BILEVEL LEARNING*

RICCARDO GRAZZI†, MASSIMILIANO PONTIL‡, SAVERIO SALZO§, AND ALAIN ZEMKOHO¶

Abstract. Bilevel learning refers to machine learning problems that can be formulated as bilevel optimization models, where decisions are organized in a hierarchical structure. This paradigm has recently gained considerable attention in machine learning, as gradient-based algorithms built on the implicit function reformulation have enabled the computation of large-scale problems involving possibly millions of variables. Despite these advances, the implicit function framework relies on restrictive assumptions, notably the requirement that the lower-level problem admit a unique optimal solution for each upper-level decision. Moreover, the computation of the derivative of the lower-level optimal solution function becomes significantly more involved when the lower-level problem includes constraints. As a result, many existing bilevel learning algorithms are effective only for relatively narrow classes of problems. This paper reviews the main algorithmic ideas underlying recent progress in bilevel learning, highlighting both the key mechanisms responsible for their scalability and the limitations that arise in more general settings. We then draw connections with the broader bilevel optimization literature and discuss algorithmic techniques that may help overcome these limitations. Our aim is to bridge the gap between bilevel learning and classical bilevel optimization, thereby supporting the development of scalable methods capable of solving more general large-scale bilevel programs.

Key words. Bilevel optimization, bilevel learning, gradient descent method, machine learning

MSC codes. 68Q25, 68R10, 68U05

1. Introduction.

Bilevel learning (BL) is a research area that lies at the intersection of *bilevel optimization* (BO) and *machine learning*. It focuses on machine learning problems with an inherent hierarchical structure that can be naturally modeled using the *Stackelberg game* framework. In this setting, two decision makers interact: a leader (or *upper-level player*) and a follower (or *lower-level player*). The defining feature of a Stackelberg game is its sequential decision structure: the leader acts first, and the follower subsequently responds after observing the leader's decision. This sequential order of play distinguishes Stackelberg games from other game-theoretic models, such as Nash or zero-sum games, in which decisions are made simultaneously. Consequently, Stackelberg games exhibit a vertical hierarchy between the players, with the leader occupying the upper level of the decision process. For this reason, they are often referred to as *hierarchical games*, in contrast to the horizontal interaction that characterizes Nash-type games. A detailed introduction to the Stackelberg framework can be found in Heinrich von Stackelberg's habilitation thesis [235] (see also the English translation [247]), as well as in the monograph [65].

To avoid early distraction by the *jungle* of bilevel optimization models, we defer a formal mathematical definition of a bilevel program to Section 2, where different formulations of the problem and related concepts are presented. For the moment, we focus on the main purpose of this paper. When necessary, we distinguish between BO and BL. The motivation for this distinction is that BL is increasingly emerging as a research field in its own right and can therefore be characterized by the tools, approaches, and questions that arise in the design and analysis of solution algorithms. In particular, while BL is largely defined by the *scale* of the problems it addresses, the traditional BO literature has been primarily concerned with the mathematical correctness of methods and theoretical properties of solution concepts. As a result, many BO approaches are tractable only for relatively small problem instances.

A key distinction between BL and BO therefore lies in the scale of the problems consid-

*Submitted to the editors DATE.

Funding: This work was funded by EPSRC through a grant with reference EP/X040909/1, and in part by the European Union, under Projects Numbers 101070617 and 101120237.

†Microsoft Research Cambridge, Cambridge, UK (riccardo.grazzi.4@gmail.com).

‡Computational Statistics and Machine Learning, Italian Institute of Technology, Genova, Italy & AI Centre, Department of Computer Science, UCL, London, UK (m.pontil@cs.ucl.ac.uk).

§DIAG, Sapienza University of Rome, Via Ariosto, 25, 00185 Roma, Italy & Computational Statistics and Machine Learning, Italian Institute of Technology, Genova, Italy (salzo@diag.uniroma1.it).

¶School of Mathematical Sciences, University of Southampton, UK (a.b.zemkoho@soton.ac.uk).

ered. Although many state-of-the-art BL methods are rooted in classical BO frameworks, the problems arising in BL can involve extremely large numbers of variables. For instance, the neural architecture design problem is modeled and solved in [174] as a bilevel optimization problem, where the resulting algorithm is applied to the CIFAR-10 and ImageNet datasets involving millions of parameters. Similarly, [185] develops a gradient-based algorithm for bilevel hyperparameter optimization in machine learning training and demonstrates its effectiveness on problems involving millions of weights and hyperparameters.

