Deep Equilibrium Network Formation

Anuraag Aekka
World Bank

March 2026

Preliminary

Please click here for latest version.

Abstract

Solving equilibrium models of network formation which feature large numbers of agents is difficult due to the high-dimensionality of an agent's state and due to the dependence of an agent's dynamics on its partners. In this paper, I introduce a new algorithm to solve such models using insights from deep learning and reinforcement learning. To address the challenge of a high-dimensional state space, policy function approximators are structured so that an agent's state is first mapped to a low-dimensional representation before outputting agent choices as a function of the agent's representation. A policy gradient method is used to improve policy functions. To address the challenge of modeling partner dynamics, the algorithm "learns" a type space and transition function from simulated data. In particular, it uses the low-dimensional representation of an agent as its type, and learns the transition function from transitions realized during simulation. The algorithm can be used to solve equilibrium models of network formation in more general settings. I apply the algorithm to a simple model of production network formation which cannot be solved using existing approaches.

1 Introduction

The role of networks in determining various outcomes of interest has received growing attention. For example, production network structure has been shown to have important implications for steady-state aggregate output and productivity, and the response of these outcomes to shocks. However, our understanding of how networks determine outcomes of interest is greatly limited by our ability to model network formation in equilibrium. Solving equilibrium models of network formation is difficult due to the high-dimensionality of an agent's state and due to the dependence of an agent's dynamics on its partners.

In the network setting, an agent's state includes the *set* of partners it is connected to in a given period. Solving for optimal policies in this high-dimensional state space is challenging due to the *curse of dimensionality*. Moreover, solving for optimal policies requires knowledge of how an agent's set of partners evolves over time. However, how a given partner evolves over time depends itself on how the partner's partners evolve. This dependence in dynamics implies that the evolution of a partner is in fact shaped by its entire upstream and downstream network. As a result, modeling partner dynamics becomes highly complex.

One approach to solving the problem of partner dynamics is to impose assumptions on the network formation process which imply a particular type space for partners and a corresponding transition function. Agents can use these objects to understand how their partners evolve over time. However, this approach is fragile. With even slight deviations in the network formation process, the type space and transition function no longer serve as sufficient objects to characterize partners. More generally, given a network formation process, it is rare that one can determine *a priori* a sufficient type space and transition function.

In this paper, I introduce a new algorithm to solve such models which addresses these challenges using insights from deep learning and reinforcement learning. To address the challenge of a high-dimensional state space, policy function approximators are structured so that an agent's state is first mapped to a low-dimensional representation before outputting agent choices as a function of the agent's representation. In addition, a policy gradient method is used to improve policy functions. To address the challenge of modeling partner dynamics, the algorithm "learns" a type space and transition function from simulated data. In particular, it uses the low-dimensional representation of an agent as its type, and learns the transition function from transitions realized during simulation. The algorithm can be used to solve equilibrium models of network formation in more general settings. To highlight the contribution of the algorithm, I apply it to a simple model of production network formation which cannot be solved using existing approaches.

The algorithm proceeds by iterating between a policy-update step and a conjecture-update step. In the policy-update step, agents improve policies taking as given current conjectures about how network formation choices determine new links and how existing partners transition through the type space over time.

Policy function approximators are structured so that an agent's state is first mapped to a low-dimensional representation via an "aggregation" function before outputting agent choices as a function of the agent's representation. Dimensionality reduction in this manner ensures that policy functions are parsimoniously approximated. This approach is motivated by the results of Han et al. (2019) and Zaheer et al. (2017), who show that certain types of aggregation functions are capable of approximating permutation-invariant functions arbitrarily well.[1]

Evaluating a policy requires computing expectations over future trajectories. However, standard numerical integration methods become computationally infeasible when the state space is high-dimensional. To overcome this, the algorithm relies on a key insight from reinforcement learning – that these expectations can be efficiently estimated via Monte Carlo simulation. In particular, the algorithm uses a policy gradient method to improve policies.

In the conjecture-update step, the algorithm takes an agent's low-dimensional representation as its type and uses choices and transitions realized during simulation to update conjectures about the choices of different types and the transition function.

The algorithm iterates on these steps until optimal policies and conjectures converge to fixed points. After convergence, equilibrium conditions can be tested to evaluate the quality of the solution.

I apply the algorithm to solve a simple model of production network formation via random search. The model features firm-level idiosyncratic productivity terms which follow a first-order autoregressive process. Though this stochastic process is quite simple and is widely used in macroeconomic models, it is not clear, *a priori*, what a sufficient type space and transition function are in this setting. Hence, existing approaches are unable to solve for an equilibrium. I choose this example to highlight the contribution of the algorithm.

Related Literature:

This paper contributes to the literature using deep learning to solve equilibrium models. Maliar et al. (2021), Fernández-Villaverde et al. (2023), Azinovic et al. (2022), and Han et al. (2021) develop deep learning algorithms to solve heterogeneous agent models with aggregate shocks. Deep learning methods perform significantly better than standard methods when models feature high-dimensional state spaces, large amounts of heterogeneity, and nonlinear

[1] A function f is permutation-invariant if $f(x_{\sigma(1)}, ..., x_{\sigma(N)}) = f(x_1, ..., x_N)$ for any permutation σ.

equilibrium functions (see Fernández-Villaverde et al. (2024) for a recent review).

This paper differs from existing work by using deep learning to solve models of network formation. Network models introduce unique challenges. Agent payoffs depend on the choices of other agents not just though an aggregate distribution, but also through the specific set of partners an agent is directly connected to. Solving for optimal policies is difficult as an agent's state is incredibly high-dimensional, and its evolution depends on how partners evolve over time.

A key component in tackling the high-dimensional state space is the mapping of agent states to low-dimensional representations. This dimensionality reduction is similar to Han et al. (2021), who use an aggregation function to summarize the aggregate distribution of agent states into a low-dimensional representation. Dimensionality reduction in this manner allows for tractable policy approximation despite the high-dimensionality of the state space. The algorithm in this paper differs from Han et al. (2021) in that each agent receives a different low-dimensional representation depending on its particular set of partners.

The low-dimensional representation is also central for addressing the challenge of modeling partner dynamics. The algorithm takes the low-dimensional representation of an agent as its type, and trains a transition function using transitions realized during simulation. Note that, in general, it is rare that one can determine *a priori* a sufficient type space and transition function to characterize the evolution of partners.

This paper also contributes to literature studying dynamic network formation with large numbers of agents. Several models in this class focus on the formation of production networks. Lim (2018) studies a model where idiosyncratic shocks affect the value of relationships over time and hence lead to dynamics in links. Huneeus (2020) builds on this framework by introducing adjustment costs which prevent firms from readjusting customers and suppliers. In Boehm et al. (2024), potential suppliers arrive randomly according to a Poisson process. Aekka and Khanna (2024) consider an environment in which firms match with new partners over time via random search. Outside of the production network literature, Dayanıklı and Laurière (2025) study a setting in which agents choose the "strength" of connection with respect to different types of agents in each period. In these settings, assumptions on the network formation technology and stochastic process governing idiosyncratic characteristics give rise to a type space and transition function which can be used to model partner dynamics. However, even with slight deviations from the imposed assumptions, these objects are no longer sufficient to characterize partners. The algorithm presented in this paper differs from existing work by learning the type space and transition function through simulation. This allows for solving more general models in which these objects cannot be determined *a*

priori.

The paper is organized as follows. In Section 2, I present a general model of network formation and describe the challenge of solving for an equilibrium. In Section 3, I introduce a deep learning algorithm to compute an approximate equilibrium. In Section 4, I apply the algorithm to solve a simple model of production network formation.

2 Network Formation Equilibria

In this section, I present a general model of network formation and describe the challenge of solving for an equilibria. I then define an approximate equilibrium which can be solved for using the algorithm introduced in Section 3.

2.1 Network Environment

Let $\{(V_t, E_t)\}_{t=0,1,2,\ldots,\infty}$ denote a sequence of time-indexed directed graphs, where V_t denotes a set of nodes in period t and E_t denotes a set of directed edges in period t. Let $V = \bigcup_{t=0}^{\infty} V_t$ denote the set of nodes across all periods. Each node $v \in V$ corresponds to an "agent". Agent v enters in period $\tau(v)$, and exogenously exits at the end of each period with probability $(1-\beta)$. Going forward, I use the terms "agent" and "node" interchangeably.

The set of edges in each period is endogenously determined by agents' actions. In each period, each agent $v \in V_t$ chooses network-formation action $a_t(v) \in \mathbb{R}^{d_a}$. Given the current set of edges, E_t, network-formation actions determine the set of edges in period $t+1$ according to:

$$\begin{aligned} E_{t+1} &= E_{t+1}^{new} \cup E_{t+1}^{cont} \\ E_{t+1}^{new} &\sim \Psi^{new}\left(\{a_t(v)\}_{v \in V_t}\right), \\ E_{t+1}^{cont} &= \{(v, v') \in E_t : v \text{ survives}, v' \text{ survives}\} \end{aligned}$$

Here E_{t+1}^{new} denotes the set of new links formed by network-formation actions, and E_t^{cont} denotes the set of links which continue from the current period. A link continues if both partners survive (do not exit).

Let $\mathcal{N}_t^+(v)$ and $\mathcal{N}_t^-(v)$ denote "out-neighborhood" and "in-neighborhood" of agent v in period t:

$$\mathcal{N}_t^+(v) = \{v' \in V : (v, v') \in E_t\}, \quad \mathcal{N}_t^-(v) = \{v' \in V : (v', v) \in E_t\}$$

I impose the following assumptions on the network formation technology. These assumptions

allow for tractable policy optimization via policy gradient methods in Section 3.

Assumption 1

Network formation technology Ψ^{new} satisfies the following conditions:

1. *The probability distribution over the number of new out-neighbors matched with by agent v, $n_{t+1}^{+,new}(v)$, is differentiable with respect to $a_t(v)$.*
2. *The probability distribution over the number of new in-neighbors matched with by agent v, $n_{t+1}^{-,new}(v)$, is differentiable with respect to $a_t(v)$.*
3. *Conditional on $n_{t+1}^{+,new}(v)$, the set of new out-neighbors matched with by agent v is independent of $a_t(v)$.*
4. *Conditional on $n_{t+1}^{-,new}(v)$, the set of new in-neighbors matched with by agent v is independent of $a_t(v)$.*

Agents choose interactions with respect to current out-neighbors, $b_t^{out}(v,v') \in \mathbb{R}^{d_{bout}}$, and with respect to current in-neighbors, $b_t^{in}(v,v') \in \mathbb{R}^{d_{bin}}$. Agent v's period payoff is given by:

$$r_t(v) = \sum_{v' \in \mathcal{N}_t^+(v)} R^{out}\Big(b_t^{out}(v,v'), b_t^{in}(v',v)\Big) + \sum_{v' \in \mathcal{N}_t^-(v)} R^{in}\Big(b_t^{in}(v,v'), b_t^{out}(v',v)\Big) \; - \; c(a_t(v))$$

where $R^{out}(\cdot)$ and $R^{in}(\cdot)$ denote payoffs from interactions with out-neighbors and in-neighbors, respectively, and $c(\cdot)$ denotes the cost of network-formation action $a_t(v)$.

Agent v's problem is then given by:

$$\begin{aligned} J(v) = \max_{\{a_t(v), b_t^{out}(v,v'), b_t^{in}(v,v')\}_{t=\tau(v)}^{\infty}} \mathbb{E}\Bigg[\sum_{t=\tau(v)}^{\infty} \beta^{t-\tau(v)} \Bigg(\sum_{v' \in \mathcal{N}_t^+(v)} R^{out}\Big(b_t^{out}(v,v'), b_t^{in}(v',v)\Big) + \\ \sum_{v' \in \mathcal{N}_t^-(v)} R^{in}\Big(b_t^{in}(v,v'), b_t^{out}(v',v)\Big) \; - \; c(a_t(v)) \Bigg) \Bigg] \qquad (1) \\ s.t. E_{t+1} = E_{t+1}^{new} \cup E_{t+1}^{cont} \\ E_{t+1}^{new} \sim \Psi^{new}\left(\{a_t(v)\}_{v \in V_t}\right), \\ E_{t+1}^{cont} = \{(v,v') \in E_t : v \text{ survives}, v' \text{ survives}\} \end{aligned}$$

2.2 Nash Equilibrium.

I now define a Nash Equilibrium for this environment.

Definition 1 ***Nash Equilibrium:*** *A Nash Equilibrium is a set of policy functions $\{a_t, b_t^{in}, b_t^{out}\}$, such that, for every $v \in V$, policy functions $\{a_t(v), b_t^{in}(v,v'), b_t^{out}(v,v')\}$ solve*

agent problem 1, given policies $\{a_t(u), b_t^{in}(u,v'), b_t^{out}(u,v')\}_{u \neq v}$.

Solving for Nash Equilibrium in agent "identities" becomes intractable as the number of agents becomes large, since it requires solving for network-formation actions for each agent and all pairwise interactions.

2.3 Rational Expectations Equilibrium

A standard approach to solving this *curse of dimensionality* is to map agents to "states" and restrict policies to be functions of an agent's state. Solving for an equilibrium then corresponds to solving for functions which satisfy equilibrium conditions. Generally as the dimension of agents and actions increase, solving for equilibrium policy functions in the state space is more tractable than solving for equilibrium policies in agent identities.

To this end, let $\omega(v) \in \Omega$ denote the state of agent $v \in V$. The state of agent v in period t is given by its out-neighborhood and in-neighborhood at the beginning of period t:

$$\omega_t(v) = \left(\mathcal{N}_t^+(v), \mathcal{N}_t^-(v)\right)$$

For agents who enter in the current period, $\tau(v) = t$, the out-neighborhood and in-neighborhood are empty sets:

$$\omega_t(v) = (\emptyset, \emptyset) \quad \text{if } t = \tau(v)$$

Furthermore, I assume that each agent can be mapped to a "type" $\phi(\omega(v)) \in \mathbb{R}^{d_x}$, where $\phi : \Omega \to \mathbb{R}^{d_x}$ denotes a type mapping. Let $x_t(v) = \phi(\omega_t(v))$ denote the type of agent v. Accordingly, redefine the out-neighborhood and in-neighborhood to be sets of types rather than identities:

$$\mathcal{N}_t^+(v) = \{x_t(v') : (v,v') \in E_t\}, \quad \mathcal{N}_t^-(v) = \{x_t(v') : (v',v) \in E_t\}$$

Redefine policies $\{b^{in}, b^{out}, a\}$ to be functions of agent states and neighbor types, rather than their identities. Furthermore, let $\{\tilde{b}^{out}, \tilde{b}^{in}\}$ denote conjectures about the interactions of neighbors. The period payoff of an agent with state $\omega = \left(\mathcal{N}_t^+, \mathcal{N}_t^-\right)$ is then given by:

$$\tilde{r}\left(\omega\right) = \sum_{x' \in \mathcal{N}_t^+} R^{out}\left(b^{out}(\omega, x'), \tilde{b}^{in}(x', \phi(\omega))\right) + \sum_{x' \in \mathcal{N}_t^-} R^{in}\left(b^{in}(\omega, x'), \tilde{b}^{out}(x', \phi(\omega))\right) - c(a(\omega))$$

Note that the conjectures depend on the *type* of agent v rather than its *state*.

Finally, let $\Gamma^+(a)$ denote a conjectured distribution over the set of new out-neighbor types matched with, conditional on choosing network-formation action a; and let $\Gamma^-(a)$ denote a conjectured distribution over the set of new in-neighbor types matched with, conditional on choosing network-formation action a. Let $\gamma^+(\mathcal{N}^+)$ denote a conjectured law of motion for the next-period types of continuing out-neighbors; and let $\gamma^-(\mathcal{N}^-)$ denote a conjectured law of motion for the next-period types of continuing in-neighbors.

The problem of an agent born in period τ is then given by:

$$\begin{aligned} \tilde{J}(\tau) = \max_{\{a,b^{out},b^{in}\}_{t=\tau}^{\infty}} \mathbb{E}\left[\sum_{t=\tau}^{\infty} \beta^{t-\tau}\tilde{r}(\omega_t)\right] \\ s.t. \quad \mathcal{N}_{t+1}^{+} = \mathcal{N}_{t+1}^{+,new} \cup \mathcal{N}_{t+1}^{+,cont}, \quad \mathcal{N}_{t+1}^{-} = \mathcal{N}_{t+1}^{-,new} \cup \mathcal{N}_{t+1}^{-,cont} \\ \mathcal{N}_{t+1}^{+,new} \sim \Gamma^+(a(\omega_t)), \quad \mathcal{N}_{t+1}^{-,new} \sim \Gamma^-(a(\omega_t)) \\ \mathcal{N}_{t+1}^{+,cont} \sim \gamma^+(\mathcal{N}_t^+), \quad \mathcal{N}_{t+1}^{-,cont} \sim \gamma^-(\mathcal{N}_t^-) \\ \omega_\tau = (\emptyset, \emptyset) \end{aligned} \tag{2}$$

I now define a Rational Expectations Equilibrium.

Definition 2 ***Rational Expectations Equilibrium:***

A rational expectations equilibrium (REE) is a type mapping, ϕ, a set of optimal policies, $\{a, b^{in}, b^{out}\}$, policy conjectures $\{\tilde{b}^{in}, \tilde{b}^{out}\}$, conjectured distributions over new neighbors, $\{\Gamma^+, \Gamma^-\}$, and conjectured laws of motion, $\{\gamma^+, \gamma^-\}$, such that:

1. *Optimal policies $\{a, b^{in}, b^{out}\}$ solve agent problem 2 given conjectures $\{\tilde{b}^{in}, \tilde{b}^{out}, \Gamma^+, \Gamma^-, \gamma^+, \gamma^-\}$.*
2. *Policy conjectures are consistent with optimal policies:*

$$\forall\, \omega, x: \quad \tilde{b}^{in}(\phi(\omega), x) = b^{in}(\omega, x)$$

$$\forall\, \omega, x: \quad \tilde{b}^{out}(\phi(\omega), x) = b^{out}(\omega, x)$$

3. *Conjectured distributions and laws of motion are consistent with optimal policies and*

network formation technology:

$$
\begin{aligned}
E_{t+1}^{new} &\sim \Psi^{new}\left(\{a(\omega_t(v))\}_{v\in V_t}\right) \\
E_{t+1}^{cont} &= \{(v,v') \in E_t : v \text{ survives}, v' \text{ survives}\} \\
\forall\, v,t: \quad \Gamma^+(a(\omega_t(v))) &\overset{d}{=} \{x_{t+1}(v') : (v,v') \in E_{t+1}^{new}\} \\
\forall\, v,t: \quad \Gamma^-(a(\omega_t(v))) &\overset{d}{=} \{x_{t+1}(v') : (v',v) \in E_{t+1}^{new}\} \\
\forall\, v,t: \quad \gamma^+\left(\mathcal{N}_t^+(v)\right) &\overset{d}{=} \{x_{t+1}(v') : (v,v') \in E_{t+1}^{cont}\} \\
\forall\, v,t: \quad \gamma^-\left(\mathcal{N}_t^-(v)\right) &\overset{d}{=} \{x_{t+1}(v') : (v',v) \in E_{t+1}^{cont}\}
\end{aligned}
$$

Solving for an equilibrium corresponds to solving for equilibrium objects which satisfy the equilibrium conditions. Generally as the dimension of agents and actions increase, solving for an REE is more tractable than solving for a Nash Equilibrium in agent identities. However, a major challenge in solving for an REE is that the functional forms of the equilibrium objects are generally unknown. I next describe an approximate REE which utilizes parametric approximators.

2.4 Approximate Rational Expectations Equilibrium

I describe an approximate REE which utilizes parametric approximators.

Let $\hat{\phi}_{\rho_\phi}$ denote a type mapping approximator, where ρ_ϕ denotes its corresponding parameters. Let $\left\{\hat{a}_{\rho_a}, \hat{b}^{in}_{\rho_{b^{in}}}, \hat{b}^{out}_{\rho_{b^{out}}}\right\}$ denote policy function approximators, with corresponding parameters $\left\{\rho_a, \rho_{b^{in}}, \rho_{b^{out}}\right\}$. Let $\left\{\hat{\tilde{b}}^{in}_{\rho_{\tilde{b}^{in}}}, \hat{\tilde{b}}^{out}_{\rho_{\tilde{b}^{out}}}\right\}$ denote policy conjecture approximators, with corresponding parameters $\left\{\rho_{\tilde{b}^{in}}, \rho_{\tilde{b}^{out}}\right\}$. Let $\left\{\hat{\Gamma}^+_{\rho_{\Gamma^+}}, \hat{\Gamma}^-_{\rho_{\Gamma^-}}\right\}$ denote approximators of new neighbor draw distributions, with corresponding parameters $\left\{\rho_{\Gamma^+}, \rho_{\Gamma^-}\right\}$.

For continuing neighbors, let the law of motion for the out-neighborhood be approximated by:

$$\{x'_j \sim \hat{\gamma}^+(\hat{\phi}(\omega_t), x_j) : x_j \in \mathcal{N}_t^+(\omega_t) \text{ and } \delta_j = 0\}$$

where $\hat{\gamma}^+$ is an out-neighbor transition approximator with parameter ρ_{γ^+}, and δ_j is a Bernoulli random variable which indicates whether the neighbor j exits between periods t and $(t+1)$. Similarly, let the law of motion for the in-neighborhood be approximated by :

$$\{x'_j \sim \hat{\gamma}^-(\hat{\phi}(\omega_t), x_j) : x_j \in \mathcal{N}_t^-(\omega_t) \text{ and } \delta_j = 0\}$$

where $\hat{\gamma}^-$ is an in-neighbor transition approximator with parameter ρ_{γ^-}, and δ_j is a Bernoulli random variable indicating whether neighbor j exits between periods t and $(t+1)$. Note that

the transition approximators draw next-period types for continuing neighbors independently, conditional on an agent's type, rather than jointly for the entire neighborhood. I discuss this choice in more detail in Section 3.

The period payoff of an agent with state $\omega = \left(\mathcal{N}_t^+, \mathcal{N}_t^-\right)$ is then given by:

$$\hat{r}\left(\omega\right) = \sum_{x' \in \mathcal{N}_t^+} R^{out}\left(\hat{b}^{out}(\omega, x'), \hat{\tilde{b}}^{in}(x', \hat{\phi}(\omega))\right) + \sum_{x' \in \mathcal{N}_t^-} R^{in}\left(\hat{b}^{in}(\omega, x'), \hat{\tilde{b}}^{out}(x', \hat{\phi}(\omega))\right) - c(\hat{a}(\omega))$$

The problem of an agent born in period τ is given by:

$$\begin{aligned}
\hat{J}(\tau) = & \max_{\{\hat{a}, \hat{b}^{out}, \hat{b}^{in}\}_{t=\tau}^{\infty}} \mathbb{E}\left[\sum_{t=\tau}^{\infty} \beta^{t-\tau} \hat{r}(\omega_t)\right] \\
s.t. \quad \mathcal{N}_{t+1}^+ = & \mathcal{N}_{t+1}^{+,new} \cup \mathcal{N}_{t+1}^{+,cont}, \quad \mathcal{N}_{t+1}^- = \mathcal{N}_{t+1}^{-,new} \cup \mathcal{N}_{t+1}^{-,cont} \\
& \mathcal{N}_{t+1}^{+,new} \sim \hat{\Gamma}^+(\hat{a}(\omega_t)), \quad \mathcal{N}_{t+1}^{-,new} \sim \hat{\Gamma}^-(\hat{a}(\omega_t)) \\
& \mathcal{N}_{t+1}^{+,cont} = \{x_j' \sim \hat{\gamma}^+(\hat{\phi}(\omega_t), x_j) : x_j \in \mathcal{N}_t^+ \text{ and } \delta_j = 0\} \\
& \mathcal{N}_{t+1}^{-,cont} = \{x_j' \sim \hat{\gamma}^-(\hat{\phi}(\omega_t), x_j) : x_j \in \mathcal{N}_t^- \text{ and } \delta_j = 0\} \\
& \omega_\tau = (\emptyset, \emptyset)
\end{aligned} \tag{3}$$

Finally, I define an Approximate Rational Expectations Equilibrium:

Definition 3 ***Approximate Rational Expectations Equilibrium:***
Given a set of approximators $\hat{\Theta} = \left\{\hat{\phi}, \hat{a}, \hat{b}^{in}, \hat{b}^{out}, \hat{\tilde{b}}^{in}, \hat{\tilde{b}}^{out}, \hat{\Gamma}^+, \hat{\Gamma}^-, \hat{\gamma}^+, \hat{\gamma}^-\right\}$,
an approximate rational expectations equilibrium (AREE) is a set of parameters $\rho = \left\{\rho_\phi, \rho_a, \rho_{b^{in}}, \rho_{b^{out}}, \rho_{\tilde{b}^{in}}, \rho_{\tilde{b}^{out}}, \rho_{\Gamma^+}, \rho_{\Gamma^-}, \rho_{\gamma^+}, \rho_{\gamma^-}\right\}$ *such that:*

