Achieving Hierarchy-Free Approximation for Bilevel Programs With Equilibrium Constraints

Jiayang Li [1] **Jing Yu** [1] **Boyi Liu** [2] **Yu Nie** [1] **Zhaoran Wang** [2]

Abstract

In this paper, we develop an approximation scheme for solving bilevel programs with equilibrium constraints, which are generally difficult to solve. Among other things, calculating the first-order derivative in such a problem requires differentiation across the hierarchy, which is computationally intensive, if not prohibitive. To bypass the hierarchy, we propose to bound such bilevel programs, equivalent to multiple-followers Stackelberg games, with two new hierarchy-free problems: a T-step Cournot game and a T-step monopoly model. Since they are standard equilibrium or optimization problems, both can be efficiently solved via first-order methods. Importantly, we show that the bounds provided by these problems — the upper bound by the T-step Cournot game and the lower bound by the T-step monopoly model — can be made arbitrarily tight by increasing the step parameter T for a wide range of problems. We prove that a small T usually suffices under appropriate conditions to reach an approximation acceptable for most practical purposes. Eventually, the analytical insights are highlighted through numerical examples.

1. Introduction

Many bilevel optimization problems arising from real-world applications can be cast as a mathematical program whose feasible region is defined by an equilibrium problem (Luo, Pang, and Ralph, 1996; Outrata, Kocvara, Zowe, and Zowe, 1998). A typical example is a Stackelberg game concerning a leader who aims to induce a desirable outcome in an economic or social system comprised of many self-interested followers, who can be seen as playing a non-cooperative game that converges to a Nash equilibrium (Dafermos, 1973; Requate, 1993; Marcotte and Marquis, 1992; Labbé, Marcotte, and Savard, 1998; Ehtamo, Kitti, and Hämäläinen, 2002). More recently, motivated by such applications, developing efficient algorithms for solving bilevel programs with equilibrium constraints has also emerged as an essential topic in machine learning (Mguni, Jennings, Sison, Valcarcel Macua, Ceppi, and Munoz de Cote, 2019; Zheng, Trott, Srinivasa, Naik, Gruesbeck, Parkes, and Socher, 2020; Liu, Li, Yang, Wai, Hong, Nie, and Wang, 2022; Maheshwari, Kulkarni, Wu, and Sastry, 2022).

In the optimization literature, a bilevel program with equilibrium constraints is often written as (Luo et al., 1996)

$$\begin{aligned} \min_{\boldsymbol{x}\in\mathbb{X},\ \boldsymbol{y}^*\in\mathbb{Y}} \quad & l(\boldsymbol{x},\boldsymbol{y}^*), \\ \text{s.t.} \quad & \langle f(\boldsymbol{x},\boldsymbol{y}^*), \boldsymbol{y}-\boldsymbol{y}^*\rangle \geq 0, \quad \forall \boldsymbol{y}\in\mathbb{Y}, \end{aligned} \tag{1}$$

where $\mathbb{X} \subseteq \mathbb{R}^m$ and $\mathbb{Y} \subseteq \mathbb{R}^n$ are two convex set; $l : \mathbb{X}\times\mathbb{Y} \to \mathbb{R}$ and $f : \mathbb{X}\times\mathbb{Y} \to \mathbb{R}^n$ are two continuously differentiable functions. The lower-level problem in Problem (1) is a variational inequality (VI) problem, which is a general formulation for many equilibrium problems (Scutari, Palomar, Facchinei, and Pang, 2010; Nagurney, 2013; Parise and Ozdaglar, 2019). Problem (1) is well known for its intractability. Indeed, it is NP-hard even when the upper-level objective function is linear, and the lower-level VI can be reduced to a linear program (LP) (Ben-Ayed and Blair, 1990), leading to a so-called bilevel linear program (Bialas, Karwan, and Shaw, 1980; Candler and Townsley, 1982; Bard and Falk, 1982).

When the lower level is not an LP, Problem (1) is usually solved via first-order methods that strive to find good local solutions (Colson, Marcotte, and Savard, 2007). Classical algorithms in this category include basic gradient descent (Friesz, Tobin, Cho, and Mehta, 1990), steepest descent built on quadratic approximation (Luo et al., 1996), and the penalty method (Aiyoshi and Shimizu, 1984). Applying a gradient descent method requires differentiation through the lower-level equilibrium problem, which is a challenging computational task. In the literature, it is often accomplished via *implicit differentiation* (ID), which requires inverting a

[1]Department of Civil and Environmental Engineering, Northwestern University, Evanston, IL, USA [2]Department of Industrial Engineering and Management Science Engineering, Northwestern University, Evanston, IL, USA. Correspondence to: Yu (Marco) Nie <y-nie@northwestern.edu>.

Proceedings of the 40th International Conference on Machine Learning, Honolulu, Hawaii, USA. PMLR 202, 2023. Copyright 2023 by the author(s).

matrix whose size scales quadratically with the dimension of the lower-level VI problem (Tobin, 1986; Dafermos, 1988; Parise & Ozdaglar, 2019). In large-scale problems, even storing such a matrix may be impractical, let alone inverting them.

The recent advance in machine learning (ML) has inspired a new class of algorithms for solving bilevel programs based on *automatic differentiation* (AD) (Griewank et al., 1989). In AD-based methods, the gradient of a bilevel program is computed in two phases (Franceschi, Donini, Frasconi, and Pontil, 2017; Franceschi, Frasconi, Salzo, Grazzi, and Pontil, 2018). In the first phase, the lower-level problem is first solved, while the computation process, along with intermediate results, is stored in a computational graph. In the second phase, the gradient of the lower-level solution is evaluated by *unrolling* the computational graph. In the literature, AD-based methods were originally proposed for ML applications, e.g., hyperparameter optimization (Maclaurin, Duvenaud, and Adams, 2015) and neural architecture search (Liu, Simonyan, and Yang, 2018), which can usually be formulated as a bilevel program whose lower level is an unconstrained optimization problem. More recently, they were also extended to handle those with equilibrium constraints (Li, Yu, Nie, and Wang, 2020). Although AD-based methods bypass implicit differentiation, they may run into another challenge: since the computational graph grows with the number of iterations required to solve the lower-level equilibrium problem, it may become too deep to unroll efficiently even with AD (Li, Yu, Wang, Liu, Wang, and Nie, 2022b) when solving the lower-level problem requires too many iterations.

In a nutshell, finding the gradient for Problem (1) remains a potential obstacle to large-scale applications, whether ID or AD is used. These difficulties have motivated many work to accelerate ID (Hong, Wai, Wang, and Yang, 2020; Chen, Sun, and Yin, 2021; Liao, Xiong, Fetaya, Zhang, Yoon, Pitkow, Urtasun, and Zemel, 2018; Grazzi, Franceschi, Pontil, and Salzo, 2020; Vicol, Lorraine, Duvenaud, and Grosse, 2021; Fung, Heaton, Li, McKenzie, Osher, and Yin, 2022; Liu et al., 2022) or approximate AD (Luketina, Berglund, Greff, and Raiko, 2016; Metz, Poole, Pfau, and Sohl-Dickstein, 2016; Finn, Abbeel, and Levine, 2017; Liu et al., 2018; Shaban, Cheng, Hatch, and Boots, 2019; Ablin, Peyré, and Moreau, 2020; Yang, Ji, and Liang, 2021; Li, Gu, and Huang, 2022a). But fundamentally, the difficulty is inherent in the *hierarchy* of the problem, or the fact that to obtain the gradient, one is obligated to solve and differentiate through the lower-level problem, which is computationally demanding in many cases. Our work is prompted by the following question: *is it possible to free the process from that obligation, or to "bypass the hierarchy"?*

We believe a hierarchy-free method is possible. Our inspiration comes from the duopoly model in economics, which concerns two firms, A and B, selling a homogeneous product in the same market. The duopoly can be organized in three ways (Shapiro, 1989). (1) *Stackelberg duopoly*: Firm A sets its output first, according to which Firm B makes the decision, giving rise to a typical bilevel program. (2) *Cournot duopoly*: Firms A and B simultaneously optimize their own output, which results in a Nash equilibrium problem. (3) *Monopoly*: Firm A becomes the only producer by taking over Firm B's business and setting the total output for both. In economics, it is well known that Firm A's optimal profit in the Stackelberg duopoly is lower than that in a monopoly but higher than that in a Cournot duopoly.

As Problem (1) can be interpreted as a Stackelberg game in which the leader and the followers control the upper- and lower-level decision variables, respectively, we reason that it may be bounded in a similar way as the Stackelberg duopoly is bounded. Specifically, instead of directly solving Problem (1), we may first solve the corresponding "Cournot game" and "monopoly model" — both of which are single-level problems — to obtain lower and upper bounds. If the two bounds are close enough, we may accept the feasible one as an approximate solution. The caveat, of course, is that the natural gap between these two models may be unacceptably large for practical purposes. Thus, the focus of this investigation is to narrow down this gap.

Our contribution. In this paper, we view Problem (1) as a Stackelberg game and develop a new Cournot game and a new monopoly model that can provide arbitrarily tight upper and lower bounds for Problem (1). The development of both models assumes the lower-level equilibrium state is the outcome of a dynamical process through which the followers improve their decisions *step-by-step* towards optimality (Weibull, 1997). The two proposed models are defined as follows: in a T-step Cournot game, the leader and followers make decisions simultaneously, but the leader *anticipates* the followers' decisions by T steps, while in a T-step monopoly model, the leader has full *control* over the followers but allows them to move on their dynamical process toward equilibrium by T steps after the leader first dictates their decision.

Our contributions are threefold. (1) We show that both models can be efficiently solved via first-order methods, and the computation cost in each iteration grows linearly with T. (2) We prove that under appropriate assumptions and by choosing a suitable T, the gap between the upper and lower bounds provided by the solutions to the two models becomes arbitrarily tight; for most practical purposes, a small T suffices to provide a high-quality approximate solution to Problem (1). (3) We demonstrate the applications of the proposed approximation scheme in a range of real-world problems.

Organization. In Section 2, we highlight a few real-world applications that motivate the present study. In Section 3, we discuss the difficulties in solving Problem (1), along with a review of how they are addressed in previous research. Section 4 motivates the proposed hierarchy-free scheme by drawing an analogy between Problem (1) and the classical Stackelberg duopoly model. Section 5 lays the foundation for the scheme: the formulations of the T-step Cournot game and the T-step monopoly model before presenting the solution algorithm and discussing the analytical properties of the scheme. Finally, Section 6 presents numerical results, and Section 7 concludes the paper.

Notation. We use $\mathbb{R}$, $\mathbb{R}_+$, and $\mathbb{N}$ to denote the set of real numbers, non-negative real numbers, and non-negative integers. The inner product of two vectors $\boldsymbol{a}, \boldsymbol{b} \in \mathbb{R}^n$ is written as $\langle \boldsymbol{a}, \boldsymbol{b} \rangle = a^\top b$. For a matrix $\boldsymbol{A} \in \mathbb{R}^{n \times m}$, we denote $\|\boldsymbol{A}\|_2$ as its matrix norm induced by the 2-norm for vectors. For a closed set $\mathbb{A}$, we denote $\operatorname{diam}(\mathbb{A}) = \max_{\boldsymbol{a}, \boldsymbol{a}' \in \mathbb{A}} \|\boldsymbol{a} - \boldsymbol{a}'\|$ as its diameter. For a finite set $\mathbb{A}$, we write $|\mathbb{A}|$ as the number of elements in $\mathbb{A}$ and $\Delta(\mathbb{A}) = \{\boldsymbol{p} \in \mathbb{R}_+^{|\mathbb{A}|} : \mathbf{1}^\top \boldsymbol{p} = 1\}$.

2. Background

We will first discuss how VI provides a unified formulation for many equilibrium problems (Section 2.1) and then introduce a few real-world applications that motivate bilevel programs with equilibrium constraints (Section 2.2)

2.1. Variational Inequalities

VI provides a unified formulation for both atomic and nonatomic games, classified according to whether the set of agents is endowed with an atomic or a nonatomic measure (Nash, 1951; Schmeidler, 1973). Simply put, each agent's decision can affect the outcome of an atomic game, but the outcome of a nonatomic game solely depends on the aggregate behavior of the agents.

Example 2.1 (Atomic game)**.** *Consider a game played by n atomic agents. Suppose that each agent i aims to select a strategy $\boldsymbol{z}_i \in \mathbb{Z}_i \subseteq \mathbb{R}^{m_i}$ to minimize its cost, which a determined by a continuously differentiable function $u_i : \mathbb{Z} \to \mathbb{R}$ where $\mathbb{Z} = \prod_i \mathbb{Z}_i$. Formally, a joint strategy $z^* \in \mathbb{Z}$ is a Nash equilibrium if $u_i(\boldsymbol{z}_i^*, \boldsymbol{z}_{-i}^*) = \min_{\boldsymbol{z}_i \in \mathbb{Z}_i} u_i(\boldsymbol{z}_i, \boldsymbol{z}_{-i}^*)$ $(i = 1, \ldots, k)$. Denote $v(\boldsymbol{z}) = (v_i(\boldsymbol{z}))_{i=1}^k$ be a function with $v_i(\boldsymbol{z}) = \nabla_{\boldsymbol{z}_i} u(\boldsymbol{z})$. Suppose that $\mathbb{Z}_i$ is convex and closed, then any Nash equilibrium $\boldsymbol{z}^* = (\boldsymbol{z}_i^*)_{i=1}^k \in \mathbb{Z}$ is also a solution to the following VI (Scutari et al., 2010)*

$$\langle v(\boldsymbol{z}^*), \boldsymbol{z} - \boldsymbol{z}^* \rangle \geq 0, \quad \forall \boldsymbol{z} \in \mathbb{Z}. \tag{2}$$

Meanwhile, the reverse also holds if each v_i is convex in $\boldsymbol{z}_i$.

Example 2.2 (Nonatomic game)**.** *Consider a game played by n classes of nonatomic agents. Let $\mathbb{A}_i$ be the discrete action set for agents in class i. Let $\boldsymbol{p}_i = (p_{ia})_{a \in \mathbb{A}_i} \in \Delta(\mathbb{A}_i)$ be the proportion of agents in class i selecting each action $a \in \mathbb{A}_i$. Suppose that each agent in class i aims to select an action a to minimize the cost determined by a continuous function $c_{ia} : \Delta(\mathbb{A}) \to \mathbb{R}$ where $\Delta(\mathbb{A}) = \prod_i \Delta(\mathbb{A}_i)$. Formally, a mass distribution $\boldsymbol{p}^* = (\boldsymbol{p}_i^*)_{i=1}^k \in \Delta(\mathbb{A})$ is a Nash equilibrium (also known as a Wardrop equilibrium) if $c_{ia'}(\boldsymbol{p}^*) = \min_{a \in \mathbb{A}_i} c_{ia}(\boldsymbol{p}^*)$ for all $a' \in \mathbb{A}_i$ satisfying $q_{ia'} > 0$ $(i = 1, \ldots, k)$. Letting $c_i(\boldsymbol{p}) = (c_{ia}(\boldsymbol{p}))_{a \in \mathbb{A}_i}$ and $c(\boldsymbol{p}) = (c_i(\boldsymbol{p}))_{i=1}^k$, then $\boldsymbol{p}^*$ is a Nash equilibrium if and only if (Bernhard, 2011)*

$$\langle c(\boldsymbol{p}^*), \boldsymbol{p} - \boldsymbol{p}^* \rangle \geq 0, \quad \forall \boldsymbol{p} \in \Delta(\mathbb{A}). \tag{3}$$

As most equilibrium problems can be cast as VI, Problem (1) provides a standard formulation for bilevel programs with equilibrium constraints (Luo et al., 1996).

2.2. Bilevel Programs with Equilibrium Constraints

The study of bilevel programs can be traced back to the Stackelberg duopoly model (von Stackelberg, 1952).

Example 2.3 (Stackelberg duopoly)**.** *Consider two firms, A and B, selling a homogeneous product. Let their outputs be denoted, respectively, as x and y, and suppose that Firm A chooses x first, and then Firm B chooses y subsequently. Let the inverse demand (i.e., price) for the product be $p = 1 - x - y$. Then the profits for firms A and B are given as $l(x, y) = x(1 - x - y)$ and $g(x, y) = y(1 - x - y)$. The optimal decisions x^* and y^* of the two firms are the solution to the following bilevel program*

$$x^* = \arg\max_{x \geq 0} \; l(x, y^*), \quad \text{s.t. } y^* = \arg\max_{y \geq 0} \; g(x, y). \tag{4}$$

The optimal solution is $x^* = 1/2$ and $y^* = 1/4$, with Firm A making an optimal profit of $1/8$.

The earliest study on bilevel programs with equilibrium constraints was motivated by Stackelberg congestion games (SCGs), which concern a leader (usually a traffic planner) who aims to induce a desirable equilibrium state in a congestion game (Wardrop, 1952; Roughgarden and Tardos, 2002) played by many self-interested followers (travelers). The network design problem (LeBlanc, 1975; Li, Yang, Zhu, and Meng, 2012) and the congestion pricing problem (Lawphongpanich and Hearn, 2004; Li, Kockelman, and Huang, 2021) are two classic examples. More recently, the study of SCGs has been influenced by the introduction and constant evolution of connected and automated vehicle (CAV) technologies (Mahmassani, 2016), leading to such applications as the design of dedicated CAV facilities (Chen, He, Zhang, and Yin, 2016; Chen, He, Yin, and Du, 2017; Bahrami and Roorda, 2020) and the control of CAVs within such facilities (Levin and Boyles, 2016; Zhang and Nie, 2018).

The question of inducing a desirable outcome in noncooperative games can be traced back to the work of Pigou

(1920) on welfare economics. Bilevel programming has long been recognized as the standard approach to such inquiries in operations research, and more recently in the ML community (Mguni et al., 2019; Zheng et al., 2020; Liu et al., 2022; Maheshwari et al., 2022). Our algorithms are focused on the applications pertinent to this question.

We hope to clarify that not all bilevel programs (and Stackelberg games) are amenable to our algorithms. The *first* class is the Stackelberg games played by one leader and one follower in which the action sets of both are finite. Such problems can be reformulated as a linear program; examples include the generalized principal-agent problem (Myerson, 1982) and the Stackelberg security game (Sinha, Fang, An, Kiekintveld, and Tambe, 2018). The *second* class is a bilevel program constrained by an LP (Bracken and McGill, 1973), which is equivalent to an NP-hard mixed-integer program (Ben-Ayed & Blair, 1990) that cannot be effectively solved via first-order methods.

3. Challenges

In this section, we will discuss the difficulties in computing the first-order gradient of Problem (1), which reads

$$\frac{\partial l(\boldsymbol{x}, \boldsymbol{y}^*)}{\partial \boldsymbol{x}} = \nabla_{\boldsymbol{x}} l(\boldsymbol{x}, \boldsymbol{y}^*) + \frac{\partial \boldsymbol{y}^*}{\partial \boldsymbol{x}} \cdot \nabla_{\boldsymbol{y}} l(\boldsymbol{x}, \boldsymbol{y}^*). \quad (5)$$

To obtain this gradient, we need to *solve* and *differentiate through* the lower-level VI problem.

Given any $\boldsymbol{x} \in \mathbb{X}$, we denote the solution set to the lower-level VI problem in Problem (1) as $\mathbb{Y}^*(\boldsymbol{x})$. We first give the following proposition for characterizing $\mathbb{Y}^*(\boldsymbol{x})$.

Proposition 3.1 (Hartman, Stampacchia, et al. (1966))**.** *Suppose that $\mathbb{Y}$ is closed. Let $h : \mathbb{X} \times \mathbb{Y} \to \mathbb{Y}$ be a function that satisfies*

$$h(\boldsymbol{x}, \boldsymbol{y}) = \arg\min_{\boldsymbol{y}' \in \mathbb{Y}} \|\boldsymbol{y}' - r \cdot f(\boldsymbol{x}, \boldsymbol{y})\|_2^2. \quad (6)$$

Then given any $\boldsymbol{x} \in \mathbb{X}$, we have $\boldsymbol{y}^* \in \mathbb{Y}^*(\boldsymbol{x})$ if and only if $\boldsymbol{y}^*$ is a fixed point of $h(\boldsymbol{x}, \cdot)$, i.e., $\boldsymbol{y}^* = h(\boldsymbol{x}, \boldsymbol{y}^*)$.

To solve $\mathbb{Y}^*(\boldsymbol{x})$, Proposition 3.1 has inspired a general class of algorithms, commonly known as the projection method Dafermos (1983); Pang and Chan (1982); Marcotte and Wu (1995), which iteratively project $\boldsymbol{y}^t$ to $\boldsymbol{y}^{t+1} = h(\boldsymbol{x}; \boldsymbol{y}^t)$, starting from some $\boldsymbol{y}^0 \in \mathbb{Y}$, until a fixed point is found.

To differentiate through $\mathbb{Y}^*(\boldsymbol{x})$, however, the lower-level VI problem must admit a unique solution; otherwise, the Jacobian matrix $\partial \boldsymbol{y}^* / \partial \boldsymbol{x}$ is not well defined. To secure the uniqueness, one often needs to assume $f(\boldsymbol{x}, \cdot)$ is strongly monotone (Mancino and Stampacchia, 1972). Our work follows many previous studies (Ghadimi and Wang, 2018; Hong et al., 2020; Chen et al., 2021; Guo, Hu, Zhang, and Yang, 2021; Ji, Yang, and Liang, 2021; Liu et al., 2022) to adopt this assumption, but we will discuss how to relax it in the appendix. The following proposition characterizes the convergence rate of the projection method when $f(\boldsymbol{x}, \cdot)$ is strongly monotone.

Proposition 3.2 (Nagurney (2013))**.** *Suppose that $f(\boldsymbol{x}, \cdot)$ is γ-strongly monotone and L-Lipschitz continuous, then*

$$\|h(\boldsymbol{x}, \boldsymbol{y}) - h(\boldsymbol{x}, \boldsymbol{y}')\|_2 \leq \eta \cdot \|\boldsymbol{y} - \boldsymbol{y}'\|_2 \quad (7)$$

for all $\boldsymbol{y}, \boldsymbol{y}' \in \mathbb{Y}$, where $\eta = (1 - 2\gamma r/\sigma + r^2 L^2/\sigma^2)^{1/2}$. Hence, starting from any $\boldsymbol{y}^0 \in \mathbb{Y}$, then the sequence $\boldsymbol{y}^{t+1} = h(\boldsymbol{x}, \boldsymbol{y}^t)$ converges to the unique point $\boldsymbol{y}^* \in \mathbb{Y}^*(\boldsymbol{x})$ at a linear rate as long as the step size $r < 2\gamma/L^2$.

In the remainder of this section, we will discuss the calculation of $\partial \boldsymbol{y}^* / \partial \boldsymbol{x}$, which is the main obstacle behind implementing any first-order methods.

3.1. Implicit Differentiation (ID) Methods

The first method to calculate $\partial \boldsymbol{y}^* / \partial \boldsymbol{x}$ is to implicitly differentiate through the fixed-point equation $\boldsymbol{y}^* = h(\boldsymbol{x}, \boldsymbol{y}^*)$, which subsequently gives rise to the following proposition.

Proposition 3.3 (Dafermos (1988))**.** *If $h(\boldsymbol{x}, \boldsymbol{y})$ is continuously differentiable and $f(\boldsymbol{x}, \cdot)$ is strongly monotone, then the unique $\boldsymbol{y}^* \in \mathbb{Y}^*(\boldsymbol{x})$ is continuously differentiable in $\boldsymbol{x}$ with the Jacobian matrix satisfying*

$$\frac{\partial \boldsymbol{y}^*}{\partial \boldsymbol{x}} = \nabla_{\boldsymbol{x}} h(\boldsymbol{x}, \boldsymbol{y}^*) \cdot (\boldsymbol{I} - \nabla_{\boldsymbol{y}} h(\boldsymbol{x}, \boldsymbol{y}^*))^{-1}. \quad (8)$$

To calculate $\partial \boldsymbol{y}^* / \partial \boldsymbol{x}$ according to Equation (8), one first needs to obtain $\nabla_{\boldsymbol{x}} h(\boldsymbol{x}, \boldsymbol{y}^*)$ and $\nabla_{\boldsymbol{y}} h(\boldsymbol{x}, \boldsymbol{y}^*)$, that is, differentiating through a Euclidean projection problem, equivalent to a quadratic program (QP). One way to perform it is using the Python package `cvxpylayers` developed by Agrawal, Amos, Barratt, Boyd, Diamond, and Kolter (2019). The computational cost of implicit differentiation (ID) is high because it requires solving the lower-level VI problem and inverting a matrix that can be prohibitively large.

Single-looped ID. To prevent repeatedly solving for $\boldsymbol{y}^*$, one could update both $\boldsymbol{x}$ and $\boldsymbol{y}$ by one gradient-descent step at each iteration in a single loop. In such schemes, instead of calculating the exact upper-level gradient via Equations (5) and (8), we replace $\boldsymbol{y}^*$ therein by the current iteration. The scheme was initially proposed by Hong et al. (2020); Chen et al. (2021) and later extended to Problem (1) by Liu et al. (2022). The single-loop scheme simplifies the overall structure of ID-based methods but does not bypass the difficulty of inverting large matrices.

Approximated ID. To prevent matrix inversion in (8), one can represent its inversion by the corresponding Neumann series and then truncate the series by keeping only its first

few terms (Liao et al., 2018; Grazzi et al., 2020; Vicol et al., 2021). The Jacobian-free scheme proposed by Fung et al. (2022) can also be interpreted through Neumann-series truncation. The scheme significantly reduces the time complexity but requires storing $\nabla_{\boldsymbol{y}} h(\boldsymbol{x}, \boldsymbol{y}^*)$.

There are other schemes designed to improve ID. For example, Bertrand, Klopfenstein, Blondel, Vaiter, Gramfort, and Salmon (2020) developed a matrix-free ID scheme for lasso-type problems; Blondel, Berthet, Cuturi, Frostig, Hoyer, Llinares-López, Pedregosa, and Vert (2021) developed a toolbox combining ID and AD benefits; Sow, Ji, and Liang (2022) proposed a method that adopts a zeroth-order-like estimator to approximate the Jacobian matrix.

3.2. Automatic Differentiation Methods

The second method to compute $\partial \boldsymbol{y}^* / \partial \boldsymbol{x}$ is to unroll the computation process for solving the lower-level VI problem via AD (Franceschi et al., 2017; 2018). For example, if the projection method is adopted, the computation process can be written as

$$\boldsymbol{y}^t = h(\boldsymbol{x}, \boldsymbol{y}^{t-1}), \quad t = 1, \ldots, T, \tag{9}$$

where T is a sufficiently large number such that the distance between $\boldsymbol{y}^T$ and $\mathbb{Y}^*(\boldsymbol{x})$ is smaller than a tolerance value. The aforementioned `cvxpylayers` proposed by Agrawal et al. (2019) package can be employed to wrap the computation process behind each $\boldsymbol{y}^t = h(\boldsymbol{x}, \boldsymbol{y}^{t-1})$ as a computational graph.

Since the computational graph grows with the number of iterations required to solve the lower-level equilibrium problem, it may become too deep to unroll efficiently even with AD, when solving the equilibrium problem requires too many iterations. Particularly for Problem (1), $h(\boldsymbol{x}, \boldsymbol{y}^{t-1})$ is equivalent to a constrained QP, which is more costly to store than in most ML applications, whose lower-level problem is typically unconstrained.

Truncated AD. The difficulty in storing a large computational graph may be bypassed by truncated AD, which, by only unrolling the last portion of the graph, settles for an approximate gradient (Shaban et al., 2019).

One-stage AD. Another approximation scheme is called one-stage AD, which updates $\boldsymbol{x}$ and $\boldsymbol{y}$ simultaneously in a single loop. Specifically, whenever $\boldsymbol{y}$ in the lower level is updated by one step, one-stage AD unrolls it to obtain the gradient for updating $\boldsymbol{x}$ in the upper level. The scheme has delivered satisfactory performance on many tasks (Luketina et al., 2016; Metz et al., 2016; Finn et al., 2017; Liu et al., 2018; Xu, Xie, Zhang, Chen, Qi, Tian, and Xiong, 2019). The method proposed by Li et al. (2022a) also shares a similar single-loop structure.

The performance of the above approximation schemes has been extensively tested (Franceschi et al., 2018; Wu, Ren, Liao, and Grosse, 2018). Ablin et al. (2020) discussed the efficiency of AD; Ji et al. (2021) analyzed the convergence rate of AD and approximated ID; Yang et al. (2021) and Dagréou, Ablin, Vaiter, and Moreau (2022) studied variance reduction in AD-based methods. For other bilevel programming algorithms for ML applications, see, e.g.,Pedregosa (2016); Lorraine and Duvenaud (2018); MacKay, Vicol, Lorraine, Duvenaud, and Grosse (2019); Bae and Grosse (2020); Ji et al. (2021); Grazzi, Pontil, and Salzo (2021); Zucchet and Sacramento (2022). The reader may also consult Liu, Gao, Zhang, Meng, and Lin (2021) for a survey.

Our scheme. A main difference between our scheme and the previous works is that we do not attempt to approximate the Jacobian matrix $\partial \boldsymbol{y}^* / \partial \boldsymbol{x}$ returned by exact ID or AD. Instead, we directly approximate the original bilevel program with two hierarchy-free models inspired by the classic economic competition theory; to the best of our knowledge, this angle is novel, even though the resulting algorithms do share some features with existing ID- and AD-based methods, as we shall see.

4. Motivation

We first discuss how the Stackelberg duopoly model (cf. Example 2.3) can be bounded by single-level models.

4.1. Classic Models

We first introduce the market structures of classic Cournot duopoly and monopoly models based on Example 2.3.

Cournot duopoly. Firm A loses its first-mover advantage and has to make decisions simultaneously with Firm B. The optimal decisions $\bar{x}$ and $\bar{y}$ of the two firms can be found by solving the following equilibrium problem

$$\begin{cases} \bar{x} = \arg\max\limits_{x \geq 0} \; l(x, \bar{y}), \\ \bar{y} = \arg\max\limits_{y \geq 0} \; g(\bar{x}, y). \end{cases} \tag{10}$$

The optimal solution is $\bar{x} = 1/3$ and $\bar{y} = 1/3$, with Firm A earning an optimal profit of $1/9$.

Monopoly. Firm B is taken over by Firm A. To find the optimal production levels $\hat{x}$ and $\hat{y}$, we need to solve the following optimization problem

$$\hat{x}, \hat{y} = \arg\max_{x, y \geq 0} \; l(x, y). \tag{11}$$

The optimal solution is $\hat{x} = 1/2$ and $\hat{y} = 0$, and the optimal profit of Firm A rises to $1/4$.

Comparision. Intuitively, when the leader gives up the first-mover advantage, its market power is weakened; on the contrary, when the leader takes over Firm B's business,

its market power is maximized. Indeed, comparing the optimal profits of Firm A, we have $1/4 > 1/8 > 1/9$, which indicates that a Stackelberg duopoly is bounded by a monopoly from above and by a Cournot duopoly from below.

4.2. Our Models

Our focus is to narrow the gap between the upper and lower bounds. To this end, we "decompose" the best response of Firm B against Firm A's decision into a dynamical process. Since Firm B faces a constrained optimization problem, the projected gradient descent method charts a natural path to optimality. Specifically, given x and a step size $r > 0$, the dynamical process through which Firm B finds its optimal decision can be described by a gradient-descent step, represented by

$$h(x, y) = \max\{y + r \cdot (1 - x - 2y), 0\}.$$

Starting with $h^{(0)}(x, y) := y$, thus, $h^{(t+1)}(x, y)$ can be defined recursively as $h(x, h^{(t)}(x, y))$ $(t = 0, 1, \ldots)$.

We are now ready to introduce the T-step Cournot duopoly model and the T-step monopoly model.

T-step Cournot duopoly. Firm A and Firm B still choose x and y simultaneously, but Firm A bases its decision on the expectation that Firm B will update its decision T times given x, i.e., from y to $h^{(T)}(x, y)$. Therefore, the optimal decisions $\bar{x}$ and $\bar{y}$ can be found by solving

$$\begin{cases} \bar{x} = \arg\max\limits_{x \geq 0} \; l(x, h^{(T)}(x, \bar{y})), \\ \bar{y} = \arg\max\limits_{y \geq 0} \; g(\bar{x}, y). \end{cases} \tag{12}$$

T-step monopoly. Firm A imposes a decision y on Firm B but allows Firm B to update its decision T steps starting from y. To find the optimal decision $\hat{x}$ and $\hat{y}$ ($\hat{y}$ refers to the production of Firm B initially dictated by Firm A), we need to solve

$$\hat{x}, \hat{y} = \arg\max_{x, y \geq 0} \; l(x, h^{(T)}(x, \bar{y})). \tag{13}$$

The question we set out to answer is the following: *will Firm A's optimal profit in the two new models be closer to that in the Stackelberg duopoly*? Intuitively, the answer is yes, because the T-step Cournot duopoly interpolates "no anticipation" and "full anticipation" on Firm B's response, whereas the T-step duopoly interpolates "full dictation" and "no dictation" on Firm B's decision. Hence, the market power of Firm A will rise and fall, respectively, in T-step monopoly and T-step Cournot, leading to an optimal profit closer to that in the Stackelberg duopoly.

A key observation. From the formulations, we find the two proposed models can be respectively viewed as a classic Cournot duopoly and a classic monopoly, with Firm A's objective function changed from $l(x, y)$ to $l^{(T)}(x, y) := l(x, h^{(T)}(x, y))$. Based on this observation, we are now ready to approximate Problem (1) with the two models.

5. Main Results

We view Problem (1) as a Stackelberg game and refer to the upper- and lower-level decision-makers therein as the leader and the followers, respectively. To extend the two models presented in Section 4, we first need a function to model the process through which the followers iteratively update their decisions towards equilibrium. As the lower level is a VI, the function $h(\boldsymbol{x}, \boldsymbol{y})$ defined by Equation (6) is a natural choice. Similarly, starting with $h^{(0)}(\boldsymbol{x}, \boldsymbol{y}) := \boldsymbol{y}$, we can recursively define $h^{(t+1)}(\boldsymbol{x}, \boldsymbol{y}) = h(\boldsymbol{x}, h^{(t)}(\boldsymbol{x}, \boldsymbol{y}))$ and $l^{(T)}(\boldsymbol{x}, \boldsymbol{y}) = l(\boldsymbol{x}, h^{(T)}(\boldsymbol{x}, \boldsymbol{y}))$.

T-step Cournot game. The leader and the followers choose $\boldsymbol{x} \in \mathbb{X}$ and $\boldsymbol{y} \in \mathbb{Y}$ simultaneously, but the leader looks ahead T steps along the followers' evolutionary path. The equilibrium state $(\bar{\boldsymbol{x}}, \bar{\boldsymbol{y}})$ thus can be found by solving

$$\begin{cases} \bar{\boldsymbol{x}} \in \arg\min\limits_{\boldsymbol{x} \in \mathbb{X}} \; l^{(T)}(\boldsymbol{x}, \bar{\boldsymbol{y}}), \\ \bar{\boldsymbol{y}} \in \mathbb{Y}^*(\bar{\boldsymbol{x}}). \end{cases} \tag{14}$$

It is a Nash-equilibrium problem: no follower has an incentive to change their decisions because $\bar{\boldsymbol{y}} \in \mathbb{Y}^*(\bar{\boldsymbol{x}})$; the leader also cannot further reduce $l^{(T)}(\boldsymbol{x}, \bar{\boldsymbol{y}})$ given $\bar{\boldsymbol{y}}$.

T-step monopoly model. The leader dictates a decision $\boldsymbol{y} \in \mathbb{Y}$ for the followers but allows them to update their decisions T steps based on $\boldsymbol{y} \in \mathbb{Y}$. To find the optimal decision $\hat{\boldsymbol{x}}$ and $\hat{\boldsymbol{y}}$, we need to solve

$$\hat{\boldsymbol{x}}, \hat{\boldsymbol{y}} \in \arg\min_{\boldsymbol{x} \in \mathbb{X},\, \boldsymbol{y} \in \mathbb{Y}.} \; l^{(T)}(\boldsymbol{x}, \boldsymbol{y}). \tag{15}$$

It is a single-level optimization problem with the leader's and the followers' decisions optimized altogether.

5.1. Optimality gaps

We first give the following proposition, which holds true regardless of the form of $h(\boldsymbol{x}, \boldsymbol{y})$ or the properties of $f(\boldsymbol{x}, \cdot)$.

Proposition 5.1. *Suppose that the function $h : \mathbb{X} \to \mathbb{Y}$ used for formulating Problems (14) and (15) satisfies the following condition*

$$\lim_{t \to \infty} h^{(t)}(\boldsymbol{x}, \boldsymbol{y}) \in \mathbb{Y}^*(\boldsymbol{x}), \quad \forall \boldsymbol{x} \in \mathbb{X},\ \boldsymbol{y} \in \mathbb{Y}. \tag{16}$$

Then given any $(\bar{\boldsymbol{x}}, \bar{\boldsymbol{y}})$ and $(\hat{\boldsymbol{x}}, \hat{\boldsymbol{y}})$ that solve Problems (14) and (15), respectively, and $(\boldsymbol{x}^*, \boldsymbol{y}^*)$ that solves Problem (1), we have $l^{(T)}(\hat{\boldsymbol{x}}, \hat{\boldsymbol{y}}) \leq l(\boldsymbol{x}^*, \boldsymbol{y}^*) \leq l^{(T)}(\bar{\boldsymbol{x}}, \bar{\boldsymbol{y}})$.

Proof. See Appendix A.2 for the proof. □

The following assumptions are needed in our analysis.

Assumption 5.2. The set $\mathbb{X}$ is closed and convex; $l(\boldsymbol{x},\boldsymbol{y})$ is twice continuously differentiable; we have $\|\nabla_{\boldsymbol{x}} l(\boldsymbol{x},\boldsymbol{y})\|_2 \leq G_x$, $\|\nabla_{\boldsymbol{y}} l(\boldsymbol{x},\boldsymbol{y})\|_2 \leq G_y$, $\|\nabla_{\boldsymbol{xy}} l(\boldsymbol{x},\boldsymbol{y})\|_2 \leq G_{xy}$, and $\|\nabla_{\boldsymbol{yy}} l(\boldsymbol{x},\boldsymbol{y})\|_2 \leq G_{yy}$ for all $\boldsymbol{x} \in \mathbb{X}$ and $\boldsymbol{y} \in \mathbb{Y}$.

Assumption 5.3. The set $\mathbb{Y}$ is closed and convex; $f(\boldsymbol{x},\boldsymbol{y})$ is continuously differentiable; there exists $\gamma > 0$ such that $f(\boldsymbol{x},\cdot)$ is γ-strongly monotone for all $\boldsymbol{x} \in \mathbb{X}$; we have $\|\nabla_{\boldsymbol{y}} f(\boldsymbol{x},\boldsymbol{y})\|_2 \leq L_y$ and $\|\nabla_{\boldsymbol{x}} f(\boldsymbol{x},\boldsymbol{y})\|_2 \leq L_x$ for all $\boldsymbol{x} \in \mathbb{X}$ and $\boldsymbol{y} \in \mathbb{Y}$.

Assumption 5.4. The function $h(\boldsymbol{x},\boldsymbol{y})$ is twice continuously differentiable; the intrinsic parameter $r < 2\gamma/L_y^2$; we have $\|\nabla_{\boldsymbol{x}} h(\boldsymbol{x},\boldsymbol{y})\|_2 \leq H_x$, $\|\nabla_{\boldsymbol{xy}} h(\boldsymbol{x},\boldsymbol{y})\|_2 \leq H_{xy}$, and $\|\nabla_{\boldsymbol{yy}} h(\boldsymbol{x},\boldsymbol{y})\|_2 \leq H_{yy}$ for all $\boldsymbol{x} \in \mathbb{X}$ and $\boldsymbol{y} \in \mathbb{Y}$.

Assumptions 5.2 and 5.3 are similar to those adopted by Ghadimi & Wang (2018) and Hong et al. (2020) with one notable difference: they require strong convexity in the lower-level optimization problem whereas we require strong monotonicity in the VI. According to Hiriart-Urruty (1982), $h(\boldsymbol{x},\boldsymbol{y})$ is differentiable as long as the boundary of $\mathbb{Y}$ is smooth; without this condition, $h(\boldsymbol{x},\boldsymbol{y})$ is still *almost everywhere* differentiable (Rademacher, 1919). Under Assumptions 5.3 and 5.4, Proposition 3.2 ensures Condition (16) be satisfied. Also, according to Proposition 3.3, a differentiable implicit function $y^*(\boldsymbol{x})$ that maps $\boldsymbol{x}$ to $\mathbb{Y}^*(\boldsymbol{x})$ can be defined. The objective function of Problem (1) then can be rewritten as $l^*(\boldsymbol{x}) = l(\boldsymbol{x}, y^*(\boldsymbol{x}))$. Under these assumptions, Example 2.1 gives the VI formulation of the two models.

Proposition 5.5. *Suppose that $(\bar{\boldsymbol{x}},\bar{\boldsymbol{y}})$ solves the T-step Cournot game (14), then for all $(\boldsymbol{x},\boldsymbol{y}) \in \mathbb{X}\times\mathbb{Y}$, we have*

$$\langle \nabla_{\boldsymbol{x}} l^{(T)}(\bar{\boldsymbol{x}},\bar{\boldsymbol{y}}), \boldsymbol{x}-\bar{\boldsymbol{x}}\rangle + \langle f(\bar{\boldsymbol{x}},\bar{\boldsymbol{y}}), \boldsymbol{y}-\bar{\boldsymbol{y}}\rangle \geq 0. \quad (17)$$

The converse also holds if $l^{(T)}(\boldsymbol{x},\boldsymbol{y})$ is convex in $\boldsymbol{x}$.

Proposition 5.6. *Suppose that $(\hat{\boldsymbol{x}},\hat{\boldsymbol{y}})$ solves the T-step monopoly model (15), then for all $(\boldsymbol{x},\boldsymbol{y}) \in \mathbb{X}\times\mathbb{Y}$, we have*

$$\langle \nabla_{\boldsymbol{x}} l^{(T)}(\hat{\boldsymbol{x}},\hat{\boldsymbol{y}}), \boldsymbol{x}-\hat{\boldsymbol{x}}\rangle + \langle \nabla_{\boldsymbol{y}} l(\hat{\boldsymbol{x}},\hat{\boldsymbol{y}}), \boldsymbol{y}-\hat{\boldsymbol{y}}\rangle \geq 0. \quad (18)$$

The converse also holds if $l^{(T)}(\boldsymbol{x},\boldsymbol{y})$ is convex in $(\boldsymbol{x},\boldsymbol{y})$.

Next, we will confirm T-step Cournot games and monopoly models respectively provide an upper and a lower bound to Problem (1). More importantly, we will prove that the gap between the bounds provided by the two models converges to 0 at a fast rate as T increases, which implies that a small T would make a good approximation.

Theorem 5.7. *Under Assumptions 5.2-5.4, suppose that $l^*(\boldsymbol{x})$ is μ-strongly convex on $\mathbb{X}$ and denote $\boldsymbol{x}^*$ as the unique minimizer of $l^*(\boldsymbol{x})$ on $\mathbb{X}$. We write $\eta = 1 - 2\gamma r + r^2 L_y^2$, $G_l = G_x + L_x G_y$ and, without loss of generality, assume $\mu = 1$ and $\gamma = 1$. For any $(\bar{\boldsymbol{x}},\bar{\boldsymbol{y}})$ that solves Problem (14), denoting $\bar{\delta}^T = l^{(T)}(\bar{\boldsymbol{x}},\bar{\boldsymbol{y}}) - l^*(\boldsymbol{x}^*)$, we then have*

$$0 \leq \bar{\delta}^T \leq G_l \cdot G_y L_x \cdot \eta^{T/2}. \quad (19)$$

Further assuming $d_{\max} = \operatorname{diam}(\mathbb{Y}) < \infty$, given any $(\hat{\boldsymbol{x}},\hat{\boldsymbol{y}})$ that solves the Problem (15), denoting $\hat{\delta}^T = l^*(\boldsymbol{x}^*) - l^{(T)}(\hat{\boldsymbol{x}},\hat{\boldsymbol{y}})$, we then have

$$\begin{aligned} 0 \leq \hat{\delta}^T &\leq M \cdot G_l \cdot \eta^{T/2} \\ &+ G_y d_{\max} \cdot (1 + H_{xy} + L_x H_{yy}) \cdot (T+1) \cdot \eta^{T/2}, \end{aligned} \quad (20)$$

where $M = 2G_y L_x + G_{xy} d_{\max} + 2L_x G_{yy} d_{\max} + G_y H_x$.

Proof. Proposition 5.1 directly guarantees $\bar{\delta}^T \geq 0$ and $\hat{\delta}^T \geq 0$; the remaining proof is given in Appendix A.2. □

Based on Theorem 5.7, we may develop the following simple procedure to obtain an approximated feasible solution to Problem (1). (1) Select an appropriate $T \in \mathbb{N}$. (2) Solve the two VI problems (17) and (18) to obtain $(\bar{\boldsymbol{x}},\bar{\boldsymbol{y}})$ and $(\bar{\boldsymbol{x}},\bar{\boldsymbol{y}})$, respectively, and then calculate $\delta^T = l^{(T)}(\bar{\boldsymbol{x}},\bar{\boldsymbol{y}}) - l^{(T)}(\hat{\boldsymbol{x}},\hat{\boldsymbol{y}})$. (3) If δ^T is small enough, accept $(\bar{\boldsymbol{x}},\bar{\boldsymbol{y}})$ as an approximated feasible solution; otherwise, increase T and return to Step (2). We expect the above procedure to converge quickly, as Theorem 5.7 guarantees the gap decreases at an exponential rate as T increases.

5.2. Solution algorithms

The two VI problems, (17) and (18), can be solved using the projection method, see Algorithms 1 and 2 respectively, for the pseudocodes.

Algorithm 1 Solving T-step Cournot game. Input: $\boldsymbol{x}^0 \in \mathbb{X}$, $\boldsymbol{y}^0 \in \mathbb{Y}$, step size α. Output: $\bar{\boldsymbol{x}} \in \mathbb{X}$ and $\bar{\boldsymbol{y}} \in \mathbb{Y}$.

1: **for** $t = 0, 1, \ldots$ **do**
2: Calculate $l_{\boldsymbol{x}} = \nabla_{\boldsymbol{x}} l^{(T)}(\boldsymbol{x}^t, \boldsymbol{y}^t)$.
3: Set $\boldsymbol{x}^{t+1} = \arg\min_{\boldsymbol{x}\in\mathbb{X}} \alpha \cdot \langle l_{\boldsymbol{x}}, \boldsymbol{x} - \boldsymbol{x}^t\rangle + \|\boldsymbol{x} - \boldsymbol{x}^t\|_2$.
4: Set $\boldsymbol{y}^{t+1} = h^{(T)}(\boldsymbol{x}^t, \boldsymbol{y}^t)$.
5: After convergence, break and return $(\bar{\boldsymbol{x}},\bar{\boldsymbol{y}}) = (\boldsymbol{x}^t, \boldsymbol{y}^t)$.
6: **end for**

Algorithm 2 Solving T-step monopoly model. Input: $\boldsymbol{x}^0 \in \mathbb{X}$, $\boldsymbol{y}^0 \in \mathbb{Y}$, step sizes α and β. Output: $\hat{\boldsymbol{x}} \in \mathbb{X}$ and $\hat{\boldsymbol{y}} \in \mathbb{Y}$.

1: **for** $t = 0, 1, \ldots$ **do**
2: Calculate $l_{\boldsymbol{x}} = \nabla_{\boldsymbol{x}} l^{(T)}(\boldsymbol{x}^t, \boldsymbol{y}^t)$ and $l_{\boldsymbol{y}} = \nabla_{\boldsymbol{y}} l^{(T)}(\boldsymbol{x}^t, \boldsymbol{y}^t)$.
3: Set $\boldsymbol{x}^{t+1} = \arg\min_{\boldsymbol{x}\in\mathbb{X}} \alpha \cdot \langle l_{\boldsymbol{x}}, \boldsymbol{x} - \boldsymbol{x}^t\rangle + \|\boldsymbol{x} - \boldsymbol{x}^t\|_2$.
4: Set $\boldsymbol{y}^{t+1} = \arg\min_{\boldsymbol{y}\in\mathbb{Y}} \beta \cdot \langle l_{\boldsymbol{y}}, \boldsymbol{y} - \boldsymbol{y}^t\rangle + \|\boldsymbol{y} - \boldsymbol{y}^t\|_2$.
5: After convergence, break and return $(\hat{\boldsymbol{x}},\hat{\boldsymbol{y}}) = (\boldsymbol{x}^t, \boldsymbol{y}^t)$.
6: **end for**

In the two algorithms, the leader and the followers evolve together, meaning they each update strategies simultaneously in every step of a shared evolution process. Hence, they break the bilevel hierarchy and turn the solution process into a single loop. At each iteration of that single loop, we need

to calculate the gradients of $l^{(T)}(\boldsymbol{x}, \boldsymbol{y}) = l(\boldsymbol{x}, h^{(T)}(\boldsymbol{x}, \boldsymbol{y}))$, which has an explicit expression. Hence, we may directly calculate its gradient via AD. To this end, we first need to feed the function $h(\boldsymbol{x}, \boldsymbol{y})$ as a differentiable layer into differentiable-programming frameworks, e.g., TensorFlow (Abadi, Agarwal, Barham, Brevdo, Chen, Citro, Corrado, Davis, Dean, Devin, et al., 2016) or PyTorch (Paszke, Gross, Massa, Lerer, Bradbury, Chanan, Killeen, Lin, Gimelshein, Antiga, et al., 2019). As discussed earlier, using the package `cvxpylayers` to code $h(\boldsymbol{x}, \boldsymbol{y})$ is always an option. In many cases, however, $h(\boldsymbol{x}, \boldsymbol{y})$ can be directly coded as a differentiable program built by elementary operations only (no differentiable optimization solver is involved). In such cases, AD guarantees the complexity of calculating the gradient of $l^{(T)}(\boldsymbol{x}, \boldsymbol{y})$ can be tightly bounded by that of calculating $l^{(T)}(\boldsymbol{x}, \boldsymbol{y})$ itself. We refer the readers to Appendix B.1 for more details.

Summary. To summarize, our main result is that Problem (1) can be tightly and efficiently bounded by two easier problems, which can be solved by AD-based first-order methods. The proposed approximation scheme thus promises to significantly reduce the computational overhead for obtaining high-quality local solutions to Problem (1).

Comparison. The proposed framework bridges two general classes of schemes for approximating bilevel programs proposed under different contexts. The first class includes several classic heuristics for solving Stackelberg congestion games (SCGs) (see Section 2.2) dated back to the 1970s. For example, Dantzig, Harvey, Lansdowne, Robinson, and Maier (1979) proposed to solve an SCG by assuming the followers work cooperatively with the leader to achieve the system-optimal state, which essentially turns the leader into a "dictator". Another is Tan, Gershwin, and Athans (1979)'s algorithm that finds the best response of the leader and followers iteratively while holding each other's decisions as a fixed input so that the leader and the followers are competing "à la Cournot". Our scheme extends the behavior assumptions underlying these two.

The second class is AD-based approximation schemes. The proposed algorithms are similar to one-stage AD (see Section 3.2) that are originally motivated by ML applications, e.g., neural architecture search (Liu et al., 2018). One-stage AD and our algorithm are structurally similar; the difference lies in how the lower-level solution (the followers' decision) is updated. In our scheme, the follower may move T steps according to either their own interest (T-step Cournot) or the leader's mandate (T-step monopoly). The former may be viewed as a natural extension of one-stage AD; the latter, however, represents a novel behavior interpretation. With this interpretation, our scheme yields adjustable upper and lower bounds on the original problem.

By bringing together the two classes of approximation schemes motivated by different applications from different disciplines, our work provides new insights for both. On the one hand, it shows that classical game-theoretic approximations for bilevel programs can be substantially refined with modest computational efforts, using ideas borrowed from the ML community. On the one hand, it provides a theoretical justification to AD-based methods from the lens of game theory.

Extensions. We leave the discussion of several extensions to the appendix. In Appendix B.2, we discuss other choices of the form of $h(\boldsymbol{x}, \boldsymbol{y})$ to formulate two proposed models. In Appendix B.3, we discuss how to search for the global solution efficiently when the global convexity assumption on $l^*(\boldsymbol{x})$ is relaxed. In Appendix B.4, we address the case when the lower-level VI admits multiple solutions by developing a heuristic based on Algorithms 1 and 2.

6. Numerical Examples

6.1. Stackelberg duopoly

To validate our analytical insights, we first test our framework on the Stackelberg duopoly (cf. Example 2.3). We solve the T-step Cournot duopoly (10) and the T-step monopoly (11) via Algorithms 1 and 2, respectively, with $r = 0.4$ and $T = 0, \ldots, 4$. Firm A's profits are then reported in Table 1, which indicates that, as T increases, Firm A's optimal profit generated by either new model quickly converges to that by the Stackelberg duopoly. The decreasing rate also meets the expectation set by Theorem 5.7.

Table 1. Firm A's profit in T-step Cournot duopoly and T-step monopoly models with different Ts.

T	0	1	2	3	4
T-step Cournot	0.111	0.124	0.125	0.125	0.125
T-step monopoly	0.250	0.150	0.130	0.126	0.125

6.2. Bilevel optimization

We then consider a bilevel program of the following form

$$\begin{aligned} \min_{\boldsymbol{a}+\boldsymbol{x}\geq 0} \ & \|\boldsymbol{x}\|_2 + \langle \boldsymbol{D}^\top(\boldsymbol{a}+\boldsymbol{x}+\boldsymbol{b}\cdot\boldsymbol{v}), \boldsymbol{w}\cdot\boldsymbol{y}^*\rangle, \\ \text{s.t. } & x_4 = 0, \quad \boldsymbol{v} = \boldsymbol{D}\boldsymbol{y}, \\ & \boldsymbol{y}^* \in \operatorname*{arg\,min}_{\boldsymbol{y}\geq 0,\ \mathbf{1}^\top\boldsymbol{y}=1} \ \langle \mathbf{1}^\top, (\boldsymbol{a}+\boldsymbol{x})\cdot\boldsymbol{v} + \frac{1}{2}\cdot\boldsymbol{b}\cdot\boldsymbol{v}^2\rangle, \end{aligned} \tag{21}$$

where $\boldsymbol{x} \in \mathbb{R}^4$, $\boldsymbol{y} \in \mathbb{R}^4$, $\boldsymbol{w} = [2, 1.1, 0.9, 0.01]^\top$, $\boldsymbol{b} = [15, 2, 8, 5]^\top$, $\boldsymbol{a} = [2, 3, 4, 5]^\top$, $\boldsymbol{D} = [\boldsymbol{e}_1+\boldsymbol{e}_3, \boldsymbol{e}_2+\boldsymbol{e}_4, \boldsymbol{e}_1+\boldsymbol{e}_4, \boldsymbol{e}_2+\boldsymbol{e}_3]$. Thus, the lower level is a convex but not strongly convex optimization, whose optimal solution set is a non-singleton. We first run Algorithm 1 and 2 for

$T = 0, \ldots, 5$. At each round, $\boldsymbol{x}^0$ is randomly sampled from a Gaussian distribution while $\boldsymbol{y}^0$ is first randomly sampled from a uniform distribution and then re-weighted to fit the constraint. The result (see Figure 1) shows the boxplot of the objective values reached from different initial points. It verifies our conclusions that T-step Cournot games may have multiple solutions with different objective values, while T-step monopoly models always have a unique optimal objective value.

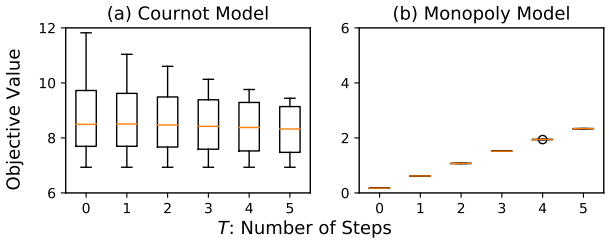

Figure 1. Comparison between Algorithm 1 and 2 starting from different initial points.

We then compare two initialization strategies. (a) Fixed initialization strategy: we follow the procedure devised at the end of Section 5.2; the initial solutions are set as $\boldsymbol{x}^0 = [0, 0, 0, 0]^\top$ and $\boldsymbol{y}^0 = [0.4, 0.3, 0.2, 0.1]$ whenever Algorithms 1 and 2 are invoked. (b) Adaptive initialization strategy: we directly run Algorithm 3 with $\boldsymbol{x}^0 = [0, 0, 0, 0]^\top$ and $\boldsymbol{y}^0 = [0.4, 0.3, 0.2, 0.1]$ as the initial input. When testing both strategies, we gradually increase T as $0, 1, 2, 3, 4, 5, 7, 10, 20, 30, 40, 50, 60, 70$. The result (see Figure 2) shows that when using the fixed-initialization strategy, the gap between upper and lower bounds is still non-eligible when $T = 70$. On the contrary, when using the adaptive-initialization strategy, a near-optimal feasible is found when T is only 1 because the previous monopoly model identifies a neighborhood of the "optimistic" solution. The gap between the two models then gradually decreases to 0, eventually settling down at the exact optimal solution.

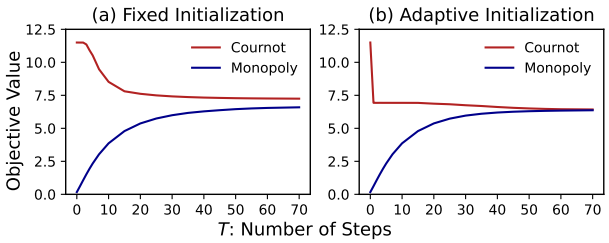

Figure 2. Comparison between fixed- and adaptive-initialization strategies (Algorithm 3).

Additional experiments. Due to space limitations, the results of other experiments are reported in Appendix C. These additional experiments are designed to test the proposed scheme on larger and harder problems that arise from diverse applications and to compare it against benchmark bilevel algorithms.

7. Conclusion

It is well known that non-cooperative games may lead to inefficient outcomes. Caused by the lack of cooperation between self-interested agents, this loss of efficiency — also known as the price of anarchy — is best illustrated by the Braess paradox (Braess, 1968) in transportation. Nevertheless, Roughgarden & Tardos (2002) proves that the total travel time experienced by travelers at *user equilibrium* (UE) is tightly bounded from the above by that achieved when they are fully cooperative, a state called system optimum (SO). Evidently, the UE state corresponds to the outcome of a *Cournot game* because everyone competes equally, whereas the SO state can be brought about only if the choices of all travelers are *monopolized*. The finding in this paper indicates that the gap between "Cournot" and "monopolized" states can not only be bounded but also be *narrowed* by simultaneously giving the leader a limited ability of anticipation in the Cournot game and the followers limited "freedom" to pursue their own interests in the monopoly model. Moreover, given the right conditions, they both can converge to the outcome of a Stackelberg game, where a compromise is arranged: the leader cannot dictate the followers' choice but can influence it indirectly; the followers enjoy the freedom of choice but must heed the leader's guidance.

The models proposed as approximations of the Stackelberg game (bilevel program) are solved by AD-based methods. This connects our work to many bilevel programming algorithms developed in the ML literature (Liu et al., 2021). Our theoretical results help answer a question that has been extensively debated recently: when and why can AD-based approximation schemes deliver satisfactory solutions? Our work contributes to this debate by revealing that — besides the power of AD itself (Ablin et al., 2020) — the underlying game-theoretic structure plays an important role in shaping the power of these methods. We hope this finding will inspire more interdisciplinary works across the domains that count bilevel programming in their toolbox.

Acknowledgements

This research is funded by US National Science Foundation's Civil Infrastructure System (CIS) Program under the award CMMI #2225087 and Energy, Power, Control, and Networks (EPCN) Program under the award ECCS #2048075. The authors are grateful for the data offered by Ms. Qianni Wang at Northwestern University.

References

Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., Corrado, G. S., Davis, A., Dean, J., Devin, M., et al. Tensorflow: Large-scale machine learning on heterogeneous distributed systems. *arXiv preprint arXiv:1603.04467*, 2016.

Ablin, P., Peyré, G., and Moreau, T. Super-efficiency of automatic differentiation for functions defined as a minimum. In *International Conference on Machine Learning*, pp. 32–41. PMLR, 2020.

Agrawal, A., Amos, B., Barratt, S., Boyd, S., Diamond, S., and Kolter, J. Z. Differentiable convex optimization layers. In *Advances in Neural Information Processing Systems*, pp. 9558–9570, 2019.

Aiyoshi, E. and Shimizu, K. A solution method for the static constrained Stackelberg problem via penalty method. *IEEE Transactions on Automatic Control*, 29(12):1111–1114, 1984.

Bae, J. and Grosse, R. B. Delta-stn: Efficient bilevel optimization for neural networks using structured response Jacobians. *Advances in Neural Information Processing Systems*, 33:21725–21737, 2020.

Bahrami, S. and Roorda, M. J. Optimal traffic management policies for mixed human and automated traffic flows. *Transportation Research Part A: Policy and Practice*, 135:130–143, 2020.

Bard, J. F. and Falk, J. E. An explicit solution to the multilevel programming problem. *Computers & Operations Research*, 9(1):77–100, 1982.

Beck, A. and Teboulle, M. Mirror descent and nonlinear projected subgradient methods for convex optimization. *Operations Research Letters*, 31(3):167–175, 2003.

Ben-Ayed, O. and Blair, C. E. Computational difficulties of bilevel linear programming. *Operations Research*, 38(3): 556–560, 1990.

Bernhard, P. ESS, population games, replicator dynamics: dynamics and games if not dynamic games. In *Advances in Dynamic Games*, pp. 291–311. Springer, 2011.

Bertrand, Q., Klopfenstein, Q., Blondel, M., Vaiter, S., Gramfort, A., and Salmon, J. Implicit differentiation of lasso-type models for hyperparameter optimization. In *International Conference on Machine Learning*, pp. 810–821. PMLR, 2020.

Bialas, W., Karwan, M., and Shaw, J. A parametric complementary pivot approach for two-level linear programming. *State University of New York at Buffalo*, 57, 1980.

Blondel, M., Berthet, Q., Cuturi, M., Frostig, R., Hoyer, S., Llinares-López, F., Pedregosa, F., and Vert, J.-P. Efficient and modular implicit differentiation. *arXiv preprint arXiv:2105.15183*, 2021.

Boyle, J. P. and Dykstra, R. L. A method for finding projections onto the intersection of convex sets in hilbert spaces. In *Advances in order restricted statistical inference*, pp. 28–47. Springer, 1986.

Bracken, J. and McGill, J. T. Mathematical programs with optimization problems in the constraints. *Operations Research*, 21(1):37–44, 1973.

Braess, D. Über ein paradoxon aus der verkehrsplanung. *Unternehmensforschung*, 12(1):258–268, 1968.

Candler, W. and Townsley, R. A linear two-level programming problem. *Computers & Operations Research*, 9(1): 59–76, 1982.

Chen, T., Sun, Y., and Yin, W. A single-timescale stochastic bilevel optimization method. *arXiv preprint arXiv:2102.04671*, 2021.

Chen, Z., He, F., Zhang, L., and Yin, Y. Optimal deployment of autonomous vehicle lanes with endogenous market penetration. *Transportation Research Part C: Emerging Technologies*, 72:143–156, 2016.

Chen, Z., He, F., Yin, Y., and Du, Y. Optimal design of autonomous vehicle zones in transportation networks. *Transportation Research Part B: Methodological*, 99:44–61, 2017.

Colson, B., Marcotte, P., and Savard, G. An overview of bilevel optimization. *Annals of operations research*, 153 (1):235–256, 2007.

Dafermos, S. An iterative scheme for variational inequalities. *Mathematical Programming*, 26(1):40–47, 1983.

Dafermos, S. Sensitivity analysis in variational inequalities. *Mathematics of Operations Research*, 13(3):421–434, 1988.

Dafermos, S. C. Toll patterns for multiclass-user transportation networks. *Transportation science*, 7(3):211–223, 1973.

Dagréou, M., Ablin, P., Vaiter, S., and Moreau, T. A framework for bilevel optimization that enables stochastic and global variance reduction algorithms. *arXiv preprint arXiv:2201.13409*, 2022.

Dantzig, G. B., Harvey, R. P., Lansdowne, Z. F., Robinson, D. W., and Maier, S. F. Formulating and solving the network design problem by decomposition. *Transportation Research Part B: Methodological*, 13(1):5–17, 1979.

Ehtamo, H., Kitti, M., and Hämäläinen, R. P. Recent studies on incentive design problems in game theory and management science. In *Optimal Control and Differential Games*, pp. 121–134. Springer, 2002.

Finn, C., Abbeel, P., and Levine, S. Model-agnostic meta-learning for fast adaptation of deep networks. In *International Conference on Machine Learning*, pp. 1126–1135. PMLR, 2017.

Franceschi, L., Donini, M., Frasconi, P., and Pontil, M. Forward and reverse gradient-based hyperparameter optimization. In *International Conference on Machine Learning*, pp. 1165–1173. PMLR, 2017.

Franceschi, L., Frasconi, P., Salzo, S., Grazzi, R., and Pontil, M. Bilevel programming for hyperparameter optimization and meta-learning. In *International Conference on Machine Learning*, pp. 1568–1577. PMLR, 2018.

Friesz, T. L., Tobin, R. L., Cho, H.-J., and Mehta, N. J. Sensitivity analysis based heuristic algorithms for mathematical programs with variational inequality constraints. *Mathematical Programming*, 48(1-3):265–284, 1990.

Fung, S. W., Heaton, H., Li, Q., McKenzie, D., Osher, S., and Yin, W. Jfb: Jacobian-free backpropagation for implicit networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2022.

Ghadimi, S. and Wang, M. Approximation methods for bilevel programming. *arXiv preprint arXiv:1802.02246*, 2018.

Grazzi, R., Franceschi, L., Pontil, M., and Salzo, S. On the iteration complexity of hypergradient computation. In *International Conference on Machine Learning*, pp. 3748–3758. PMLR, 2020.

Grazzi, R., Pontil, M., and Salzo, S. Convergence properties of stochastic hypergradients. In *International Conference on Artificial Intelligence and Statistics*, pp. 3826–3834. PMLR, 2021.

Griewank, A. et al. On automatic differentiation. *Mathematical Programming: recent developments and applications*, 6(6):83–107, 1989.

Guo, Z., Hu, Q., Zhang, L., and Yang, T. Randomized stochastic variance-reduced methods for multi-task stochastic bilevel optimization. *arXiv preprint arXiv:2105.02266*, 2021.

Hartman, P., Stampacchia, G., et al. On some non-linear elliptic differential-functional equations. *Acta mathematica*, 115:271–310, 1966.

Hiriart-Urruty, J.-B. At what points is the projection mapping differentiable? *The American Mathematical Monthly*, 89(7):456–458, 1982.

Hong, M., Wai, H.-T., Wang, Z., and Yang, Z. A two-timescale framework for bilevel optimization: Complexity analysis and application to actor-critic. *arXiv preprint arXiv:2007.05170*, 2020.

Ji, K., Yang, J., and Liang, Y. Bilevel optimization: Convergence analysis and enhanced design. In *International Conference on Machine Learning*, pp. 4882–4892. PMLR, 2021.

Labbé, M., Marcotte, P., and Savard, G. A bilevel model of taxation and its application to optimal highway pricing. *Management science*, 44(12-part-1):1608–1622, 1998.

Lawphongpanich, S. and Hearn, D. W. An mpec approach to second-best toll pricing. *Mathematical programming*, 101(1):33–55, 2004.

LeBlanc, L. J. An algorithm for the discrete network design problem. *Transportation Science*, 9(3):183–199, 1975.

Levin, M. W. and Boyles, S. D. A cell transmission model for dynamic lane reversal with autonomous vehicles. *Transportation Research Part C: Emerging Technologies*, 68:126–143, 2016.

Li, C., Yang, H., Zhu, D., and Meng, Q. A global optimization method for continuous network design problems. *Transportation Research Part B: Methodological*, 46(9): 1144–1158, 2012.

Li, J., Yu, J., Nie, Y. M., and Wang, Z. End-to-end learning and intervention in games. *Advances in Neural Information Processing Systems*, 33, 2020.

Li, J., Gu, B., and Huang, H. A fully single loop algorithm for bilevel optimization without Hessian inverse. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2022a.

Li, J., Yu, J., Wang, Q., Liu, B., Wang, Z., and Nie, Y. M. Differentiable bilevel programming for Stackelberg congestion games. *arXiv preprint arXiv:2209.07618*, 2022b.

Li, W., Kockelman, K. M., and Huang, Y. Traffic and welfare impacts of credit-based congestion pricing applications: An Austin case study. *Transportation Research Record*, 2675(1):10–24, 2021.

Liao, R., Xiong, Y., Fetaya, E., Zhang, L., Yoon, K., Pitkow, X., Urtasun, R., and Zemel, R. Reviving and improving recurrent back-propagation. In *International Conference on Machine Learning*, pp. 3082–3091. PMLR, 2018.

Liu, B., Li, J., Yang, Z., Wai, H. T., Hong, M., Nie, Y., and Wang, Z. Inducing equilibria via incentives: Simultaneous design-and-play ensures global convergence. In *Advances in Neural Information Processing Systems*, 2022.

Liu, H., Simonyan, K., and Yang, Y. Darts: Differentiable architecture search. *arXiv preprint arXiv:1806.09055*, 2018.

Liu, R., Gao, J., Zhang, J., Meng, D., and Lin, Z. Investigating bi-level optimization for learning and vision from a unified perspective: A survey and beyond. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2021.

Loridan, P. and Morgan, J. Weak via strong Stackelberg problem: new results. *Journal of global Optimization*, 8 (3):263–287, 1996.

Lorraine, J. and Duvenaud, D. Stochastic hyperparameter optimization through hypernetworks. *arXiv preprint arXiv:1802.09419*, 2018.

Luketina, J., Berglund, M., Greff, K., and Raiko, T. Scalable gradient-based tuning of continuous regularization hyperparameters. In *International conference on machine learning*, pp. 2952–2960. PMLR, 2016.

Luo, Z.-Q., Pang, J.-S., and Ralph, D. *Mathematical programs with equilibrium constraints*. Cambridge University Press, 1996.

MacKay, M., Vicol, P., Lorraine, J., Duvenaud, D., and Grosse, R. Self-tuning networks: Bilevel optimization of hyperparameters using structured best-response functions. *arXiv preprint arXiv:1903.03088*, 2019.

Maclaurin, D., Duvenaud, D., and Adams, R. Gradient-based hyperparameter optimization through reversible learning. In *International conference on machine learning*, pp. 2113–2122. PMLR, 2015.

Maheshwari, C., Kulkarni, K., Wu, M., and Sastry, S. S. Inducing social optimality in games via adaptive incentive design. In *2022 IEEE 61st Conference on Decision and Control (CDC)*, pp. 2864–2869. IEEE, 2022.

Mahmassani, H. S. 50th anniversary invited article—autonomous vehicles and connected vehicle systems: Flow and operations considerations. *Transportation Science*, 50(4):1140–1162, 2016.

Mancino, O. and Stampacchia, G. Convex programming and variational inequalities. *Journal of Optimization Theory and Applications*, 9(1):3–23, 1972.

Marcotte, P. Network design problem with congestion effects: A case of bilevel programming. *Mathematical programming*, 34(2):142–162, 1986.

Marcotte, P. and Marquis, G. Efficient implementation of heuristics for the continuous network design problem. *Annals of Operations Research*, 34(1):163–176, 1992.

Marcotte, P. and Wu, J. H. On the convergence of projection methods: application to the decomposition of affine variational inequalities. *Journal of Optimization Theory and Applications*, 85(2):347–362, 1995.

Mertikopoulos, P. and Zhou, Z. Learning in games with continuous action sets and unknown payoff functions. *Mathematical Programming*, 173(1-2):465–507, 2019.

Metz, L., Poole, B., Pfau, D., and Sohl-Dickstein, J. Unrolled generative adversarial networks. *arXiv preprint arXiv:1611.02163*, 2016.

Mguni, D., Jennings, J., Sison, E., Valcarcel Macua, S., Ceppi, S., and Munoz de Cote, E. Coordinating the crowd: Inducing desirable equilibria in non-cooperative systems. In *Proceedings of the 18th International Conference on Autonomous Agents and MultiAgent Systems*, pp. 386–394, 2019.

Myerson, R. B. Optimal coordination mechanisms in generalized principal–agent problems. *Journal of mathematical economics*, 10(1):67–81, 1982.

Nagurney, A. *Network economics: A variational inequality approach*, volume 10. Springer Science & Business Media, 2013.

Nash, J. Non-cooperative games. *Annals of mathematics*, pp. 286–295, 1951.

Nemirovskij, A. S. and Yudin, D. B. *Problem Complexity and Method Efficiency in Optimization*. A Wiley-Interscience publication. Wiley, 1983.

Outrata, J., Kocvara, M., Zowe, J., and Zowe, J. *Nonsmooth Approach to Optimization Problems with Equilibrium Constraints: Theory, Applications and Numerical Results*, volume 28. Springer Science & Business Media, 1998.

Pang, J.-S. and Chan, D. Iterative methods for variational and complementarity problems. *Mathematical programming*, 24(1):284–313, 1982.

Parise, F. and Ozdaglar, A. A variational inequality framework for network games: Existence, uniqueness, convergence and sensitivity analysis. *Games and Economic Behavior*, 114:47–82, 2019.

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., et al. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32, 2019.

Pedregosa, F. Hyperparameter optimization with approximate gradient. In *International conference on machine learning*, pp. 737–746. PMLR, 2016.

Pigou, A. C. *The economics of welfare*. Palgrave Macmillan, 1920.

Rademacher, H. Über partielle und totale differenzierbarkeit von funktionen mehrerer variabeln und über die transformation der doppelintegrale. *Mathematische Annalen*, 79 (4):340–359, 1919.

Requate, T. Pollution control in a Cournot duopoly via taxes or permits. *Journal of Economics*, 58(3):255–291, 1993.

Roughgarden, T. and Tardos, É. How bad is selfish routing? *Journal of the ACM*, 49(2):236–259, 2002.

Schmeidler, D. Equilibrium points of nonatomic games. *Journal of statistical Physics*, 7(4):295–300, 1973.

Scutari, G., Palomar, D. P., Facchinei, F., and Pang, J.-S. Convex optimization, game theory, and variational inequality theory. *IEEE Signal Processing Magazine*, 27 (3):35–49, 2010.

Shaban, A., Cheng, C.-A., Hatch, N., and Boots, B. Truncated back-propagation for bilevel optimization. In *The 22nd International Conference on Artificial Intelligence and Statistics*, pp. 1723–1732. PMLR, 2019.

Shapiro, C. Theories of oligopoly behavior. *Handbook of industrial organization*, 1:329–414, 1989.

Sinha, A., Fang, F., An, B., Kiekintveld, C., and Tambe, M. Stackelberg security games: Looking beyond a decade of success. In *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI-18*, pp. 5494–5501. International Joint Conferences on Artificial Intelligence Organization, 2018.

Sow, D., Ji, K., and Liang, Y. On the convergence theory for Hessian-free bilevel algorithms. In *Advances in Neural Information Processing Systems*, 2022.

Tan, H.-N., Gershwin, S. B., and Athans, M. Hybrid optimization in urban traffic networks. Technical report, Massachusetts Institute of Technology, 1979.

Tobin, R. L. Sensitivity analysis for variational inequalities. *Journal of Optimization Theory and Applications*, 48(1): 191–204, 1986.

Vicol, P., Lorraine, J., Duvenaud, D., and Grosse, R. Implicit regularization in overparameterized bilevel optimization. In *ICML 2021 Beyond First Order Methods Workshop*, 2021.

von Stackelberg, H. *The theory of the market economy*. Oxford University Press, 1952.

Wardrop, J. G. Some theoretical aspects of road traffic research. In *Proceedings of the Institution of Civil Engineers*, volume 1, pp. 325–362, 1952.

Weibull, J. W. *Evolutionary game theory*. MIT press, 1997.

Wu, Y., Ren, M., Liao, R., and Grosse, R. Understanding short-horizon bias in stochastic meta-optimization. *arXiv preprint arXiv:1803.02021*, 2018.

Xu, Y., Xie, L., Zhang, X., Chen, X., Qi, G.-J., Tian, Q., and Xiong, H. Pc-darts: Partial channel connections for memory-efficient architecture search. *arXiv preprint arXiv:1907.05737*, 2019.

Yang, J., Ji, K., and Liang, Y. Provably faster algorithms for bilevel optimization. *Advances in Neural Information Processing Systems*, 34:13670–13682, 2021.

Zhang, K. and Nie, Y. M. Mitigating the impact of selfish routing: An optimal-ratio control scheme (orcs) inspired by autonomous driving. *Transportation Research Part C: Emerging Technologies*, 87:75–90, 2018.

Zheng, S., Trott, A., Srinivasa, S., Naik, N., Gruesbeck, M., Parkes, D. C., and Socher, R. The AI economist: Improving equality and productivity with ai-driven tax policies. *arXiv preprint arXiv:2004.13332*, 2020.

Zucchet, N. and Sacramento, J. Beyond backpropagation: implicit gradients for bilevel optimization. *arXiv preprint arXiv:2205.03076*, 2022.

A. Proofs in Section 5

A.1. Proof of Proposition 5.1

Proof of Proposition 5.1. First, $(\bar{\boldsymbol{x}}, \bar{\boldsymbol{y}})$ is a feasible solution to Problem (1) since $(\bar{\boldsymbol{x}}, \bar{\boldsymbol{y}}) \in \{(\boldsymbol{x}, \boldsymbol{y}) : \boldsymbol{x} \in \mathbb{X}, \boldsymbol{y} \in \mathbb{Y}^*(\boldsymbol{x})\}$. Thus, we can directly conclude $l(\boldsymbol{x}^*, \boldsymbol{y}^*) \leq l(\bar{\boldsymbol{x}}, \bar{\boldsymbol{y}}) = l^{(T)}(\bar{\boldsymbol{x}}, \bar{\boldsymbol{y}})$. Next, as $l(\boldsymbol{x}, \boldsymbol{y}) = l(\boldsymbol{x}, h^{(T)}(\boldsymbol{x}, \boldsymbol{y})) = l^{(T)}(\boldsymbol{x}, \boldsymbol{y})$ for all $(\boldsymbol{x}, \boldsymbol{y}) \in \{(\boldsymbol{x}, \boldsymbol{y}) : \boldsymbol{x} \in \mathbb{X}, \boldsymbol{y} \in \mathbb{Y}^*(\boldsymbol{x})\}$, we claim $(\boldsymbol{x}^*, \boldsymbol{y}^*)$ also solves the following optimization

$$\boldsymbol{x}^*, \boldsymbol{y}^* \in \underset{\boldsymbol{x} \in \mathbb{X},\, \boldsymbol{y} \in \mathbb{Y}^*(\boldsymbol{x}).}{\arg\min} \quad l^{(T)}(\boldsymbol{x}, \boldsymbol{y}). \tag{22}$$

Eventually, $\{(\boldsymbol{x}, \boldsymbol{y}) : \boldsymbol{x} \in \mathbb{X}, \boldsymbol{y} \in \mathbb{Y}^*(\boldsymbol{x})\} \subseteq \mathbb{X} \times \mathbb{Y}$ implies that $l^{(T)}(\hat{\boldsymbol{x}}, \hat{\boldsymbol{y}}) \leq l^{(T)}(\boldsymbol{x}^*, \boldsymbol{y}^*) = l(\boldsymbol{x}^*, \boldsymbol{y}^*)$. □

A.2. Proof of Theorem 5.7

We first provide the following lemmas.

Lemma A.1. *If $l^*(\boldsymbol{x}) = l(\boldsymbol{x}, y^*(\boldsymbol{x}))$ is μ-strongly convex on $\boldsymbol{x} \in \mathbb{X}$, then we have*

$$\mu \cdot \|\boldsymbol{x}^* - \boldsymbol{x}\|_2^2 \leq \langle \nabla l^*(\boldsymbol{x}), \boldsymbol{x} - \boldsymbol{x}^* \rangle, \quad \forall \boldsymbol{x} \in \mathbb{X}. \tag{23}$$

Proof. As $l^*(\boldsymbol{x})$ is μ-strongly convex, we have

$$\begin{cases} l^*(\boldsymbol{x}^*) \geq l^*(\boldsymbol{x}) + \langle \nabla l^*(\boldsymbol{x}), \boldsymbol{x}^* - \boldsymbol{x} \rangle + \dfrac{\mu}{2} \cdot \|\boldsymbol{x}^* - \boldsymbol{x}\|_2^2, \\ l^*(\boldsymbol{x}) \geq l^*(\boldsymbol{x}^*) + \langle \nabla l^*(\boldsymbol{x}^*), \boldsymbol{x} - \boldsymbol{x}^* \rangle + \dfrac{\mu}{2} \cdot \|\boldsymbol{x}^* - \boldsymbol{x}\|_2^2. \end{cases} \tag{24}$$

Adding these two equations, we then have

$$0 \geq \langle \nabla l^*(\boldsymbol{x}), \boldsymbol{x}^* - \boldsymbol{x} \rangle + \langle \nabla l^*(\boldsymbol{x}^*), \boldsymbol{x} - \boldsymbol{x}^* \rangle + \mu \cdot \|\boldsymbol{x}^* - \boldsymbol{x}\|_2^2 \geq \langle \nabla l^*(\boldsymbol{x}), \boldsymbol{x}^* - \boldsymbol{x} \rangle + \mu \cdot \|\boldsymbol{x}^* - \boldsymbol{x}\|_2^2, \tag{25}$$

where the second inequality comes from the fact that $\boldsymbol{x}^*$ minimizes $l^*(\boldsymbol{x})$. □

Lemma A.2. *For all $\boldsymbol{x} \in \mathbb{X}$, we have $\|\nabla y^*(\boldsymbol{x})\|_2 \leq L_x/\gamma$.*

Proof. Let $\boldsymbol{x}$ and $\boldsymbol{x}'$ be two arbitrary points in $\mathbb{X}$. Denote $\boldsymbol{y}^* = y^*(\boldsymbol{x})$ and $\boldsymbol{y}'^* = y^*(\boldsymbol{x}')$. Then we have

$$\begin{cases} \langle f(\boldsymbol{x}, \boldsymbol{y}^*), \boldsymbol{y}'^* - \boldsymbol{y}^* \rangle \geq 0, \\ \langle f(\boldsymbol{x}', \boldsymbol{y}'^*), \boldsymbol{y}^* - \boldsymbol{y}'^* \rangle \geq 0. \end{cases} \tag{26}$$

Adding these two inequalities, we then have

$$\langle f(\boldsymbol{x}', \boldsymbol{y}'^*) - f(\boldsymbol{x}, \boldsymbol{y}'^*) + f(\boldsymbol{x}, \boldsymbol{y}'^*) - f(\boldsymbol{x}, \boldsymbol{y}^*), \boldsymbol{y}^* - \boldsymbol{y}'^* \rangle \geq 0. \tag{27}$$

We can further obtain that

$$\langle f(\boldsymbol{x}', \boldsymbol{y}'^*) - f(\boldsymbol{x}, \boldsymbol{y}'^*), \boldsymbol{y}^* - \boldsymbol{y}'^* \rangle \geq \langle f(\boldsymbol{x}, \boldsymbol{y}^*) - f(\boldsymbol{x}, \boldsymbol{y}'^*), \boldsymbol{y}^* - \boldsymbol{y}'^* \rangle \geq \gamma \cdot \|\boldsymbol{y}^* - \boldsymbol{y}'^*\|_2^2, \tag{28}$$

where the first inequality comes from (27) and the second inequality comes from the assumption that f is strongly monotone with respect to $\|\cdot\|_2$. Noting that $f(\cdot, \boldsymbol{y})$ is L_x-Lipschitz continuous and using the Cauchy-Schwartz inequality, we eventually have

$$\|\boldsymbol{y}^* - \boldsymbol{y}'^*\|_2 \leq \frac{1}{\gamma} \cdot \|f(\boldsymbol{x}', \boldsymbol{y}'^*) - f(\boldsymbol{x}, \boldsymbol{y}'^*)\|_2 \leq \frac{L_x}{\gamma} \cdot \|\boldsymbol{x} - \boldsymbol{x}'\|_2. \tag{29}$$

Letting $\boldsymbol{x}' \to \boldsymbol{x}$, we then conclude the proof. □

Lemma A.3. *The function $l^*(\boldsymbol{x})$ is $(G_x + L_x G_y/\gamma)$-Lipschitz continuous with respect to $\|\cdot\|_2$.*

Proof. Noting that $l(\boldsymbol{x}, \cdot)$, $l(\cdot, \boldsymbol{y})$ and $y^*(\boldsymbol{x})$ are respectively G_y-, G_x- and L_x/γ-Lipschitz continuous, by directly applying the triangle inequality of norms, we can obtain that

$$\begin{aligned}\|\nabla l^*(\boldsymbol{x})\| &= \|\nabla_{\boldsymbol{x}} l(\boldsymbol{x}, y^*(\boldsymbol{x})) + \nabla y^*(\boldsymbol{x}) \cdot \nabla_{\boldsymbol{y}} l(\boldsymbol{x}, y^*(\boldsymbol{x}))\| \\ &\leq \|\nabla_{\boldsymbol{x}} l(\boldsymbol{x}, y^*(\boldsymbol{x}))\| + \|\nabla y^*(\boldsymbol{x})\| \cdot \|\nabla_{\boldsymbol{y}} l(\boldsymbol{x}, y^*(\boldsymbol{x}))\| \leq G_x + \frac{L_x G_y}{\gamma}.\end{aligned} \tag{30}$$

We thus conclude the proof. □

Lemma A.4. *We can directly obtain the following formulas by applying the chain rule*

$$\nabla l^*(\boldsymbol{x}) = \nabla_{\boldsymbol{x}} l(\boldsymbol{x}, y^*(\boldsymbol{x})) + \nabla y^*(\boldsymbol{x}) \cdot \nabla_{\boldsymbol{y}} l(\boldsymbol{x}, y^*(\boldsymbol{x})), \tag{31}$$

$$\nabla_{\boldsymbol{x}} l^{(T)}(\boldsymbol{x}, \boldsymbol{y}) = \nabla_{\boldsymbol{x}} l(\boldsymbol{x}, h^{(T)}(\boldsymbol{x}, \boldsymbol{y})) + \nabla_{\boldsymbol{x}} h^{(T)}(\boldsymbol{x}, \boldsymbol{y}) \cdot \nabla_{\boldsymbol{y}} l(\boldsymbol{x}, h^{(T)}(\boldsymbol{x}, \boldsymbol{y})), \tag{32}$$

$$\nabla_{\boldsymbol{x}} h^{(t)}(\boldsymbol{x}, \boldsymbol{y}) = \nabla_{\boldsymbol{x}} h(\boldsymbol{x}, h^{(t-1)}(\boldsymbol{x}, \boldsymbol{y})) + \nabla_{\boldsymbol{x}} h^{(t-1)}(\boldsymbol{x}, \boldsymbol{y}) \cdot \nabla_{\boldsymbol{y}} h(\boldsymbol{x}, h^{(t-1)}(\boldsymbol{x}, \boldsymbol{y})), \tag{33}$$

$$\nabla_{\boldsymbol{y}} h^{(t)}(\boldsymbol{x}, \boldsymbol{y}) = \nabla_{\boldsymbol{y}} h(\boldsymbol{x}, h^{(t-1)}(\boldsymbol{x}, \boldsymbol{y})) \cdot \nabla_{\boldsymbol{y}} h^{(t-1)}(\boldsymbol{x}, \boldsymbol{y}). \tag{34}$$

Proposition A.5. *For any $\boldsymbol{x} \in \mathbb{X}$ and $\boldsymbol{y}^* = y^*(\boldsymbol{x})$, we have*

$$\|\nabla y^*(\boldsymbol{x}) - \nabla_{\boldsymbol{x}} h^{(T)}(\boldsymbol{x}, \boldsymbol{y}^*)\|_2 \leq \frac{L_x}{\gamma} \cdot \eta^{T/2}. \tag{35}$$

Proof. By reformulating Equation (8) in Proposition 3.2, we have

$$\nabla y^*(\boldsymbol{x}) = \nabla_{\boldsymbol{x}} h(\boldsymbol{x}, \boldsymbol{y}^*) + \nabla y^*(\boldsymbol{x}) \cdot \nabla_{\boldsymbol{y}} h(\boldsymbol{x}, \boldsymbol{y}^*). \tag{36}$$

Based on Equations (33), (34), and (36), we can then iteratively obtain

$$\begin{aligned}\nabla y^*(\boldsymbol{x}) - \nabla_{\boldsymbol{x}} h^{(T)}(\boldsymbol{x}, \boldsymbol{y}^*) &= \left(\nabla y^*(\boldsymbol{x}) - \nabla_{\boldsymbol{x}} h^{(T-1)}(\boldsymbol{x}, \boldsymbol{y}^*)\right) \cdot \nabla_{\boldsymbol{y}} h^{(1)}(\boldsymbol{x}, \boldsymbol{y}^*) \\ &= \left(\nabla y^*(\boldsymbol{x}) - \nabla_{\boldsymbol{x}} h^{(T-2)}(\boldsymbol{x}, \boldsymbol{y}^*)\right) \cdot \nabla_{\boldsymbol{y}} h^{(2)}(\boldsymbol{x}, \boldsymbol{y}^*) \\ &= \cdots = \nabla y^*(\boldsymbol{x}) \cdot \nabla_{\boldsymbol{y}} h^{(T)}(\boldsymbol{x}, \boldsymbol{y}^*).\end{aligned} \tag{37}$$

Using the triangle inequality of norms and Applying Lemma A.2, we then obtain

$$\|\nabla y^*(\boldsymbol{x}) - \nabla_{\boldsymbol{x}} h^{(T)}(\boldsymbol{x}, \boldsymbol{y}^*)\|_2 \leq \|\nabla y^*(\boldsymbol{x})\|_2 \cdot \|\nabla_{\boldsymbol{y}} h^{(T)}(\boldsymbol{x}, \boldsymbol{y}^*)\|_2 \leq \frac{L_x}{\gamma} \cdot \|\nabla_{\boldsymbol{y}} h^{(T)}(\boldsymbol{x}, \boldsymbol{y}^*)\|_2. \tag{38}$$

We eventually conclude the proof by applying Proposition 3.2. □

The remaining proof will be decomposed into two parts, Part I and Part II, in which we will prove the upper bounds given to $\bar{\delta}^T = l^{(T)}(\bar{\boldsymbol{x}}, \bar{\boldsymbol{y}}) - l^*(\boldsymbol{x})$ and $\hat{\delta}^T = l^*(\boldsymbol{x}) - l^{(T)}(\hat{\boldsymbol{x}}, \hat{\boldsymbol{y}})$, respectively.

A.2.1. Part I of the Remaining Proof

By applying Lemma A.3, we can obtain that

$$l(\bar{\boldsymbol{x}}, \bar{\boldsymbol{y}}) - l^*(\boldsymbol{x}^*) = l^*(\bar{\boldsymbol{x}}) - l^*(\boldsymbol{x}^*) \leq \left(G_x + \frac{L_x G_y}{\gamma}\right) \cdot \|\bar{\boldsymbol{x}} - \boldsymbol{x}^*\|_2 \tag{39}$$

Therefore, we only need to bound $\|\bar{\boldsymbol{x}} - \boldsymbol{x}^*\|_2$. Based on Proposition 5.5, we have

$$\langle \nabla_{\boldsymbol{x}} l^{(T)}(\bar{\boldsymbol{x}}, \bar{\boldsymbol{y}}), \boldsymbol{x}^* - \bar{\boldsymbol{x}} \rangle \geq 0. \tag{40}$$

Then we can set $\boldsymbol{x} = \bar{\boldsymbol{x}}$ in Equation (23) and combine it with Equation (40), which gives

$$\mu \cdot \|\boldsymbol{x}^* - \bar{\boldsymbol{x}}\|_2^2 \leq \langle \nabla l^*(\bar{\boldsymbol{x}}), \bar{\boldsymbol{x}} - \boldsymbol{x}^* \rangle + \langle \nabla_{\boldsymbol{x}} l^{(T)}(\bar{\boldsymbol{x}}, \bar{\boldsymbol{y}}), \boldsymbol{x}^* - \bar{\boldsymbol{x}} \rangle = \langle \nabla l^*(\bar{\boldsymbol{x}}) - \nabla_{\boldsymbol{x}} l^{(T)}(\bar{\boldsymbol{x}}, \bar{\boldsymbol{y}}), \bar{\boldsymbol{x}} - \boldsymbol{x}^* \rangle \tag{41}$$

Using the Cauchy-Schwartz inequality, we then obtain

$$\|\boldsymbol{x}^* - \bar{\boldsymbol{x}}\|_2 \leq \frac{1}{\mu} \cdot \|\nabla l^*(\bar{\boldsymbol{x}}) - \nabla_{\boldsymbol{x}} l^{(T)}(\bar{\boldsymbol{x}}, \bar{\boldsymbol{y}})\|_2. \tag{42}$$

Noting that $\bar{\boldsymbol{y}} = y^*(\bar{\boldsymbol{x}}) = h^{(T)}(\bar{\boldsymbol{x}}, \bar{\boldsymbol{y}})$, substituting $\nabla l^*(\bar{\boldsymbol{x}})$ and $\nabla_{\boldsymbol{x}} l^{(T)}(\bar{\boldsymbol{x}}, \bar{\boldsymbol{y}})$ by Equations (31) and (32), and then applying Proposition A.5, we eventually have

$$\|\boldsymbol{x}^* - \bar{\boldsymbol{x}}\|_2 \leq \frac{1}{\mu} \cdot \|\nabla y^*(\bar{\boldsymbol{x}}) - \nabla_{\boldsymbol{x}} h^{(T)}(\bar{\boldsymbol{x}}, \bar{\boldsymbol{y}})\|_2 \cdot \|\nabla_{\boldsymbol{y}} l(\bar{\boldsymbol{x}}, \bar{\boldsymbol{y}})\|_2 \leq \frac{G_y L_x}{\mu\gamma} \cdot \eta^{T/2}. \tag{43}$$

We can hence conclude the proof by combining Equations (39) and (43).

A.2.2. Part II of the Remaining Proof

We need an additional lemma to prove the upper bound given to $\hat{\delta}^T = l^*(\boldsymbol{x}) - l^{(T)}(\hat{\boldsymbol{x}}, \hat{\boldsymbol{y}})$.

Lemma A.6. *For any $\boldsymbol{x} \in \mathbb{X}$ and $\boldsymbol{y} \in \mathbb{Y}$, we have*

$$\|\nabla_{\boldsymbol{x}} h^{(T)}(\boldsymbol{x}, \boldsymbol{y}) - \nabla y^*(\boldsymbol{x})\|_2 \leq \left(\|\nabla_{\boldsymbol{x}} h(\boldsymbol{x}, \boldsymbol{y})\|_2 + \delta^0 T \left(H_{xy} + \frac{L_x}{\gamma} \cdot H_{yy}\right)\right) \eta^{T/2}, \tag{44}$$

where $\delta^0 = \|\boldsymbol{y} - y^*(\boldsymbol{x})\|_2$.

Proof. In the sequel, we write $\boldsymbol{y}^t = h^{(t)}(\boldsymbol{x}, \boldsymbol{y})$, $\boldsymbol{D}^{(t)} = \nabla_{\boldsymbol{x}} h^{(t)}(\boldsymbol{x}, \boldsymbol{y}^0) - \nabla y^*(\boldsymbol{x})$, $\boldsymbol{E}^{(t)} = \nabla_{\boldsymbol{x}} h(\boldsymbol{x}, \boldsymbol{y}^{t-1}) - \nabla_{\boldsymbol{x}} h(\boldsymbol{x}, \boldsymbol{y}^*)$ and $\boldsymbol{F}^{(t)} = \nabla_{\boldsymbol{y}} h(\boldsymbol{x}, \boldsymbol{y}^{t-1}) - \nabla_{\boldsymbol{y}} h(\boldsymbol{x}, \boldsymbol{y}^*)$, $\boldsymbol{H}^{(t)} = \nabla_{\boldsymbol{y}} h^{(t)}(\boldsymbol{x}, \boldsymbol{y}^{T-t})$ and $\boldsymbol{J} = \nabla y^*(\boldsymbol{x})$. By recursively applying (33) and (36), we then have

$$\begin{aligned}
\boldsymbol{D}^{(T)} &= \boldsymbol{D}^{(T-1)} \cdot \boldsymbol{H}^{(1)} + \boldsymbol{E}^{(T)} + \boldsymbol{J} \cdot \boldsymbol{F}^{(T)} \\
&= \boldsymbol{D}^{(T-2)} \cdot \boldsymbol{H}^{(2)} + \boldsymbol{E}^{(T-1)} \cdot \boldsymbol{H}^{(1)} + \boldsymbol{E}^{(T)} + \boldsymbol{J} \cdot (\boldsymbol{F}^{(T-1)} \cdot \boldsymbol{H}^{(1)} + \boldsymbol{F}^{(T)}) \\
&= \cdots = \boldsymbol{D}^{(0)} \cdot \boldsymbol{H}^{(T)} + \sum_{t=0}^{T} \boldsymbol{E}^{(T-t)} \cdot \boldsymbol{H}^{(t)} + \boldsymbol{J} \cdot \sum_{t=0}^{T} \boldsymbol{F}^{(T-t)} \cdot \boldsymbol{H}^{(t)}
\end{aligned} \tag{45}$$

Applying the triangle inequality of norms, we then have

$$\|\boldsymbol{D}^{(T)}\|_2 \leq \|\boldsymbol{D}^{(0)}\|_2 \cdot \|\boldsymbol{H}^{(T)}\|_2 + \sum_{t=0}^{T} (\|\boldsymbol{E}^{(T-t)}\|_2 + \|\boldsymbol{J}\|_2 \cdot \|\boldsymbol{F}^{(T-t)}\|_2) \cdot \|\boldsymbol{H}^{(t)}\|_2. \tag{46}$$

Lemma A.2 and Proposition 3.2 then imply that $\|\boldsymbol{J}\|_2 \leq L_x/\gamma$, $\|\boldsymbol{E}^{(t)}\|_2 \leq H_{xy} \cdot \|\boldsymbol{y}^{t-1} - \boldsymbol{y}^*\| \leq H_{xy}\delta^0\eta^{t/2}$, $\|\boldsymbol{F}^{(t)}\|_2 \leq H_{yy} \cdot \|\boldsymbol{y}^{t-1} - \boldsymbol{y}^*\| \leq H_{yy}\delta^0\eta^{t/2}$, and also $\|\boldsymbol{H}\|^t \leq \eta^{t/2}$. Thus, we eventually have

$$\begin{aligned}
\|\boldsymbol{D}^{(T)}\|_2 &\leq \eta^{t/2}\|\boldsymbol{D}^{(0)}\|_2 + \sum_{t=0}^{T} (H_{xy}\delta^0\eta^{(T-t)/2} + \frac{L_x}{\gamma} \cdot H_{yy}\delta^0\eta^{(T-t)/2}) \cdot \eta^{t/2} \\
&= \left(\|\boldsymbol{D}^{(0)}\|_2 + \delta^0(T+1)\left(H_{xy} + \frac{L_x}{\gamma} \cdot H_{yy}\right)\right)\eta^{T/2}.
\end{aligned} \tag{47}$$

□

Now we are ready to complete the proof. Using the triangle inequality and applying Lemma A.3, we obtain

$$\begin{aligned}
l^*(\boldsymbol{x}^*) - l(\hat{\boldsymbol{x}}, h^{(T)}(\hat{\boldsymbol{x}}, \hat{\boldsymbol{y}})) &\leq |l^*(\boldsymbol{x}^*) - l^*(\hat{\boldsymbol{x}})| + |l(\hat{\boldsymbol{x}}, y^*(\hat{\boldsymbol{x}})) - l(\hat{\boldsymbol{x}}, h^{(T)}(\hat{\boldsymbol{x}}, \hat{\boldsymbol{y}}))|_2 \\
&\leq \left(G_x + \frac{L_x G_y}{\gamma}\right) \cdot \|\hat{\boldsymbol{x}} - \boldsymbol{x}^*\|_2 + G_y \cdot \|y^*(\hat{\boldsymbol{x}}) - h^{(T)}(\hat{\boldsymbol{x}}, \hat{\boldsymbol{y}})\|_2
\end{aligned} \tag{48}$$

We first bound $\|y^*(\hat{\boldsymbol{x}}) - h^{(T)}(\hat{\boldsymbol{x}}, \hat{\boldsymbol{y}})\|_2$. We can obtain that

$$\|y^*(\hat{\boldsymbol{x}}) - h^{(T)}(\hat{\boldsymbol{x}}, \hat{\boldsymbol{y}})\|_2 \leq \eta^{T/2} \cdot \|\hat{\boldsymbol{y}} - y^*(\hat{\boldsymbol{x}})\|_2 \leq d_{\max} \cdot \eta^{T/2}, \tag{49}$$

where $d_{\max} = \operatorname{diam}(\mathbb{Y}) < \infty$.

We then bound $\|\hat{\boldsymbol{x}} - \boldsymbol{x}^*\|$. Based on Proposition 5.6, we have

$$\langle \nabla_{\boldsymbol{x}} l^{(T)}(\hat{\boldsymbol{x}}, \hat{\boldsymbol{y}}), \boldsymbol{x}^* - \hat{\boldsymbol{x}} \rangle \geq 0. \tag{50}$$

Combining it with Equation (23) by setting $\boldsymbol{x} = \hat{\boldsymbol{x}}$ and using the Cauchy-Schwartz inequality, we then have

$$\begin{aligned}
&\mu \cdot \|\boldsymbol{x}^* - \hat{\boldsymbol{x}}\|_2^2 \\
&\leq \langle \nabla l^*(\hat{\boldsymbol{x}}) - \nabla_{\boldsymbol{x}} l^{(T)}(\hat{\boldsymbol{x}}, \hat{\boldsymbol{y}}), \hat{\boldsymbol{x}} - \boldsymbol{x}^* \rangle \\
&= \langle \nabla l^*(\hat{\boldsymbol{x}}) - \nabla_{\boldsymbol{x}} l^{(T)}(\hat{\boldsymbol{x}}, y^*(\hat{\boldsymbol{x}})) + \nabla_{\boldsymbol{x}} l^{(T)}(\hat{\boldsymbol{x}}, y^*(\hat{\boldsymbol{x}})) - \nabla_{\boldsymbol{x}} l^{(T)}(\hat{\boldsymbol{x}}, \hat{\boldsymbol{y}}), \hat{\boldsymbol{x}} - \boldsymbol{x}^* \rangle \\
&\leq (\|\nabla l^*(\hat{\boldsymbol{x}}) - \nabla_{\boldsymbol{x}} l^{(T)}(\hat{\boldsymbol{x}}, y^*(\hat{\boldsymbol{x}}))\|_2 + \|\nabla_{\boldsymbol{x}} l^{(T)}(\hat{\boldsymbol{x}}, y^*(\hat{\boldsymbol{x}})) - \nabla_{\boldsymbol{x}} l^{(T)}(\hat{\boldsymbol{x}}, \hat{\boldsymbol{y}})\|_2) \cdot \|\hat{\boldsymbol{x}} - \boldsymbol{x}^*\|_2.
\end{aligned} \tag{51}$$

Thus, we have

$$\|\hat{\boldsymbol{x}} - \boldsymbol{x}^*\|_2 \leq \frac{1}{\mu} \cdot (\|\nabla l^*(\hat{\boldsymbol{x}}) - \nabla_{\boldsymbol{x}} l^{(T)}(\hat{\boldsymbol{x}}, y^*(\hat{\boldsymbol{x}}))\|_2 + \|\nabla_{\boldsymbol{x}} l^{(T)}(\hat{\boldsymbol{x}}, y^*(\hat{\boldsymbol{x}})) - \nabla_{\boldsymbol{x}} l^{(T)}(\hat{\boldsymbol{x}}, \hat{\boldsymbol{y}})\|_2). \tag{52}$$

By applying Proposition A.5, we then have

$$\|\nabla l^*(\hat{\boldsymbol{x}}) - \nabla_{\boldsymbol{x}} l^{(T)}(\hat{\boldsymbol{x}}, y^*(\hat{\boldsymbol{x}}))\|_2 \leq \|\nabla y^*(\hat{\boldsymbol{x}}) - \nabla_{\boldsymbol{x}} h^{(T)}(\hat{\boldsymbol{x}}, y^*(\hat{\boldsymbol{x}}))\|_2 \cdot \|\nabla_{\boldsymbol{y}} l(\hat{\boldsymbol{x}}, y^*(\hat{\boldsymbol{x}}))\|_2 \leq \frac{G_y L_x \eta^{T/2}}{\gamma}. \tag{53}$$

Meanwhile, using the triangle inequality of norms, we have

$$\begin{aligned}
\|\nabla_{\boldsymbol{x}} l^{(T)}(\hat{\boldsymbol{x}}, y^*(\hat{\boldsymbol{x}})) - \nabla_{\boldsymbol{x}} l^{(T)}(\hat{\boldsymbol{x}}, \hat{\boldsymbol{y}})\|_2 &= \|\nabla_{\boldsymbol{x}} l(\hat{\boldsymbol{x}}, y^*(\hat{\boldsymbol{x}})) - \nabla_{\boldsymbol{x}} l(\hat{\boldsymbol{x}}, h^{(T)}(\hat{\boldsymbol{x}}, \hat{\boldsymbol{y}}))\|_2 \\
&+ \|\nabla_{\boldsymbol{x}} h^{(T)}(\hat{\boldsymbol{x}}, y^*(\hat{\boldsymbol{x}}))\|_2 \cdot \|\nabla_{\boldsymbol{y}} l(\hat{\boldsymbol{x}}, y^*(\hat{\boldsymbol{x}})) - \nabla_{\boldsymbol{y}} l(\hat{\boldsymbol{x}}, h^{(T)}(\hat{\boldsymbol{x}}, \hat{\boldsymbol{y}})\|_2 \\
&+ \|\nabla_{\boldsymbol{x}} h^{(T)}(\hat{\boldsymbol{x}}, y^*(\hat{\boldsymbol{x}})) - \nabla_{\boldsymbol{x}} h^{(T)}(\hat{\boldsymbol{x}}, \hat{\boldsymbol{y}})\|_2 \cdot \|\nabla_{\boldsymbol{y}} l(\hat{\boldsymbol{x}}, h^{(T)}(\hat{\boldsymbol{x}}, \hat{\boldsymbol{y}}))\|_2.
\end{aligned} \tag{54}$$

Firstly, noted that $\nabla_{\boldsymbol{x}} l(\boldsymbol{x}, \cdot)$ is G_{xy}-Lipschitz continuous, we have

$$\|\nabla_{\boldsymbol{x}} l(\hat{\boldsymbol{x}}, y^*(\hat{\boldsymbol{x}})) - \nabla_{\boldsymbol{x}} l(\hat{\boldsymbol{x}}, h^{(T)}(\hat{\boldsymbol{x}}, \hat{\boldsymbol{y}}))\|_2 \leq G_{xy} \cdot \|y^*(\hat{\boldsymbol{x}}) - h^{(T)}(\hat{\boldsymbol{x}}, \hat{\boldsymbol{y}})\|_2 \leq G_{xy} d_{\max} \eta^{T/2}, \tag{55}$$

where the second inequality comes from Equation (49).

Secondly, by applying the triangle inequality, we can obtain

$$\|\nabla_{\boldsymbol{x}} h^{(T)}(\hat{\boldsymbol{x}}, y^*(\hat{\boldsymbol{x}}))\|_2 \leq \|\nabla_{\boldsymbol{x}} h^{(T)}(\hat{\boldsymbol{x}}, y^*(\hat{\boldsymbol{x}})) - \nabla_{\boldsymbol{x}} y^*(\hat{\boldsymbol{x}})\|_2 + \|\nabla y^*(\hat{\boldsymbol{x}})\|_2 \leq \frac{L_x}{\gamma}(1 + \eta^{T/2}). \tag{56}$$

where the second inequality comes from Lemma A.2 and 3.2. Noting that $\nabla_{\boldsymbol{y}} l(\boldsymbol{x}, \cdot)$ is G_{yy}-Lipschitz continuous, we also have

$$\|\nabla_{\boldsymbol{y}} l(\hat{\boldsymbol{x}}, y^*(\hat{\boldsymbol{x}})) - \nabla_{\boldsymbol{y}} l(\hat{\boldsymbol{x}}, h^{(T)}(\hat{\boldsymbol{x}}, \hat{\boldsymbol{y}})\|_2 \leq G_{yy} \cdot \|y^*(\hat{\boldsymbol{x}}) - h^{(T)}(\hat{\boldsymbol{x}}, \hat{\boldsymbol{y}})\|_2 \leq G_{yy} d_{\max} \eta^{T/2}, \tag{57}$$

where the second inequality also comes from (49).

Thirdly, we have $\|\nabla_{\boldsymbol{y}} l(\hat{\boldsymbol{x}}, h^{(T)}(\hat{\boldsymbol{x}}, \hat{\boldsymbol{y}}))\|_2 \leq G_y$. Meanwhile, we also have

$$\begin{aligned}
&\|\nabla_{\boldsymbol{x}} h^{(T)}(\hat{\boldsymbol{x}}, y^*(\hat{\boldsymbol{x}})) - \nabla_{\boldsymbol{x}} h^{(T)}(\hat{\boldsymbol{x}}, \hat{\boldsymbol{y}})\|_2 \\
&\qquad \leq \|\nabla_{\boldsymbol{x}} h^{(T)}(\hat{\boldsymbol{x}}, y^*(\hat{\boldsymbol{x}})) - \nabla_{\boldsymbol{x}} y^*(\hat{\boldsymbol{x}})\|_2 + \|\nabla_{\boldsymbol{x}} h^{(T)}(\hat{\boldsymbol{x}}, \hat{\boldsymbol{y}}) - \nabla_{\boldsymbol{x}} y^*(\hat{\boldsymbol{x}})\|_2 \\
&\qquad \leq \frac{L_x \eta^{T/2}}{\gamma} + \left(H_x + T d_{\max}\left(H_{xy} + \frac{L_x}{\gamma} \cdot H_{yy}\right)\right) \eta^{T/2},
\end{aligned} \tag{58}$$

where the second inequality comes from Lemma A.6.

Combing these, we obtain that

$$\begin{aligned}\mu\|\hat{\boldsymbol{x}}^* - \boldsymbol{x}^*\|_2 \leq \frac{G_y L_x \eta^{T/2}}{\gamma} + G_{xy} d_{\max} \cdot \eta^{T/2} + \frac{L_x}{\gamma}(1+\eta^{T/2}) G_{yy} d_{\max} \cdot \eta^{T/2} + \\ G_y \left(\frac{L_x \eta^{T/2}}{\gamma} + \left(H_x + T d_{\max} \left(H_{xy} + \frac{L_x}{\gamma} \cdot H_{yy} \right) \right) \cdot \eta^{T/2} \right).\end{aligned} \tag{59}$$

Without loss of generality, we assume that $\gamma = 1$ and $\mu = 1$. Then we have

$$\begin{aligned}\|\hat{\boldsymbol{x}}^* - \boldsymbol{x}^*\|_2 &\leq G_y L_x \cdot \eta^{T/2} + G_{xy} d_{\max} \cdot \eta^{T/2} + L_x (1+\eta^{T/2}) G_{yy} d_{\max} \cdot \eta^{T/2} \\ &\quad + G_y \left(L_x \cdot \eta^{T/2} + (H_x + T d_{\max} (H_{xy} + L_x \cdot H_{yy})) \cdot \eta^{T/2} \right) \\ &\leq (2 G_y L_x + G_{xy} d_{\max} + 2 L_x G_{yy} d_{\max} + G_y H_x) \cdot \eta^{T/2} \\ &\quad + G_y (H_{xy} + L_x H_y y) \cdot (T+1) \cdot \eta^{T/2}.\end{aligned} \tag{60}$$

Eventually, we obtain that

$$l^*(\boldsymbol{x}^*) - l(\hat{\boldsymbol{x}}, h^{(T)}(\hat{\boldsymbol{x}}, \hat{\boldsymbol{y}})) \leq G_y (H_{xy} + L_x H_y y) \cdot (T+1) \cdot \eta^{T/2} + M \cdot \eta^{T/2}, \tag{61}$$

where $M = (G_x + L_x G_y) \cdot (2 G_y L_x + G_{xy} d_{\max} + 2 L_x G_{yy} d_{\max} + G_y H_x) + G_y d_{\max}$.

B. More Discussion on the Results in Section 5

B.1. Gradient Calculation in Algorithms 1 and 2

Below we briefly discuss how to code $h(\boldsymbol{x}, \boldsymbol{y})$ so that it can be fed into differentiable-programming frameworks. For simplicity, we denote $g(\boldsymbol{z}) = \arg\min_{\boldsymbol{y} \in \mathbb{Y}} \|\boldsymbol{y} - \boldsymbol{z}\|_2$, then $h(\boldsymbol{x}, \boldsymbol{y}) = g(\boldsymbol{y} - r \cdot f(\boldsymbol{x}, \boldsymbol{y}))$, where $\boldsymbol{z} = \boldsymbol{y} - r \cdot f(\boldsymbol{x}, \boldsymbol{y})$. Thus, the problem reduces to how to code $g(\boldsymbol{z})$. Since $g(\boldsymbol{z})$ is equivalent to a Euclidean projection problem, it can always be coded via `cvxpylayers` (Agrawal et al., 2019), as discussed earlier. Here we consider some other coding strategies. For example, there are three cases when $g(\boldsymbol{z})$ is analytic, so that we can directly code $g(\boldsymbol{z})$ according to its analytic expression.

- (i) When $\mathbb{Y} = \mathbb{R}^n$ (the full space), we always have $g(\boldsymbol{z}) = \boldsymbol{z}$.
- (ii) When $\mathbb{Y} = \{\boldsymbol{y} : \boldsymbol{a} \leq \boldsymbol{y} \leq \boldsymbol{b}\}$ (a box constraint), we have $g(\boldsymbol{z}) = \min\{\max\{\boldsymbol{z}, \boldsymbol{a}\}, \boldsymbol{b}\}$.
- (iii) When $\mathbb{Y} = \{\boldsymbol{y} : \mathbf{1}^\top \boldsymbol{y} = 1\}$ (an equity constraint), we have $g(\boldsymbol{z}) = \boldsymbol{z} - (\mathbf{1}^\top \boldsymbol{z} - 1)/n$.

We are now ready to handle a more complicated case when $\mathbb{Y}$ is probability simplex as in many practical applications.

- (iv) When $\mathbb{Y} = \{\boldsymbol{y} \geq \mathbf{0} : \mathbf{1}^\top \boldsymbol{y} = 1\}$ (a *probability simplex constraint*), we can rewrite $\mathbb{Y} = \{\boldsymbol{y} : \boldsymbol{y} \geq \mathbf{0}\} \cap \{\boldsymbol{y} : \mathbf{1}^\top \boldsymbol{y} = 1\}$, which is the intersection of a box constraint and an equity constraint. Thus, we can use Dykstra's projection algorithm (Boyle and Dykstra, 1986) to solve $g(\boldsymbol{z})$, which iteratively projects $\boldsymbol{z}$ onto two sets until a fixed point is found. This alternating projection process is differentiable because the projections on both sets respectively correspond to cases (ii) and (iii), both of which are analytic.

Subsequently, we can move to a more complicated case when $\mathbb{Y}$ is rewritten as $\mathbb{Y} = \mathbb{Y}_1 \times \cdots \times \mathbb{Y}_n$, i.e., the Cartesian product of some other sets, the projection then can be carried out on $\mathbb{Y}_i$ ($i = 1, \ldots, n$) independently. Hence, as long as every $\mathbb{Y}_i$ falls into the four cases discussed above, the function $g(\boldsymbol{z})$ can still be coded as a differentiable program easily. Decomposing the projection problem $g(\boldsymbol{z})$ into n parallel sub-problems can also accelerate AD.

Remark B.1. Here we mark that a "differentiable program" is not necessarily everywhere differentiable. Instead, it only means that the program can be fed to differentiable-programming frameworks. For example, in case (ii), the operation $\min\{\max\{\boldsymbol{z}, \boldsymbol{a}\}, \boldsymbol{b}\}$ is not differentiable at $\boldsymbol{z} = \boldsymbol{a}$ and $\boldsymbol{z} = \boldsymbol{b}$; at those points, a sub-gradient can be used to run AD.

B.2. Other Functions for Formulating the Two Models

To formulate the two proposed models, the selection of $h(\boldsymbol{x}, \boldsymbol{y})$ is not absolute. Indeed, we have shown that whenever $h(\boldsymbol{x}, \boldsymbol{y})$ charts a provably convergent path to solve the lower-level VI in Problem (1), the resulting T-step Cournot games and monopoly models will provide an upper and a lower bound to Problem (1), respectively. For example, the mirror descent method (Nemirovskij and Yudin, 1983) gives

$$h(\boldsymbol{x}, \boldsymbol{y}) = \operatorname*{arg\,min}_{\boldsymbol{y}' \in \mathbb{Y}} \; r \cdot \langle f(\boldsymbol{x}, \boldsymbol{y}), \boldsymbol{y}' - \boldsymbol{y} \rangle + D_\phi(\boldsymbol{y}', \boldsymbol{y}), \tag{62}$$

where $D_\phi(\boldsymbol{y}', \boldsymbol{y}) = \phi(\boldsymbol{y}') - \phi(\boldsymbol{y}) - \langle \nabla \phi(\boldsymbol{y}), \boldsymbol{y}' - \boldsymbol{y} \rangle$ is the Bregman divergence between $\boldsymbol{y}'$ and $\boldsymbol{y}$ induced by a strongly convex function $\phi : \mathbb{Y} \to \mathbb{R}$. We refer the readers to Mertikopoulos and Zhou (2019) for sufficient conditions under which the mirror descent algorithm converges to the solution to a VI. When $\mathbb{Y}$ is a probability simplex (or the Cartesian product of several probability simplices) and $D_\phi(\boldsymbol{y}', \boldsymbol{y})$ is specified as the KL divergence, $h(\boldsymbol{x}, \boldsymbol{y})$ becomes analytic (Beck and Teboulle, 2003) so that it can be more easily differentiated through via AD. We refer the readers to Li et al. (2022b) for a more detailed discussion on how to unroll the mirror descent method.

B.3. Global Optimization Strategies

In Theorem 5.7, we assume $l^*(\boldsymbol{x})$ to be globally convex; in general, this assumption does not hold for most practical applications. In this section, we briefly discuss how to search for the global minimizer of Problem (1). Typically, when an optimization problem is not convex, then we have no choice but to accept the best local solution after running an appropriate first-order method multiple times with different initial solutions. But for bilevel programs, repeatedly searching for local solutions is not easy. Nevertheless, with our framework, we may first fix T as a small value (e.g., 0 or 1) and first run Algorithm 1 and/or Algorithm 2 many times, each time with a randomly generated initial solution. This step may be viewed as a quick-and-dirty "scan" of the geometry of the overall problem. Afterward, the best local solution to either T-step Cournot games or monopoly models then can be used to roughly estimate where the global optimal solution to (1) might be located. Subsequently, we can search for the optimal solution to Problem (1) around that region, following the procedure that we devised at the end of Section 5.1.

B.4. Extension to Bilevel Programs with Multiple Lower-Level Solutions

We then discuss the case when $\mathbb{Y}^*(\boldsymbol{x})$ is not a singleton, which renders a principled rule for selecting a $\boldsymbol{y}^* \in \mathbb{Y}^*(\boldsymbol{x})$ to evaluate $l(\boldsymbol{x}, \boldsymbol{y}^*)$ as a necessity. To this end, the optimistic and pessimistic principles respectively select the best and worst solutions, as judged by the leader's objective. In other words, they respectively solve

$$\min_{\boldsymbol{x} \in \mathbb{X}} \; \min_{\boldsymbol{y}^* \in \mathbb{Y}^*(\boldsymbol{x})} \; l(\boldsymbol{x}, \boldsymbol{y}^*) \quad \text{and} \quad \min_{\boldsymbol{x} \in \mathbb{X}} \; \max_{\boldsymbol{y}^* \in \mathbb{Y}^*(\boldsymbol{x})} \; l(\boldsymbol{x}, \boldsymbol{y}^*). \tag{63}$$

The above two problems are commonly referred to as a *strong* and a *weak* Stackelberg game respectively (Loridan and Morgan, 1996). Denoting the solutions to those two problems as $(\boldsymbol{x}^*_{\text{opt}}, \boldsymbol{y}^*_{\text{opt}})$ and $(\boldsymbol{x}^*_{\text{pes}}, \boldsymbol{y}^*_{\text{pes}})$, then we have $l(\boldsymbol{x}^*_{\text{opt}}, \boldsymbol{y}^*_{\text{opt}}) \leq l(\boldsymbol{x}^*_{\text{pes}}, \boldsymbol{y}^*_{\text{pes}})$. In the literature, the strong Stackelberg games is more frequently regarded as the "standard" formulation (Luo et al., 1996). The original formulation (1) is indeed a strong Stackelberg game because it allows the leader to pick any $\boldsymbol{y}^* \in \mathbb{Y}^*(\boldsymbol{x})$. Thus, the leader can always pick the most favorable (i.e., the optimistic) one. According to Proposition 5.1, the solutions given by T-step Cournot games and monopoly models always provide an upper and a lower bound for $l(\boldsymbol{x}^*_{\text{opt}}, \boldsymbol{y}^*_{\text{opt}})$, respectively. We are hence motivated to ask whether our models would still provide *arbitrarily tight* bounds to $l(\boldsymbol{x}^*_{\text{opt}}, \boldsymbol{y}^*_{\text{opt}})$ when $T \to \infty$.

Unfortunately, we cannot answer this question with a certain yes. Indeed, in a T-step Cournot game, the followers' decision $\bar{\boldsymbol{y}}$ is not necessarily the one that minimizes $l(\bar{\boldsymbol{x}}, \boldsymbol{y}^*)$ among all $\boldsymbol{y}^* \in \mathbb{Y}^*(\bar{\boldsymbol{x}})$. In other words, a T-step Cournot game is not intrinsically optimistic; instead, it has to let the followers pick an equilibrium by themselves. Hence, even though T is sufficiently large, a T-step Cournot game may still admit multiple solutions. If one solution $(\bar{\boldsymbol{x}}, \bar{\boldsymbol{y}})$ accidentally satisfies $\bar{\boldsymbol{y}} \in \arg\min_{\boldsymbol{y}^*} l(\bar{\boldsymbol{x}}, \boldsymbol{y}^*)$, then we expect $l(\bar{\boldsymbol{x}}, \bar{\boldsymbol{y}})$ may be close to $l(\boldsymbol{x}^*_{\text{opt}}, \boldsymbol{y}^*_{\text{opt}})$. Otherwise, if $\bar{\boldsymbol{y}}$ is the solution in $\mathbb{Y}^*(\bar{\boldsymbol{x}})$ against the leader's interest, the gap may still be unacceptably large. Below we give an example.

Example B.2. *Consider the following problem*

$$\begin{aligned} \min_{1\geq x\geq 0.5} \quad & (x+y^*-1)^2 \\ \text{s.t.} \quad & y^* \in \arg\min_{y\in\mathbb{R}} \; g(x,y) = \begin{cases} (y-x-0.25)^2, & \text{if } y \geq x+0.25, \\ 0, & \text{if } y \in (x-0.25, x+0.25), \\ (y-x+0.25)^2, & \text{if } y \leq x-0.25, \end{cases} \end{aligned} \tag{64}$$

which is equivalent to the following one

$$\begin{aligned} \min_{1\geq x\geq 0.5} \quad & (x+y^*-1)^2 \\ \text{s.t.} \quad & y^* \in [x-0.25, x+0.25]. \end{aligned} \tag{65}$$

Under this setting, the optimal solution set to (65) is $\mathbb{Z}^* = \{(x,y) : x \in [0.5,1], y = 1-x\}$ and the solution set to the corresponding 0-step Cournot game is $\mathbb{Z}^{(0)} = \mathbb{Z}^* \cup \{(x,y) : x = 0.5, y \in [0.5, 0.75]\}$. We then prove that the solution sets to T-step Cournot models are still $\mathbb{Z}^{(0)}$ for alll $T > 0$.

$$\nabla_y g(x,y) = \begin{cases} 2(y-x-0.25), & \text{if } y \geq x+0.25, \\ 0, & \text{if } y \in (x-0.25, x+0.25), \\ 2(y-x+0.25), & \text{if } y \leq x-0.25. \end{cases} \tag{66}$$

We can then define

$$h(x,y) = \arg\min_{1\geq x\geq 0.5} \; y - r\cdot\nabla_y g(x,y) \tag{67}$$

to model the dynamics, which satisfies $h(x,y) = y$ for any $x \in [0.5,1]$ and $y \in [x-0.25, x+0.25]$. Thus, for any $\bar{y} \in [0.5, 0.75]$, the optimal solution to

$$\min_{1\geq x\geq 0.5} \; (x + h^{(T)}(x,\bar{y}) - 1)^2 = (x+\bar{y}-1)^2 \tag{68}$$

is still $\bar{x} = 0.5$ for all $T > 0$. Therefore, we have $\mathbb{Z}^{(T)} = \mathbb{Z}^* \cup \{(x,y) : x = 0.5, y \in [0.5, 0.75]\}$ for all $T \in \mathbb{N}$. Among these solutions, the worst one is $(\bar{x},\bar{y}) = (0.5, 0.75)$, whose corresponding objective value is $0.0625 > 0$.

In contrast, a T-step monopoly model, to some extent, is essentially optimistic. Intuitively, if $(\hat{\boldsymbol{x}},\hat{\boldsymbol{y}})$ solves a T-step monopoly model with a sufficiently large T, we may expect that $h^{(T)}(\hat{\boldsymbol{x}},\hat{\boldsymbol{y}})$ is close to the one that minimizes $l(\hat{\boldsymbol{x}},\boldsymbol{y}^*)$ among all $\boldsymbol{y}^* \in \mathbb{Y}^*(\hat{\boldsymbol{x}})$, as the leader has the authority to steer the follower's evolutionary path by manipulating their initial strategy.

Motivated by the above discussion, we design a heuristic for approximating Problem (1) by iteratively solving T-step Cournot games and monopoly models. Specifically, setting T as a small number and starting with an arbitrary $(\boldsymbol{x}^0,\boldsymbol{y}^0) \in \mathbb{X}\times\mathbb{Y}$, we may first run run Algorithms 1 and 2 to obtain $(\bar{\boldsymbol{x}},\bar{\boldsymbol{y}})$ and $(\hat{\boldsymbol{x}},\hat{\boldsymbol{y}})$, respectively. If the gap between $l^{(T)}(\bar{\boldsymbol{x}},\bar{\boldsymbol{y}})$ and $l^{(T)}(\hat{\boldsymbol{x}},\hat{\boldsymbol{y}})$ is sufficiently small, then we can accept $(\bar{\boldsymbol{x}},\bar{\boldsymbol{y}})$ as an approximated solution to Problem (1). Otherwise, we can increase T and repeat the above procedure. But during the next iteration, we may use a warm-start initial solution, rather than the original $(\boldsymbol{x}^0,\boldsymbol{y}^0)$, to run Algorithms 1 and 2 to reduce the number of iterations required for convergence. Particularly, that initial solution could set as $(\hat{\boldsymbol{x}}, h^{(T)}(\hat{\boldsymbol{x}},\hat{\boldsymbol{y}}))$, where $(\hat{\boldsymbol{x}},\hat{\boldsymbol{y}})$ is the output of Algorithm 2 in the previous iteration. As discussed earlier, the pair $(\hat{\boldsymbol{x}}, h^{(T)}(\hat{\boldsymbol{x}},\hat{\boldsymbol{y}}))$ is intrinsically optimistic. Hence, if in the next iteration, we run Algorithm 1 starting from $(\hat{\boldsymbol{x}}, h^{(T)}(\hat{\boldsymbol{x}},\hat{\boldsymbol{y}}))$, the new output $(\bar{\boldsymbol{x}},\bar{\boldsymbol{y}})$ is more likely to within the neighborhood of $(\boldsymbol{x}^*_{\text{opt}},\boldsymbol{y}^*_{\text{opt}})$. The pseudocode code is provided in Algorithm 3. We are not sure whether the gap $\delta^{(T)}$ in Algorithm 3 would eventually converge to 0 or a sufficiently small value for all practical purposes; we leave a theoretical analysis of the algorithm to future work. But later, we will numerically test it.

C. Numerical Experiments

In Experiment II (Appendix C.1), we study the network design problem on the Braess network (Braess, 1968). We will compare the solutions provided by the T-step Cournot games and monopoly models formulated by two types of $h(\boldsymbol{x},\boldsymbol{y})$: the projection method and the mirror descent method (see Appendix B.2). Eventually, in Experiment III (Appendix C.2), we study the network problem on a real transportation network and compare the proposed scheme with many existing bilevel programming algorithms.

Algorithm 3 Adaptive-initialization strategy for approximating Problem (1).

Input: $(\boldsymbol{x}^0, \boldsymbol{y}^0) \in \mathbb{X} \times \mathbb{Y}$, T as a small integer
while True **do**
  Starting from $(\boldsymbol{x}^0, \boldsymbol{y}^0)$, run Algorithm 1 to obtain $(\bar{\boldsymbol{x}}, \bar{\boldsymbol{y}})$ and then run Algorithm 2 to obtain $(\hat{\boldsymbol{x}}, \hat{\boldsymbol{y}})$.
  If $l^{(T)}(\bar{\boldsymbol{x}}, \bar{\boldsymbol{y}}) - l^{(T)}(\hat{\boldsymbol{x}}, \hat{\boldsymbol{y}})$ is sufficiently small, break and return $(\bar{\boldsymbol{x}}, \bar{\boldsymbol{y}})$; otherwise, increase T and set $(\boldsymbol{x}^0, \boldsymbol{y}^0) = (\hat{\boldsymbol{x}}, h^{(T)}(\hat{\boldsymbol{x}}, \hat{\boldsymbol{y}}))$.
end while

C.1. Experiment II

We then test our algorithms on a classic network design problem, which studies how to add capacities in a traffic network to reduce congestion. We model the network as a directed graph $\mathcal{G}(\mathbb{N}, \mathbb{A})$, where $\mathbb{N}$ and $\mathbb{A}$ are the set of nodes and arcs. We write $\mathbb{W} \subseteq \mathbb{N} \times \mathbb{N}$ as the set of origin-destination (OD) pairs and $\mathbb{K} \subseteq 2^{\mathbb{A}}$ as the set of paths connecting the OD pairs. We assume that each OD pair $w \in \mathbb{W}$ is associated with q_w vehicles. Let $\mathbb{K}_w \subseteq \mathbb{K}$ be the set of all paths connecting w. We assume that the traffic planner's objective is a weighted sum of the total monetary cost associated with the capacity enhancement and the total travel time experienced by the vehicles. We denote the original arc capacity as $\boldsymbol{s} \in \mathbb{R}_+^{|\mathbb{A}|}$ and the capacity enhancement added by the planning agent as $\boldsymbol{x}$. We assume that the capacities can only be added to selected arcs, denoted as $\tilde{\mathbb{A}} \subseteq \mathbb{A}$. The feasible region for $\boldsymbol{x}$ then becomes $\mathbb{X} = \{\boldsymbol{x} \in \mathbb{R}_+^{|\mathbb{A}|} : x_a = 0, a \in \mathbb{A} \setminus \tilde{\mathbb{A}}\}$. We write $m(\boldsymbol{x}) = \langle \boldsymbol{b}, \boldsymbol{x}^2 \rangle$ as the total monetary cost associated with the capacity enhancement, where $\boldsymbol{b} \in \mathbb{R}_+^{|\mathbb{A}|}$ are cost parameters. We assume that the arc travel time is given by $u(\boldsymbol{x}, \boldsymbol{v}) = \boldsymbol{u}_0 \cdot \big(1 + 0.15 \cdot (\boldsymbol{v}/(\boldsymbol{s} + \boldsymbol{x}))^4\big)$, where $\boldsymbol{v}, \boldsymbol{u}_0 \in \mathbb{R}_+^{|\mathbb{A}|}$ represent arc vehicle flows and free-flow travel times, respectively. Meanwhile, we write the vehicles' route choices as a vector $\boldsymbol{y} = (y_k)_{k \in \mathbb{K}}$ with y_k equals the proportion of vehicles between w using the path $k \in \mathbb{K}_w$, and the travel times for using each path as $\boldsymbol{c} = (c_k)_{k \in \mathbb{K}}$. Denote $\Sigma_{w,k}$ as the OD-path incidence, with $\Sigma_{w,k}$ equals 1 if the path $k \in \mathbb{K}_w$ and 0 otherwise. Meanwhile, denote $\Lambda_{a,k}$ as the arc-path incidence, with $\Lambda_{a,k}$ equals 1 if $a \in \mathbb{A}_k$ and 0 otherwise. For notational convenience, we write $\boldsymbol{\Lambda} = (\Lambda_{a,k})_{a \in \mathbb{A}, k \in \mathbb{K}}$ and $\boldsymbol{\Sigma} = (\Sigma_{w,k})_{w \in \mathbb{W}, k \in \mathbb{K}}$. Let $\boldsymbol{d} = (d_k)_{k \in \mathbb{K}}$ be a vector with $d_k = q_w$ if $k \in \mathbb{K}_w$. The feasible region for $\boldsymbol{y}$ can then be written as $\mathbb{Y} = \{\boldsymbol{y} \geq 0 : \boldsymbol{\Sigma}\, \boldsymbol{y} = \boldsymbol{1}\}$. Meanwhile, we also have $\boldsymbol{v} = \boldsymbol{\Lambda}(\boldsymbol{d} \cdot \boldsymbol{y})$ and $\boldsymbol{c} = \boldsymbol{\Lambda}^\top u(\boldsymbol{x}, \boldsymbol{v})$. The results in Example 2.2 imply that the set of Wardrop equilibria $\mathbb{Y}^*(\boldsymbol{x})$ is the solution set to the following VI problem:

$$\langle \boldsymbol{\Lambda}^\top u(\boldsymbol{x}, \boldsymbol{\Lambda}(\boldsymbol{d} \cdot \boldsymbol{y}^*)), \boldsymbol{y} - \boldsymbol{y}^* \rangle \geq 0, \quad \forall \boldsymbol{y} \in \mathbb{Y}. \tag{69}$$

The network design problem then has the following form

$$\begin{aligned} \min_{\boldsymbol{x} \in \mathbb{X}} \quad & \langle u(\boldsymbol{x}, \boldsymbol{v}^*)), \boldsymbol{v}^*) \rangle + \gamma \cdot m(\boldsymbol{x}), \\ \text{s.t.} \quad & \boldsymbol{v}^* = \boldsymbol{\Lambda}(\boldsymbol{d} \cdot \boldsymbol{y}^*), \quad \boldsymbol{y}^* \in \mathbb{Y}^*(\boldsymbol{x}). \end{aligned} \tag{70}$$

We first solve the network design problem on the Braess network (Braess, 1968) as shown in Figure 3. The network has three paths connecting the origin (node 1) and the destination (node 4): path 1 uses links 1 and 3, path 2 uses links 1, 4, and 5, and path 3 uses links 2 and 5. The Braess paradox (Braess, 1968) implies that no capacities should be added to link 4 (the bridge link), otherwise it would increase the total travel time experienced by the travelers at equilibrium.

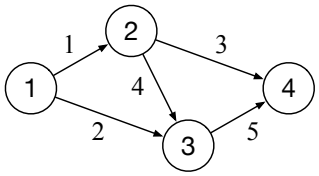

Figure 3. The Braess network.

We set $m(\boldsymbol{x}) = \langle \boldsymbol{w}, \boldsymbol{x}^2 \rangle$, $\boldsymbol{d} = 6$, $\boldsymbol{u}_0 = [1, 3, 3, 0.5, 1]^\top$, $\boldsymbol{v}_0 = [2, 4, 4, 1, 2]^\top$, $\boldsymbol{w} = [1, 3, 3, 0.5, 1]^\top$, and $\gamma = 1$. The problem is first solved via two existing AD-based methods proposed by Li et al. (2020) and Li et al. (2022b), both of which unroll the full computational process for solving the lower-level VI problem. The difference lies in the algorithm being unrolled: the first unrolls the projection method while the second unrolls the mirror descent method (see Section B.2). In the following experiments, the solutions returned by Li et al. (2020)'s and Li et al. (2022b)'s methods will be used as the benchmark.

When testing our scheme, we also formulate T-step Cournot games and monopoly models with two types of $h(\boldsymbol{x}, \boldsymbol{y})$: the projection method and the mirror descent method. To investigate whether their solutions can make a nice approximation to the benchmark solutions, we progressively increase T in the experiment until the gap is closed. Our models are solved by Algorithms 1 and 2; the same hyperparameters are employed for all tested algorithms (including both the benchmarks and our algorithms), except the learning rate r, which is set to be 0.1 in the projection-method version and 0.25 in the mirror-descent version. Table 2 reports the solutions (upper-level: capacity enhancement; lower-level: route choice), the corresponding objective function values as well as the total CPU (2.9 GHz Quad-Core Intel Core i7) time and the number of iterations required to obtain the solutions (for notational simplicity, we use "S" to represent the benchmarks and use "C-" and "M-" to represent T-step Cournot games and monopoly models with different Ts).

Table 2. Performance of T-step Cournot games and T-step monopoly models when solving the continuous network design problem on the Braess network.

(a) Setting A: Modeling the lower-level solution process using the projection method.

Method	Time (s)	Iterations	Value	Capacity enhancement					Route choice		
				$a=1$	$a=2$	$a=3$	$a=4$	$a=5$	$k=1$	$k=2$	$k=3$
C-0	1.06	80	38.786	2.075	0	0	2.826	2.075	0.000	1.000	0.000
C-1	6.68	109	28.920	0.936	0.016	0.016	0	0.936	0.339	0.321	0.339
S	12.5	92	28.920	0.928	0.016	0.016	0	0.928	0.333	0.333	0.333
M-4	8.22	46	28.920	0.928	0.016	0.016	0	0.928	0.329	0.343	0.329
M-3	35.1	339	26.745	0.867	0.027	0.027	0.192	0.867	0.156	0.688	0.156
M-2	2.16	43	26.789	0.744	0.039	0.039	0.009	0.744	0.217	0.565	0.217
M-1	0.31	40	27.142	0.603	0.066	0.066	0.000	0.603	0.225	0.549	0.225
M-0	0.03	70	26.722	0.821	0.030	0.030	0.113	0.821	0.424	0.151	0.424

(b) Setting B: Modeling the lower-level solution process using the mirror descent method.

Method	Time (s)	Iterations	Value	Capacity enhancement					Route choice		
				$a=1$	$a=2$	$a=3$	$a=4$	$a=5$	$k=1$	$k=2$	$k=3$
C-0	0.103	264	38.786	2.075	0	0	2.830	2.075	0.000	1.000	0.000
C-1	0.148	254	28.925	0.966	0.015	0.015	0	0.966	0.339	0.322	0.339
C-2	0.208	270	28.920	0.939	0.016	0.016	0	0.939	0.339	0.321	0.339
S	0.393	232	28.920	0.928	0.016	0.016	0	0.928	0.333	0.333	0.333
M-5	0.292	232	28.920	0.928	0.016	0.016	0	0.928	0.319	0.363	0.319
M-4	0.562	512	26.722	0.821	0.03	0.03	0.083	0.821	0.169	0.663	0.169
M-3	0.459	440	26.722	0.821	0.03	0.03	0.084	0.821	0.179	0.642	0.179
M-2	0.323	381	26.722	0.821	0.03	0.03	0.085	0.821	0.190	0.620	0.190
M-1	0.235	341	26.722	0.82	0.03	0.03	0.088	0.82	0.202	0.596	0.202
M-0	0.151	447	26.722	0.821	0.03	0.03	0.095	0.821	0.425	0.151	0.425

The solutions returned by the two benchmark algorithms are identical; no capacity is added on link 4 as indicated by the Braess paradox. However, if we adopt T-step Cournot games and monopoly models to approximate the original problem, then some capacity would be added on link 4 when T is overly too small, which falls into the trap. However, a slightly larger T will fix the problem in both models, no matter which method is used to model the lower-level solution process. It is worth noting that when T is the same, the solutions to the models formulated by the projection method have better quality. However, the slight improvement in solution quality can hardly offset the increase in computational cost. Take C-1 for instance. Using the projection method instead of the mirror descent method could decrease the objective value by merely $(28.92 - 28.925)/28.925 = 0.170\%$, but the required CPU time is increased by $(6.68 - 0.148)/0.148 = 4410\%$. Eventually, we remark that a classic result is that C-0 and M-0 can deliver satisfactory solutions on many networks (Marcotte, 1986); here the Braess network is intentionally selected as a counterexample. Our goal is to show that if our models are able to make good approximations on the Braess network, then we can be more confident in applying it to other problems.

C.2. Experiment III

We then move to the network in the City of Sioux Fall, South Dakota; the network data (topology, travel demand, arc travel time function) are downloaded from the *Transportation Network* GitHub repository[1]. For this network, we have $|\mathbb{N}| = 24$, $|\mathbb{A}| = 76$, $|\mathbb{K}| = 638$. We select 10 arcs for expanding the capacities (see Table 3).

Table 3. Selected arcs for adding capacities and their corresponding cost parameters.

a	16	19	17	20	25	26	29	48	39	74
b_a	26.0	26.0	40.0	40.0	25.0	25.0	48.0	48.0	34.0	34.0

In the experiment, we set $\gamma = 0.01$. Based on the results given by Experiment II (see Appendix C.1), we use the mirror descent method to formulate the proposed models. We compare our approaches with some previous bilevel programming methods studied in the optimization and ML literature.

- "c0": Algorithm 1 with $T = 0$. It solves a 0-step Cournot game. The classic single-level approximation scheme proposed by (Tan et al., 1979) essentially solves the same model but uses different algorithms.
- "c1": Algorithm 1 with $T = 1$. It solves a 1-step Cournot game. It may be viewed as a straightforward extension of one-stage AD (Liu et al., 2018).
- "c10": Algorithm 1 with $T = 10$. It solves a 10-step Cournot game.
- "m0": Algorithm 2 with $T = 0$. It solves a 0-step monopoly model. The classic single-level approximation scheme proposed by (Dantzig et al., 1979) essentially solves the same model.
- "m45": Algorithm 2 with $T = 45$. It solves a 45-step monopoly model.
- "ad": The Algorithm proposed by Li et al. (2022b).
- "tad": An extension of truncated AD (Shaban et al., 2019). It is similar to "ad"; the difference is that we only unroll the last 10 iterations of the dynamical process for solving the lower-level problem.
- "id": The implicit differentiation scheme proposed by Li et al. (2020).
- "aid": An extension of approximated ID. It shares the same overall structure as "id"; the difference is that we only keep the first 10 terms in the Neumann series for matrix inversion.
- "sid": We extend the two-timescale single-looped method proposed by Hong et al. (2020).

We appropriately design hyperparameters in all of these algorithms and stop running the algorithms based on similar termination conditions. The results are shown in Figure 4, in which we report the total CPU (2.9 GHz Quad-Core Intel Core i7) time, the final optimality gaps, the total iteration number as well as the CPU time per iteration.

Observation 1. We first note that "c0" and Tan et al. (1979) essentially solve the same models. Compared with the previous approach, our methods can provide more accurate solutions (see "c1" and "c10"). Particularly, the optimality gap induced by "c10" is almost the same as "tad" and "aid", the two approximation schemes proposed in the machine learning literature. Meanwhile, it is just slightly larger than the two exact methods, namely, "id" and "ad".

Observation 2. The total CPU time required by "c10" is significantly lower than "ad", "tad", "id" and "aid". Specifically, the total number of iterations increases by 5-6 times, but the CPU time per iteration is reduced from 2.5-5.5s to just 0.007s (roughly 400 times). Hence, "c10" is more efficient.

Observation 3. The single-looped scheme "sid" is more special. Its CPU time per iteration is obviously much lower than "ad", "tad", "id" and "aid" because it also bypasses the difficulty in repeatedly solving for the lower-level solution. However, it is still 30 times higher than our approach, mainly because it still builds on implicit differentiation, which involves storing and inverting large matrices.

[1] https://github.com/bstabler/TransportationNetworks/tree/master/SiouxFalls

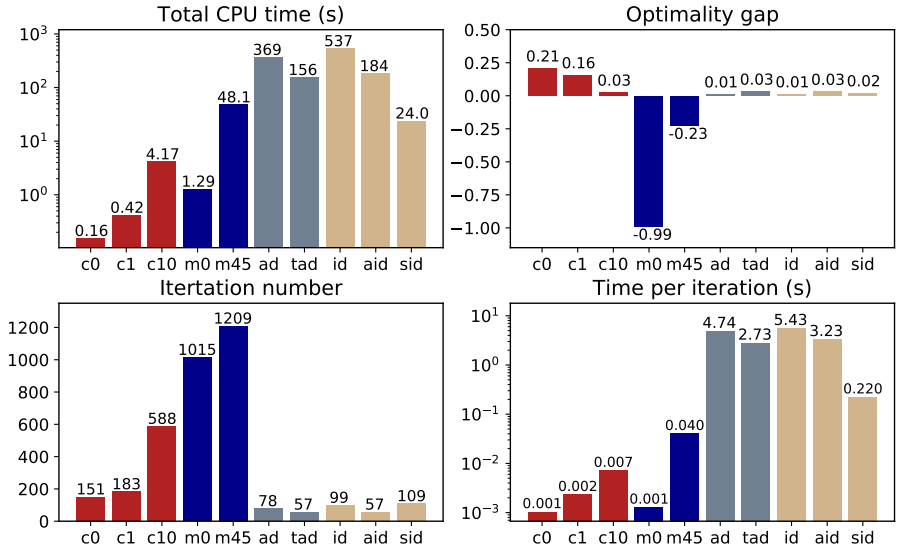

Figure 4. Comparison between our methods and previous methods.

Observation 4. The lower bound provided by "m45" is not that accurate compared with the upper bound provided by "c10". Nevertheless, it answers how good the upper bound is. Specifically, "(upper bound - lower bound) / lower bound" would be a reasonable estimation of the accuracy. If it is smaller than the tolerance, then we are sure that we already find a sufficiently good solution. To the best of our knowledge, no previous method in the machine learning literature can provide such a lower bound for bilevel programs. The classic scheme (Dantzig et al., 1979) in the optimization literature could provide such a bound (see "m0"). However, the gap between the accurate solution and this lower bound is too large. If we use this lower bound to evaluate the upper bound, it may be over too pessimistic.