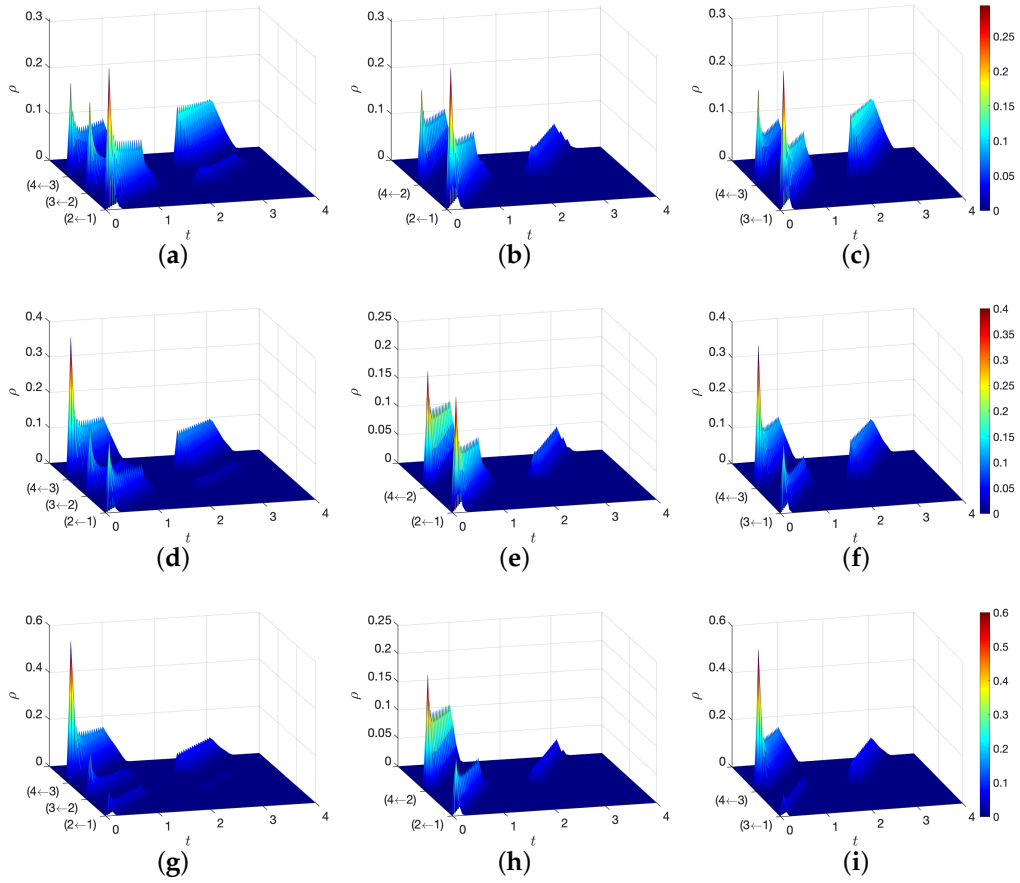


Figure 4. Population density $\boldsymbol{\rho}$ along each path on the road network with various travel demands $[d_1, d_2, d_3]$. (**a**) $1 \to 2 \to 3 \to 4$ (**b**) $1 \to 2 \to 4$, demand: $0.6, 0.4, 0.2$ (**c**) $1 \to 3 \to 4$ (**d**) $1 \to 2 \to 3 \to 4$ (**e**) $1 \to 2 \to 4$, demand: $0.4, 0.4, 0.4$ (**f**) $1 \to 3 \to 4$ (**g**) $1 \to 2 \to 3 \to 4$ (**h**) $1 \to 2 \to 4$, demand: $0.2, 0.4, 0.6$ (**i**) $1 \to 3 \to 4$.