In contrast, many algorithms in the current BO literature remain largely conceptual or are designed primarily for small-scale problems (see, e.g., [76]). Consequently, their direct applicability to large-scale BL settings is often limited. The *first objective* of this paper is therefore to highlight, for the BO community, the algorithmic machinery that has recently enabled implicit-function–based gradient methods to scale to machine learning problems involving very large numbers of variables, as illustrated in the examples above. Note that the implicit function model, which underlies many state-of-the-art BL methods, consists essentially of substituting the lower-level optimal solution function, when it is well-defined as a vector-valued function, into the upper-level objective function. This single-level reformulation then enables the development of gradient-based algorithms.

It is worth recalling that the implicit function approach was already introduced in the early 1990s as a method for solving BO problems. In fact, it forms the basis of the principal solution approaches presented in two classical monographs on bilevel optimization, namely [65, 203]. However, this framework did not gain widespread practical adoption, largely due to the difficulties involved in guaranteeing the existence of the lower-level optimal solution function and computing its Jacobian when it exists; see Sections 5 and 6 for further discussion. What has recently enabled the implicit function model to become a mainstream approach in BL is the introduction of derivative approximation approaches, inspired by advances in *automatic differentiation* techniques, tools that have played a central role in the development of modern deep learning methods; see, e.g., [14, 115].

Beyond derivative approximation techniques, another major departure from classical BO methods is the use of approximate solutions of the lower-level problem. In traditional BO approaches, the lower-level problem is typically assumed to be solved exactly, often to global optimality. In contrast, BL methods frequently rely on approximate solutions obtained through iterative optimization procedures. The recent resurgence of the implicit function framework in BL therefore comes with certain modeling simplifications. In particular, lower-level constraints are often avoided in order to facilitate the computation of gradient descent directions. However, many bilevel learning problems cannot realistically be formulated with unconstrained lower-level problems; see, e.g., [269, 101, 177, 262, 259].

Moreover, the implicit function reformulation itself can be conceptually problematic, since the lower-level problem can typically admit multiple optimal solutions for some upper-level variable(s). Taking this issue into account—together with the fact that computing the Jacobian of the lower-level optimal solution function typically requires second-order information—the *lower-level value function reformulation* has recently emerged as an important alternative in the BL literature (see, e.g., [171, 156, 155, 100, 269]). A key advantage of this approach is that it requires only first-order information to compute descent directions within gradient-based schemes, as it will be outlined in Section 8.

The *second objective* of this paper is to contribute to the acceleration of such developments by providing researchers in BL with a broad overview of algorithmic techniques from the BO literature. By bridging these two bodies of work, we aim to facilitate the development of enhanced tools capable of efficiently addressing large-scale BL problems that remain beyond the reach of existing approaches. More broadly, the goal of this article is to propose a unified perspective that brings together key algorithmic ideas from both BL and BO, thereby fostering the design of efficient algorithms for very large-scale problems and promoting new theoretical insights into the numerical behavior of bilevel programs.

1.1. Related work. A number of survey and overview papers on bilevel optimization and related topics already exist in the literature. On the BO side, to the best of our knowledge, the most recent and comprehensive collection of surveys is the edited volume [76]. The topics covered in that volume are intentionally broad and include connections with other problem classes such as game theory and multiobjective optimization, as well as general theoretical questions and algorithms designed for particular classes of bilevel optimization problems. In contrast, the BO perspective adopted in this paper focuses primarily on algorithms for continuous bilevel optimization problems, while exploring the opportunities and challenges that arise when attempting to adapt or scale these methods to bilevel learning settings. In addition, several earlier BO surveys and bibliographic reviews have provided high-level overviews of algorithmic developments for particular classes of bilevel optimization problems; see, for example, [231, 57, 144, 66, 246]. On the BL side, a number of survey papers have appeared more recently. These works typically focus on specific machine learning applications of bilevel optimization and on tailored variants of gradient-based algorithms, together with empirical studies of their performance characteristics; see, for instance, [175, 55, 59, 48, 96]. A more recent overview aimed at a broader machine learning audience is provided in [276], which explicitly discusses the connections between bilevel optimization and modern signal processing and machine learning applications.