1. *Policy parameters* $\{\rho_a, \rho_{b^{in}}, \rho_{b^{out}}\}$ *solve agent problem 3 given conjectures* $\left\{\hat{\tilde{b}}^{in}, \hat{\tilde{b}}^{out}, \hat{\Gamma}^+, \hat{\Gamma}^-, \hat{\gamma}^+, \hat{\gamma}^-\right\}$.
2. *Policy conjectures are consistent with optimal policies:*

$$\forall\, \omega, x: \quad \hat{\tilde{b}}^{in}(\hat{\phi}(\omega), x) = \hat{b}^{in}(\omega, x)$$

$$\forall\, \omega, x: \quad \hat{\tilde{b}}^{out}(\hat{\phi}(\omega), x) = \hat{b}^{out}(\omega, x)$$

3. *Conjectured distributions and laws of motion are consistent with optimal policies and*

network formation technology:

$$E_{t+1}^{new} \sim \Psi^{new}\left(\{\hat{a}(\omega_t(v))\}_{v \in V_t}\right)$$

$$E_{t+1}^{cont} = \{(v, v') \in E_t : v \text{ survives}, v' \text{ survives}\}$$

$$\forall\, v, t: \quad \hat{\Gamma}^{+}(\hat{a}(\omega_t(v))) \overset{d}{=} \{x_{t+1}(v') : (v, v') \in E_{t+1}^{new}\}$$

$$\forall\, v, t: \quad \hat{\Gamma}^{-}(\hat{a}(\omega_t(v))) \overset{d}{=} \{x_{t+1}(v') : (v', v) \in E_{t+1}^{new}\}$$

$$\forall\, v, t, x \in \mathcal{N}_t^{+}(v): \quad \hat{\gamma}^{-}(x, \hat{\phi}(\omega_t(v))) \overset{d}{=} \mathcal{L}\left(\hat{\phi}(\omega_{t+1}(v)) \;\middle|\; \hat{\phi}(\omega_t(v)),\, x \in \mathcal{N}_t^{+}(v)\right)$$

$$\forall\, v, t, x \in \mathcal{N}_t^{-}(v): \quad \hat{\gamma}^{+}(x, \hat{\phi}(\omega_t(v))) \overset{d}{=} \mathcal{L}\left(\hat{\phi}(\omega_{t+1}(v)) \;\middle|\; \hat{\phi}(\omega_t(v)),\, x \in \mathcal{N}_t^{-}(v)\right)$$

where $\mathcal{L}\left(\hat{\phi}(\omega_{t+1}(v)) \;\middle|\; \hat{\phi}(\omega_t(v)),\, x \in \mathcal{N}_t^{+}(v)\right)$ *and* $\mathcal{L}\left(\hat{\phi}(\omega_{t+1}(v)) \;\middle|\; \hat{\phi}(\omega_t(v)),\, x \in \mathcal{N}_t^{-}(v)\right)$ *denote the conditional laws of motion for* $\hat{\phi}(\omega)$ *under the policies and conjectures.*

Solving for an approximate equilibrium corresponds to solving for approximator parameters which satisfy the equilibrium conditions. An equilibrium here is "approximate" here due to the fact that transition approximators are required to be consistent with conditional laws of motion for individual agent transitions, rather than the joint transition of an entire neighborhood. In the case that conditioning on an agent's type is sufficient for capturing correlation in neighbor transitions, these two consistency requirements are equivalent.

3 Deep Learning an Approximate Equilibrium

In this section, I introduce an algorithm to solve for AREE. The algorithm uses deep neural networks to approximate equilibrium objects and proceeds by iterating between a policy-update step and a conjecture-update step for simulated batches of agents.

The algorithm needs to address two key challenges. First, approximators need to be sufficiently flexible so that they approximate the equilibrium objects well. This is particularly challenging in the network setting due to the fact that an agent's state includes its set of partners, and hence the certain equilibrium functions are *set* functions. Deep neural networks serve as attractive approximators here due "universal approximation theorems" which show that they are able to approximate functions in various function spaces arbitrarily well.

Second, the algorithm needs to determine a type mapping, as generally, this mapping is unknown *a priori*. To address this challenge, policy function approximators are structured so that an agent's state is first mapped to a low-dimensional representation before agent choices are outputted as functions of the representation. The algorithm uses this low-dimensional representation as the type of an agent. Deep learning is well-suited for this problem due to

its ability to learn low-dimensional representations of high-dimensional objects.

The algorithm consists of four components (1) a set of approximators for equilibrium objects (2) a loss function to evaluate the quality of a given approximation (3) a training set to evaluate the loss over (4) and a method to update approximator parameters. I describe these components in sequence.

Equilibrium Object Approximators:

The algorithm relies on deep neural networks to approximate equilibrium objects. "Universal approximation theorems" have shown that deep neural networks are capable of approximating functions in various function spaces arbitrarily well.

To approximate policy functions, the algorithm first maps agent states to low-dimensional representations via an "aggregation" function before outputting agent choices as a function of the low-dimensional representation. The aggregation function is given by:

$$\hat{\phi}_{\rho_\phi}\left(\mathcal{N}_t^+, \mathcal{N}_t^-\right) = \hat{\phi}_{\rho_{\phi,0}}\left(\hat{\phi}^{out,1}_{\rho_{out,1}}\left(\sum_{x' \in \mathcal{N}_t^+} \hat{\phi}^{out,0}_{\rho_{out,0}}(x')\right), \hat{\phi}^{in,1}_{\rho_{in,1}}\left(\sum_{x' \in \mathcal{N}_t^-} \hat{\phi}^{in,0}_{\rho_{in,0}}(x')\right)\right) \tag{4}$$

where $\{\hat{\phi}_{\rho_{\phi,0}}, \hat{\phi}^{out,1}_{\rho_{out,1}}, \hat{\phi}^{out,0}_{\rho_{out,0}}, \hat{\phi}^{in,1}_{\rho_{in,1}}, \hat{\phi}^{in,0}_{\rho_{in,0}}, \}$ are fully-connected feed-forward neural networks with parameters $\rho_\phi = \{\rho_{\phi,0}, \rho_{out,0}, \rho_{out,1}, \rho_{in,0}, \rho_{in,1}\}$.

This approach is motivated by results of Han et al. (2019) and Zaheer et al. (2017), who show that permutation-invariant functions can be approximated arbitrarily well by aggregation functions similar to 4. Han et al. (2021) use a similar aggregation function to summarize the aggregate distribution of agent states into a low-dimensional representation. The mapping here differs from Han et al. (2021), in that an agent's low-dimensional representation depends on its particular set of partners.

Policy functions then output agent choices as a function of an agent's low-dimensional representation. In particular I approximate policy functions with fully-connected feed-forward neural networks $\{\hat{a}_{\rho_a}(x), \hat{b}^{in}_{\rho_{b^{in}}}(x), \hat{b}^{out}_{\rho_{b^{out}}}(x)\}$, where $\{\rho_a, \rho_{b^{in}}, \rho_{b^{out}}\}$ denote the corresponding neural network parameters.

In order to approximate policy conjectures, I use fully-connected feed-forward neural networks $\left\{\hat{b}^{in}_{\tilde{\rho}_{b^{in}}}(x), \hat{b}^{out}_{\tilde{\rho}_{b^{out}}}(x)\right\}$, where $\{\rho_{\tilde{b}^{in}}, \rho_{\tilde{b}^{out}}\}$ denote the corresponding parameters. Hornik et al. (1989) show that feed-forward neural networks are capable of approximating Borel measurable functions arbitrarily well.

Finally, in order to approximate laws of motion $\{\Gamma^+, \Gamma^-, \gamma^+, \gamma^-\}$, the algorithm uses *normalizing flows*, $\left\{\hat{\Gamma}^+_{\rho_{\Gamma^+}}, \hat{\Gamma}^-_{\rho_{\Gamma^-}}, \hat{\gamma}^+_{\rho_{\gamma^+}}, \hat{\gamma}^-_{\rho_{\gamma^-}}\right\}$, where $\left\{\rho_{\Gamma^+}, \rho_{\Gamma^-}, \rho_{\gamma^+}, \rho_{\gamma^-}\right\}$ denote the corresponding parameters. A normalizing flow approximates a probability distribution by taking a simple

"base" distribution and passing it through several transformations to generate an output distribution. The transformations are implemented using neural networks and maintain properties which allow for efficient sampling and likelihood evaluation.[2] Huang et al. (2018) show that certain classes of normalizing flows are capable of approximating any strictly positive and continuous probability distribution arbitrarily well.

Note that the transition approximators differ from the "true" neighborhood laws of motion in that they draw transitions for each neighbor of an agent independently, conditional on the agent's type, rather than jointly for the entire neighborhood. However, the transitions of neighbors may be correlated since they are all connected to the agent and their types depend on the agent's type. The approximator attempts to capture this correlation by conditioning on the agent's type. This approach is motivated by the intuition that, with a large number of agents, partners of the same agent are likely only connected through the agent.

This approach is also motivated by the low computational cost of training $\hat{\gamma}^+$ and $\hat{\gamma}^-$. As will be described in more detail below, $\hat{\gamma}^+$ and $\hat{\gamma}^-$ are trained using simulated trajectories of agents and their direct partners. The loss function for each approximator is computed by comparing actual transitions of agent types to transitions conjectured by $\hat{\gamma}^+$ or $\hat{\gamma}^-$. Computing the actual type of an agent in a given period is computationally cheap, as an agent's type is just a nonlinear function of its partners types. Computing the conjectured type via $\hat{\gamma}^+$ or $\hat{\gamma}^-$ requires only the agent's type in the previous period and a partner's type in the previous period. Thus one needs to only simulate trajectories for a batch of agents and their direct partners to compute the loss. On the other hand, suppose one used a transition approximator that predicted the transition of the entire partner set jointly. To train this via simulated data, one would need to simulate the entire network, and then compare actual transitions in firm types to conjectured ones. This becomes prohibitively expensive as the number of agents, N, increases, as computing the actual types requires solving a nonlinear system of N equations in each period.

Loss Function:

The algorithm proceeds by iterating between a policy-update step and a conjecture-update step and uses a separate loss function for each.

For the policy-update step, the algorithm uses the expected lifetime value to train policy parameters.

[2]See Papamakarios et al. (2021) for an introduction to normalizing flows.

$$
\begin{aligned}
& & L^{pol}(\rho_{pol}) &= \mathbb{E}\left[\sum_{t=0}^{\infty} \beta^t \hat{r}(\omega_t)\right] \\
s.t. \quad & \mathcal{N}^{+}_{t+1} = \mathcal{N}^{+,new}_{t+1} \cup \mathcal{N}^{+,cont}_{t+1}, \quad & \mathcal{N}^{-}_{t+1} &= \mathcal{N}^{-,new}_{t+1} \cup \mathcal{N}^{-,cont}_{t+1} \\
& \mathcal{N}^{+,new}_{t+1} \sim \hat{\Gamma}^{+}(\hat{a}(\omega_t)), \quad & \mathcal{N}^{-,new}_{t+1} &\sim \hat{\Gamma}^{-}(\hat{a}(\omega_t)) \\
& \mathcal{N}^{+,cont}_{t+1} = \{x'_j \sim \hat{\gamma}^{+}(\hat{\phi}(\omega_t), x_j) : x_j \in \mathcal{N}^{+}_t \text{ and } \delta_j = 0\} & & \\
& \mathcal{N}^{-,cont}_{t+1} = \{x'_j \sim \hat{\gamma}^{-}(\hat{\phi}(\omega_t), x_j) : x_j \in \mathcal{N}^{-}_t \text{ and } \delta_j = 0\} & & \\
& \omega_0 = (\emptyset, \emptyset) & &
\end{aligned}
\tag{5}
$$

Evaluating the loss requires computing expectations over future trajectories. However, standard numerical integration methods become computationally infeasible when the state space is high-dimensional. To overcome this, the algorithm relies on a key insight from reinforcement learning – that these expectations can be efficiently estimated via Monte Carlo simulation. The algorithm simulates a batch of B agents for T periods under current policies and estimates the gradient of the loss function via REINFORCE. Assumption 1 ensures that this gradient can be estimated tractably via policy gradient methods.

At the end of the policy-update step, the algorithm stores the sequence of realized agent states and types. These sets will be used in the conjecture-update step.

$$
\chi_\omega = \{\omega_{i,t}\}_{i=1,2,...,B;\ t=0,1,...,T-1}, \quad \chi_\phi = \left\{\hat{\phi}(\omega_{i,t})\right\}_{i=1,2,...,B;\ t=0,1,...,T-1}
$$

Note that the policy loss function depends on both policy parameters and conjecture parameters. I omit reference to the conjecture parameters, however, for the sake of brevity. During training, the policy loss function is only used to update policy parameters. I follow a similar notation and training approach for conjecture parameters.

In the conjecture-update step, the algorithm uses agent transitions realized in the policy-update step to train conjectured objects. The loss with respect to the out-neighbor transition approximator $\hat{\gamma}^{+}$ is given by:

$$
L^{\gamma^+}(\rho_{\gamma^+}) = -\frac{1}{B}\sum_{i=1}^{B}\sum_{t=0}^{T-2}\sum_{x' \in \mathcal{N}^{-}_{i,t}} log\, p_{\hat{\gamma}^+}\left(\hat{\phi}(\omega_{i,t+1}) \,|\, x', \hat{\phi}(\omega_{i,t})\right)
$$

where $p_{\hat{\gamma}^+}\left(\hat{\phi}(\omega_{i,t+1}) \,|\, x', \hat{\phi}(\omega_{i,t})\right)$ is the likelihood under $\hat{\gamma}^{+}$ of an out-neighbor transitioning from $\hat{\phi}(\omega_{i,t})$ to $\hat{\phi}(\omega_{i,t+1})$ for an x' agent. Similarly, the loss with respect to the in-neighbor

transition approximator $\hat{\gamma}^-$ is given by:

$$L^{\gamma^-}(\rho_{\gamma^-}) = -\frac{1}{B}\sum_{i=1}^{B}\sum_{t=0}^{T-2}\sum_{x'\in\mathcal{N}_{i,t}^+} log\, p_{\hat{\gamma}^-}\left(\hat{\phi}(\omega_{i,t+1}) \,|\, x', \hat{\phi}(\omega_{i,t})\right)$$

where $p_{\hat{\gamma}^-}\left(\hat{\phi}(\omega_{i,t+1}) \,|\, x', \hat{\phi}(\omega_{i,t})\right)$ is the likelihood under $\hat{\gamma}^-$ of an in-neighbor transitioning from $\hat{\phi}(\omega_{i,t})$ to $\hat{\phi}(\omega_{i,t+1})$ for an x' agent.

Note that the loss functions with respect to $\hat{\gamma}^+$ and $\hat{\gamma}^-$ can be evaluated using only simulated trajectories of a batch of agents and their direct partners. This makes evaluating the loss function computationally cheap.

I assume that the losses of $\hat{\Gamma}^+$ and $\hat{\Gamma}^-$ can be evaluated by the likelihood of observing certain moments of the realized set of types, χ_ϕ, under $\hat{\Gamma}^+$ and $\hat{\Gamma}^-$. This is the case for the application I present in Section 4. I denote the corresponding loss function with $L^{\Gamma}(\rho_{\Gamma^+}, \rho_{\Gamma^-})$.

The interaction conjecture parameters do not have a corresponding loss function as they will be updated to slowly track the current optimal interactions.

Again, in general, all loss functions depend on all parameters. However, each set of parameters is updated using only its corresponding loss function.

Training Set:

The simulation method used to estimate the gradient of the policy loss function also defines the training set. Both policies and conjectures are trained using points in the state space reached by simulated agents under the current policies. As the current policies approaches the optimal policies, the losses will be trained using points likely to be reached under the optimal policies, i.e. the ergodic set. As argued by Judd et al. (2011), this type of stochastic simulation method is ideal for high-dimensional applications as it only solves the model in the area of the state space where agents actually "live".

Approximator Update:

Approximator parameters are updated via stochastic gradient descent. The algorithm works by iterating between a policy-update step and a conjecture-update step.

In the policy update-step, policy parameters ρ_{pol} are updated using policy loss, $L^{pol}(\rho_{pol})$.

At the end of the policy-update step, the algorithm stores agent states and types realized during simulation to update conjectures.

Interaction conjecture parameters are adjusted slowly to match the current optimal policies:

$$\rho^m_{\tilde{b}^{out}} = \kappa\rho^m_{b^{out}} + (1-\kappa)\rho^{m-1}_{\tilde{b}^{out}}, \quad \rho^m_{\tilde{b}^{in}} = \kappa\rho^m_{b^{in}} + (1-\kappa)\rho^{m-1}_{\tilde{b}^{in}}$$

where m denotes the training iteration and $\kappa \in (0, 1]$.

New neighbor distributions and law of motion conjectures $\left\{\rho_{\Gamma^+}, \rho_{\Gamma^-}, \rho_{\gamma^+}, \rho_{\gamma^-}\right\}$ are updated using respective loss functions, $\left\{L^{\Gamma}(\rho_{\Gamma^+}, \rho_{\Gamma^-}), L^{\gamma^+}(\rho_{\gamma^+}), L^{\gamma^-}(\rho_{\gamma^-})\right\}$.

Again, though all loss functions depend on all parameters, each set of parameters is only updated using its corresponding loss function.

The policy-update and conjecture-update steps are repeated until convergence. After convergence, equilibrium conditions can be tested to evaluate the quality of the solution.

Full Algorithm.

The full algorithm is summarized in Algorithm 1.

Algorithm 1 Solution Algorithm

Initialize policy parameters, $\rho_{pol} = \left\{\rho_\phi, \rho_a, \rho_{b^{out}}, \rho_{b^{in}}\right\}$, interaction conjecture parameters $\left\{\rho_{\tilde{b}^{out}}, \rho_{\tilde{b}^{in}}\right\}$, and new neighbor and LOM conjecture parameters, $\left\{\rho_{\Gamma^+}, \rho_{\Gamma^-}, \rho_{\gamma^+}, \rho_{\gamma^-}\right\}$.
for $n_m = 1$ to N_m **do**
    Initialize batch of agents with $\omega_{i,0} = (\emptyset, \emptyset)$ for $i = 1, 2, ..., B$.
(Policy Update)
    for $n_o = 1$ to N_o **do**
        Simulate agents for T periods to estimate gradient of $L^{pol}(\rho_{pol})$ via REINFORCE.
        Update policy parameters, ρ_{pol}, via gradient descent using loss $L^{pol}(\rho_{pol})$.
    end for
    Store final sequences of agent states and types in χ_ω and χ_ϕ.
(Conjecture Update)
    Update interaction conjecture parameters $\{\rho_{\tilde{b}^{in}}, \rho_{\tilde{b}^{out}}\}$:

$$\rho^m_{\tilde{b}^{in}} = \kappa\rho^m_{b^{in}} + (1-\kappa)\rho^{m-1}_{\tilde{b}^{in}}, \quad \rho^m_{\tilde{b}^{in}} = \kappa\rho^m_{b^{in}} + (1-\kappa)\rho^{m-1}_{\tilde{b}^{in}}$$

    for $n_e = 1$ to N_e **do**
        Update ρ_{γ^+} via gradient descent using loss $L^{\gamma^+}(\rho_{\gamma^+})$.
        Update ρ_{γ^-} via gradient descent using loss $L^{\gamma^-}(\rho_{\gamma^-})$.
        Update $\left\{\rho_{\Gamma^+}, \rho_{\Gamma^-}\right\}$ via gradient descent using loss $L^{\Gamma}(\rho_{\Gamma^+}, \rho_{\Gamma^-})$.
    end for
end for

4 Application: Production Network Formation

In this section, I apply the algorithm introduced in Section 3 to solve a model of production network formation. The production network model cannot be solved using existing approaches despite being fairly simple. I choose this example to highlight the contribution of the algorithm.

4.1 Model Outline:

I briefly outline the important details of the model. The economy is inhabited by a representative household which inelastically supplies labor and a continuum of firms that produce differentiated varieties. Firms are heterogeneous in productivity and operate production technologies that use labor and varieties produced by other firms as inputs. The economy features matching frictions, which entail that a firm can only trade with other firms it is connected to. In order to connect with trading partners, firms, in each period, exert search effort and links are formed via an aggregate matching function. Suppliers choose prices to charge customers, and customers choose quantities to purchase.

Firms. There exists a continuum I of firms. Firm $i \in I$ produces a differentiated variety using a constant returns to scale production technology:

$$y_t(i) = z_t(i) \left(\alpha l_t(i)^{\frac{\theta-1}{\theta}} + (1-\alpha) m_t(i)^{\frac{\theta-1}{\theta}} \right)^{\frac{\theta}{\theta-1}}$$

Here, $z_t(i)$ denotes the firm's productivity in period t, $l_t(i)$ denotes the quantity of labor it uses in time period t, and $m_t(i)$ denotes an intermediate bundle:

$$m_t(i) = \left(\sum_{j \in \mathcal{N}_t^-} \nu_t(i,j)^{\frac{\sigma-1}{\sigma}} \right)^{\frac{\sigma}{\sigma-1}}$$

where $\{\nu_t(i,j)\}_{j \in \mathcal{N}_t^-}$ denotes quantities of intermediate inputs the firm demands from its suppliers in period t, $\mathcal{N}_t^-$. Varieties are imperfect substitutes with constant elasticity of substitution, $\sigma \geq 1$. Firm productivity evolves according to:

$$log(z_t(i)) = \zeta \, log(z_{t-1}(i)) + \sigma_\epsilon \varepsilon_t(i), \quad \varepsilon_t(i) \sim \mathcal{N}(0,1)$$

Initial productivities are drawn from log-normal distribution, Lognormal $\left(\mu_z, \sigma_z^2\right)$.

Supplier choose prices to charge their customers, and customers choose quantities to pur-

chase. Given the set of suppliers in period t, firm i minimizes the cost of satisfying downstream demand $\bar{y}_t(i)$:

$$C\Big(\{p_t(i,j)\}_{j\in\mathcal{N}_t^-},\, \bar{y}_t(i)\Big) = \min_{l(i,t),\{\nu_t(i,j)\}_{j\in\mathcal{N}_t^-}} l_t(i) + \sum_{j\in\mathcal{N}_t^-} p_t(i,j)\nu_t(i,j)$$
$$s.t. \quad z_t(i)\left(\alpha l_t(i)^{\frac{\theta-1}{\theta}} + (1-\alpha)m_t(i)^{\frac{\theta-1}{\theta}}\right)^{\frac{\theta}{\theta-1}} \geq \bar{y}_t(i) \tag{6}$$

where the wage is the numeraire and $p_t(i,j)$ denotes the price firm j charges firm i.

Firms sell their variety to the representative household and to other firms. A firm's set of firm-to-firm customers is denoted by $\mathcal{N}_t^+$.

At the end of every period, firms exogenously exit with probability $(1-\beta)$.

Representative Household. There exists a representative household that inelastically supplies its labor endowment $L = 1$, owns all firms, and consumes differentiated varieties. The household maximizes utility subject to a budget constraint. Household utility is given by a composite of varieties it consumes:

$$W(t) = \max_{\left\{\nu_t^f(i)\right\}_{i\in I}} \left(\int_{i\in I} \nu_t^f(i)^{\frac{\sigma-1}{\sigma}} di\right)^{\frac{\sigma}{\sigma-1}} \quad s.t. \quad \int_{i\in I} p_t^f(i)\nu_t^f(i)di \leq E_t$$

where varieties are imperfect substitutes with constant elasticity of substitution $\sigma \geq 1$, $p_t^f(i)$ denotes the price firm i charges the household, and household income, E_t, is the sum of labor income and firm profits:

$$E_t = 1 + \int_{i\in I} \pi_t(i)di$$

Network . There exist search and matching frictions which entail that a firm can only trade with other firms it has matched with. Firms enter without any firm-to-firm links and slowly match with partners over time using a random search technology.

Specifically, in period t, firm i exerts matching effort for finding customers, $u_t(i) \in \mathbb{R}_+$, and for finding suppliers, $v_t(i) \in \mathbb{R}_+$. Exerting effort $u_t(i)$ results in $\tilde{u}_t(i)$ customer "advertisements" being posted:

$$\tilde{u}_t(i) \sim B\left(n, \min\left\{\frac{u_t(i)}{n},\, 1\right\}\right)$$

where $B(\cdot)$ denotes the binomial distribution, and n denotes the number of trials.

Similarly exerting effort $v_t(i)$ results in $\tilde{v}_t(i)$ supplier "advertisements" being posted:

$$\tilde{v}_t(i) \sim B\left(n, \min\left\{\frac{v_t(i)}{n}, 1\right\}\right)$$

Note that $\mathbb{E}[\tilde{u}_t(i)|u_t(i)] = u_t(i)$ for $u_t(i) \leq n$, and $\mathbb{E}[\tilde{v}_t(i)|v_t(i)] = v_t(i)$ for $v_t(i) \leq n$.

Exerting effort $u_t(i)$ costs $\varphi_c(u_t(i))$ units of labor and exerting effort $v_t(i)$ costs the firm $\varphi_s(v_t(i))$ units of labor. I assume $\frac{\partial \varphi_c(u)}{\partial u} > 0$, $\frac{\partial^2 \varphi_c(u)}{\partial u^2} > 0$, $\frac{\partial \varphi_s(v)}{\partial v} > 0$, and $\frac{\partial^2 \varphi_s(v)}{\partial v^2} > 0$. Given the advertisements posted by firms, the number of new matches formed is given by aggregate matching function:

$$\mathcal{M}_t = \mathcal{V}_t(1 - \exp(-\kappa \mathcal{U}_t/\mathcal{V}_t))$$
$$\mathcal{U}_t = \left(\int_{i\in I} \tilde{u}_t(i)di\right), \quad \mathcal{V}_t = \left(\int_{i\in I} \tilde{v}_t(i)di\right)$$

The probability of a customer advertisement resulting in a new customer match is given by $\frac{\mathcal{M}_t}{\mathcal{U}_t}$ and the probability of a supplier advertisement resulting in a new supplier match is given by $\frac{\mathcal{M}_t}{\mathcal{V}_t}$. Conditional on matching with a new customer, the probability that the new customer is in measurable set $\mathcal{C}$ is given by $\left(\int_{i\in\mathcal{C}} \tilde{v}_t(i)di\right)/\mathcal{V}_t$. Similarly, conditional on matching with a new supplier, the probability that the new supplier is in measurable set $\mathcal{C}$ is given by $\left(\int_{i\in\mathcal{C}} \tilde{u}_t(i)di\right)/\mathcal{U}_t$.

In each period, each existing link is destroyed with probability δ.

Note that the random matching technology implies that the probability of posting $\tilde{u}$ customer advertisements is differentiable in u (almost everywhere), and that the probability of posting $\tilde{v}$ supplier advertisements is differentiable in v (almost everywhere). Furthermore, note that the set of new customers matched with is independent of u, conditional on $\tilde{u}$, and that the set of new suppliers matched with is independent of v, conditional on $\tilde{v}$. These properties allow for tractable policy optimization via policy gradient methods.

Timing. For clarity, I summarize the sequence of events which occurs in each time period:

1. Measure n_t^e of new firms enter.
2. Firms exert matching effort, produce output, and sell to the household and other firms.
3. Share δ of existing links exogenously destroyed and share $(1-\beta)$ of existing firms exogenously exit.
4. Matches formed with new customers and suppliers.

4.2 AREE for Production Network Model:

Despite its simplicity, existing approaches are unable to solve for an equilibrium of the production network model described in Section 4.1. The main challenge is that, due to the stochastic evolution of firm productivity and neighbors, it is difficult to determine a type-mapping and transition function for agents *a priori*. Without these objects, one has to solve the model in agent "identities", which becomes infeasible as the number of agents becomes large. An alternative strategy sometimes feasible is to use the "history" of shocks experienced by an agent as its type. However, this is intractable in the current setting due to the dimensionality of shocks. In each period, firms realize shocks in their idiosyncratic productivity, in the number of advertisements posted, and in the partners matched with conditional on posting.

I solve for an AREE of the production network using Algorithm 1. Again, the key difference is that rather than determining the type-mapping and transition function *a priori*, Algorithm 1 learns these objects through simulation.

For reference, I display the values chosen for structural parameters in Table 1.

Table 1: Structural Parameters

Parameter	Description	Value
β	exogenous firm survival rate	0.90
ρ	productivity persistence	0.95
σ_ϵ	productivity innovation std.	0.05
μ_z	initial productivity mean	1.00
σ_z	initial productivity std.	0.25
α	labor share parameter	0.50
θ	labor-intermediate EoS	3.00
σ	intermediate variety EoS	3.00
μ	constant markup	1.50
n	maximum advertisments	50
κ	matching function efficiency	0.50
ξ	level of search costs	1.00
ζ	curvature of search costs	4.00
δ	exogenous partner separation	0.01

Solution Evaluation. After Algorithm 1 converges, one can test whether equilibrium conditions hold by evaluating various moments. In Table 2, I present the moments used to evaluate the solution quality in the current setting. I describe the moments in detail below.

Interactions between agents in this setting are the the prices firms charge their customers and the quantities of inputs they demand from their suppliers. As parameters of interaction

conjectures are slowly updated to match the parameters of corresponding optimal policies, consistency of interaction conjectures is ensured by the convergence of optimal policies.

The model also requires consistency of conjectured household demand. Conjectured household demand depends on the household's price index - which can be computed using optimal policies - and on conjectured household income. Thus, to evaluate the consistency of conjectured household demand, I compare conjectured household income to household income realized under optimal policies. Specifically, I simulate a large batch of agents for many periods and construct a cross-section of firm profits from the simulated trajectories. Realized household income is then computed by aggregating profits from the cross-section and adding labor income. I compute the symmetric mean absolute percentage error (SMAPE) between conjectured household income and realized household income.[3]

To evaluate the consistency of transition approximators, I compare agent transitions realized during simulation to corresponding conjectured transitions. For the out-neighbor transition approximator, $\hat{\gamma}^+$, I first take realized firm types, $\{\hat{\phi}(\omega_{i,t})\}_{i=1,2,...,B;\, t=0,1,...,T}$, as well as in-neighbors for each agent in each period, $\{x' \in \mathcal{N}^-_{i,t}\}_{i=1,2,...,B;\, t=0,1,...,T}$. I use $\hat{\gamma}^+$ to draw a sample of conjectured transitions, $\{\hat{\gamma}^+(x', \hat{\phi}(\omega_{i,t}))\}_{x' \in \mathcal{N}^-_{i,t};\, i=1,2,...,B;\, t=0,1,...,T-1}$ and then compute the maximum mean discrepancy (MMD) between this conjectured sample and the realized sample of transitions $\{\hat{\phi}(\omega_{i,t+1})\}_{x' \in \mathcal{N}^-_{i,t};\, i=1,2,...,B;\, t=0,1,...,T-1}$. If the conjecture is consistent, the two samples should be similar and the MMD between them should be low.[4] Similarly, for the in-neighbor transition approximator, $\hat{\gamma}^-$, I take out-neighbors for each agent in each period, use $\hat{\gamma}^-$ to draw a sample of conjectured transitions for the agents, and compute the MMD between the conjectured sample and the realized sample of transitions.

Given the random matching technology, one can compute the probability of matching with a new neighbor of a particular type given the stationary distribution of types $F(x)$ and optimal search policies. Thus consistency of new partner draw conjectures $\{\hat{\Gamma}^+, \hat{\Gamma}^-\}$ in the current setting is equivalent to the the consistency of the stationary distribution. To evaluate the consistency of the stationary distribution, I draw a large sample from the stationary distribution, transition it forward one period, and compute the MMD between the samples in the two time periods. In the case the stationary distribution conjecture is consistent, the two samples should be similar and the MMD should be low. In order to transition the stationary distribution forward, I use a conjecture on the unconditional transition of types

[3] $SMAPE\,(y, \hat{y}) \equiv \frac{|y-\hat{y}|}{(|y|+|\hat{y}|)/2}$

[4] Given samples $\hat{X} \sim P$, $\hat{Y} \sim Q$, $(MMD(\hat{X}, \hat{Y}))^2$ estimates $\sup_{f \in \mathcal{F}_k} \mathbb{E}_{X \sim P}\left[f(X)\right] - \mathbb{E}_{Y \sim Q}\left[f(Y)\right]$, where $\mathcal{F}_k \equiv \{f \in \mathcal{H} : ||f||_{\mathcal{H}} \leq 1\}$ and $\mathcal{H}$ denotes the Reproducing Kernel Hilbert Space associated with kernel function k. I use the RBF kernel $k(X, Y) = \exp\left(-\frac{||X-Y||_2^2}{2\eta^2}\right)$

Table 2: AREE Evaluation:

Moment	Value
E simulated vs. conjectured SMAPE	0.02
$\gamma^+(x, x')$ simulated vs. conjectured MMD	0.04
$\gamma^-(x, x')$ simulated vs. conjectured MMD	0.03
$g(x, x')$ simulated vs. conjectured MMD	0.01
$F^e(x)$ simulated vs. conjectured MMD	0.08
$F(x)$ Pre vs. Post-Transition MMD	< 0.01

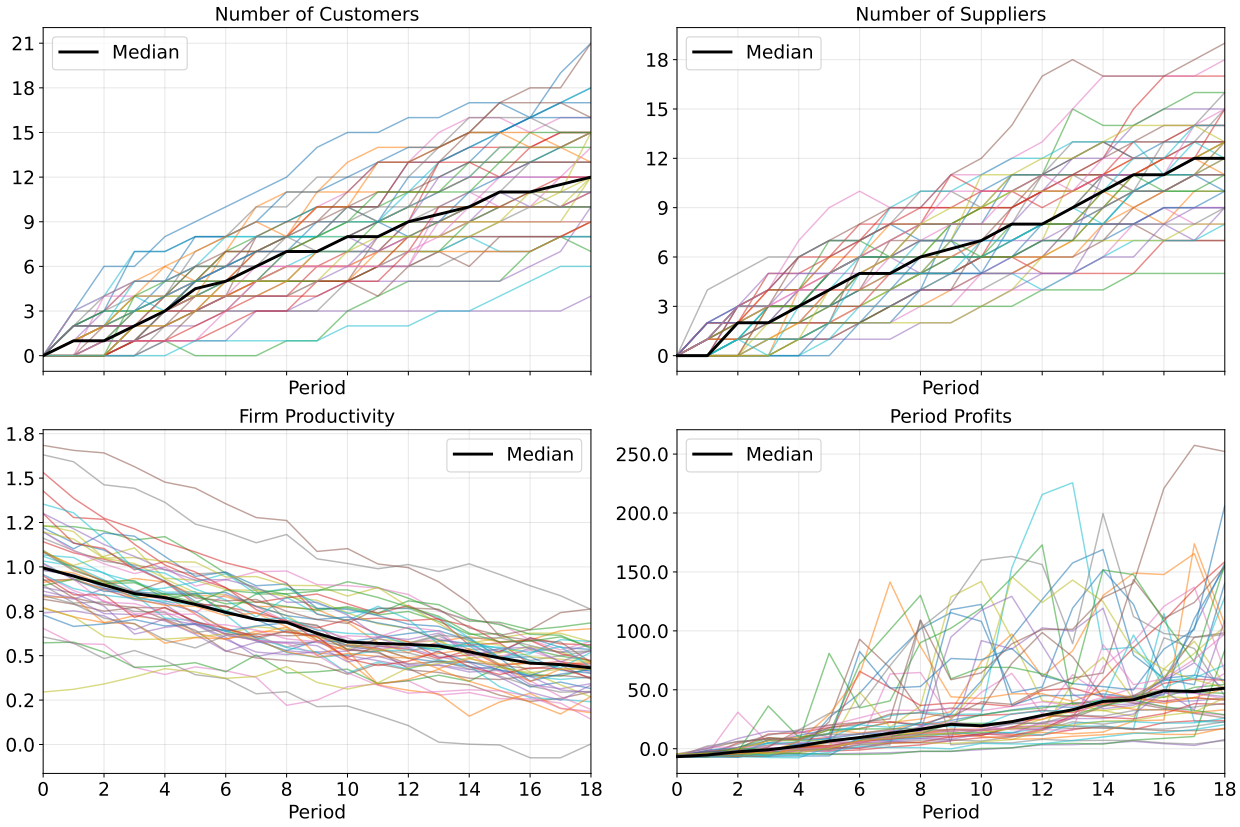
$g(x, x')$, as well as a conjecture on the distribution of entrant firms $F^e(x)$. These conjectures are also trained via Algorithm 1 using simulated trajectories. Similar to the conditional transition conjectures, I evaluate the consistency of g by computing the MMD between agent transitions realized during simulation and a corresponding sample of conjectured transitions. To evaluate the consistency of F^e, I compute the MMD between the realized entrant types during simulation and a corresponding sample of conjectured types.

As shown in Table 2, all evaluation moments are small in magnitude. The SMAPE between simulated and conjectured household income is 0.02, implying consistency of conjectured household demand. The MMD statistics comparing conjectured and simulated transitions are all below 0.05, indicating that transitions generated by $\hat{\gamma}^+$, $\hat{\gamma}^-$, and g align closely with those realized under optimal policies. The MMD between conjectured and realized entrant distributions is 0.08, implying consistency of conjectured and realized entrant types. Finally, the MMD between the pre- and post-transition stationary distributions is close to zero, indicating stationarity. Taken together, these moments indicate that equilibrium conditions are approximately satisfied by the solution returned by Algorithm 1.

4.3 Model Output

In this section, I describe output from the quantitative model. In Figure 1, I plot trajectories of firm attributes for a simulated batch of 50 firms. The top two panels display number of customers and number of suppliers, while the bottom panels display log firm productivity and period profits. Firms accumulate customers and suppliers over time through search. A larger customer base expands sales on the extensive margin, while a larger supplier base lowers marginal cost. Together, these dynamics generate rising profitability over the lifecycle.

Figure 1: Firm Attributes over the Lifecycle



Note: I plot trajectories for number of customers, number of suppliers, log firm productivity, and period profits for a simulated batch of 50 firms.

5 Conclusion

This paper introduces a new algorithm for solving dynamic models of network formation using insights from deep learning and reinforcement learning. The algorithm addresses the challenges of solving such models by mapping high-dimensional agent states to low-dimensional representations, improving agent policies via a policy gradient method, and learning a type space and transition function to characterize partners from simulated data. The algorithm can be used to solve equilibrium models of network formation in more general settings.

References

Aekka, A. and G. Khanna (2024). Endogenous production networks and firm dynamics.".

Azinovic, M., L. Gaegauf, and S. Scheidegger (2022). Deep equilibrium nets. *International Economic Review 63*(4), 1471–1525.

Boehm, J., R. South, E. Oberfield, and M. Waseem (2024). The network origins of firm dynamics: Contracting frictions and dynamism with long-term relationships.

Dayanıklı, G. and M. Laurière (2025). A game-theoretic framework for network formation in large populations. In *2025 IEEE 64th Conference on Decision and Control (CDC)*, pp. 6832–6837.

Fernández-Villaverde, J., S. Hurtado, and G. Nuno (2023). Financial frictions and the wealth distribution. *Econometrica 91*(3), 869–901.

Fernández-Villaverde, J., G. Nuño, and J. Perla (2024). Taming the curse of dimensionality: quantitative economics with deep learning. Technical report, National Bureau of Economic Research.

Han, J., Y. Li, L. Lin, J. Lu, J. Zhang, and L. Zhang (2019). Universal approximation of symmetric and anti-symmetric functions. *arXiv preprint arXiv:1912.01765*.

Han, J., Y. Yang, et al. (2021). Deepham: A global solution method for heterogeneous agent models with aggregate shocks. *arXiv preprint arXiv:2112.14377*.

Hornik, K., M. Stinchcombe, and H. White (1989). Multilayer feedforward networks are universal approximators. *Neural networks 2*(5), 359–366.

Huang, C.-W., D. Krueger, A. Lacoste, and A. Courville (2018). Neural autoregressive flows. In *International conference on machine learning*, pp. 2078–2087. PMLR.

Huneeus, F. (2020). Production network dynamics and the propagation of shocks.

Judd, K. L., L. Maliar, and S. Maliar (2011). Numerically stable and accurate stochastic simulation approaches for solving dynamic economic models. *Quantitative Economics 2*(2), 173–210.

Lim, K. (2018). Endogenous production networks and the business cycle.

Maliar, L., S. Maliar, and P. Winant (2021). Deep learning for solving dynamic economic models. *Journal of Monetary Economics 122*, 76–101.

Papamakarios, G., E. Nalisnick, D. J. Rezende, S. Mohamed, and B. Lakshminarayanan (2021). Normalizing flows for probabilistic modeling and inference. *Journal of Machine Learning Research 22*(57), 1–64.

Zaheer, M., S. Kottur, S. Ravanbakhsh, B. Poczos, R. R. Salakhutdinov, and A. J. Smola (2017). Deep sets. *Advances in neural information processing systems 30*.