6. Conclusions

In this paper, we propose a scalable learning framework, $\mathcal{G}$-MFGs. Existing numerical methods have to recompute MFE when the initial population density changes. To avoid recomputing MFE inefficiently, this work proposes a learning method, which utilizes graph neural networks to establish a functional mapping between the initial population density and MFE over the entire spatial temporal domain. This learning framework allows us to obtain MFEs corresponding to each initial population distribution without recomputing them. We demonstrate the efficiency of this method in autonomous driving games. Our contribution lies in the scalability of PIGNO to handle various initial population densities without recomputing MFEs. Our framework offers a memory- and data-efficient approach for solving $\mathcal{G}$-MFGs.

Author Contributions: Conceptualization, X.C.; methodology, S.L. and X.C.; validation, S.L. and X.C.; writing—original draft preparation, S.L. and X.C.; writing—review and editing, X.C. and X.D.; visualization, S.L.; supervision, X.D.; project administration, X.D.; funding acquisition, X.D.; S.L. and X.C. have the equal contribution to this paper. All authors have read and agreed to the published version of the manuscript.

Funding: This work is partially supported by the National Science Foundation CAREER under award number CMMI-1943998.

Data Availability Statement: No new data were created or analyzed in this study. Data sharing is not applicable to this article.

Conflicts of Interest: The authors declare no conflict of interest. All authors have approved the manuscript and agree with its submission to the journal "Games".

Appendix A

Appendix A.1. Mean Field Games on Graph

Optimal control of a representative agent: On a graph $\mathcal{G}$, a generic agent moves from her initial position to a destination, aiming to solve optimal control to minimize its cost connecting its origin to the destination. Assume there is a single destination $s \in \mathcal{N}$ for all the agents. One with multiple destinations forms a multi-population MFG and will be left for future research. The agent's state at time t can be specified by two scenarios, either in the interior of an edge or at a node. In the interior of edge $l \in \mathcal{L}$: (x,t) is the agent's position on edge l at time t where $x \in [0, \text{len}(l)]$ and $\text{len}(l)$ is the length of edge l. $u_l(x,t)$ is the velocity of the agent at position x at time t when navigating edge l. Note that $\mathcal{G}$-MFG is *non-stationary*, thus, the optimal velocity evolves as time progresses. $r_l(u,\rho)$ is the congestion cost arising from the agent population on edge l, which is increasingly monotone in ρ indicating the congestion effect. $V_l(x,t)$ is the minimum cost of the representative agent starting from position x at time t. $V_l(x,t)$ is modeled by a HJB equation: $\forall l \in \mathcal{L}$,

$$\begin{aligned} \partial_t V_l(x,t) + \min_u \{r_l(u,\rho) + u\partial_x V_l(x,t)\} = 0, \\ V_l(\text{len(l)},t) = \pi_i(t), l \in IN(i), \end{aligned} \tag{A1}$$

where, $\partial_t V_l(x,t), \partial_x V_l(x,t)$ are partial derivatives of $V_l(x,t)$ with respect to t,x, respectively. $IN(i)$ represent the edges going into node i. π_i is the minimum travel cost of agents staring from node i (i.e., the end of edge l) at time t, which is also the boundary condition of Equation (A1). At node $i \in \mathcal{N}$: $\beta(i,l,t)$ represents the probability of choosing the next-go-to edge $l \in OUT(i)$ where $OUT(i)$ represent the edges coming out of node i. We have $\sum_{l \in \mathcal{L}^O(i)} \beta(i,l,t) = 1$. $\beta(i,l,t)$ can be interpreted as the proportion of agents selecting edge l (or turning ratio) at node i at time t. β determines the boundary condition of population evolution on edges, which will be defined in Equation (A7). $\pi_i(t)$ is the minimum traverse cost starting from node i at time t. $\pi_i(t)$ satisfies

$$\begin{aligned} \pi_i(t) = \min_\beta \sum\nolimits_{l \in OUT(i)} \beta(i,l,t) \cdot V_l(0,t), \\ \pi_s(t) = 0. \end{aligned} \tag{A2}$$

where, $V_l(0,t)$ is the minimum cost entering edge l (i.e., $x = 0$) at time t. $\pi_s(t)$ is the terminal cost at destination s.

We now show Equations (A1) and (A2) can be reformulated into Equation (1b) in a discrete-time setting. We first discretize the spatiotemporal domain. On a spatiotemporal mesh grid, denote $\Delta x, \Delta t$ as the spatial and temporal mesh sizes, respectively. Denote $\mathcal{G}^D = \{\mathcal{N}^D, \mathcal{L}^D\}$ as the discretized representation of $\mathcal{G}$. To construct $\mathcal{G}^D$ from $\mathcal{G}$, we first discretize edges on a graph. Each edge $l = (i,j) \in \mathcal{L}$ is divided into a sequence of adjacent edge cells, denoted as $l^D = \{(i,i_1),(i_1,i_2),\dots(i_{|l^D|-1},j)\}$, where $|l^D| = \frac{\text{len}(l)}{\Delta x}$ is the number of adjacent edge cells. The node set $\mathcal{N}^D$ is created by augmenting $\mathcal{N}$ with auxiliary nodes $i_1, i_2, \cdots, |\mathcal{L}^D| - 1$ that separate newly split edge cells. In summary, a spatially discretized directed graph $\mathcal{G}^D$ is a collection of edge cells and augmented nodes linked by directed arrows. It preserves the topology of the original graph $\mathcal{G}$ but with more edges and nodes. We discretize the time interval into $[\cdots, \tau\Delta t, \cdots], \tau = 0, 1, \dots$, where τ represents the discretized time instant. The relation between the spatial and temporal resolutions

needs to fulfill the Courant–Friedrichs–Lewy (CFL) condition to ensure numerical stability: $\Delta t \cdot u_{max} \leqslant \Delta x$, where u_{max} is the maximum velocity. We first reformulate Equation (A1) on spatiotemporal grids where $\Delta t \cdot u_{max} \leqslant \Delta x$ as follows

$$\begin{aligned}
&\frac{V_l(x,t+\Delta t) - V_l(x,t)}{\Delta t} + \min_u \{f_l(u,\rho) + \\
&u\frac{V_l(x+\Delta x, t+\Delta t) - V_l(x,t+\Delta t)}{\Delta x}\} = 0 \\
&\rightarrow V_l(x,t) = \min_u \{f_l(u,\rho)\Delta t + (1 - \frac{\Delta t}{\Delta x}u)V_l(x,t+\Delta t) \\
&+ uV_l(x+\Delta x, t+\Delta t).\}, u \equiv u(x,t)
\end{aligned}$$

On the graph $\mathcal{G}^D = \{\mathcal{N}^D, \mathcal{L}^D\}$, $\forall (i,j) \in \mathcal{L}^D, j \notin \mathcal{N}$. We have $V_l(x,t) = V_{ij}^\tau, V_l(x,t+\Delta t) = V_{ij}^{\tau+1}, V_l(x+\Delta x, t+\Delta t) = V_{jk}^{\tau+1} = \pi_j^{\tau+1}$ where (j,k) is the successor edge cell of (i,j) in the interior of an edge. Therefore,

$$V_{ij}^\tau = \min_u \{V_{ij}^{\tau+1}(1 - \frac{\Delta t}{\Delta x}u_{ij}^\tau) + V_{jk}^{\tau+1}\frac{\Delta t}{\Delta x}u_{ij}^\tau + r_{ij}^\tau\}$$

where, $r_{ij}^\tau = \Delta t f(u_{ij}^\tau, \rho_{ij}^\tau)$.

Edge cell $(i,j) \in \mathcal{L}^D, j \in \mathcal{N}$ are the last cell on an edge and j is the end node of the edge. It means j is also the start node of the next-go-to edge. We have $V(x+\Delta x, t+\Delta t) = \pi_j^{\tau+1}$. Accordingly, $\forall (i,j) \in \mathcal{L}^D$,

$$V_{ij}^\tau = \min_u \{V_{ij}^{\tau+1}(1 - \frac{\Delta t}{\Delta x}u_{ij}^\tau) + \pi_j^{\tau+1}\frac{\Delta t}{\Delta x}u_{ij}^\tau + r_{ij}^\tau\}. \tag{A3}$$

We then look into Equation (A2). We denote $\beta_{i,l,t} = \beta_{ij}^\tau$ and we assume that agents entering node i and making route choice at time τ will exist the node at time $\tau+1$. We then have $\pi_i(t) = \pi_i^{\tau+1}, V_l(0,t) = Vij^{\tau+1}$. Therefore,

$$\pi_i^{\tau+1} = \min_\beta \sum_{j:(i,j)\in\mathcal{L}^D} \beta_{ij}^\tau \cdot V_{ij}^{\tau+1}, \forall i \in \mathcal{N}^D. \tag{A4}$$

We substitute π in Equation (A3) with V according to Equation (A4) and obtain Equation (1b). We have $\forall (i,j) \in \mathcal{L}^D$.

$$P^\tau_{|\mathcal{L}^D|\times|\mathcal{L}^D|} = \begin{bmatrix} \dots & \dots & \dots & \dots & \dots \\ \dots & \dots & \dots & \frac{\Delta t}{\Delta x}u_{mi}^\tau\beta_{ij}^\tau & \dots \\ \dots & \dots & \dots & \dots & \dots \\ \dots & \frac{\Delta t}{\Delta x}u_{ij}^\tau\beta_{jk}^\tau & \dots & 1 - \frac{\Delta t}{\Delta x}u_{ij}^\tau & \dots \\ \dots & \dots & \dots & \dots & \dots \end{bmatrix}. \tag{A5}$$

where, (j,k) is a successor edge cell of $(i,j) \in \mathcal{L}^D$. Below we show that P^τ is a transition matrix. For diagonal elements of P^τ, $\forall (i,j) \in \mathcal{L}^D$, we have

$$\begin{aligned}
&1 - \frac{\Delta t}{\Delta x}u_{ij}^\tau \leqslant 1, \\
&1 - \frac{\Delta t}{\Delta x}u_{ij}^\tau \geqslant 1 - \frac{\Delta t}{\Delta x}u_{max} \geqslant 1 - 1 = 0.
\end{aligned}$$

For off-diagonal elements,

$$0 \leqslant \frac{\Delta t}{\Delta x}u_{ij}^\tau \cdot \beta_{jk}^\tau \leqslant \frac{\Delta t}{\Delta x}u_{max} \cdot 1 \leqslant 1, (i,j), (j,k) \in \mathcal{L}^D.$$

The sum of elements in each row is:

$$1 - \frac{\Delta t}{\Delta x} u_{ij}^{\tau} + \sum_{k:(j,k)\in\mathcal{L}^D} \frac{\Delta t}{\Delta x} u_{ij}^{\tau} \cdot \beta_{jk}^{\tau} = 1, \forall (i,j) \in \mathcal{L}^D.$$

Therefore, P^{τ} is a transition matrix satisfying: (1) each element is between 0 and 1, and (2) the sum of elements in each row equals 1.

Population dynamics: When all agents follow the optimal control, the population density distribution on edge l, denoted as $\rho_l(x,t), \forall l \in \mathcal{L}$, evolves over the graph. It is solved by a deterministic (or first-order) FPK, given the velocity control $u_l(x,t)$ of agents:

$$\begin{aligned} \partial_t \rho_l(x,t) + \partial_x [\rho_l(x,t) \cdot u_l(x,t)] = 0, \\ \rho_l(x,0) = \rho_0(x), \end{aligned} \tag{A6}$$

where $\partial_t \rho_l(x,t), \partial_x \rho_l(x,t)$ are partial derivatives of $\rho_l(x,t)$ with respect to t, x, respectively. Since agents may not appear or disappear randomly, there is no stochasticity in this equation. $\rho_l(x,0)$ is the initial population density. At the starting node of edge l or the starting position on edge l, agents move to the next-go-to edge based on their route choice. Therefore, the *boundary condition* is:

$$\rho_l(0,t) = \beta(i,l,t)\{ \sum_{h\in IN(i)} [\rho_h(\text{len}(h),t) \cdot u_h(\text{len}(h),t)\}, \tag{A7}$$

where, $l \in OUT(i)$ and $\rho_l(0,t)$ is the influx entering edge l at time t. For a source node where new agents appear, this node can be treated as a dummy edge where agents exit this edge at a speed of u_{max} to enter a downstream edge. In the above boundary condition, there is no need to distinguish between an intermediate and a source node explicitly without loss of generality.

We now show Equations (A6) and (A7) can be reformulated into Equation (1a). We reformulate Equation (A6) as

$$\begin{aligned} &\frac{\rho_l(x,t+\Delta t) - \rho_l(x,t)}{\Delta t} \\ &= \frac{\rho_l(x-\Delta x,t)u_l(x-\Delta x,t) - \rho_l(x,t)u_l(x,t)}{\Delta x} \\ &\rho_l(x,t+\Delta t) = \rho_l(x,t) \\ &+ \frac{\Delta t}{\Delta x}[\rho_l(x-\Delta x,t)u_l(x-\Delta x,t) - \rho_l(x,t)u_l(x,t)] \end{aligned}$$

We denote $\rho_l(x,t+\Delta t) = \rho_{ij}^{\tau+1}, \rho_l(x,t) = \rho_{ij}^{\tau}, \forall (i,j) \in \mathcal{L}^D, i \notin \mathcal{N}$. $\rho_l(x-\Delta x,t)$ represents agents entering edge cell (i,j). If $i \notin \mathcal{N}$, $\rho_l(x-\Delta x,t) = \beta_{mi}^{\tau}\rho_{mi}^{\tau}, \beta_{mi}^{\tau} \equiv 1$. If $i \in \mathcal{N}$, agents entering (i,j) is calculated as $\beta_{ij}^{\tau} \sum_{m:(m,i)\in\mathcal{L}^D} \rho_{mi}^{\tau} u_{mi}^{\tau}$. Therefore,

$$\rho_{ij}^{\tau+1} = \rho_{ij}^{\tau} + \frac{\Delta t}{\Delta x}(\beta_{ij}^{\tau} \sum_{m:(m,i)\in\mathcal{L}^D} \rho_{mi}^{\tau} u_{mi}^{\tau} - \rho_{ij}^{\tau} u_{ij}^{\tau}) \tag{A8}$$

Therefore, Equation (1a) holds.

Appendix A.2. Toy Example

To further demonstrate the linkage of these MFGs on graphs, below we present a toy network (Figure A1) and show how each model is formulated on this network.

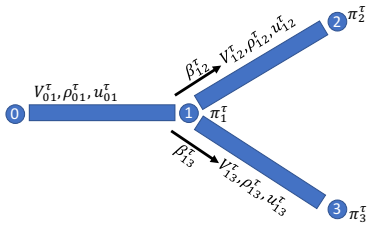


Figure A1. Toy example.

The $\mathcal{G}^D$-dMFG on the toy network is first presented. We assume $\pi_2^{\tau+1} = \tilde{V}_2^{\tau+1}$ and $\pi_3^{\tau+1} = \tilde{V}_3^{\tau+1}$. We have

$$
\begin{aligned}
\rho_{01}^{\tau+1} &= (1 - \frac{\Delta t}{\Delta x} u_{01}^{\tau}) \rho_{01}^{\tau} \\
\rho_{12}^{\tau+1} &= (1 - \frac{\Delta t}{\Delta x} u_{12}^{\tau}) \rho_{12}^{\tau} + \beta_{12}^{\tau} \rho_{01}^{\tau} \frac{\Delta t}{\Delta x} u_{01}^{\tau} \\
\rho_{13}^{\tau+1} &= (1 - \frac{\Delta t}{\Delta x} u_{13}^{\tau}) \rho_{13}^{\tau} + \beta_{13}^{\tau} \rho_{01}^{\tau} \frac{\Delta t}{\Delta x} u_{01}^{\tau} \\
V_{01}^{\tau} &= \min_{u} \{ r_{01}^{\tau} + (1 - \frac{\Delta t}{\Delta x} u_{01}^{\tau}) V_{01}^{\tau+1} + \frac{\Delta t}{\Delta x} u_{01}^{\tau} \pi_1^{\tau+1} \} \\
V_{12}^{\tau} &= \min_{u} \{ r_{12}^{\tau} + (1 - \frac{\Delta t}{\Delta x} u_{12}^{\tau}) V_{12}^{\tau+1} + \frac{\Delta t}{\Delta x} u_{12}^{\tau} \pi_2^{\tau+1} \} \\
V_{13}^{\tau} &= \min_{u} \{ r_{13}^{\tau} + (1 - \frac{\Delta t}{\Delta x} u_{13}^{\tau}) V_{13}^{\tau+1} + \frac{\Delta t}{\Delta x} u_{13}^{\tau} \pi_3^{\tau+1} \} \\
\pi_1^{\tau+1} &= \min_{\beta} \{ \beta_{12}^{\tau} V_{12}^{\tau+1} + \beta_{13}^{\tau} V_{13}^{\tau+1} \} \\
\pi_2^{\tau+1} &= \tilde{V}_2^{\tau+1} \\
\pi_3^{\tau+1} &= \tilde{V}_3^{\tau+1}
\end{aligned}
$$

We have

$$
\begin{aligned}
&[\mathcal{G}\text{-MFG}] \\
&(FPK)\ \boldsymbol{\rho}^{\tau+1} = [P^{\tau}]^T \boldsymbol{\rho}^{\tau}, \\
&(HJB)\ \boldsymbol{V}^{\tau} = \min_{\boldsymbol{u}, \boldsymbol{\beta}} P^{\tau} \boldsymbol{V}^{\tau+1} + \boldsymbol{r}^{\tau}, \\
&P_{5\times 5}^{\tau} = \\
&\begin{bmatrix}
1 - \frac{\Delta t}{\Delta x} u_{01}^{\tau} & \frac{\Delta t}{\Delta x} u_{01}^{\tau} \beta_{12}^{\tau} & \frac{\Delta t}{\Delta x} u_{01}^{\tau} \beta_{13}^{\tau} & 0 & 0 \\
0 & 1 - \frac{\Delta t}{\Delta x} u_{12}^{\tau} & 0 & \frac{\Delta t}{\Delta x} u_{12}^{\tau} & 0 \\
0 & 0 & 1 - \frac{\Delta t}{\Delta x} u_{13}^{\tau} & 0 & \frac{\Delta t}{\Delta x} u_{13}^{\tau} \\
0 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 & 1
\end{bmatrix}
\end{aligned}
$$

References

1. Lasry, J.M.; Lions, P.L. Mean field games. *Jpn. J. Math.* **2007**, *2*, 229–260. [CrossRef]
2. Huang, M.; Malhamé, R.P.; Caines, P.E. Large population stochastic dynamic games: Closed-loop McKean-Vlasov systems and the Nash certainty equivalence principle. *Commun. Inf. Syst.* **2006**, *6*, 221–252.
3. Yang, J.; Ye, X.; Trivedi, R.; Xu, H.; Zha, H. Deep Mean Field Games for Learning Optimal Behavior Policy of Large Populations. In Proceedings of the International Conference on Learning Representations, Vancouver, BC, Canada, 30 April–3 May 2018.
4. Elamvazhuthi, K.; Berman, S. Mean-field models in swarm robotics: A survey. *Bioinspir. Biomim.* **2019**, *15*, 015001. [CrossRef] [PubMed]
5. Calderone, D.; Sastry, S.S. Markov Decision Process Routing Games. In Proceedings of the 8th International Conference on Cyber-Physical Systems, ICCPS '17, Pittsburgh, PA, USA, 18–20 April 2017; Association for Computing Machinery: New York, NY, USA, 2017.

6. Cabannes, T.; Laurière, M.; Perolat, J.; Marinier, R.; Girgin, S.; Perrin, S.; Pietquin, O.; Bayen, A.M.; Goubault, E.; Elie, R. Solving N-Player Dynamic Routing Games with Congestion: A Mean-Field Approach. In Proceedings of the 21st International Conference on Autonomous Agents and Multiagent Systems, AAMAS '22, Auckland, New Zealand, 9–13 May 2022.
7. Guéant, O. Existence and uniqueness result for mean field games with congestion effect on graphs. *Appl. Math. Optim.* **2015**, *72*, 291–303. [CrossRef]
8. Guo, X.; Hu, A.; Xu, R.; Zhang, J. Learning Mean-Field Games. In *Advances in Neural Information Processing Systems (NeurIPS 2019)*; Curran Associates, Inc.: New York, NY, USA, 2019.
9. Subramanian, J.; Mahajan, A. Reinforcement Learning in Stationary Mean-Field Games. In Proceedings of the 18th International Conference on Autonomous Agents and MultiAgent Systems, AAMAS '19, Montreal, QC, Canada, 13–17 May 2019; pp. 251–259.
10. Perrin, S.; Laurière, M.; Pérolat, J.; Élie, R.; Geist, M.; Pietquin, O. Generalization in Mean Field Games by Learning Master Policies. *Proc. Aaai Conf. Artif. Intell.* **2022**, *36*, 9413–9421. [CrossRef]
11. Lauriere, M.; Perrin, S.; Girgin, S.; Muller, P.; Jain, A.; Cabannes, T.; Piliouras, G.; Perolat, J.; Elie, R.; Pietquin, O.; et al. Scalable Deep Reinforcement Learning Algorithms for Mean Field Games. In Proceedings of the 39th International Conference on Machine Learning, PMLR, Baltimore, MD, USA, 17–23 July 2022; Volume 162, pp. 12078–12095.
12. Ruthotto, L.; Osher, S.J.; Li, W.; Nurbekyan, L.; Fung, S.W. A machine learning framework for solving high-dimensional mean field game and mean field control problems. *Proc. Natl. Acad. Sci. USA* **2020**, *117*, 9183–9193. [CrossRef] [PubMed]
13. Carmona, R.; Laurière, M. Convergence Analysis of Machine Learning Algorithms for the Numerical Solution of Mean Field Control and Games I: The Ergodic Case. *SIAM J. Numer. Anal.* **2021**, *59*, 1455–1485. [CrossRef]
14. Germain, M.; Mikael, J.; Warin, X. Numerical resolution of McKean-Vlasov FBSDEs using neural networks. *Methodol. Comput. Appl. Probab.* **2022**, *24*, 2557–2586. [CrossRef]
15. Chen, X.; Fu, Y.; Liu, S.; Di, X. Physics-Informed Neural Operator for Coupled Forward-Backward Partial Differential Equations. In Proceedings of the 1st Workshop on the Synergy of Scientific and Machine Learning Modeling@ICML2023, Honolulu, HI, USA, 28 July 2023.
16. Chen, X.; Liu, S.; Di, X. Learning Dual Mean Field Games on Graphs. In Proceedings of the 26th European Conference on Artificial Intelligence, ECAI '23, Kraków, Poland, 30 September–5 October 2023.
17. Li, Z.; Kovachki, N.; Azizzadenesheli, K.; Liu, B.; Bhattacharya, K.; Stuart, A.; Anandkumar, A. Multipole Graph Neural Operator for Parametric Partial Differential Equations. In Proceedings of the 34th International Conference on Neural Information Processing Systems, Online, 6–12 December 2020; Curran Associates Inc.: New York, NY, USA 2020; NIPS'20.