1.2. Main contributions. In summary, the main contributions of this paper are as follows: (i) to provide an overview of state-of-the-art numerical schemes tailored to BL that have been developed to address medium- to large-scale problems, while highlighting the key features that drive their efficiency; (ii) to identify the limitations and weaknesses of existing BL algorithms, thereby enabling researchers in BO interested in machine learning applications to better understand the challenges and shortcomings of current approaches; (iii) to present a collection of state-of-the-art algorithmic techniques from the broader BO literature for general bilevel programs, allowing BL researchers to quickly familiarize themselves with potential methodological directions for addressing problems that current BL algorithms cannot effectively solve; and (iv) to provide a reference resource for researchers interested in (or beginning work on) bilevel optimization and its applications in machine learning.

1.3. Organization of the paper. In the next section, we introduce the mathematical description of the *bilevel optimization* concept and its different formulations. This step is necessary because the term bilevel optimization is used to describe a variety of problems that may differ in structure while still preserving the defining characteristics of a Stackelberg game model, as outlined above. It is therefore useful to begin by presenting the main variants of the problem and clarifying how they are related and how they build upon one another. Subsequently, in Section 3, we provide a brief historical overview of BO and illustrate how many problems arising in BL naturally exhibit a Stackelberg structure, even though this connection was not always explicitly recognized or formally linked to the BO literature.

For readers who are less familiar with applications of BO in machine learning, Section 4 presents a concise overview of bilevel programming applications in machine learning (i.e., an overview of *BL problems*). We also discuss in greater detail two representative examples—hyperparameter optimization and adversarial learning—which correspond to special classes of optimistic and pessimistic bilevel programs, respectively (these concepts will be formally defined in Section 2). Section 5 focuses on the state-of-the-art methods currently used in BL, which are largely built upon the implicit function reformulation. This framework relies on the well-posedness of the lower-level problem, typically requiring the existence of a unique optimal solution for every upper-level variable. We discuss the main assumptions underlying this approach—such as strong convexity and the absence of lower-level constraints—as well as the algorithmic mechanisms that have enabled its practical success, including approximation strategies for derivatives and lower-level solutions. Convergence paradigms and performance evaluation techniques are also reviewed in this section.

In Section 6, we examine several important limitations and challenges associated with the state-of-the-art methods described in Section 5, and discuss possible directions for over-coming these difficulties. Section 7 then provides an overview of algorithmic techniques from the BO literature that can be used to address problems in which the lower-level problem is convex—but not necessarily strongly convex—and may include constraints. Subsequently, in Section 8, we relax the convexity assumption on the lower-level problem and explore potential avenues for developing algorithms for BO that rely purely on first- or second-order information. Such approaches are not directly applicable within the framework discussed in Section 5, since the Jacobian of the lower-level optimal solution function typically requires second-order information about the lower-level problem.

Overall, the study in this paper is mainly focused on the *standard optimistic* bilevel optimization problem (see next section for the definition), and the algorithmic discussions throughout Sections 5–8 are dedicated to different transformations of the model; namely, the implicit function, Karush-Kuhn-Tucker, and lower-level value reformulations. In Section 9, a comparison of these different approaches is provided. Finally, Section 10 concludes the paper with a set of observations and perspectives on how the ideas discussed in Sections 7 and 8 could be further explored in the context of BL, where they remain largely underdeveloped.

2. What is a bilevel optimization problem?.

This is possibly one of the most fascinating questions, as there are multiple concepts labeled as/or related to *bilevel optimization*, depending on what interpretation one makes, or also the specific area of application. The first and most widely understood, and which will also be at the center of the attention of this paper, because it is the one generally used in machine learning, is the problem

$$\text{(BOP)} \qquad \min_{x \in X} F(x,y) \;\text{ s.t. }\; y \in S(x) := \operatorname*{argmin}_{y \in Y(x)} f(x,y),$$

where the functions $F : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}$ and $f : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}$ represent the upper- and lower-level objective functions, respectively. Similarly, $X \subset \mathbb{R}^n$ is the upper-level feasible set, while the set-valued mapping $Y : X \rightrightarrows \mathbb{R}^m$ describes the lower-level feasible set. Overall, (BOP) represents the upper-level (or leader's) problem, while the set $S(x)$ collects all the optimal solutions of the lower-level (or follower's) problem (for all $x \in X$):

$$\text{(LL)} \qquad \min_{y} \; f(x,y) \quad \text{s.t.} \quad y \in Y(x).$$

Problem (BOP) will be said to be well-posed if one assumes that for any choice $x \in X$ of the leader, the follower has a single optimal solution. Precisely, this means that the following condition, where $|C|$ stands for the cardinality of the set C, is satisfied:

$$\{x \in X : \; |S(x)| = 1\} = X; \tag{2.1}$$

i.e., we have $S(x) = \{y(x)\}$ for all $x \in X$ with $y(\cdot) : X \to \mathbb{R}^m$ being the optimal solution function of the lower-level problem. In this case, problem (BOP) reduces to

$$\text{(P}_\text{i}\text{)} \qquad \min_{x \in X} \mathcal{F}(x) := F(x, y(x)).$$

This model, known in the classical bilevel optimization literature as *implicit function reformulation* (see [150] for one of the very first studies based on the approach) is the state of the art working framework in *bilevel learning*.

However, there is also another very rich set of solution concepts for (BOP) when the lower-level optimal solution set-valued mapping S satisfies the condition

$$\{x \in X : \; |S(x)| > 1\} \neq \emptyset. \tag{2.2}$$

This corresponds to the situation where the lower-level problem has more than one optimal solution for some choices of the upper-level player. In the context of (2.2), there are two radically opposed solution concepts for problem (BOP). The first, and more commonly used one, is to assume that each time the leader picks an $x \in X$ that leads to $|S(x)| > 1$, the follower selects a value $y \in S(x)$ that is in favorable to the leader. This leads to the (original) *optimistic* bilevel optimization problem

$$(\mathrm{P_o}) \qquad \min_{x\in X} \min_{y\in S(x)} F(x,y),$$

also known as the cooperative model. The optimistic bilevel program is the most studied class of the problem; however, investigations are almost never on the formulation $(\mathrm{P_o})$, but rather on the following version of the problem, labeled in [73, 271] as *standard* optimistic bilevel optimization problem in opposition to the original optimistic $(\mathrm{P_o})$:

$$(\mathrm{P}) \qquad \min_{x,\,y} F(x,y) \;\text{ s.t. }\; x \in X, \;\; y \in S(x).$$

It can be seen that here, full control over the leader and follower's variables x and y is given to the upper-level player. Although it might sound intuitive that problems $(\mathrm{P_o})$ and (P) are equivalent, it was shown in [73] that this is true only if global optimal solutions are considered. But locally, both problems are not equivalent. For a local optimal solution $\bar{x}$ of problem $(\mathrm{P_o})$, any point $(\bar{x}, \bar{y})$ with $\bar{y} \in S(\bar{x})$ is a local optimal solution of problem (P). However, if $(\bar{x}, \bar{y})$ is locally optimal for problem (P), one needs the set-valued mapping

$$S_{\mathrm{o}}(x) := \operatorname*{argmin}_{y\in S(x)} F(x,y)$$

to be *inner semicontinuous* at $(\bar{x}, \bar{y})$ to ensure that $\bar{x}$ is locally optimal for problem $(\mathrm{P_o})$. This is quite a strong assumption; for its definition, and more details on the relationship between these two problems, interested readers are referred to [73].

The second option, which is a bit less researched, is to assume that each time the leader picks an $x \in X$ such that $S(x) > 1$, the follower selects a value $y \in S(x)$ that is antagonistic to the leader. To anticipate on unfavorable choices from the follower, the leader solves the so-called *pessimistic* bilevel optimization problem

$$(\mathrm{P_p}) \qquad \min_{x\in X} \max_{y\in S(x)} F(x,y).$$

This is a more challenging problem to solve, and in the hope of promoting the implementation of ideas from the abundant literature on the standard optimistic bilevel program (P), the *standard pessimistic* bilevel optimization problem

$$(2.3) \qquad \min_{x,\,y} F(x,y) \;\text{ s.t. }\; x \in X, \;\; y \in S_{\mathrm{p}}(x)$$

was recently investigated in [160]. Note that here, the *two-level* optimal solution set-valued mapping $S_{\mathrm{p}} : X \rightrightarrows \mathbb{R}^m$ is defined by

$$S_{\mathrm{p}}(x) := \operatorname*{argmax}_{y\in S(x)} F(x,y).$$

A combination of *Karush-Kuhn-Tucker* and some *lower-level value function*–type reformulation (see Section 7 and Section 8 for some relevant details) is used to address S_{p} in order to get a single–level transformation of problem (2.3).

The optimistic and pessimistic bilevel optimization problems $(\mathrm{P_o})$ and $(\mathrm{P_p})$, respectively, represent two very extreme positions, either the leader and follower cooperate (optimistic problem) or they do not (pessimistic problem). For this reason, a number of works (see, e.g., [44, 157]) have suggested a compromise model, where in a nutshell, if the follower has multiple options to pick from, for some choices of $x \in X$, they select one that represents a compromise between the leader and the follower; i.e., the problem to be solved is

$$(\mathrm{P_{op}}) \qquad \min_{x\in X} \lambda\varphi_{\mathrm{o}}(x) + (1-\lambda)\varphi_{\mathrm{p}}(x),$$

with $\lambda \in [0, 1]$ denoting the degree of cooperation between the upper- and lower-level players [228]. In problem (P_{op}), the functions φ_o and φ_p are respectively defined by

$$\varphi_o(x) := \min \{F(x, y) : y \in S(x)\} \text{ and } \varphi_p(x) := \max \{F(x, y) : y \in S(x)\}. \tag{2.4}$$

Labeled as *two-level* optimal value functions and studied in detail in [73], where suitable conditions for their local Lipschitz conditions are established.

Finally, [2, 1] introduces the following *weak-strong Stackelberg/bilevel optimization* problem as a generalization of the above optimistic and pessimistic bilevel programs:

$$\min_{x \in X} \min_{y \in S(x)} \max_{z \in S(x)} \mathfrak{F}(x, y, z). \tag{2.5}$$

Here, the function $\mathfrak{F} : \mathbb{R}^n \times \mathbb{R}^m \times \mathbb{R}^m \to \mathbb{R}$ represents the upper-level objective function, while $S : X \rightrightarrows \mathbb{R}^m$ describes the lower-level optimal solution set-valued mapping defined in (BOP). It might be useful to recall that the optimistic (resp. pessimistic) bilevel program (P_o) (resp. (P_p)) is also called *weak* (resp. *strong*) Stackelberg/bilevel optimization problem. Hence, the reason why the problem is referred to as weak-strong Stackelberg problem. Similarly, it could therefore be labeled as *optimistic-pessimistic* bilevel optimization problem. To see why this vocabulary makes sense, observe that if $\mathfrak{F}(x, y, z) := F(x, y)$, we get the original optimistic model (P_o), while having $\mathfrak{F}(x, y, z) := F(x, z)$ leads to the pessimistic bilevel optimization problem (P_p). Moreover, setting $\mathfrak{F}(x, y, z) := \lambda F(x, y) + (1-\lambda)F(x, z)$, we can observe that we get the partial cooperation model in (P_{op}) for a fixed $\lambda \in [0, 1]$.

3. A short (possibly) shared history.

Bilevel optimization emerged from the habilitation thesis of von Stackelberg in 1934 [235]. First, it attracted interest mainly from economists (see, e.g., [130, 151, 49]; for a thorough economic perspective on von Stackelberg's work, see Volume 23 Issue 5/6 of the Journal of Economic Studies specifically dedicated to his research contribution and its influence) until 1973 when it was introduced to the field of mathematical optimization by Bracken and McGill [36]. It is however important to note that the problem introduced in Bracken and McGill's first paper on the subject instead corresponds to what is known today as a semi-infinite programming problem [122]. Candler and Norton in their reports [41] and [42] might have been the first to connect the dots between the bilevel optimization model that emerged in the mathematical optimization literature and the work of von Stackelberg, and can also be credited for coining the expression "bilevel optimization" (namely, multilevel for optimization problems with more than two levels).

The pioneering works of operations research and mathematical optimization experts from around the early 70s came with two important things: (i) the emergence of applications of bilevel optimization outside of economics and pure game theory; for example, the initial works of Bracken and McGill focused on military and defense applications [36, 37], while Candler and Norton highlighted the applicability of the problem in areas such as engineering, biology, and policy design and implementation in agriculture [41, 42]. (ii) A huge interest in the development of solution algorithms to solve bilevel optimization problems. Works on solution algorithms started to intensify around the early 1980s, as it can be seen in this first survey on the subject by Kolstad [148] in 1985. Overall, the algorithms developed so far could be generally classified in the four categories outlined in following subsections.

3.1. Methods for bilevel programs with fully linear functions.

The problem (BOP) will be said to be fully linear if the functions F and f are linear in (x, y) and the sets X and $Y(x)$, for all $x \in X$, are defined by functions that are linear in x and (x, y), respectively. Problem (P) in this case has an interesting geometric structure, in the sense that at least one optimal solution of the problem, if one exists, occurs at a vertex of the polyhedron the set $(X \times \mathbb{R}^m) \cap \text{gph}Y$. Note that here and in the sequel, for a set-valued mapping $\Gamma : \mathbb{R}^n \rightrightarrows \mathbb{R}^m$, its *graph* is defined by

$$\text{gph}\,\Gamma := \{(x, y) \in \mathbb{R}^n \times \mathbb{R}^m \mid y \in \Gamma(x)\}.$$

This result, first discovered by Bials and Karwan [29], has been the basis for the development of multiple enumerative algorithms for fully linear bilevel programs; see, e.g., [43, 208, 201]. Unfortunately, as we will see in the general overview of bilevel learning problems, it is not clear that there are any applications of a fully linear model in the context machine learning, except perhaps that it might happen that for some problems, such cases could appear as subproblems for some algorithmic problems given that, for example, as it will be clear in Section 5, the process of computing the directional derivative for the lower-level problem can be seen as solving a quadratic linear bilevel program.

3.2. Algorithms tailored to problems with convex lower-level problems. The lower-level problem (LL) is said to be convex if the objective function $f(x,.)$ and feasible set $Y(x)$ are both convex for all $x \in X$. Of course, the fully linear case above is a special case of this one, but in this more general setting, the construction of numerical methods has generally relied on the so-called Karush-Kuhn-Tucker (KKT) reformulation, which replaces the lower-level problem in (P) by its corresponding KKT conditions. If the lower-level problem has inequality constraints, this reformulation leads to a problem with complementarity constraints, which introduces a very high-level of complexity in the problems.

Various methods have been proposed to solve this problem, with the main focus usually being on how to handle the complementary constraints; early related papers include the work of Fortuny-Amatand McCarl [94], who introduce the famous big M method, which has been very influential in the field for many year, the work of Bard and Falk [12], which introduces a branch and bound–type method, which consists to solve convex approaximations of the KKT reformulation at each iteration, as well as the work by Bialas and Karwan [28], where a pivot-type method is developed to compute an approximation of the KKT reformulation as a problem of finding the solution of a mixed-complementary system. As it will be shown in Section 7, despite being fundamentally a nonconvex constrained optimization problem, the KKT reformulation has a potential for bilevel learning problems, as it is less restrictive in terms of the required framework.

3.3. Techniques based on strongly convex lower-level problems. This is the framework that enables the implicit function model ($\mathrm{P_i}$) to be well-defined, and has been the main based for the development of numerical algorithms in the context of BL. One of the main motivations of this paper has come from witnessing the huge interest that the implicit model ($\mathrm{P_i}$) has attracted in the context of solving bilevel programs appearing in machine learning. The resurgence of methods based on the implicit has been a source of curiosity, as progress on the use of such methods seemed to have stalled in the more general field of BO. Considering the performance of this approach and the depth of analysis of the corresponding algorithms in BL, we aim to identifying the reasons of this success and draw attention to lessons that could be learned for other applications of BO. Before we come back to the technicalities of such methods and reasons for their success in Section 5, we would like to point out that gradient descent approach, which has been the main algorithmic technique in this context, has been in existence in bilevel optimization since the early development of mathematical optimization–based numerical methods for the problem.

The PhD thesis of de Silva [63] completed under the supervision of Garth McCormick, who was at the forefront of the development of sensitivity analysis for optimal solutions of parametric optimization problem, was probably the first work in this context, implementing mainstream implicit function results for the calculation of $\nabla y(x)$ for a gradient descent scheme for a problem of the form ($\mathrm{P_i}$); see the paper [80] with some of the related results. Around the same period, Shimizu and Aiyoshi [229] propose another gradient descent scheme, where a barrier approach is used to eliminate lower-level constraints, before an implicit function technique is applied to the resulting Fermat rule of the new penalized problem. However, the work of Kolstad and Lasdon [150], well-known for a gradient descent scheme for problem ($\mathrm{P_i}$), is probably the first article that introduced an approximation approach for $\nabla y(x)$, which was then applied to solve relatively large size problems at the time (see details in [149]). Bundle methods for ($\mathrm{P_i}$) have also been very prominent in solving problem ($\mathrm{P_i}$), and represent the main focus of the book [203]; it was also an important component of the exposition in other books on bilevel optimization such as [11, 65].

3.4. General bilevel program without any explicit convexity requirement. In this case, the natural way to transform problem (P) into a numerically tractable problem has been through the lower-level value function consisting to replace the inclusion $y \in S(x)$ by its definition. Initial ideas in this context can be traced back to [38, 103]. More recent effort in this include the works [194, 252, 147, 209], which largely exploit the connection of the value function reformulation to semi-infinite programming to build methods to compute global optimal solutions. Considering the underlying techniques, the approaches in these papers are unlikely to scale well to the size of problems encountered in machine learning. This is because the schemes in the latter articles either rely on some branch-and-bound techniques ([194, 147, 209]) or semi-infinite programming–type discretization techniques ([194, 252, 147, 209]). A second class of method that has emerged recently in the literature (see [273, 89, 93]), building on standard continuous nonlinear optimization theory, and to be overviewed in Section 8, has more potential in the context of bilevel learning.

3.5. Some bilevel learning history and connections to bilevel optimization. The BL history can go as far behind as well, if we consider the multiple problems that are now well-understood as bilevel optimization problems, but had stayed in the shadows of the field for a very long time. For example, in Bengio's paper [20], where he studies the gradient–based approach for hyperparameter optimization in machine learning, which is simply an early version of the now classical gradient descent method for the implicit function model ($\mathrm{P_i}$), discussed in Section 5 of the bielevel optimization problem, the Akaike Information Criterion (AIC) model is referred to as one of the long standing techniques for hyperparameter computation [3]. If we look closely at the AIC model in statistics, one of its applications is for cross-validation in the context of training the autoregressive integrated moving average (ARIMA) model for forecasting, it is used to find the order (p, d, q), and this problem can be written as the bilevel program

$$\min_{p,d,q} \mathrm{AIC}(p, d, q, \theta, \phi) \ \text{ s.t. } \ (\theta, \phi) \in \operatorname*{argmin}_{\theta,\phi} f(p, d, q, \theta, \phi), \tag{3.1}$$

where the order $(p, d, q) \in \mathbb{N}^3$ and θ and ϕ represent the AR and MA model parameters, respectively. The AIC model dates back to 1974 [3] and even if it has probably not yet been written in the form (3.1), it can naturally be translated as such. Similar BO models can be written to compute hyperparameters in a wide range of statistics problems, including design of experiments [84], regression analysis [83], as well the estimation of parameters in various areas of engineering; see, e.g., [193, 32, 107]. It might be worth mentioning here that hyperparameter optimization in machine learning is one of the most prominent areas of applications of BO in machine learning, as we will discuss in the next section.

In terms of direct connections between machine learning and BO, it seems like links started to become clearer in the works of Bengio [20] and Chapelle et al. [47], where they developed gradient descent–type algorithms for hyperparameter computation based on the implicit function model (see Fig. 1), where the implicit function model is used, which is now the classical framework for bilevel learning algorithms, as it will be discussed in Section 5.

It must be said that in the articles [20] and Chapelle et al. [47], there is no mention of bilevel optimization or Stackelberg games; moreover, no relevant articles on the subject seems to be mentioned. So, their works could potentially be cast as independent discoveries of bilevel optimization. To the best of our knowledge, Kristin Bennett and her co-authors in a series of papers mainly on hyperparameter optimization for support vector machines [21, 25, 152, 22, 23, 153, 195, 24] can be credited as the first to clearly make the connection between bilevel learning and bilevel optimization. Since the publication of this series of papers between 2006 and 2010, bilevel optimization has literally exploded in machine learning, and seems to be have become an area of research in its own right, within the field of machine learning. To illustrate this, we can mention, for example, the surveys [175, 59, 48, 275], just

1. Initialize $\boldsymbol{\theta}$ to some value.
2. Using a standard SVM algorithm, find the maximum of the quadratic form W:

$$\boldsymbol{\alpha}^0(\boldsymbol{\theta}) = \arg\max_{\boldsymbol{\alpha}} W(\boldsymbol{\alpha}, \boldsymbol{\theta}).$$

$$\boldsymbol{\theta}^0 = \arg\min_{\boldsymbol{\theta}} T(\boldsymbol{\alpha}^0, \boldsymbol{\theta})$$

3. Update the parameters $\boldsymbol{\theta}$ such that T is minimized. This is typically achieved by a gradient step (see below).
4. Go to step 2 or stop when the minimum of T is reached.

Fig. 1: Algorithm for bilevel hyperparameter optimization by Chapelle et al. [47]

in the last 5 years, and a couple of hundreds of papers on the subject have been published in the last few years. So many applications of bilevel optimization have been discovered in machine learning; see next section for a sample of them.

A key focus of this paper is to look closely at the technical aspects of the main BL algorithmic framework, in order to highlight what can be learned from it to tackle other bilevel programs. Conversely, as most algorithms in the current BL literature are based on the implicit function model, this paper aims to draw the attention of BL researchers to the multitude of approaches that could be used to address a wide range of problems for which the model $(\mathrm{P_i})$ is not well-defined, because of the failure of condition (2.1).

4. A flavor of bilevel optimization applications in machine learning. There is a wide range of applications of BO in machine learning, and the number has been growing steadily in recent years; see, e.g., [175, 55, 59, 48] for surveys focused around applications. Our aim here is to give a flavor of how typical such applications look like, while focusing on where the challenges of BL problems lie. Recall that the major difference that sets apart these applications from the ones usually considered in the classical BO literature is the scale of the considered problems. Indeed, the number of lower-level variables, the number of samples in the dataset, and in some cases even the number of upper-level variables can reach millions or even billions [97, 185, 206]. In this setting, scalability becomes a core priority for the development of practically relevant algorithms.

We first focus our attention on two standard examples representing the extreme modeling viewpoints discussed in Section 2 occurring when condition (2.2) holds; i.e., the optimistic and pessimistic models, represented by the *hyperparameter optimization* and *data poisoning* problems, respectively. Let $\mathcal{D}_{\mathrm{tr}}$ denote the *training set* used to optimize the lower-level model parameters $y \in \mathbb{R}^m$ (e.g., neural network weights), and let $\mathcal{D}_{\mathrm{val}}$ denote the *validation set* used to evaluate the upper-level decision variables $x \in \mathbb{R}^n$ (e.g., hyperparameters). In the supervised setting, the datasets consist of input-target pairs; for instance, $\mathcal{D}_{\mathrm{val}} = \{\xi_i\}_{i=1}^{N_{val}}$ where each $\xi_i = (u_i, v_i)$ represents a data point u_i and its label v_i for $i = 1, \ldots, N_{val}$.

4.1. Hyperparameter optimization in machine learning. In a hyperparameter optimization (HPO) problem, the goal is to select hyperparameters $x \in X$ that minimize a validation loss F (upper-level objective function), subject to the model parameters y minimizing a training loss f (lower-level objective function). In this cooperative setting, the leader assumes the follower selects the $y \in S(x)$ most favorable to the upper-level objective. The optimistic formulation of the HPO problem reads as

$$\min_{x \in X} \inf_{y \in S(x)} F(x, y) := \sum_{\xi \in \mathcal{D}_{\mathrm{val}}} \ell_{\mathrm{val}}(y, x; \xi), \tag{4.1}$$

where $S : X \rightrightarrows \mathbb{R}^m$ describes the optimal solution set-valued mapping of the training (or lower-level) problem; to be more precise,

$$S(x) := \underset{z \in \mathbb{R}^m}{\arg\min} f(x, z) := \sum_{\zeta \in \mathcal{D}_{\mathrm{tr}}} \ell_{\mathrm{tr}}(z, x; \zeta) + \mathcal{R}(z, x). \tag{4.2}$$

Here, $\mathcal{R}$ is a regularizer (e.g., ℓ_2 regularization). Here, $\zeta \in \mathcal{D}_{\mathrm{tr}}$ represents a training sample pair. The functions ℓ_{tr} and ