Solving Structured Hierarchical Games Using Differential Backward Induction

Abstract

From large-scale organizations to decentralized political systems, hierarchical strategic decision making is commonplace. We introduce a novel class of *structured hierarchical games (SHGs)* that formally capture such hierarchical strategic interactions. In an SHG, each player is a node in a tree, and strategic choices of players are sequenced from root to leaves, with root moving first, followed by its children, then followed by their children, and so on until the leaves. A player's utility in an SHG depends on its own decision, and on the choices of its parent and *all* the tree leaves. SHGs thus generalize simultaneous-move games, as well as Stackelberg games with many followers. We leverage the structure of both the sequence of player moves as well as payoff dependence to develop a novel gradient-based back propagation-style algorithm, which we call *Differential Backward Induction (DBI)*, for approximating equilibria of SHGs. We provide a sufficient condition for convergence of DBI and demonstrate its efficacy in finding approximate equilibrium solutions to several SHG models of hierarchical policy-making problems.

1 INTRODUCTION

The COVID-19 pandemic has revealed considerable strategic tension among the many parties involved in decentralized hierarchical policy-making. For example, recommendations by the World Health Organization are sometimes heeded, and other times discarded by nations, while subnational units, such as provinces and urban areas, may in turn take a policy stance (such as on lockdowns, mask mandates, or vaccination priorities) that is not congruent with national policies. Similarly, in the US, policy recommendations at the federal level can be implemented in a variety of ways by the states, while counties and cities, in turn, may comply with state-level policies, or not, potentially triggering litigation [15]. Central to all these cases is that, besides this strategic drama, what ultimately determines infection spread is how policies are implemented *at the lowest level*, such as by cities and towns, or even individuals. Similar strategic encounters routinely play out in large-scale organizations, where actions throughout the management hierarchy are ultimately reflected in the decisions made at the lowest level (e.g., by the employees who are ultimately involved in production), and these lowest-level decisions play a decisive role in the organizational welfare.

We propose a novel model of hierarchical decision making which is a natural stylized representation of strategic interactions of this kind. Our model, which we term *structured hierarchical games (SHGs)*, represents each player by a node in a tree hierarchy. The tree plays two roles in SHGs. First, it captures the sequence of moves by the players: the root (the lone member of level 1 of the hierarchy) makes the first strategic choice, its children (i.e., all nodes in level 2) observe the root's choice and follow, their children then follow in turn, and so on, until we reach the leaf node players who move upon observing their predecessors' choices. Second, the tree partially captures strategic dependence: a player's utility depends on its own strategy, that of its parent, and the strategies of *all of the leaf nodes*. The sequence of moves in our model naturally captures the typical sequence of decisions in hierarchical policy-making settings, as well as in large organizations, while the utility structure captures the decisive role of leaf nodes (e.g., individual compliance with vaccination policies), as well as hierarchical dependence (e.g., employee dependence on a manager's approval of their performance, or state dependence on federal funding). Significantly, the *SHG* model generalizes a number of well-established models of strategic encounters, including (a) simultaneous-move games (captured by a 2-level SHG with the root having a single dummy action), (b) Stackelberg (leader-follower) games (a 2-level game with a single leaf node) [11, 37], and (c) single-leader multi-follower Stackel-

Submitted to the 38th Conference on Uncertainty in Artificial Intelligence (UAI 2022). **To be used for reviewing only**.

berg games (e.g., a Stackelberg security game with a single defender and many attackers) [5, 8].

Our second contribution is a novel gradient-based algorithm for approximately computing subgame-perfect equilibria of *SHGs*. Specifically, we propose *Differential Backward Induction (DBI)*, which is a backpropagation-style gradient ascent algorithm that leverages both the sequential structure of the game, as well as the utility structure of the players. As *DBI* involves simultaneous gradient updates of players in the same level (particularly at the leaves), convergence is not guaranteed in general (as is also the case for best-response dynamics [12]). Viewing *DBI* as a dynamical system, we provide a sufficient condition for its convergence to a stable point. Our results also imply that in the special case of two-player zero-sum Stackelberg games, *DBI* converges to a local Stackelberg equilibrium [11, 39].

Finally, we demonstrate the efficacy of DBI in finding approximate equilibrium solutions to several classes of SHGs. First, we use a highly stylized class of SHGs with polynomial utility functions to compare DBI with five baseline gradient-based approaches from prior literature. Second, we use DBI to solve a recently proposed game-theoretic model of 3-level hierarchical epidemic policy making. Third, we apply DBI to solve a hierarchical variant of a public goods game, which naturally captures the decentralization of decision making in public good investment decisions, such as investments in sustainable energy. Fourth, we evaluate DBI in the context of a hierarchical security investment game, where hierarchical decentralization (e.g., involving federal government, industry sectors, and particular organizations) can also play a crucial role. In all of these, we show that DBI significantly outperforms the state of the art approaches that can be applied to solve games with hierarchical structure.

Related Work SHGs generalize both simultaneous-move games and Stackelberg games with multiple followers [5, 23]. They are also related to *graphical games* [20] in capturing utility dependence structure, although SHGs also capture sequential structure of decisions. Several prior approaches use gradient-based methods for solving games with particular structure. A prominent example is generative adversarial networks (GANs), though these are zero-sum games [9, 14, 19, 29, 30, 31]. Ideas from learning GANs have been adopted in gradient-based approaches to solve multi-player general-sum games [4, 7, 17, 22, 25, 28, 29]. However, all of these approaches assume a simultaneous-move game. A closely-related thread to our work considers gradient-based methods for bi-level optimization [24, 35]. Several related efforts consider gradient-based learning in Stackelberg games, and also use the implicit function theorem to derive gradient updates [2, 11, 32, 38, 39]. We significantly generalize these ideas by considering an arbitrary hierarchical game structure.

Jia et al. [18] recently considered a stylized 3-level SHG for pandemic policy making, and proposed several non-gradient-based algorithms for this problem. We compare with their approach in Section 4.

2 STRUCTURED HIERARCHICAL GAMES

Notation We use bold lower-case letters to denote vectors. Let f be a function of the form $f(\boldsymbol{x}, \boldsymbol{y}) : \mathbb{R}^d \times \mathbb{R}^{d'} \to \mathbb{R}^{d''}$. We use $\nabla_{\boldsymbol{x}} f$ to denote the partial derivative of f with respect to $\boldsymbol{x}$. When there is functional dependency between $\boldsymbol{x}$ and $\boldsymbol{y}$, we use $D_{\boldsymbol{x}} f$ to denote the total derivative of $f(\boldsymbol{x}, \boldsymbol{y}(\boldsymbol{x}))$ with respect to $\boldsymbol{x}$. We use $\nabla^2_{\boldsymbol{x},\boldsymbol{x}} f$ and $\nabla^2_{\boldsymbol{x},\boldsymbol{y}} f$ to denote the second-order partial derivatives and $D^2_{\boldsymbol{x},\boldsymbol{x}} f$ to denote the second-order total derivative of f. For a mapping $f : \mathbb{R}^d \to \mathbb{R}^d$, we use $f^t(\boldsymbol{x})$ to denote t iterative applications of f on $\boldsymbol{x}$. For mappings $f_1 : \mathbb{R}^d \to \mathbb{R}^d$ and $f_2 : \mathbb{R}^d \to \mathbb{R}^d$, we define $(f_1 \circ f_2)(\boldsymbol{x}) \triangleq f_1(f_2(\boldsymbol{x}))$ and $(f_1 + f_2)(\boldsymbol{x}) \triangleq f_1(\boldsymbol{x}) + f_2(\boldsymbol{x})$. Moreover, for a given $\epsilon \in \mathbb{R}^{\geqslant 0}$ and $\boldsymbol{x} \in \mathbb{R}^d$, we define the ϵ-ball around $\boldsymbol{x}$ as $\mathbb{B}_\epsilon(\boldsymbol{x}) = \{\boldsymbol{x}' \in \mathbb{R}^d \mid \|\boldsymbol{x} - \boldsymbol{x}'\|_2 < \epsilon\}$. Finally, $\boldsymbol{I}$ denotes an identity matrix.

Formal Model A structured hierarchical game (SHG) $\mathcal{G}$ consists of the set $\mathscr{N}$ of n players. Each player i is associated with a set of actions $\mathcal{X}_i \subseteq \mathbb{R}^{d_i}$. The players are partitioned across L levels, where $\mathscr{N}_l$ is the set of n_l players occupying level l. Let l_i denote the level occupied by player i. This *hierarchical* structure of the game is illustrated in Figure 1 where players correspond to nodes and levels are marked by dashed boundaries. The hierarchy plays two crucial roles: 1) it determines the order of moves, and 2) it partly determines utility dependence among players. Specifically, the temporal pattern of actions is as follows: level 1 has a single player, the *root*, who chooses an action first, followed by all players in level 2 making simultaneous choices, followed in turn by players in level 3, and so on until the *leaves* in the final level L. Players of level l only observe the actions chosen by all players of levels $1, 2, ..., l-1$, but not their peers in the same level. So, for example, pandemic social

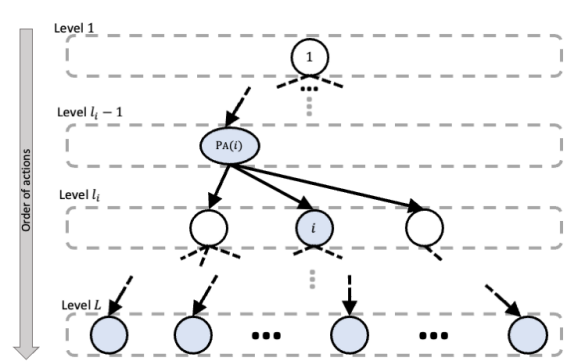


Figure 1: Schematic representation of an SHG. The utility of player i can have direct functional dependence *only* on the joint action of *all* shaded players.

distancing and vaccination policies in the US are initiated by the federal government (including the Centers for Disease Control and Prevention), with states subsequently instituting their own policies, counties reacting to these by determining their own, and behavior of people ultimately influenced, but not determined, by the guidelines and enforcement policies by the local county/city.

Next, we describe the utility structure of the game as entailed by the SHG hierarchy. Each player i in level $l_i > 1$ (i.e., any node other than the root) has a *unique parent* in level $l_i - 1$; we denote the parent of node i by $\text{PA}(i)$. A player's utility function is determined by 1) its own action, 2) the action of its parent, and 3) the actions of *all* players in level L (i.e., all *leaf* players). To formalize, let $\boldsymbol{x}_l$ denote the joint action profile of all players in level l. Player i's utility function then has the form $u_i(x_i, \boldsymbol{x}_L)$ if $l_i = 1$, $u_i(x_i, x_{\text{PA}(i)}, \boldsymbol{x}_L)$ if $1 < l_i < L$, and $u_i(x_i, x_{\text{PA}(i)}, \boldsymbol{x}_{L,-i})$ if $l_i = L$, where $\boldsymbol{x}_{L,-i}$ is the action profile of all players in level L other than i. For example, in our running pandemic policy example, the utility of a county depends on both the policy and enforcement strategy of its state (its *parent*) and on the ultimate pandemic spread and economic impact within it, both determined largely by the behavior of the county residents (*leaf nodes*). Note the considerable generality of the SHG model. For example, an arbitrary simultaneous-move game is a SHG with 2 levels and a "dummy" root node (utilities of all leaves depend on one another's actions), and an arbitrary Stackelberg game (e.g., Stackelberg security game), even with many followers, can be modeled as a 2-level SHG with the leader as root and followers as leaves. Furthermore, while we have defined SHGs with respect to real-vector player action sets, it is straightforward to represent mixed strategies of finite-action games in this way by simply using a softmax function to map an arbitrary real vector into a valid mixed strategy.

Solution Concept Since an SHG has important sequential structure, it is natural to consider the *subgame perfect equilibrium (SPE)* as the solution concept [33]. Here, we focus on pure-strategy equilibria. To begin, we note that in SHGs, the strategies of players in any level $l > 1$ are, in general, functions of the complete history of play in levels $1, \ldots, l-1$, which we denote by $h_{<l} = (\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_{l-1})$. Formally, a (pure) strategy of a player i is denoted by $s_i(h_{<l})$, which deterministically maps an arbitrary history $h_{<l}$ into an action $x_i \in \mathcal{X}_i$. A *Nash equilibrium* of an SHG is then a strategy profile $\boldsymbol{s} = (s_1, \ldots, s_i, \ldots, s_n)$ such that for all $i \in \mathcal{N}$, $u_i(s_i, \boldsymbol{s}_{-i}) \geqslant u_i(s'_i, \boldsymbol{s}_{-i})$ for all possible alternative strategies for i, s'_i. Here, we denote the realized payoff of i from profile $\boldsymbol{s}$ by $u_i(s_i, \boldsymbol{s}_{-i})$. Next, we define a *level-l-subgame* given $h_{<l}$ as an SHG that includes only players at levels $\geqslant l$, with actions chosen in levels $< l$ fixed to $h_{<l}$. A strategy profile $\boldsymbol{s}$ is a *subgame perfect equilibrium* of SHG if it is a Nash equilibrium of every level-l-subgame of SHG for every l and history $h_{<l}$. We prove in appendix A that our definition of SPE is equivalent to the standard SPE in an extensive-form representation of SHG.

While in principle we can compute an SPE of an SHG using backward induction, this cannot be done directly (i.e., by complete enumeration of actions of all players) as actions are real vectors. Moreover, even discretizing actions is of little help, as the hierarchical nature of the game leads to exponential explosion of the search space. We now present a gradient-based approach for approximating SPE along the equilibrium path in an SHG that leverages the game structure to derive backpropagation-style gradient updates.

3 DIFFERENTIAL BACKWARD INDUCTION

In this section, we describe our gradient-based algorithm, Differential Backward Induction (DBI), for approximating an SPE, and then analyze its convergence. Just as gradient ascent does not, in general, identify a globally optimal solution to a non-convex optimization problem, DBI in general yields a solution which only satisfies first-order conditions (see Section 3.2 for further details). Moreover, we leverage the structure of the utility functions to focus computation on an SPE in which strategies of players are only a function of their immediate parents.[1] In this spirit, we define *local* best response functions $\phi_i : \mathbb{R}^{d_{\text{PA}(i)}} \to \mathbb{R}^{d_i}$ mapping a player i's parent's action $x_{\text{PA}(i)}$ to i's action x_i; note that the notation ϕ_i is distinct from s_i above for i's strategy to emphasize the fact that ϕ_i is only locally optimal. Now, suppose that a player i is in the last level L. Local optimality of ϕ_i implies that if $x_i = \phi_i(x_{\text{PA}(i)})$, then $\nabla_{x_i} u_i\left(\boldsymbol{x}_i, \boldsymbol{x}_{\text{PA}(i)}, \boldsymbol{x}_{L,-i}\right) = 0$ and $\nabla^2_{x_i,x_i} u_i\left(\boldsymbol{x}_i, \boldsymbol{x}_{\text{PA}(i)}, \boldsymbol{x}_{L,-i}\right) \prec 0$.[2]

Let ϕ_l denote the local best response for all the players in level l given the actions of all players in level $l-1$. We can compose these local best response functions to define the function $\Phi_l := \phi_L \circ \phi_{L-1} \circ \ldots \circ \phi_{l+1} : \mathbb{R}^{d_{n_l}} \to \mathbb{R}^{d_{n_L}}$ i.e., the local best response of players in the last level L given the actions of the players in level l.[3] Then for any player (i) in level $l_i < L$, $D_{x_i} u_i\left(x_i, x_{\text{PA}(i)}, \Phi_l\left(\langle x_i, \boldsymbol{x}_{l,-i}\rangle\right)\right) = 0$ and $D^2_{x_i,x_i} u_i\left(x_i, x_{\text{PA}(i)}, \Phi_l\left(\langle x_i, \boldsymbol{x}_{l,-i}\rangle\right)\right) \prec 0$, where D_{x_i} is the total derivative with respect to x_i (as $\Phi_l(\langle x_i, \boldsymbol{x}_{l,-i}\rangle)$ is also a function of x_i). Note that the functions ϕ and Φ are *implicit*, capturing the functional dependencies between actions of players in different levels at the local equilibrium.

Throughout, we make the following standard assumption on the utility functions [10, 39].

[1] Note that while we cannot guarantee that an SPE exists in SHGs in general, let alone those possessing the assumed structure, we find experimentally that our approach often yields good SPE approximations.

[2] For simplicity, we omit degenerate cases where $\nabla^2_{\boldsymbol{x}_i,\boldsymbol{x}_i} u_i = 0$ and assume all local maxima are strict.

[3] Note that in particular $\Phi_L = \phi_L$.

Assumption 1. *For any $x_i \in \mathcal{X}_i$, the second-order partial derivatives of the form $\nabla^2_{x_i,x_i} u_i$ are non-singular.*

3.1 ALGORITHM

The DBI algorithm works in a bottom-up manner, akin to back-propagation: for each level l, we compute the total derivatives (gradients) of the utility functions and local best response maps (ϕ, Φ) based on analytical expressions that we derive below. We then propagate this information up to level $l-1$, as it is used to compute gradients for that level, and so on until level 1. Algorithm 1 gives the full DBI algorithm. In this algorithm, $\textsc{Chd}(i)$ denotes the set of children of player i (i.e., nodes in level $l_i + 1$ for whom i is the parent). DBI works in a backward message-passing manner, comparable to back-propagation: after each player has computed its total derivative, it passes (back-propagates) $D_{x_i}\Phi_l$ to its direct parent; this information is, in turn, used by the parent to compute its own total derivative, which is passed to its own parent, and so on.

Algorithm 1 Differential Backward Induction (DBI)

Input: An SHG instance $\mathcal{G}$
Parameters: Learning rate α, maximum number of iterations T for gradient update
Output: A strategy profile
Randomly initialize $\boldsymbol{x}^0 = \langle \boldsymbol{x}_1^0, \ldots, \boldsymbol{x}_L^0 \rangle$
for $t = 1, 2, \ldots, T$ **do**
  for $l = L, L-1, \ldots, 1$ **do**
    for $i = 1, 2, \ldots, n_l$ **do**
      if $l = L$ **then**
        Back-propagate $D_{x_i}\Phi_i = \boldsymbol{I}$ to $\textsc{Pa}(i)$
        Set $x_i^t \leftarrow x_i^{t-1} + \alpha \nabla_{x_i} u_i$
      else
        Compute $\nabla_{x_i} u_i, \nabla_{\boldsymbol{x}_L} u_i$ at $\boldsymbol{x}^{t-1}$
        Compute $D_{x_i}\phi_j, \forall j \in \textsc{Chd}(i)$ (Eqn. (5))
        Compute $D_{x_i}\Phi_l$ (Eqn. (4))
        Back-propagate $D_{x_i}\Phi_l$ to $\textsc{Pa}(i)$
        Compute $D_{x_i} u_i = \nabla_{x_i} u_i + \nabla_{\boldsymbol{x}_L} u_i D_{x_i}\Phi_l$
        Set $x_i^t \leftarrow x_i^{t-1} + \alpha D_{x_i} u_i$
Return $\boldsymbol{x}^T$

Algorithm 1 takes the total derivates as given. We now derive closed-form expressions for these. We start from the last level L. Given the actions of players in level $L-1$, the total derivative of a player $i \in \mathcal{N}_L$ with respect to x_i is

$$D_{x_i} u_i \left(x_i, x_{\textsc{Pa}(i)}, \boldsymbol{x}_{L,-i}\right) = \nabla_{x_i} u_i. \tag{1}$$

For a player i in level $L-1$, the total derivative (at a local best response) is

$$\begin{aligned} D_{x_i} u_i(x_i, x_{\textsc{Pa}(i)}, &\phi_L(\langle x_i, \boldsymbol{x}_{L-1,-i}\rangle)) \\ &= \nabla_{x_i} u_i + \left(\nabla_{\boldsymbol{x}_L} u_i\right)\left(D_{x_i}\phi_L\right), \end{aligned} \tag{2}$$

where $\nabla_{\boldsymbol{x}_L} u_i$ is a $1 \times d_{n_L}$ vector and $D_{x_i}\phi_L$ is a $d_{n_L} \times d$ matrix. The technical challenge here is to derive the term $D_{x_i}\phi_L$ for $i \in \mathcal{N}_{L-1}$. Recall that ϕ_L is the vectorized concatenation of the ϕ_j functions for $j \in \mathcal{N}_L$. Since the local best response strategy of a player in level L only depends on its parent in level $L-1$, the only terms in ϕ_L that depend on x_i are the actions of $\textsc{Chd}(i)$ in level L. Consequently, it suffices to derive $D_{x_i}\phi_j$ for $j \in \textsc{Chd}(i)$. Note that for these players j, $\nabla_{x_j} u_j = 0$ (by local optimality of ϕ_L). We will use this first-order condition to derive the expression for the total derivative using the *implicit function theorem*.

Theorem 1 (Implicit Function Theorem (IFT) [10, Theorem 1B.1])**.** . *Let $f(\boldsymbol{x}_1, \boldsymbol{x}_2) : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}^d$ be a continuously differentiable function in a neighborhood of $(\boldsymbol{x}_1^*, \boldsymbol{x}_2^*)$ such that $f(\boldsymbol{x}_1^*, \boldsymbol{x}_2^*) = 0$. Also suppose $\nabla_{\boldsymbol{x}_2} f$, the Jacobian of f with respect to $\boldsymbol{x}_2$, is non-singular at $(\boldsymbol{x}_1^*, \boldsymbol{x}_2^*)$. Then around a neighborhood of $\boldsymbol{x}_1^*$, we have a local diffeomorphism $\boldsymbol{x}_2^*(\boldsymbol{x}_1) : \mathbb{R}^d \to \mathbb{R}^d$ such that $D_{\boldsymbol{x}_1}\boldsymbol{x}_2 = -\left(\nabla_{\boldsymbol{x}_2} f\right)^{-1} \nabla_{\boldsymbol{x}_1} f$.*

To use Theorem 1, we set $f = \nabla_{x_j} u_j$ (which satisfies the conditions of Theorem 1 by Assumption 1), $\boldsymbol{x}_1 = x_i$ and $x_2 = \boldsymbol{x}_j$ (recall that $j \in \textsc{Chd}(i)$). By IFT, there exists $\phi_j(x_i)$ such that $D_{x_i}\phi_j = -(\nabla^2_{x_j,x_j} u_j)^{-1}\nabla^2_{x_j,x_i} u_j$. Define $\nabla^2_j := \nabla^2_{x_j,x_i} u_j$. Then

$$\begin{aligned} \left(\nabla_{\boldsymbol{x}_L} u_i\right)\left(D_{x_i}\phi_L\right) &= -\sum_{j \in \textsc{Chd}(i)} \left(\nabla_{x_j} u_i\right) D_{x_i}\phi_j \\ &= -\sum_{j \in \textsc{Chd}(i)} \left(\nabla_{x_j} u_i\right)\left(\nabla^2_{x_j,x_j} u_j\right)^{-1}\nabla^2_j. \end{aligned}$$

Plugging this into Equation (2), we obtain

$$\begin{aligned} &D_{x_i} u_i \left(x_i, x_{\textsc{Pa}(i)}, \phi_L\left(\boldsymbol{x}_{L-1}\right)\right) \\ &\quad = \nabla_{x_i} u_i - \sum_{j \in \textsc{Chd}(i)} \left(\nabla_{x_j} u_i\right)\left(\nabla^2_{x_j,x_j} u_j\right)^{-1}\nabla^2_j. \end{aligned} \tag{3}$$

For a level $l < L-1$, the total derivative of player $i \in \mathcal{N}_l$ in a local best response is $D_{x_i} u_i = \nabla_{x_i} u_i + \left(\nabla_{\boldsymbol{x}_L} u_i\right)\left(D_{x_i}\Phi_l\right)$, where

$$\begin{aligned} D_{x_i}\Phi_l &= \left(D_{\boldsymbol{x}_{l+1}}\Phi_{l+1}\right)\left(D_{x_i}\boldsymbol{x}_{l+1}\right) \\ &= \sum_{j \in \textsc{Chd}(i)} \left(D_{x_j}\Phi_{l+1}\right)\left(D_{x_i}\phi_j\right). \end{aligned} \tag{4}$$

Applying IFT, we get

$$D_{x_i}\phi_j = -(\nabla^2_{x_j,x_j} u_j)^{-1}\nabla^2_{x_j,x_i} u_j, \tag{5}$$

for $j \in \textsc{Chd}(i)$. We can apply the above procedure recursively for $D_{\boldsymbol{x}_{l+1}}\Phi_{l+1}$ to derive the total derivative for players $i \in \mathcal{N}_l$ for $l < L-1$:

$$\begin{aligned} D_{x_i} u_i = \nabla_{x_i} u_i + \Bigg(&\sum_{j \in \textsc{Leaf}(i)} (-1)^{L-l} \nabla_{x_j} u_i \\ &\prod_{\eta \in \textsc{Path}(j \to i)} \left(\nabla^2_{x_\eta,x_\eta} u_\eta\right)^{-1} \nabla^2_{x_\eta,\boldsymbol{x}_{\textsc{Pa}(\eta)}} u_\eta \Bigg), \end{aligned} \tag{6}$$

where $\text{PATH}(j \to i)$ is an ordered list of nodes (players) lying on the unique path from j to i, excluding i. Note that Equation (6) is a generalization of Equation (3) where the PATH only consists of the leaf player.

While the above derivation assumes the ϕ and Φ functions are local best responses, in our algorithm in each iteration we evaluate these functional expressions for the total derivatives *at the current joint action profile*. This significantly reduces computational complexity and ensures that Algorithm 1 satisfies the first-order conditions upon convergence.

3.2 CONVERGENCE ANALYSIS

Our next task is to provide sufficient conditions for the DBI algorithm to converge to a stable point. A few notes are in order. First, as we remarked earlier, stable points of DBI are not guaranteed to be SPE just as stable points of gradient ascent are not guaranteed to be globally optimal with general non-convex objective functions. Furthermore, DBI algorithm entails what are effectively iterative better-response updates by players, and it is well-known that best response dynamic processes in games will in general lead to cycles [12].

To begin, we observe that the gradient updates in DBI can be interpreted as a discrete dynamical system, $\boldsymbol{x}^{t+1} = F(\boldsymbol{x}^t)$, with $F(\boldsymbol{x}^t) = (\boldsymbol{I} + \alpha G)(\boldsymbol{x}^t)$ where G is an update gradient vector. This discrete system can be viewed as an approximation of a continuous limit dynamical system $\dot{\boldsymbol{x}} = G(\boldsymbol{x})$ (i.e., letting $\alpha \to 0$). A standard solution concept for such dynamical systems is a *locally asymptotic stable point (LASP)*.

Definition 1 ([13])**.** *A continuous (or discrete) dynamical system $\dot{\boldsymbol{x}} = G(\boldsymbol{x})$ (or $\boldsymbol{x}^{t+1} = F(\boldsymbol{x}^t)$) has a locally asymptotic stable point (LASP) $\boldsymbol{x}^*$ if $\exists \epsilon > 0, \lim_{t\to\infty} \boldsymbol{x}^t = \boldsymbol{x}^*, \forall \boldsymbol{x}^0 \in \mathbb{B}_\epsilon(\boldsymbol{x}^*)$.*

There are well-known necessary and sufficient conditions for the existence of an LASP.

Proposition 1 (Characterization of LASP [40, Theorem 1.2.5, Theorem 3.2.1])**.** *A point $\boldsymbol{x}^*$ is an LASP for the continuous dynamical system $\dot{\boldsymbol{x}} = G(\boldsymbol{x})$ if $G(\boldsymbol{x}^*) = 0$ and all eigenvalues of Jacobian matrix $\nabla_{\boldsymbol{x}} G$ at $\boldsymbol{x}^*$ have negative real parts. Furthermore, for any $\boldsymbol{x}^*$ such that $G(\boldsymbol{x}^*) = 0$, if $\nabla_{\boldsymbol{x}} G$ has eigenvalues with positive real parts at $\boldsymbol{x}^*$, then $\boldsymbol{x}^*$ cannot be an LASP.*

Note that an LASP of DBI is an action profile of all players that satisfies the first-order conditions, i.e., it has the property that no player can improve their utility through a local gradient update. While the existence of an LASP depends on game structure, we show that under Assumption 1, and as long as the sufficient conditions for LASP existence in Proposition 1 are satisfied, DBI converges to LASP. We defer all the omitted proofs to Appendix C.

Proposition 2. *Let $\lambda_1, \ldots, \lambda_d$ denote the eigenvalues of the updating Jacobian $\nabla_{\boldsymbol{x}} G$ at an LASP $\boldsymbol{x}^*$ and define $\lambda^* = \arg\max_{i\in[d]} Re(\lambda_i)/|\lambda_i|^2$, where Re is the real part operator. Then with a learning rate $\alpha < -2Re(\lambda^*)/|\lambda^*|^2$, and an initial point $\boldsymbol{x}^0 \in \mathbb{B}_\epsilon(\boldsymbol{x}^*)$ for some $\epsilon > 0$ around $\boldsymbol{x}^*$, DBI converges to an LASP. Specifically, if the choice of learning rate equals α^* and the modulus of matrix $\rho(\boldsymbol{I} + \alpha^* \nabla_{\boldsymbol{x}} G) = 1 - \kappa < 1$, then the dynamics converge to $\boldsymbol{x}^*$ with the rate of $O((1-\kappa/2)^t)$.*

Proposition 2 states that there exists a region such that, if the initial point is in that region, then DBI will converge to an LASP. We next show that if we assume first-order Lipschitzness for the update rule, then we can also characterize the region of initial points which converge to an LASP.

Proposition 3. *Suppose G is L-Lipschitz.*[4] *Then for all $\boldsymbol{x}^0 \in \mathbb{B}_{\kappa/2L}(\boldsymbol{x}^*)$, $\epsilon > 0$ and after T rounds of gradient update, DBI will output a point $\boldsymbol{x}^T \in \mathbb{B}_\epsilon(\boldsymbol{x}^*)$ as long as $T \geqslant \lceil \frac{2}{\kappa} \log \left\|\boldsymbol{x}^0 - \boldsymbol{x}^*\right\| / \epsilon \rceil$ where κ is as defined in Proposition 2.*

We further show that through random initialization, the probability of reaching a *saddle point* is 0, which means that with probability 1, DBI converges to an LASP in which players are playing *local* best responses.

Proposition 4. *Suppose G is L-Lipschitz. Let $\alpha < 1/L$ and define the saddle points of the dynamics G as $\mathcal{X}^*_{sad} = \{\boldsymbol{x}^* \in \mathcal{X} \mid \boldsymbol{x}^* = (\boldsymbol{I} + \alpha G)(\boldsymbol{x}^*), \rho((\boldsymbol{I} + \alpha \nabla_{\boldsymbol{x}} G)(\boldsymbol{x}^*)) > 1\}$. Also let $\mathcal{X}^0_{sad} = \{\boldsymbol{x}^0 \in \mathcal{X} \mid \lim_{t\to\infty} (\boldsymbol{I} + \alpha G)^t(\boldsymbol{x}^0) \in \mathcal{X}^*_{sad}\}$ denote the set of initial points that converge to a saddle point. Then $\mu(\mathcal{X}^0_{sad}) = 0$, where μ is Lebesgue measure.*

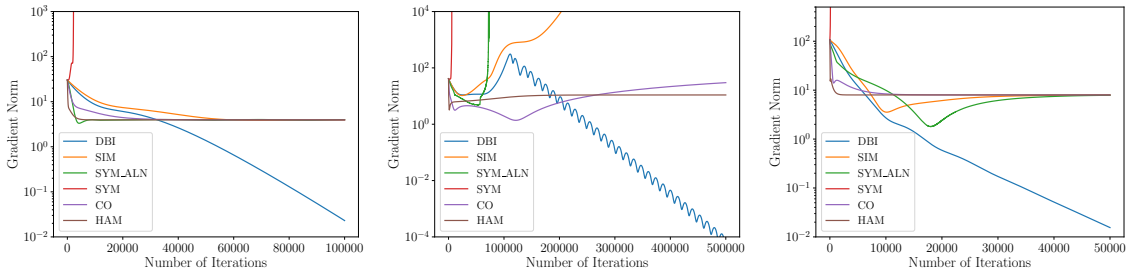
While our convergence analysis does not guarantee convergence to an approximate SPE, our experiments show that DBI is in fact quite effective in doing so in practice.

4 EXPERIMENTS

In this section, we empirically investigate the following questions: (1) the convergence rate of DBI, (2) the solution quality of DBI, (3) the behavior of DBI in games where we can verify global stability. All our code is written in python. We ran our experiments on an Intel(R) Core(TM) i7-7700HQ CPU @ 2.80GHz to obtain the results in Sections 4.1 and C.3, and on an Intel(R) Core(TM) i9-9820X CPU @ 3.30GHz for the rest of the experiments.

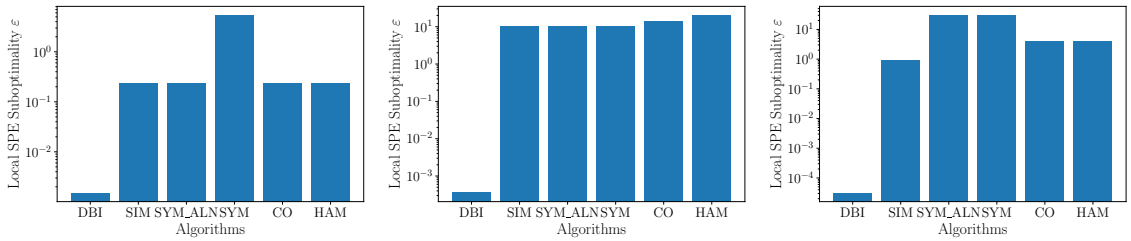
We evaluate the performance in terms of quality of equilibrium approximation as a function of the number of iterations of a given algorithm, or its running time. Ideally, given a collection of actions $\boldsymbol{x}$ played by players along the (approximate) equilibrium path computed, we wish to find the

[4] Formally, this means that $\exists L > 0$ such that $\forall \boldsymbol{x}, \boldsymbol{x}' \in \mathcal{X}, \|G(\boldsymbol{x}) - G(\boldsymbol{x}')\|_2 \leqslant L\|\boldsymbol{x} - \boldsymbol{x}'\|_2$.



(a) (b) (c)

Figure 2: Convergence behaviors on (a) a $(1, 1, 1)$ game with 1-d actions (b) a $(1, 1, 2)$ game with 1-d actions (c) a $(1, 1, 1)$ game with 3-d actions.



(a) (b) (c)

Figure 3: Solution qualities on (a) a $(1, 1, 1)$ game with 1-d actions (b) a $(1, 1, 2)$ game with 1-d actions (c) a $(1, 1, 1)$ game with 3-d actions.

largest utility gain any player can have by deviating from this path, which we denote by $\epsilon(\boldsymbol{x})$. However, this computation is impossible in our setting, as it would need to consider all possible histories as well, whereas our approach and alternatives only return $\boldsymbol{x}$ along the path of play (moreover, considering all possible histories is itself intractable).

Therefore, we consider two heuristic alternatives. The first, which we call *local SPE regret*, runs DBI for every player i starting with $\boldsymbol{x}$, and returns the greatest benefit that any player can thereby obtain; we use this in Section 4.1. In the rest of this section, we use the second alternative, which we call *global SPE regret*. It considers for each player i in level l a discrete grid of alternative actions, and uses best response dynamics to compute an approximate SPE of the level-$(l + 1)$ subgame to evaluate player i's utility for each such deviation. This approach then returns the highest regret among all players computed in this way.

Our evaluation considers three SHG scenarios. We begin by comparing DBI to a number of baselines on simple, stylized SHG models, then move on to three complex hierarchical game models motivated by concrete applications.

4.1 POLYNOMIAL GAMES

We begin by considering instances of SHGs to which we can readily apply several state-of-the-art baselines, allowing us a direct comparison to previous work. Specifically, we consider 3 SHG instances with different game properties: (a) a three-level chain structure (or the $(1, 1, 1)$ game) with 1-d actions (b) a "ㅅ" shape tree (or the $(1, 1, 2)$ game) with 1-d action spaces, and (c) and $(1, 1, 1)$ game with 3-d actions. In all the games, the payoffs are polynomial functions of $\boldsymbol{x}$ with randomly generated coefficients (we can think of these as proxies for a Taylor series approximation of actual utility functions). Details are in Appendix C.

We compare DBI with the following five baselines: 1) simultaneous partial gradient ascent (SIM) [7, 28], 2) symplectic gradient dynamics with or 3) without alignment (SYM_ALN and SYM, respectively) [4], 4) consensus optimization (CO) [30], and 5) Hamilton gradient (HAM) [1, 26]. SIM, SYM_ALN, SYM, CO and HAM are all designed to compute a local Nash equilibrium [4, 7].

We start by comparing convergence behavior of DBI to the baselines. We run all algorithm with the same initial point and learning rate. The results are in Figure 2 where we plot the L_2 norm of total gradient for each of the algorithms (Y axis) against the number of iterations (X axis).

In all cases, DBI converges to a critical point that meets the first-order conditions while the baseline algorithms fail to do so in most cases. In Figures 2(a) and (c), all baselines have converged to a point with finite norm for the total gradients. In (b), however, only CO and HAM converge to a stationary point while SIM, SYM, SYM_ALN all diverge. For scenario (b), DBI appears to be on an inward spiral to a critical point. We further check the second-order condition (see Appendix C) and verify that DBI has actually converged to local maxima of individual payoffs in all three games.

Next, we investigate solution quality in terms of *local regret* of DBI compared to baselines. As shown in Figure 3, across all three game instances, DBI outputs a profile of actions (along the path of play) with near-zero local regret while other algorithm fail to do so.

4.2 DECENTRALIZED EPIDEMIC POLICY GAME

Next, we consider DBI for solving a class of games inspired by hierarchical decentralized policy-making in the context of epidemics such as COVID-19 [18]. The hierarchy has levels corresponding to the (single) federal government, multiple states, and county administrations under each state. Each player's action (policy) is a scalar in $[0, 1]$ that represents, for example, the extent of social distancing recommended or mandated by a player (e.g., a state) for its administrative subordinates (e.g., counties). Crucially, these subordinates have considerable autonomy about setting their own policies, but incur a non-compliance cost for significantly deviating from recommendations made by the level immediately above (of course, non-compliance costs are not relevant for the root player). The full cost function of each player additionally includes infection prevalence within the geographic territory of interest to the associated entity (e.g., within the state), as well as the socio-economic cost of the policy itself (complete details are provided in Appendix C).

Since the actions are in a one-dimensional compact space and the depth of the hierarchy is at most 3, our baseline is the best response dynamics (BRD) algorithm proposed by Jia et al. [18] (detailed in Appendix C), and we use *global regret* as a measure of efficacy in comparing the proposed DBI algorithm with BRD. The results of this comparison are shown in Figure 4 for two-level (government and states) and three-level (government, states, counties) variants of this game. We consider two-level games with 20 and 50 leaves (states), and three-level games with 2 players in level 2 (states) and 4 and 10 leaves (counties).

As we can see from the top plots in Figure 4, BRD can have poor convergence behavior in terms of global regret, whereas DBI appears to converge quite reliably to a path of play with a considerably lower global regret. Notably, the improvement in solution quality becomes more substantial

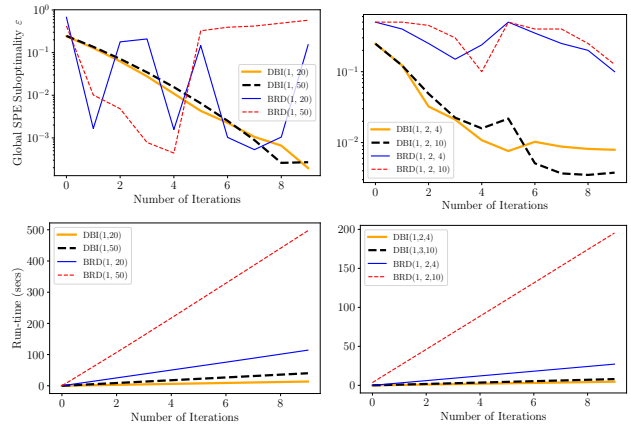


Figure 4: Global regret (top) and running time (bottom) for the decentralized epidemic policy game. Left (resp., right) column corresponds to result for games with 2(resp., 3) levels.

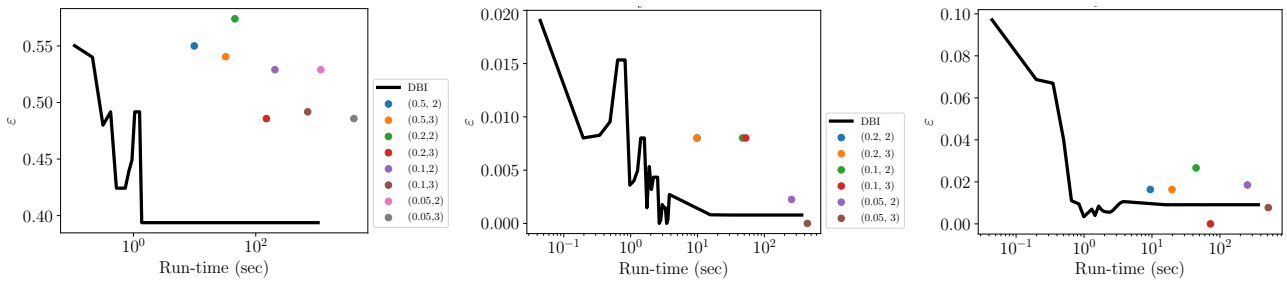
as we increase the game complexity either in terms of scale (number of leaves) or in terms of the level of hierarchy (moving from 2- to 3-level games).

Running time (in seconds) demonstrates the relative efficacy of DBI even further (bottom plots in Figure 4). In particular, observe the significant increase in the running time of BRD as we increase the number of leaves. In contrast, DBI is far more scalable: indeed, even more than doubling the number of players appears to have little impact on its running time. Moreover, BRD is several orders of magnitude slower than DBI for the more complex games.

4.3 HIERARCHICAL PUBLIC GOODS GAMES

Next, we consider *hierarchical public goods games*. A conventional networked public goods game endows each player i with a utility function $u_i(x_i, x_{-i}) = a_i + b_i x_i + \sum_j g_{ji} x_i x_j - c_i(x_i)$, where g_{ji} is the impact of player j on player i (often represented as a weighted edge on a network), and $x_i \in [0, 1]$ the level of investment in the public good by player i [6]. We construct a 3-level hierarchical variant of such games by starting with the karate club network [43] which represents friendships among 34 individuals. Level-2 nodes are obtained by partitioning the network into two (sub)clubs, with leaves (level-3 nodes) representing all the individuals. The utility of level-2 nodes is the sum of utilities of individual members of associated clubs, with the utility of the root being the sum of the utilities of all individuals. Furthermore, we introduce non-compliance costs with investment policies in the level immediately above, as we did in the decentralized epidemic policy game (Section 4.2). Further details are provided in Appendix C.6.

Figure 5(a) presents the global regret as a function of running time for DBI (black line) and BRD with different levels



(a) Networked Public Goods Game (b) Security game $\kappa = 0.1$. (c) Security game $\kappa = 0.5$.

Figure 5: (a) Performance (ϵ) in the Public Goods Game (Section 4.3); the scatter points show the results of BRD with discretized factors $0.5, 0.2, 0.1, 0.05$, and best response rounds $2, 3$. (b), (c) Results on $(1, 3, 6)$ hierarchical security games (Section 4.4); legend is shared.

of discretization (dots). We observe that DBI yields considerably lower regret in these games than BRD even as we discretize the latter finely. Moreover, DBI reaches smaller regret orders of magnitude faster than BRD.

4.4 HIERARCHICAL SECURITY GAMES

In the final set of experiments, we evaluate DBI on a hierarchical extension of *interdependent security games* [3]. In these games, n defenders can each invest $x_i \geqslant 0$ in security. If defender i is attacked, the probability that the attack succeeds is $1/(1 + x_i)$. Furthermore, defenders are interdependent, so that a successful attack on defender i cascades to defender j with probability q_{ji}. In the variant we adopt, the attacker strategy is a uniform distribution over defenders (e.g., the "attacker" is just nature, with attacks representing stochastic exogenous failures). The utility of the defender is the probability of surviving the attack less the cost of security investment.

We extend this simultaneous-move game to a hierarchical structure consisting of one root player (e.g., government), three level-2 players (e.g., sectors), and six leaf players (e.g., organizations). The policy-makers in the first two levels of the game recommend an investment policy to the level below, and aim to maximize total welfare (sum of utilities) among the leaf players in their subtrees. Just as in both hierarchical epidemic and public goods games, whenever a player in level l does not act according to the recommendation of their parent in level $l-1$, they incur a non-compliance cost. Complete model details are deferred to Appendix C.7. We conduct experiments with two weights κ that determine the relative importance of non-compliance costs in the decisions of non-root players in the game: $\kappa \in \{0.1, 0.5\}$.

Figures 5(b) and 5(c) present the results of comparing DBI with BRD on this class of games, where BRD is again evaluated with different levels of action space discretization (note, moreover, that in this setting discretizing actions is not enough, since these are unbounded, and we also had to impose an upper bound). We can observe that for either value of κ, DBI yields high-quality SPE approximation (in terms of global SPE regret) far more quickly than BRD. In particular, when we use relatively coarse discretization, BRD is approximately an order of magnitude slower, and yields significantly higher regret. In contrast, if we use finer discretization for BRD, global regret for BRD and DBI becomes comparable, but now BRD is several orders of magnitude slower. For example, DBI converges within several seconds, whereas if we discretize x_i into multiples of 0.02, BRD takes nearly 2 hours, while discretization at the level of 0.01 results in BRD taking nearly 7 hours.

5 CONCLUSION

We introduced a novel class of hierarchical games, proposed a new game-theoretic solution concept and designed an algorithm to compute it. We assume a specific form of utility dependency between players and our solution concept only guarantees local stability. Improvement on each of these two fronts is an interesting direction for future work.

Given the generality of our framework, our approach can be used for many applications characterized by a hierarchy of strategic agents e.g., pandemic policy making. However, our modeling requires the full knowledge of the true utility functions of all players and our analysis assumes full rationality for all the players. Although the model we have addressed here is already challenging, these assumptions are unlikely to hold in many real-world applications. Therefore, further analysis is necessary to fully gauge the robustness of our approach before deployment.

References

[1] Jacob Abernethy, Kevin Lai, and Andre Wibisono. Last-iterate convergence rates for min-max optimization: convergence of Hamiltonian gradient descent and consensus optimization. In *Thirty-Second Interna-*

tional Conference on Algorithmic Learning Theory, pages 3–47, 2021.

[2] Kareem Amin, Satinder Singh, and Michael Wellman. Gradient methods for Stackelberg security games. In *Thirty-Second Conference on Uncertainty in Artificial Intelligence*, pages 2–11, 2016.

[3] Yoram Bachrach, Moez Draief, and Sanjeev Goyal. Contagion and observability in security domains. In *2013 51st Annual Allerton Conference on Communication, Control, and Computing (Allerton)*, pages 1364–1371. IEEE, 2013.

[4] David Balduzzi, Sebastien Racaniere, James Martens, Jakob Foerster, Karl Tuyls, and Thore Graepel. The mechanics of n-player differentiable games. In *Thirty-Fifth International Conference on Machine Learning*, pages 354–363, 2018.

[5] Nicola Basilico, Stefano Coniglio, and Nicola Gatti. Methods for finding leader-follower equilibria with multiple followers. In *Fifteenth International Conference on Autonomous Agents and Multiagent Systems*, pages 1363–1364, 2016.

[6] Yann Bramoullé and Rachel Kranton. Public goods in networks. *Journal of Economic theory*, 135(1):478–494, 2007.

[7] Benjamin Chasnov, Lillian Ratliff, Eric Mazumdar, and Samuel Burden. Convergence analysis of gradient-based learning in continuous games. In *Thirty-Sixth Conference on Uncertainty in Artificial Intelligence*, pages 935–944, 2020.

[8] Stefano Coniglio, Nicola Gatti, and Alberto Marchesi. Computing a pessimistic stackelberg equilibrium with multiple followers: The mixed-pure case. *Algorithmica*, 82(5):1189–1238, 2020.

[9] Constantinos Daskalakis and Ioannis Panageas. The limit points of (optimistic) gradient descent in min-max optimization. In *Thirty-Second International Conference on Neural Information Processing Systems*, pages 9256–9266, 2018.

[10] Asen Dontchev and Tyrrell Rockafellar. *Implicit functions and solution mappings*. Springer, 2009.

[11] Tanner Fiez, Benjamin Chasnov, and Lillian Ratliff. Implicit learning dynamics in Stackelberg games: Equilibria characterization, convergence analysis, and empirical study. In *Thirty-Seventh International Conference on Machine Learning*, pages 3133–3144, 2020.

[12] Drew Fudenberg, Fudenberg Drew, David K Levine, and David K Levine. *The theory of learning in games*, volume 2. MIT press, 1998.

[13] Oded Galor. *Discrete dynamical systems*. Springer Science & Business Media, 2007.

[14] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial networks. In *Twenty-Seventh Conference on Neural Information Processing Systems*, pages 2672–2680, 2014.

[15] Michael Hill and Frederic Varone. *The public policy process*. Routledge, 2021.

[16] Roger Horn and Charles Johnson. *Matrix analysis*. Cambridge University Press, 2012.

[17] Adam Ibrahim, Waıss Azizian, Gauthier Gidel, and Ioannis Mitliagkas. Linear lower bounds and conditioning of differentiable games. In *Thirty-Seventh International Conference on Machine Learning*, pages 4583–4593, 2020.

[18] Feiran Jia, Aditya Mate, Zun Li, Shahin Jabbari, Mithun Chakraborty, Milind Tambe, Michael Wellman, and Yevgeniy Vorobeychik. A game-theoretic approach for hierarchical policy-making. *CoRR*, abs/2102.10646, 2021.

[19] Chi Jin, Praneeth Netrapalli, and Michael Jordan. What is local optimality in nonconvex-nonconcave minimax optimization? In *Thirty-Seventh International Conference on Machine Learning*, pages 4880–4889, 2020.

[20] Michael Kearns, Michael Littman, and Satinder Singh. Graphical models for game theory. In *Seventeenth Conference on Uncertainty in Artificial Intelligence*, pages 253–260, 2001.

[21] John Kelley. *General topology*. Springer, 1955.

[22] Alistair Letcher. On the impossibility of global convergence in multi-loss optimization. In *Ninth International Conference on Learning Representations*, 2021.

[23] Sven Leyffer and Todd Munson. Solving multi-leader-follower games. *Preprint ANL/MCS-P1243-0405*, 2005.

[24] Shu Li and Tamer Başar. Distributed algorithms for the computation of noncooperative equilibria. *Automatica*, 23(4):523–533, 1987.

[25] Tianyi Lin, Zhengyuan Zhou, Panayotis Mertikopoulos, and Michael Jordan. Finite-time last-iterate convergence for multi-agent learning in games. In *Thirty-Seventh International Conference on Machine Learning*, pages 6161–6171, 2020.

[26] Nicolas Loizou, Hugo Berard, Alexia Jolicoeur-Martineau, Pascal Vincent, Simon Lacoste-Julien, and Ioannis Mitliagkas. Stochastic Hamiltonian gradient methods for smooth games. In *Thirty-Seventh International Conference on Machine Learning*, pages 6370–6381, 2020.

[27] Andreu Mas-Colell, Michael Dennis Whinston, Jerry R Green, et al. *Microeconomic theory*, volume 1. Oxford university press New York, 1995.

[28] Eric Mazumdar, Lillian Ratliff, and Shankar Sastry. On gradient-based learning in continuous games. *SIAM Journal on Mathematics of Data Science*, 2(1):103–131, 2020.

[29] Panayotis Mertikopoulos and Zhengyuan Zhou. Learning in games with continuous action sets and unknown payoff functions. *Mathematical Programming*, 173(1): 465–507, 2019.

[30] Lars Mescheder, Sebastian Nowozin, and Andreas Geiger. The numerics of GANs. In *Thirty-First International Conference on Neural Information Processing Systems*, pages 1823–1833, 2017.

[31] Vaishnavh Nagarajan and Zico Kolter. Gradient descent GAN optimization is locally stable. In *Thirty-First International Conference on Neural Information Processing Systems*, pages 5591–5600, 2017.

[32] Thanh Nguyen, Arunesh Sinha, and He He. Partial adversarial behavior deception in security games. In *Twenty-Nineth International Joint Conference on Artificial Intelligence*, pages 283–289, 2020.

[33] Martin J Osborne and Ariel Rubinstein. *An introduction to game theory*. Oxford university press New York, 2004.

[34] R Selten. Reexamination of the perfectness concept for equilibrium points in extensive games. *International Journal of Game Theory*, 4(1):25–55, 1975.

[35] Amirreza Shaban, Ching-An Cheng, Nathan Hatch, and Byron Boots. Truncated back-propagation for bilevel optimization. In *Twenty-Second International Conference on Artificial Intelligence and Statistics*, pages 1723–1732, 2019.

[36] Michael Shub. *Global stability of dynamical systems*. Springer, 1987.

[37] Heinrich Von Stackelberg. *The theory of the market economy*. Oxford University Press, 1952.

[38] Kai Wang, Lily Xu, Andrew Perrault, Michael Reiter, and Milind Tambe. Coordinating followers to reach better equilibria: End-to-end gradient descent for stackelberg games. *CoRR*, abs/2106.03278, 2021.

[39] Yuanhao Wang, Guodong Zhang, and Jimmy Ba. On solving minimax optimization locally: A follow-the-ridge approach. In *Eighth International Conference on Learning Representations*, 2020.

[40] Stephen Wiggins. *Introduction to applied nonlinear dynamical systems and chaos*. Springer Science & Business Media, 2003.

[41] Bryan Wilder, Marie Charpignon, Jackson Killian, Han-Ching Ou, Aditya Mate, Shahin Jabbari, Andrew Perrault, Angel Desai, Milind Tambe, and Maimuna Majumder. Modeling between-population variation in COVID-19 dynamics in Hubei, Lombardy, and New York City. *Proceedings of the National Academy of Sciences*, 117(41):25904–25910, 2020.

[42] Stephen Wolfram. *The Mathematica*. Cambridge University Press, 1999.

[43] Wayne W Zachary. An information flow model for conflict and fission in small groups. *Journal of anthropological research*, 33(4):452–473, 1977.

A OMITTED DETAILS FROM SECTION 2

Proposition 5. *The SPE notion we defined for an SHG model corresponds exactly to the SPE of its extensive-form game (EFG) representation.*

Proof. First we show how to construct the EFG representation of an SHG. Since within level l players act simultaneously, we can designate a canonical player order for player in level l, say we label them as $(l,1),(l,2),\ldots,(l,n_l)$. Then in the EFG representation, we sequentialize the playing order as always be $(1,1),(2,1),(2,2),\ldots,(3,1),\ldots,(L,1),\ldots,(L,n_L)$. For every $(l,1), l=1,\ldots,L$, its information sets are all in the form of some singleton $\{h_{<l}\}$ for some history $h_{<l}$. While for $(l,n_l), l>1$, its information sets are sets that consist of some $h_{<l}$ concatenated with all possible moves of $(l,1),\ldots,(l,n_{l-1})$. Then according the the classical definition of the subgame notion [27, 34], the only well-defined subgames are those rooted at $(l,1), \forall l$ since these are the only ones with singleton information set. So every level-l subgames corresponds to a subgame in the EFG representation, and every subgame in the EFG representation maps to a level-l subgame of the EFG by investigating the player index. Then the SPE for both representations are also equivalent. □

B OMITTED DETAILS FROM SECTION 3.2

Proof of Proposition 2. The learning dynamics of DBI can be written as $\boldsymbol{x}^t = (\boldsymbol{I}+\alpha G)(\boldsymbol{x}^{t-1})$. Since G has eigenvalues $\lambda_1,\ldots,\lambda_d$ at $\boldsymbol{x}^*$, the matrix $\boldsymbol{I}+\alpha\nabla_{\boldsymbol{x}}G$ at a stationary point $\boldsymbol{x}^*$ has eigenvalues $1+\alpha\lambda_1,\ldots,1+\alpha\lambda_d$. Since the set of LASPs is non-empty, $Re(\lambda_i)<0$ for all λ_i. Then for the choice of α^* in the Proposition, the modulus of the Jacobian

$$\rho(\boldsymbol{I}+\alpha^*\nabla_{\boldsymbol{x}}G) = \max_{i\in[d]} |1+\alpha^*\lambda_i| = \max_{i\in[d]} \sqrt{1+2\alpha^* Re(\lambda_i)+(\alpha^*)^2\left|\lambda_i\right|^2} < 1.$$

Proofs for the convergence rate of $O((1-\kappa/2)^t)$ can be found in previous work (e.g., Fiez et al. [11, Proposition F.1] or Wang et al. [39, Proposition 4])). For completeness, we provide a proof. Since $\rho((\boldsymbol{I}+\alpha\nabla_{\boldsymbol{x}}G)(\boldsymbol{x}^*)) = 1-\kappa$, according to Horn and Johnson [16, Lemma 5.6.10], there exists a matrix norm $\|\cdot\|$ such that $\|\boldsymbol{I}+\alpha\nabla_{\boldsymbol{x}}G\| < 1-\kappa+\epsilon$, for $\forall\epsilon>0$. We choose $\epsilon=\frac{\kappa}{4}$ The Taylor expansion of $(\boldsymbol{I}+\alpha G)$ at $\boldsymbol{x}^*$ is

$$(\boldsymbol{I}+\alpha G)(\boldsymbol{x}) = (\boldsymbol{I}+\alpha G)(\boldsymbol{x}^*) + (\boldsymbol{I}+\alpha\nabla_{\boldsymbol{x}}G)(\boldsymbol{x}^*)(\boldsymbol{x}-\boldsymbol{x}^*) + R(\boldsymbol{x}-\boldsymbol{x}^*),$$

where $R(\boldsymbol{x}-\boldsymbol{x}^*) = o(\|\boldsymbol{x}-\boldsymbol{x}^*\|)$. Let $R_1(\boldsymbol{x}-\boldsymbol{x}^*) = \frac{1}{\alpha}R(\boldsymbol{x}-\boldsymbol{x}^*)$, Then we have $\lim_{\boldsymbol{x}\to\boldsymbol{x}^*}\frac{R_1(\boldsymbol{x}-\boldsymbol{x}^*)}{\|\boldsymbol{x}-\boldsymbol{x}^*\|}=0$. Then we can choose $\delta>0$ such that $\|R_1(\boldsymbol{x}-\boldsymbol{x}^*)\| \leqslant \frac{\kappa}{4}\|\boldsymbol{x}-\boldsymbol{x}^*\|$ when $\|\boldsymbol{x}-\boldsymbol{x}^*\|<\delta$.

$$\begin{aligned}
\|G(\boldsymbol{x})-G(\boldsymbol{x}^*)\| &\leqslant \|\nabla_{\boldsymbol{x}}G(\boldsymbol{x}^*)(\boldsymbol{x}-\boldsymbol{x}^*)\| + \|R_1(\boldsymbol{x}-\boldsymbol{x}^*)\| \\
&\leqslant \|\nabla_{\boldsymbol{x}}G(\boldsymbol{x}^*)\|\,\|(\boldsymbol{x}-\boldsymbol{x}^*)\| + \frac{\kappa}{4}\|\boldsymbol{x}-\boldsymbol{x}^*\| \\
&\leqslant (1-\frac{\kappa}{2})\|\boldsymbol{x}-\boldsymbol{x}^*\|.
\end{aligned}$$

This shows the operator $(\boldsymbol{I}+\alpha G)$ is a contract mapping with contraction constant $(1-\frac{\kappa}{2})$. Therefore the convergence rate is $O((1-\kappa/2)^t)$ □

Before we present the proof of Proposition 3, we state the following lemma.

Lemma 1. *The update gradient vector G is L-Lipchitz if and only if $\|\nabla_{\boldsymbol{x}}G\| \leqslant L$ at all $\boldsymbol{x}\in\mathcal{X}$.*

Proof. First we prove for "if" direction. Consider $\boldsymbol{x}_1,\boldsymbol{x}_2\in\mathcal{X}$,

$$\begin{aligned}
G(\boldsymbol{x}_2)-G(\boldsymbol{x}_1) &= \int_0^1 \nabla_{\boldsymbol{x}}G(x_1+\tau(\boldsymbol{x}_2-\boldsymbol{x}_1))(\boldsymbol{x}_2-\boldsymbol{x}_1)d\tau \\
\Rightarrow \|G(\boldsymbol{x}_2)-G(\boldsymbol{x}_1)\| &= \left\|\int_0^1 \nabla_{\boldsymbol{x}}G(x_1+\tau(\boldsymbol{x}_2-\boldsymbol{x}_1))(\boldsymbol{x}_2-\boldsymbol{x}_1)d\tau\right\| \\
&\leqslant \left\|\int_0^1 \nabla_{\boldsymbol{x}}G(x_1+\tau(\boldsymbol{x}_2-\boldsymbol{x}_1))d\tau\right\| \|\boldsymbol{x}_2-\boldsymbol{x}_1\|
\end{aligned}$$

$$\leqslant \int_0^1 \|\nabla_{\boldsymbol{x}} G(x_1 + \tau(\boldsymbol{x}_2 - \boldsymbol{x}_1))\| \, d\tau \, \|\boldsymbol{x}_2 - \boldsymbol{x}_1\|$$
$$\leqslant L \|\boldsymbol{x}_2 - \boldsymbol{x}_1\| .$$

Then we prove for the "only if" direction. Take $\epsilon > 0$, then for $\forall \boldsymbol{x}_1, \boldsymbol{x}_2 \in \mathcal{X}$,

$$\left\| \int_0^\epsilon (\nabla_{\boldsymbol{x}} G(\boldsymbol{x}_1 + \tau \boldsymbol{x}_2)) \cdot \boldsymbol{x}_2 d\tau \right\| = \|G(\boldsymbol{x}_1 + \epsilon \boldsymbol{x}_2) - G(\boldsymbol{x}_1)\| \leqslant \epsilon L \|\boldsymbol{x}_2\|$$
$$\Rightarrow \lim_{\epsilon \to 0} \frac{\left\| \int_0^\epsilon (\nabla_{\boldsymbol{x}} G(\boldsymbol{x}_1 + \tau \boldsymbol{x}_2)) \cdot \boldsymbol{x}_2 d\tau \right\|}{\epsilon \|\boldsymbol{x}_2\|} = \frac{\|\nabla_{\boldsymbol{x}} G(\boldsymbol{x}_1) \cdot \boldsymbol{x}_2\|}{\|\boldsymbol{x}_2\|} \leqslant L.$$

Since it holds for any $\boldsymbol{x}_2$, it must be $\|\nabla_{\boldsymbol{x}} G(\boldsymbol{x}_1)\| \leqslant L$. And since it applies for any $\boldsymbol{x}_1 \in \mathcal{X}$, this completes the proof. □

Proof of Proposition 3. From the proof from Proposition 2 notice that in order to make $\|\boldsymbol{x}^t - \boldsymbol{x}^*\| \leqslant \epsilon$ we only have to let $t \geqslant \lceil \frac{2}{\kappa} \log \left\|\boldsymbol{x}^0 - \boldsymbol{x}^*\right\| / \epsilon \rceil$ since now $\|\boldsymbol{x}^t - \boldsymbol{x}^*\| \leqslant (1 - \frac{\kappa}{2})^t \left\|\boldsymbol{x}^0 - \boldsymbol{x}^*\right\| \leqslant \exp(-\kappa/2t) \leqslant \epsilon$. Now we need to characterize the region of initial points that can converge to $\boldsymbol{x}^*$ by characterizing the maximum possible radius of such initial point to $\boldsymbol{x}^*$. Recall in the proof of Proposition 2, this is captured by the. parameter δ. On one hand by using the Lipschitzness, we can bound the residual function by

$$\|R_1(\boldsymbol{x} - \boldsymbol{x}^*)\| \leqslant \int_0^1 \|\boldsymbol{I} + \alpha \nabla_{\boldsymbol{x}} G(\boldsymbol{x}^* + \tau(\boldsymbol{x} - \boldsymbol{x}^*)) - (\boldsymbol{I} + \alpha \nabla_{\boldsymbol{x}} G(\boldsymbol{x}^*))\| \, \|\boldsymbol{x} - \boldsymbol{x}^*\| \, d\tau \leqslant \frac{L}{2} \|\boldsymbol{x} - \boldsymbol{x}^*\|^2 .$$

One the other hand to maintain this convergence rate we should let $\|R_1(\boldsymbol{x} - \boldsymbol{x}^*)\| \leqslant \frac{\kappa}{4} \|\boldsymbol{x} - \boldsymbol{x}^*\|$. Then we simply let $\leqslant \frac{L}{2} \|\boldsymbol{x} - \boldsymbol{x}^*\|^2 \leqslant \frac{\kappa}{4} \|\boldsymbol{x} - \boldsymbol{x}^*\|$ we get an initial point should satisfy $\|\boldsymbol{x} - \boldsymbol{x}^*\| \leqslant \frac{\kappa}{2L}$. □

To prove Proposition 4, we need a few more machinery.

Proposition 6. *With $\alpha < 1/L$, $\boldsymbol{I} + \alpha G$ is a diffeomorphism.*

Proof. First we show $(\boldsymbol{I} + \alpha G)$ is invertible. Suppose $\boldsymbol{x}_1, \boldsymbol{x}_2 \in \mathcal{X}$ such that $\boldsymbol{x}_1 \neq \boldsymbol{x}_2$ and $(\boldsymbol{I} + \alpha G)(\boldsymbol{x}_1) = (\boldsymbol{I} + \alpha G)(\boldsymbol{x}_2)$. Then $\boldsymbol{x}_1 - \boldsymbol{x}_2 = \alpha(G(\boldsymbol{x}_2) - G(\boldsymbol{x}_1))$. And by $\alpha < 1/L$ we have $\|\boldsymbol{x}_1 - \boldsymbol{x}_2\| \leqslant \alpha L \|\boldsymbol{x}_1 - \boldsymbol{x}_2\| < \|\boldsymbol{x}_1 - \boldsymbol{x}_2\|$, which is a contraction.

Next we show we show its invert function is well-defined on any point of $\mathcal{X}$. Notice that $\rho(\alpha \nabla_{\boldsymbol{x}} G) \leqslant \|\alpha \nabla_{\boldsymbol{x}} G\| \leqslant \alpha L < 1$. And notice that since the eigenvalues of $\boldsymbol{I} + \alpha \nabla_{\boldsymbol{x}} G$ is the eigenvalues of $\alpha \nabla_{\boldsymbol{x}} G$ plus 1, the only way that makes $\det(\boldsymbol{I} + \alpha \nabla_{\boldsymbol{x}} G) = 0$ is to have one of the eigenvalues of $\alpha \nabla_{\boldsymbol{x}}$ to be -1. This contradicts to $\rho(\alpha \nabla_{\boldsymbol{x}} G) < 1$. Therefore by the implicit function theorem, $(\boldsymbol{I} + \alpha G)$ is a local diffeomorphism on any point of $\mathcal{X}$, and therefore $(\boldsymbol{I} + \alpha G)^{-1}$ is well defined on $\mathcal{X}$. □

Theorem 2 (Center and Stable Manifold [36, Theorem III.7, Chapter 5])**.** *Suppose $\boldsymbol{x}^* = h(\boldsymbol{x}^*)$ is a critical point for the C^r local diffeomorphism $h : \mathcal{X} \to \mathcal{X}$. Let $\mathcal{X} = \mathcal{X}_s \oplus \mathcal{X}_u$, where $\mathcal{X}_s$ is the stable center eigenspace belonging to those eigenvalues of $\nabla_{\boldsymbol{x}} h(\boldsymbol{x}^*)$ whose modulus is no greater than 1, and $\mathcal{X}_s$ is the unstable eigenspace belonging to those whose modulus is greater than 1. Then there exists a C^r embeded disk $W^{cs}_{loc}(\boldsymbol{x}^*)$ that is tangent to $\mathcal{X}_s$ at $\boldsymbol{x}^*$ called the local stable center manifold. Moreover there $\exists \epsilon > 0$, $h(W^{cs}_{loc}(\boldsymbol{x}^*)) \bigcap \mathbb{B}_\epsilon(\boldsymbol{x}^*) \subset W^{cs}_{loc}$ and $\bigcap_{t=0}^\infty h^{-t}(\mathbb{B}_\epsilon(\boldsymbol{x}^*)) \subset W^{cs}_{loc}(\boldsymbol{x}^*)$.*

Proof of Proposition 4. For $\forall \boldsymbol{x}^* \in \mathcal{X}^*_{sad}$, let $\epsilon(\boldsymbol{x}^*) > 0$ be the radius of neighborhood provided by Theorem 2 for diffeomorphism $h = \boldsymbol{I} + \alpha G$ and point $\boldsymbol{x}^*$. Then define $\mathscr{B} = \bigcup_{\boldsymbol{x}^* \in \mathcal{X}^*_{sad}} \mathbb{B}_{\epsilon(\boldsymbol{x}^*)}(\boldsymbol{x}^*)$. And since $\mathcal{X}$ is a subset of Euclidean space so it is second-countable. By Lindelöf's Lemma [21], which stated that every open cover there is a countable subcover, we can actually write $\mathscr{B} = \bigcup_{i=1}^\infty \mathbb{B}_{\epsilon(\boldsymbol{x}^*_i)}(\boldsymbol{x}^*_i)$ for a countable family of saddle points $\{\boldsymbol{x}^*_i\}_{i=1}^\infty \subseteq \mathcal{X}^*_{sad}$. Therefore for $\forall \boldsymbol{x}^0 \in \mathcal{X}^0_{sad}$ that converges to a saddle point, it must converge to $\boldsymbol{x}^*_i$ for some i. And $\exists T(\boldsymbol{x}^0) > 0$, such that $\forall t > T(\boldsymbol{x}^0), h^t(\boldsymbol{x}^0) \in \mathbb{B}_{\epsilon(\boldsymbol{x}^*_i)}(\boldsymbol{x}^*_i)$. From Theorem 2 we have $h^t(\boldsymbol{x}^0) \in W^{sc}_{loc}(\boldsymbol{x}^*_i)$. Since h is a diffeomorphism on $\mathcal{X}$ we have $\boldsymbol{x}^0 \in h^{-t}(W^{sc}_{loc}(\boldsymbol{x}^*_i))$. We furthermore union over all finite time step $\boldsymbol{x}^0 \in \bigcup_{t=0}^\infty h^{-t}(W^{sc}_{loc}(\boldsymbol{x}^*_i))$. Then we have $\mathcal{X}^0_{sad} \subseteq \bigcup_{i=1}^\infty \bigcup_{t=0}^\infty h^{-t}(W^{sc}_{loc}(\boldsymbol{x}^*_i))$. For each i since $\boldsymbol{x}^*_i$ is a saddle point, it has an eigenvalue greater than 1, so the dimension of unstable eigenspace $dim(\mathcal{X}_u(\boldsymbol{x}^*_i)) \geqslant 1$. Therefore the dimension of $W^{sc}_{loc}(\boldsymbol{x}^*_i)$ is less than full dimension. This leads to $\mu(W^{sc}_{loc}(\boldsymbol{x}^*_i)) = 0$ for any i. Again since h is a diffeomorphism, h^{-1} is locally Lipschitz which are null set preserving. Then $\mu(h^{-t}(W^{sc}_{loc}(\boldsymbol{x}^*_i))) = 0$ for $\forall i, t$. And since the countable union of measure zero sets is measure zero, $\mu(\bigcup_{i=1}^\infty \bigcup_{t=0}^\infty h^{-t}(W^{sc}_{loc}(\boldsymbol{x}^*_i))) = 0$. So $\mu(\mathcal{X}^0_{sad}) = 0$. □

B.1 LOCAL SUBGAME PERFECT EQUILIBRIUM

Our DBI algorithm essentially is designed to satisfy a first order condition $\forall(l,i), D_{\boldsymbol{x}_{l,i}} u_{l,i} = 0$. If we furthermore impose a second-order condition $\forall(l,i), D^2_{\boldsymbol{x}_{l,i},\boldsymbol{x}_{l,i}} u_{l,i} \prec 0$, then at such point every player reaches their local maxima in payoff, which is effectively corresponding to an local equilibrium notion. In fact, in two-level, one-agent each level case this is consistent with the local Stackelberg equilibrium notion in previous literature [11, 39]. We next formally define the notion of local subgame perfect equilibrium (LSPE).

Definition 2 (LSPE)**.** $\boldsymbol{x}^* = (\boldsymbol{x}^*_1, \ldots, \boldsymbol{x}^*_L)$ *is a* local subgame perfect equilibrium (LSPE) *if*

1. *For all players* (L,i) *at* $\boldsymbol{x}^*_{L,i}$, $\nabla_{\boldsymbol{x}_{L,i}} u_{L,i}(\boldsymbol{x}_{L,i}, \boldsymbol{x}^*_{\mathrm{PA}(L,i)}, \boldsymbol{x}^*_L) = 0$ *and* $\nabla^2_{\boldsymbol{x}_{L,i},\boldsymbol{x}_{L,i}} u_{L,i}(\boldsymbol{x}_{L,i}, \boldsymbol{x}^*_{\mathrm{PA}(L,i)}, < \boldsymbol{x}_{L,i}, \boldsymbol{x}*_{L,-i} >) \prec 0$.
2. *For all players* (i) *where* $l < L$ *at* $\boldsymbol{x}^*_{l,i}$, $D_{\boldsymbol{x}_i} u_i(\boldsymbol{x}_i, \boldsymbol{x}^*_{\mathrm{PA}(i)}, \Phi_l(< \boldsymbol{x}_i, \boldsymbol{x}^*_{l,-i} >)) = 0$ *and* $D^2_{\boldsymbol{x}_i,\boldsymbol{x}_i} u_i(\boldsymbol{x}_i, \boldsymbol{x}^*_{\mathrm{PA}(i)}, \Phi_l(< \boldsymbol{x}_i, \boldsymbol{x}^*_{l,-i} >)) \prec 0$.

Proposition 2 only guarantees that DBI will find an LASP when there exists one. So a natural question is to characterize the relationship between LSPEs and LASPs. Fiez et al. [11] shows that for two-player zero-sum Stackelberg games, the set of LASPs and the set of LSPEs coincide. Unfortunately, we next show that this is not the case for general SHGs.

Proposition 7. *There exists a SHG* $\mathcal{G}$ *such that LSPE*$(\mathcal{G}) \not\subset$ *LASP*$(\mathcal{G})$ *and LASP*$(\mathcal{G}) \not\subset$ *LSPE*$(\mathcal{G})$.

Proof of Proposition 7. Consider a two layer game with one player each level (also the Stackelberg game [11, 39]). Suppose the action for player 1 is x and for 2 is y, where $x, y \in \mathbb{R}$. The analytical form for first-order gradient is

$$D_x y = -\frac{\frac{\partial^2}{\partial y \partial x} u_2}{\frac{\partial^2}{\partial y^2} u_2}, D_x u_1 = \frac{\partial u_1}{\partial x} - \frac{\partial u_1}{\partial y} \frac{\frac{\partial^2}{\partial y \partial x} u_2}{\frac{\partial^2}{\partial y^2} u_2}, \text{ and, } D_y u_2 = \frac{\partial u_2}{\partial y}.$$

The total second-order closed-form is

$$\begin{aligned}
D^2_{x,x} u_1 = D_x(D_x u_1) &= \frac{\partial}{\partial x} D_x u_1 + \left(\frac{\partial}{\partial y} D_x u_1\right) \cdot D_x y \\
&= \frac{-\frac{\partial}{\partial y} u_1 \frac{\partial^3}{\partial y^3} u_2 (\frac{\partial^2}{\partial x \partial y} u_2)^2}{(\frac{\partial^2}{\partial y^2} u_2)^3} + \frac{\frac{\partial^2}{\partial x \partial y} u_2 (\frac{\partial^2}{\partial y^2} u_1 \frac{\partial^2}{\partial x \partial y} u_2 + 2 \frac{\partial}{\partial y} u_1 \frac{\partial^3}{\partial x \partial y^2} u_2)}{(\frac{\partial^2}{\partial y^2} u_2)^2} \\
&+ \frac{-2 \frac{\partial^2}{\partial x \partial y} u_1 \frac{\partial^2}{\partial x \partial y} u_2 + \frac{\partial}{\partial y} u_1 \frac{\partial^3}{\partial x^2 \partial y} u_2}{\frac{\partial^2}{\partial y^2} u_2} + \frac{\partial^2}{\partial x^2} u_1, \text{ and,} \\
D^2_{y,y} u_2 &= \frac{\partial^2}{\partial y^2} u_2.
\end{aligned}$$

The Jacobian matrix of the dynamics of Algorithm 1 is

$$\begin{pmatrix} \nabla_x(D_x u_1) & \nabla_y(D_x u_1) \\ \nabla_x(D_y u_2) & \nabla_y(D_y u_2) \end{pmatrix},$$

Where

$$\begin{aligned}
\nabla_x(D_x u_1) &= \frac{\frac{\partial}{\partial y} u_1 \frac{\partial^2}{\partial x \partial y} u_2 \frac{\partial^3}{\partial x \partial y^2} u_2}{(\frac{\partial^2}{\partial y^2} u_2)^2} - \frac{\frac{\partial^2}{\partial x \partial y} u_1 \frac{\partial^2}{\partial x \partial y} u_2 + \frac{\partial}{\partial y} u_1 \frac{\partial^3}{\partial x^2 \partial y} u_2}{\frac{\partial^2}{\partial y^2} u_2} + \frac{\partial^2}{\partial x^2} u_1, \\
\nabla_y(D_x u_1) &= \frac{\frac{\partial}{\partial y} u_1 \frac{\partial^2}{\partial x \partial y} u_2 \frac{\partial^3}{\partial y^3} u_2}{(\frac{\partial^2}{\partial y^2} u_2)^2} - \frac{\frac{\partial^2}{\partial y^2} u_1 \frac{\partial^2}{\partial x \partial y} u_2 + \frac{\partial}{\partial y} u_1 \frac{\partial^3}{\partial x \partial y^2} u_2}{\frac{\partial^2}{\partial y^2} u_2} + \frac{\partial^2}{\partial x \partial y} u_1, \\
\nabla_x(D_y u_2) &= \frac{\partial^2}{\partial x \partial y} u_2, \text{ and,}
\end{aligned}$$

$$\nabla_y(D_y u_2) = \frac{\partial^2}{\partial y^2} u_2.$$

Then consider payoff function $u_1(x,y) = -8x^4+8x^3y-5x^2y^2+5xy^3+6y^4+3x^3+3x^2y+6xy^2+3y^3-8x^2-xy-3y^2+6x$ and $u_2(x,y) = 4x^4 - x^3y + 5x^2y^2 - 10xy^3 + 10y^4 + 7x^3 - 9x^2y - 7xy^2 - 2y^3 - 3x^2 - 9xy - 4y^2 - 7x - 2y$.

We can derive that $D_x u_1 = 6 - 16x + 9x^2 - 32x^3 - y + 6xy + 24x^2y + 6y^2 - 10xy^2 + 5y^3 - ((-9 - 18x - 3x^2 - 14y + 20xy - 30y^2)(-x + 3x^2 + 8x^3 - 6y + 12xy - 10x^2y + 9y^2 + 15xy^2 + 24y^3))/(-8 - 14x + 10x^2 - 12y - 60xy + 120y^2)$ and $D_y u_2 = -2 - 9x - 9x^2 - x^3 - 8y - 14xy + 10x^2y - 6y^2 - 30xy^2 + 40y^3$. Then there are two critical points $(x_1^*, y_1^*) \approx (-0.35225, 0.09948)$ and $(x_2^*, y_2^*) \approx (-0.175443, -0.188133)$.

At point (x_1^*, y_1^*), $D^2_{x,x} u_1 = 28350.9, D^2_{y,y} u_2 = 0.268741$ the Jacobian $\begin{pmatrix} \nabla_x(D_x u_1) & \nabla_y(D_x u_1) \\ \nabla_x(D_y u_2) & \nabla_y(D_y u_2) \end{pmatrix} \approx \begin{pmatrix} -1464.55 & 1477.72 \\ -5.42229 & 0.268741 \end{pmatrix}$. And it can be verified that it is an LASP but not an LSPE.

At point (x_2^*, y_2^*), $D_{x,x} u_1 = -3413.93, D_{y,y} u_2 = -0.711531$ the Jacobian $\begin{pmatrix} \nabla_x(D_x u_1) & \nabla_y(D_x u_1) \\ \nabla_x(D_y u_2) & \nabla_y(D_y u_2) \end{pmatrix} \approx \begin{pmatrix} 27.9321 & 661.496 \\ -3.70221 & -0.711531 \end{pmatrix}$. And it can be verified that it is an LSPE but not an LASP. □

C OMITTED DETAILS FROM SECTION 4

C.1 DERIVATION OF TOTAL SECOND-ORDER DERIVATIVES

In our experiments we may need to check the second-order total derivative at a point. This section provides our approach to calculate such quantity.

For player of level L, $D^2_{\boldsymbol{x}_{l,i},\boldsymbol{x}_{l,i}} u_{l,i} = \nabla^2_{\boldsymbol{x}_{l,i},\boldsymbol{x}_{l,i}} u_{l,i}$.

For level $l < L$, notice that $D_{\boldsymbol{x}_{l,i}} u_{l,i}$ is an explicit function of the action variables of players that are descendants of (l,i). Denote this set of players as $DES(l,i)$ (excluding (l,i)), then we have $D^2_{\boldsymbol{x}_{l,i},\boldsymbol{x}_{l,i}} u_{l,i} = \nabla_{\boldsymbol{x}_{l,i}}(D_{\boldsymbol{x}_{l,i}} u_{l,i}) + \sum_{(\lambda,\eta)\in DES(l,i)} \nabla_{\boldsymbol{x}_{\lambda,\eta}}(D_{\boldsymbol{x}_{l,i}} u_{l,i}) D_{\boldsymbol{x}_{l,i}} \boldsymbol{x}_{\lambda,\eta}$.

Here $D_{\boldsymbol{x}_{l,i}} \boldsymbol{x}_{\lambda,\eta} = (-1)^{\lambda - l} \prod_{\substack{(\lambda',\eta')\in \\ PATH((\lambda,\eta) \\ \to (l,i))}} \left(\nabla^2_{\boldsymbol{x}_{\lambda',\eta'},\boldsymbol{x}_{\lambda',\eta'}} u_{\lambda',\eta'}\right)^{-1} \nabla^2_{\boldsymbol{x}_{\lambda',\eta'},\boldsymbol{x}_{PA(\lambda',\eta')}} u_{\lambda',\eta'}$.

The terms $\nabla_{\boldsymbol{x}_{l,i}}(D_{\boldsymbol{x}_{l,i}} u_{l,i})$ and $\nabla_{\boldsymbol{x}_{\lambda,\eta}}(D_{\boldsymbol{x}_{l,i}} u_{l,i})$ involve thrice derivatives. In practice we use either numerical finite-difference methods to approximate this value or use symbolic algebra feature in software tools such as Mathematica [42] to calculate the analytical form of this derivative.

C.2 DETAILS OF CONVERGENCE ANALYSIS EXPERIMENTS

Setup We compare the following algorithms, such that the gradient update vectors are as follows

DBI: $G^{DBI} = (D_{\boldsymbol{x}_1} u_1, \ldots, D_{\boldsymbol{x}_n} u_n)$. SIM: $G^{SIM} = (\nabla_{\boldsymbol{x}_1} u_1, \ldots, \nabla_{\boldsymbol{x}_n} u_n)$. SYM: $G^{SYM} = G^{SIM} + G^{SIM} A^{SIM}$, where $A^{SIM} = (J^{SIM} - (J^{SIM})^T)/2$, and J^{SIM} is the Jacobian matrix of G^{SIM}. HAM: $G^{HAM} = -\nabla_{\boldsymbol{x}} \left\|G^{SIM}\right\|^2$. CO: $G^{CO} = G^{SIM} + \gamma G^{HAM}$, where we let $\gamma = 0.1$ for all experiments. SYM_ALN: $G^{SYM} = G^{SIM} + \zeta G^{SIM} A^{SIM}$, where ζ is the sign of $\frac{1}{2d}(G^{HAM}(G^{SIM})^T)(G^{HAM}(G^{SIM} A^{SIM})^T) + 0.1$.

Experimental Details for (1, 1, 1), 1-d Action Games We use $u_1(x,z) = -7x^2 + 9xz + x - z$, $u_2(x,y,z) = -2y^2 - 4yz - 10x^2 + 2xz - 3z^2 + 4y + 7x - 8z - 8xyz$, and, $u_3(y,z) = -10z^2 - 9yz + 9y^2 - 5z - 2y$. Here (x,y,z) are action variables for players 1, 2, 3. We use the learning rate of $\alpha = 1e-5$ for all algorithms. We verify that DBI converges to an LSPE $(x,y,z) = (-0.34, 1.85, -1.08)$.

Experimental Details for (1, 1, 2), 1-d Action Games We use $u_1(x, y, z) = -2x^2 - 3xy + y^2 + 5x + 7y + 3xz - 10yz + 5xyz - 6z$, $u_2(w, x, y, z) = 2w^2 - wx - 3wy - 5x^2 + 9xy + 2y^2 + 3w + 5x - 4y + 5z^2 + 8wz + 7xz - 9yz - 10z$, $u_3(w, y, z) = -5y^2 - 8yz + z^2 + 8y - 9z - 2wy - 4wz - w^2 - 8wyz - 2w$, and, $u_4(w, y, z) = -10z^2 - 2yz + 5y^2 - 7z - 6y - 3wz - 8wy - 10wyz + 5w$ Here (x, w, y, z) are action variables for players 1, 2, 3, 4. We use $\alpha = 4e - 6$ for all algorithms. We verify that DBI converges to an LSPE $(x, w, y, z) = (4.70, -2.13, 10.27, 9.93)$.

Experimental Details for (1, 1, 1), 3-d Action Games We use $u_1(\boldsymbol{x}, \boldsymbol{z}) = -7\sum_{i=1}^{3} x_i^2 + 9(\sum_{i=1}^{3} x_i)(\sum_{i=1}^{3} z_i) + \sum_{i=1}^{3} x_i - \sum_{i=1}^{3} z_i$, $u_2(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z}) = -2\sum_{i=1}^{3} y_i^2 - 4(\sum_{i=1}^{3} y_i)(\sum_{i=1}^{3} z_i) - 10(\sum_{i=1}^{3} x_i)(\sum_{i=1}^{3} z_i) + 2\sum_{i=1}^{3} x_i^2 - 3\sum_{i=1}^{3} z_i^2 + 4(\sum_{i=1}^{3} x_i)(\sum_{i=1}^{3} y_i)(\sum_{i=1}^{3} z_i) + 7(\sum_{i=1}^{3} x_i) - 8(\sum_{i=1}^{3} y_i) - 8(\sum_{i=1}^{3} z_i)$, and, $u_3(\boldsymbol{y}, \boldsymbol{z}) = -10\sum_{i=1}^{3} z_i^2 - 9(\sum_{i=1}^{3} y_i)(\sum_{i=1}^{3} z_i) + 9\sum_{i=1}^{3} y_i^2 - 5\sum_{i=1}^{3} y_i - 2\sum_{i=1}^{3} z_i$, where $\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z}$ are action variables of players 1, 2, 3. We use the learning rate $\alpha = 1e - 5$ for all algorithms. We verify that DBI converges to an LSPE $(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z}) = ((-0.39, -0.39, -0.39), (0.29, 0.29, 0.29), (-0.58, -0.58, -0.58))$.

The time cost of each algorithm is measured in Table (1)

	DBI	SIM	SIM_ALN	SYM	CO	HAM
(1,1,1)-1d	1.65	1.26	5.92	1.61	4.91	4.58
(1,1,2)-1d	2.93	2.07	11.9	3.57	14.9	13.3
(1,1,1)-3d	12.4	10.9	60.1	9.63	49.4	53.9

Table 1: Comparison of the running times of DBI and baselines from experiment 1.

A note on the physical significance of the above utility functions Although our methodology is applicable in principle to any set of agents' utility function satisfying the definition of SHGs, it is reasonable to ask whether there exists a game model, grounded in some real-world domain, that exhibits the SHG hierarchy as well as the polynomial multi-variable utility structure above. To that end, consider the **COVID-19 Game** delineated below. In the experiments described below, each agent (a policy-maker in a hierarchy such as the federal government, state governments, and county governments) has a non-dimensional action restricted to closed interval $[0, 1]$, interpreted as a social distancing factor due to policy intervention (either recommended or implemented). However, in general, a policy intervention (hence, and agent's action) may take the form of a multi-dimensional real vector, e.g. factor for social distancing, factor for vaccination roll-out, etc.; hence, in general, the overall cost (negative payoff) of each agent is a function of $2 + n_L$ vectors rather than $2 + n_L$ scalars (correspoding to the agent, its parent, and all leaves) as in Equation (7). Although the actual functional form is not necessarily polynomial, we can always take a polynomial approximation in our model via curve-fitting or a Taylor series expansion. The above polynomial functions would then belong to the resulting class of functions; in our experiments for Convergence Analysis experiments are randomly chosen since the focus of this paper is not on modeling any real-world multi-agent interaction scenario to a high degree of accuracy but on assessing the performance of our algorithm on reasonable models consistent with the SHG structure. This line of thinking can be also applied to the utility functions used in the experiments on the relationship between LASP and LSPE described at the end of the appendix.

C.3 ADDITIONAL EXPERIMENTS ON THE RELATIONSHIP BETWEEN LASP AND LSPE

In this additional experiment, We will study the chance of having a convergent algorithm as well as the probability of finding an equilibrium upon convergence on several classes of SHGs. We design game classes of SHGs in a way that every game class $\mathcal{F}$ has the same parameter space on both its topology and payoff structure. For each class, we generate N instances, and for each instance, we first find the set of critical points by numerically solving the first order condition $D_{\boldsymbol{x}_i} u_i = 0, \forall(i)$. For each of these critical points, we determine whether it is an LASP of DBI by checking whether the Jacobian of updating gradient G have eigenvalues all of which have negative real part. If this is the case, we classify the critical point as an LASP according to Proposition 1. Finally, for each of the LASPs, we further check whether it is an LSPE by checking whether $D^2_{\boldsymbol{x}_i, \boldsymbol{x}_i} u_i \prec 0, \forall(i)$. Finally, across the N instances we compute the fraction of games for which an LASP was found (% LASP), and for those with an LASP, the fractions of the instances where an LASP is also an LSPE (% LSPE). We call these two numbers the measure properties of a game class.

We denote by our game classes by $\mathcal{F}^{\text{super}}_{\text{sub}}$ where the subscript is a vector listing the number of players in each SHG level from top to bottom, and the superscript indicates the parameters of the game. For example $\mathcal{F}^{C}_{1,1}$ is an SHG class with two levels and one player in each level (a Stackelberg game) with $C \in \mathbb{R}^+$ determining payoffs as follows: $u_i = \sum_{\alpha+\beta \leqslant 4, \alpha, \beta \in \mathbb{N}} c_{i,\alpha,\beta} x^\alpha y^\beta$,

where $x, y \in \mathbb{R}$ are 1-d action variables for the player ($i = 1$) and the second level-2 player ($i = 2$), respectively; the player-specific coefficients $c_{i,\alpha,\beta}$ are integers generated uniformly in $[-C, C]$ for different values of C. When $C = \infty$, we generate the coefficients from a continuous uniform distribution in $[-1, 1]$. See Appendix C for more information (e.g., the utility structure) of the other game classes.

$\mathcal{F}$	$\mathcal{F}^1_{1,1}$	$\mathcal{F}^{10}_{1,1}$	$\mathcal{F}^{\infty}_{1,1}$	$\mathcal{F}^1_{1,2}$	$\mathcal{F}^{10}_{1,2}$	$\mathcal{F}^{\infty}_{1,2}$	$\mathcal{F}^1_{1,1,1}$	$\mathcal{F}^{10}_{1,1,1}$	$\mathcal{F}^{\infty}_{1,1,1}$	$\mathcal{F}^1_{1,1,2}$	$\mathcal{F}^{10}_{1,1,2}$	$\mathcal{F}^{\infty}_{1,1,2}$
% LASP	52.6	57.3	59.7	32.0	35.0	36.8	47.0	51.0	50.3	13.6	21.3	21.2
% LSPE	88.8	87.9	87.1	65.6	64.3	60.0	66.5	64.3	67.0	62.1	60.7	60.2

Table 2: Results for $\mathcal{F}^C_{1,1}$ $\mathcal{F}^C_{1,2}$, $\mathcal{F}^C_{1,1,1}$ and $\mathcal{F}^C_{1,1,2}$ averaged over $N = 10^5$ instances.

The results are shown in Table 2. First, we notice for the same game topology, C does not appear to substantially affect the measure property. Parameter C essentially controls the granularity of a uniform discrete distribution and approaches a uniform continuous distribution as it becomes large. The measure properties will also be quite similar as C grows. The second observation is that as the topology becomes much more complex, it is less probable for DBI to find a stable point. The relevant probabilities degrade from $52\% \sim 59\%$ for Stackelberg games to $13\% \sim 21\%$ for 4-player 3-level games. This suggests a limitation of our algorithm in facing complex game topologies with more intricate back-propagation. However, we observe that the probability of an LASP being an LSPE does not decay speedily, as it achieves $87\% \sim 88\%$ for the structure of $(1, 1)$, $60\% \sim 67\%$ for $(1, 2)$, $(1, 1, 2)$ and $(1, 1, 1)$.

C.4 DETAILS OF THE EXPERIMENT

We elaborate our experimental designs here. For each function class $\mathcal{F}$, we generates N instances. For each instances, we solve equations $D_{\boldsymbol{x}_{l,i}} = 0, \forall (l, i)$, using numerical methods provided by Mathematica [42]. For each critical point found, we check whether the Jacobian of the updating gradient has eigenvalues all of which have negative real parts. If it is the case, we classify it as an LASP. For each of such LASP, we can determine whether it is an LSPE by checking whether $D^2_{\boldsymbol{x}_{l,i},\boldsymbol{x}_{l,i}} < 0$. Then for each instance, we can have two numbers: (1) an indicator demonstrating whether it is found an LASP (2) if there is an LASP, the fraction of these LASPs being LSPE. Then across these N instances, we define $\%LASP \triangleq \#(\text{Instances found LASP})/N$ and $\%LSPE \triangleq \sum(\text{Fraction of these LASPs being LSPE in an instance})/\#(\text{Instances found LASP})$.

$\mathcal{F}^C_{1,1}$ is detailed in the main body. For $\mathcal{F}^C_{1,2}$, the payoff structure is $u_{1,1}(x, y, z) = \sum_{\alpha+\beta+\gamma\leqslant 3,\alpha,\beta,\gamma\in\mathbb{N}} c_{(1,1),\alpha,\beta,\gamma} x^\alpha y^\beta z^\gamma$, $u_{2,1}(x, y, z) = \sum_{\alpha+\beta+\gamma\leqslant 3,\alpha,\beta,\gamma\in\mathbb{N}} c_{(2,1),\alpha,\beta,\gamma} x^\alpha y^\beta z^\gamma$, $u_{2,2}(x, y, z) = \sum_{\alpha+\beta+\gamma\leqslant 3,\alpha,\beta,\gamma\in\mathbb{N}} c_{(2,2),\alpha,\beta,\gamma} x^\alpha y^\beta z^\gamma$, where x, y, z are action variables for players $(1, 1), (2, 1), (2, 2)$ respectively.

For $\mathcal{F}^C_{1,1,1}$, $u_1(x, z) = \sum_{\alpha+\beta\leqslant 4,\alpha,\beta\in\mathbb{N}} c_{1,\alpha,\beta} x^\alpha z^\beta$, $u_2(x, y, z) = \sum_{\alpha+\beta+\gamma\leqslant 3,\alpha,\beta,\gamma\in\mathbb{N}} c_{2,\alpha,\beta} x^\alpha y^\beta z^\gamma$, $u_3(y, z) = \sum_{\alpha+\beta\leqslant 4,\alpha,\beta,\gamma\in\mathbb{N}} c_{3,\alpha,\beta} y^\alpha z^\beta$, where x, y, z are action variables for players $(1, 1), (2, 1), (3, 1)$ respectively.

For $\mathcal{F}^C_{1,1,2}$, $u_{1,1}(x, y, z) = \sum_{\alpha+\beta+\gamma\leqslant 3,\alpha,\beta,\gamma\in\mathbb{N}} c_{1,\alpha,\beta,\gamma} x^\alpha y^\beta z^\gamma$, $u_{2,1}(w, x, y, z) = \sum_{\alpha+\beta+\gamma+\iota\leqslant 2,\alpha,\beta,\gamma,\iota\in\mathbb{N}} c_{2,\alpha,\beta} x^\alpha y^\beta z^\gamma w^\iota$, $u_{3,1}(w, y, z) = \sum_{\alpha+\beta+\gamma\leqslant 3,\alpha,\beta,\gamma\in\mathbb{N}} c_{(3,1),\alpha,\beta,\gamma} w^\alpha y^\beta z^\gamma$, $u_{3,2}(w, y, z) = \sum_{\alpha+\beta+\gamma\leqslant 3,\alpha,\beta,\gamma\in\mathbb{N}} c_{(3,2),\alpha,\beta,\gamma} w^\alpha y^\beta z^\gamma$, where (x, w, y, z) are action variables for players $(1, 1), (2, 1), (3, 1), (3, 2)$.

C.5 DETAILS OF COVID-19 GAME EXPERIMENTS

Global Subgame Perfect Equilibrium in an SHG Before we analyze the COVID-model, we should formally define the notion of global subgame perfect equilibrium in an SHG. Denote the set of players as $\mathscr{N}$. Roughly speaking, a profile $\boldsymbol{x}^*$ is an SPE, if for every (l, i), given actions of players belonging to $\mathscr{N}\backslash(\{(l, i)\} \cup DES(l, i))$, player (l, i) reaches its maximum payoff by choosing $\boldsymbol{x}^*_{l,i}$, while assuming $DES(l, i)$ also had reached an SPE. We only consider pure SPE here so we assume $\boldsymbol{x}^* \in \mathcal{X}$.

However, there are a couple of issues here. First let's imagine when solving a two-level games with n_2 players at the second-level. Given $(1, 1)$'s action choice, the problem of computing SPE in the second-level is equivalent to computing a Nash equilibrium in the second-level. However, an exact pure Nash equilibrium may not exist. So which profile should we choose to propagate back to $(1, 1)$? To resolve this issue, we define ε-Nash equilibrium.

Definition 3. *For a simultaneous-move game, a profile $\boldsymbol{x}^*$ is an ε-Nash if for any player n, $\forall \boldsymbol{x}'_n \in \mathcal{X}_n, u_n(\boldsymbol{x}'_n, \boldsymbol{x}^*_{-n}) \leqslant u_n(\boldsymbol{x}^*) + \varepsilon$.*

In another word, an ε-Nash is a profile where for every player a unilateral deviation cannot offer benefit more than ε while fixing other's profile. In the context of our example for the two level game, given $(1,1)$'s action, we select the profile with the minimum ε of the simultaneous-move game defined on level 2 as an SPE back to $(1,1)$.

We now generalize this example to formally define the notion of ε-SPE in an SHG. First let us define $\Phi_{l,i}(\boldsymbol{x}) \in \mathbb{R}^{d_L}$ that returns the equilibrated profile at level L. In this profile, leaves that are not descendants of (l,i) are fixed in $\boldsymbol{x}$, while $LEAF(l,i)$ moved to a profile that corresponding to an SPE of $\mathcal{G}_{l,i}(\boldsymbol{x})$ with an minimum ε. Then we define the $\varepsilon_{l,i}(\boldsymbol{x})$ as $\max_{\boldsymbol{x}'_{l,i} \in \mathcal{X}_{l,i}} u_{l,i}(\boldsymbol{x}'_{l,i}, \boldsymbol{x}_{\text{PA}(l,i)}, \Phi(\boldsymbol{x}'_{l,i})) - u_{l,i}(\boldsymbol{x}_{l,i}, \boldsymbol{x}_{\text{PA}(l,i)}, \Phi(\boldsymbol{x}_{l,i}))$ and define $\varepsilon(\boldsymbol{x}) = \max_{l,i} \varepsilon(\boldsymbol{x})$ as the ε of profile $\boldsymbol{x}$ in an SHG $\mathcal{G}$.

Algorithm 2 Procedures for computing ε of profile $\boldsymbol{x}$ and an algorithm for computing an approximate SPE

Input: An SHG instance $\mathcal{G}$
Parameters: Best response iterations T

```
procedure SEARCH(x, (l, i))
    for x'_{l,i} ∈ X_{l,i} do
        u^{re-eq}_{l,i}(x'_{l,i}), ε_{DES(l,i)}(x'_{l,i}), x^{re-eq(l,i)}(x'_{l,i}) ← RE-EQ((l, i), x : x_{l,i} → x'_{l,i})
    Find x*_{l,i} ← arg max_{x'_{l,i}} u^{re-eq}_{l,i}(x'_{l,i})
    return x*_{l,i}, u^{re-eq}_{l,i}(x*_{l,i}), x^{re-eq(l,i)}(x*_{l,i})
end procedure
procedure RE-EQ(G, x, (l, i))
    ε ← 0
    if l < L then
        x, ε ← SHG_SOLVE((l, i), x)
    return u_{l,i}(x), ε, x
end procedure
procedure SHG_SOLVE((l, i), x)
    x^0 ← x
    ∀(l + 1, j) ∈ CHD(l, i), replace its action in x^0 with other random initialization
    for t = 1, 2, ..., T do                                          ▷ level-wise best response
        for (l + 1, j) ∈ CHD(l, i) do
            u^{re-eq}_{l+1,j}, ε_{DES(l+1,j)}, x^{re-eq(l+1,j)} ← RE-EQ(G, x^{t-1}, (l + 1, j))
            x'_{l+1,j}, u^{re-eq-max}_{l+1,j}, x^{re-eq-max(l+1,j)} ← SEARCH(x^{t-1}, (l + 1, j))
            ε^{t-1}_{l+1,j} = max{ε_{DES(l+1,j)}, u^{re-eq-max}_{l+1,j} − u^{re-eq}_{l+1,j}}
        x^t ← x^{t-1}
        for ∀(l + 1, j) ∈ CHD(l, i) do
            Replace dimensions of x^t belonging to DES(l + 1, j) with the ones in x^{re-eq-max(l+1,j)}
        ε^{t-1}_{l+1} ← max_i ε^{t-1}_{l+1,j}
    t* ← arg min_t ε^t_{l+1}
    return x^{t*}, ε^{t*}_{l+1}
end procedure
procedure COMPUTE_ε(x)
    for (l, i) do
        x*_{l,i}, u*_{l,i}, x^{re-eq(l,i)} ← SEARCH(x, (l, i))
        ε_{l,i} ← u*_{l,i} − u_{l,i}(x)
    return max_{l,i} ε_{l,i}
end procedure
```

We next provide an approximate algorithm (see Algorithm 2) to compute ϵ in an SHG. Pay attention here that $\boldsymbol{x} : \boldsymbol{x}_i \to \boldsymbol{x}'_i$ means replacing $\boldsymbol{x}_i$ of $\boldsymbol{x}$ with $\boldsymbol{x}'_i$. The functions in Algorithm 2 operate as follows. The procedure SEARCH takes a given joint profile $\boldsymbol{x}$, player index (l,i) and returns a best response profile for a given player (l,i). Our approach is that for each action in $\boldsymbol{x}'_{l,i} \in \mathcal{X}_{l,i}$, we re-equilibriate subgames $\mathcal{G}_{l,i}(\boldsymbol{x} : \boldsymbol{x}_{l,i} \to \boldsymbol{x}'_{l,i})$, and then compute the corresponding payoff. In our actual implementation, we discretize $\mathcal{X}_{l,i}$ in a bucket of grid points, and search within such bucket. The procedure RE-EQ returns the re-equilibrated profile of $\mathcal{G}_{l,i}(\boldsymbol{x})$ given (l,i) and $\boldsymbol{x}$. The procedure SHG_SOLVE solve a simultaneous-move game at level $l+1$. It applies an iterative best response approach for $\text{CHD}(l,i)$ to generate diverse profiles, and select the

one with minimum possible ε. In each iteration, for each $(l+1, j) \in \text{CHD}(l, i)$, it computes its best response action against the previous joint profile $\boldsymbol{x}^{t-1}$. And then in the next iteration it replace those descendant-profiles of $(l+1, j)$ in $\boldsymbol{x}^{t-1}$ by computed re-equilibrated profiles when $(l+1, j)$ selected its best response. Then it will return the joint profile with the minimum ε found so-far.

To solve the whole game, we just call SHG_SOLVE$((0,0), \boldsymbol{x})$, where $\boldsymbol{x}$ is some other action profile. To compute the ε of a given profile, we just call COMPUTE_ ε, where it just compute the maximum unilateral deviation for every player using SEARCH to compute the best response action and payoff.

Structured Game Model Inspired by COVID-19 Policy-Making We will now describe in detail the particular subclass of SHGs that we studied in our experiments reported in Section 4.2. This class is based on the SHG proposed in Jia et al. [18] which we describe in detail here. The exposition is in terms of a *cost function* $\mathtt{C}_i(\boldsymbol{x}_i, \boldsymbol{x}_{\text{PA}(i)}, \boldsymbol{x}_L)$ for each player, which is more natural in this context, rather than the payoff function $u_{l,i}$ introduced in Section 2, with the understanding that $u_i \equiv -\mathtt{C}_i$.

There are $L = 3$ levels in the hierarchy such that player $(1,1)$ represents the federal government (or, simply, government), the players $(2, i)$, $i \in \{1, 2, \ldots, n_2\}$ are state governments (or, simply, states), and the players $(3, i)$, $i \in \{1, 2, \ldots, n_3\}$ are county governments (or, simply, counties) partitioned into groups such that each group shares a single state as a parent.

Each player (i) takes a bounded, scalar action $x_i \in [0, 1]$ which is a *social-distancing factor* that (multiplicatively) reduces the proportion of post-intervention contacts among individuals — a lower number implies a stronger policy intervention, hence a lower number of infections but a higher cost of implementation (see below). The actions taken by counties represent policies that get *actually implemented* (hence directly impact the realized cost of every player — one of the defining attributes of SHGs) while those taken by the government and states are *recommendations*. Similar to Jia et al. [18], we also study a restricted variant where each county is non-strategic and constrained to comply with the action (recommendation) of its parent-state, effectively reducing the model to a 2-level hierarchy. We call this special case a *two-level game* (Figure 4(a)) and the more general model a *three-level game* (Figure 4(b)).

The cost function of each player (i) has, in general, three components: a *policy impact cost* $\mathtt{C}_i^{\text{inc}}(\boldsymbol{x}_L)$ which we will elaborate on below; a *policy implementation cost* $\mathtt{C}_i^{\text{dec}}(\boldsymbol{x}_L)$, e.g. economic and psychological costs of a lockdown; and, for each player in levels $l > 1$, a *non-compliance cost* $\mathtt{C}_i^{\text{NC}}(\boldsymbol{x}_i, \boldsymbol{x}_{\text{PA}(i)})$, a penalty incurred by a policy-maker for deviating from the recommendation of its parent in the hierarchy (e.g., a fine, litigation costs, or reputation harm).

Let $N_{(3,i)} > 0$ denote the fixed population under the jurisdiction of county $(3, i)$ for every $i \in \{1, 2, \ldots, n_3\}$. By construction, the population under each state is given $N_{(2,i)} = \sum_{(3,j) \in \text{CHD}(2,i)} N_{(3,j)}$ and the population under the government is $N_{(1,1)} = \sum_{i=1}^{n_2} N_{(2,i)}$. We next define the expressions for each component of the cost function.

Policy Impact Cost: This cost component is a quadratic closed-form approximation to the agent-based model introduced by Wilder et al. [41]. This is inspired by the infection cost computation approach in Jia et al. [18] but they used a different closed-form approximation. For each each county $(3, i)$, let $N_{(3,i)}^{init}$ denote the number of infected individuals within the population of the county prior to policy intervention; thus, the number of *post-intervention susceptible* individuals is $(N_{(3,i)} - N_{(3,i)}^{init}) x_{3,i}$. Another parameter in the game is the the *transport matrix* $R = \{r_{aa'}\}_{a,a' \in (3,1),(3,2),\ldots,(3,n_3)}$, where $r_{aa'} \geqslant 0$ is the proportion of the population of county a' that is active in county a in the absence of an intervention. Thus, in the number of post-intervention infected individuals of county a' that is active in county a is $r_{aa'} N_{a'}^{init} x_{a'}$. The last parameter in the model is M, the average number of contacts with active individuals that a susceptible individual makes, and finally μ is the probability that a susceptible individual gets infected upon contact with an active infected individual is $\mu \in (0, 1)$.

Putting these together, the policy impact cost is defined by the fraction of post-intervention infected individuals in county $a = (3, i)$, $i \in \{1, 2, \ldots, n_3\}$:

$$\mathtt{C}_a^{\text{inc}}(\boldsymbol{x}_L) = \mu M x_a \frac{N_a - N_a^{init}}{N_a^2} \Big(\sum_{a'} r_{aa'} N_{a'}^{init} x_{a'}\Big).$$

For a higher-level player (i),

$$\mathtt{C}_i^{\text{inc}}(\boldsymbol{x}_L) = \frac{1}{N_{(i)}} \sum_{a \in \text{CHD}(i)} N_a \mathtt{C}_a^{\text{inc}}(\boldsymbol{x}_L).$$

Policy Implementation Cost: For each county $(3, i)$, the policy implementation cost is given by

$$\mathtt{C}^{\text{dec}}_{3,i}(\boldsymbol{x}_L) = 1 - x_{3,i}.$$

For a higher-level player (i),

$$\mathtt{C}^{\text{dec}}_{i}(\boldsymbol{x}_L) = \frac{1}{N_{(i)}} \sum_{a \in \text{CHD}(i)} N_a \mathtt{C}^{\text{dec}}_{a}(\boldsymbol{x}_L).$$

Non-Compliance Cost: The non-compliance cost of player (i) for $l \in \{2, 3\}$ is given by Euclidean distance between its action and that of its parent:

$$\mathtt{C}^{\text{NC}}_{i}(x_i, x_{\text{PA}(i)}) = (x_i - x_{\text{PA}(i)})^2.$$

Finally, each player (i) for $l > 1$ has an idiosyncratic set of *weights* $\kappa_i \geqslant 0$ and $\eta_i \geqslant 0$ that trade its three cost components off against each other via a convex combination, and account for differences in ideology; the overall cost of such a player is given by

$$\mathtt{C}_i(x_i, x_{\text{PA}(i)}, \boldsymbol{x}_L) = \kappa_i \mathtt{C}^{\text{inc}}_{i}(\boldsymbol{x}_L) + \eta_i \mathtt{C}^{\text{dec}}_{i}(\boldsymbol{x}_L) + (1 - \kappa_i - \eta_i)\mathtt{C}^{\text{NC}}_{i}(x_i, x_{\text{PA}(i)}). \tag{7}$$

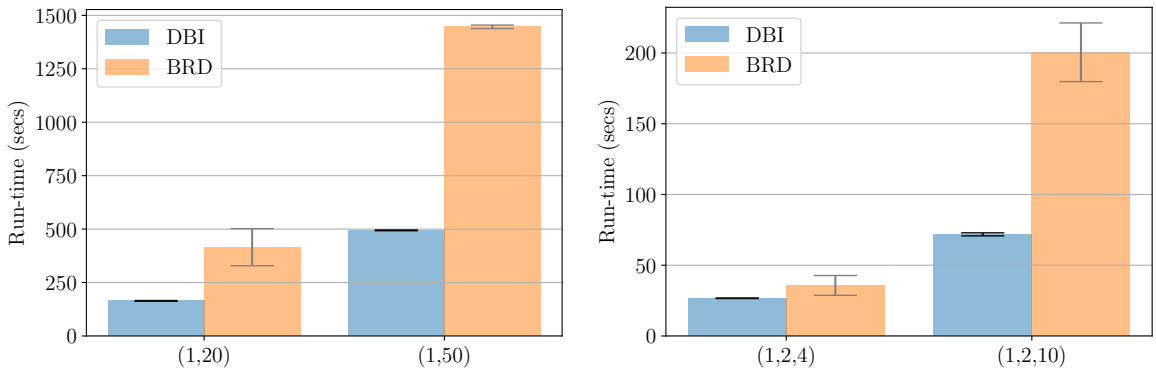
The player $(1, 1)$ obviously has no non-compliance issues, hence it has only one weight $\kappa_{1,1} > 0$, its overall cost being

$$\mathtt{C}_{1,1}(x_i, \boldsymbol{x}_L) = \kappa_{1,1}\mathtt{C}^{\text{inc}}_{1,1}(\boldsymbol{x}_L) + (1 - \kappa_{1,1})\mathtt{C}^{\text{dec}}_{1,1}(\boldsymbol{x}_L).$$

In our experiments, we set $r_{aa'} = 1/n_3$ for every pair of counties (a, a'), $M = 20$ and $\mu = 0.3$.

Experimental Setup and Further Results We formally define the algorithm BRD as the procedure SHG_SOLVE$((0, 0), \boldsymbol{x})$ for some randomly initilized $\boldsymbol{x}$. We start by defining the concept of one full *algorithm iteration* for each of DBI and BRD. For DBI(1, 20), DBI(1, 50), DBI(1, 2, 4), DBI(1, 2, 10), one algrithm iteration consists of 50 steps of gradient ascent, with a learning rate of 0.01. For BRD(1, 20) and BRD(1, 50), one algorithm iteration consists of 20 iterations of level-wise best response during the recursive procedure in SHG_SOLVE; for BRD(1, 2, 4) and BRD(1, 2, 10), it corresponds to 500 and 200 iterations of level-wise best response, respectively. For DBI we adopt a projector operator that project the resulted action into the nearest point in $[0, 1]$.

We discretize each action space uniformly into 101 grid points for two-level games, and 11 grid points for three-level games. We let $T = 100$ for BRD(1, 20) and BRD(1, 50) and $T = 20$ for BRD(1, 2, 4) and BRD(1, 2, 10). In the three-level experiments, the κ is set to be 0.5 for counties and the states and 0.8 for the government. The η is set to be 0.2 for the states, 0.3 for the counties in (1,2,4) setting , and 0.2 for (1,2,10) experiment. In the two-level experiments, the κ is set to be 0.2 for the government, 0.5 for counties and the states. The η is set to be 0.2 for counties and states.



(a) Two-level games.

(b) Three-level games.

Figure 6: The run-time (in secs) on COVID-19 Game Models. T = 20 for BRD experiments.

To further study the scalability of the BRD and DBI algorithms, we compare the run-time that (1) the DBI algorithm converges and (2) the BRD algorithm terminates. The convergence of DBI is defined as the convergence of the action profile.

When DBI converges, the action profile remains unchanged. In the projected gradient descent method, if all the gradients go to zero, DBI converges. Besides, it is also likely to hit the boundary of the constraints. The DBI is possible to converge with a non-zero gradient norm. The BRD algorithm terminates either T achieves, or ϵ goes to zero for all players. Under these conditions, the two algorithms find their optimal solutions. Figure 6 demonstrates the run-time results. We conduct each experiment four times with different random seeds. In two-level problems, the DBI algorithm is more than two times faster than BRD algorithms. Although the action spaces are discretized to 11 grid points for three-level games rather than 101 grids, DBI algorithms still perform better. In practice, discretizing the action spaces in 101 grids for three-level games is computational intensively. The performance and run-time of the BRD algorithm are more dependent on randomization and the initial points for the best responses at each level. When we face many players or multiple levels, the DBI algorithm is a natural choice. The DBI algorithm is significantly more efficient and more stable than the BRD algorithm.

C.6 DETAILS OF HIERARCHICAL PUBLIC GOODS GAME EXPERIMENTS

We extend the described public good game on Zachary's Karate club network to a (1-2-34) hierarchical game by introducing the non-compliance costs similar to the COVID-19 game model. The overall utility of the club individuals should be the combination of public good utility u_i and $\mathrm{C}_i^{\mathrm{NC}}(x_i, x_{\text{PA(i)}}) = (x_i - x_{\text{PA(i)}})^2$.

$$U_{3,i}(x_i, x_{\text{PA(i)}}, \mathbf{x}_L) = (1 - \kappa_{3,i}) u_i(x_i, \mathbf{x}_L) - \kappa_{3,i} \mathrm{C}_i^{\mathrm{NC}}(x_i, x_{\text{PA(i)}}). \tag{8}$$

In the second level, the administrator and instructor's utility is composed of the total public good utility of their group members, and their own non-compliance costs.

$$U_{2,i}(x_i, x_{\text{PA(i)}}, \mathbf{x}_L) = (1 - \kappa_{2,i}) \sum_{j \in \text{CHD}(i)} u_j(x_j, \mathbf{x}_L) - \kappa_{2,i} \mathrm{C}_i^{\mathrm{NC}}(x_i, x_{\text{PA(i)}}). \tag{9}$$

The root player's utility is considered as the social welfare of the whole club.

$$U_{1,1}(x_i, x_{\text{PA(i)}}, \mathbf{x}_L) = \sum_j u_j(x_j, \mathbf{x}_L) \tag{10}$$

In our experiments, we set $\kappa_{2,i} = \kappa_{3,j} = 0.5,\ \forall i, j$. Other parameters of the public good utility are set to be $a_i = 0, b_i = 1, c_i = 6,\ \forall i$. We use learning rate 0.1 in DBI. We project the results to $[0, 1]$. To compare with the DBI, we use BRD with discretized factors $0.5, 0.2, 0.1, 0.05$, and best response rounds $2, 3$.

C.7 DETAILS OF HIERARCHICAL SECURITY GAME EXPERIMENTS

Let $a = (a_1, \ldots, a_n)$ be the attacker strategy, the probability distribution over which defenders are attacked. In our experiments, a is set to follow a logit distribution $\mathrm{softmax}(\lambda(1 - \mathbf{x}_L))$, with defenders having lower security investment more likely to be attacked. Let $c_i(x_i) = c_i x_i$ be the cost of security investment for defender i, which is assumed to be linear. The utility of defender i is then as follows:

$$u_i(x_i, \mathbf{x}_L) = a_i \frac{x_i}{1 + x_i} + \sum_{j \neq i} a_j (1 - q_{ji} \frac{1}{1 + x_j} \frac{1}{1 + x_i}) - c_i(x_i)$$

.

We then build the (1,3,6) structured hierarchical game by introducing the non-compliance costs $\mathrm{C}_i^{\mathrm{NC}}(x_i, x_{\text{PA(i)}}) = (x_i - x_{\text{PA(i)}})^2$. The utility of each agent in different levels are the same as that of public good game (Eq. 8, 9, 10).

In the experiments, we consider the agents who almost obey the parent's suggestion ($\kappa = 0.1$) and the somewhat selfish agents ($\kappa = 0.5$). The influence probability $q_{ji},\ \forall i, j$ are set to be equal among the agents, and other parameters are set to $c_i = 0.2$, $\lambda = 5$. We use a learning rate of 0.1 in DBI. To compare with the DBI, we use BRD with discretized factors $0.2, 0.1, 0.05$ and best response rounds $2, 3$. Since there is no upper bound of the strategy profile, we set $x_i \leqslant 1.0$ according to the performance. Then we search the strategy space $x \in [0, 1]$ in the BRD execution.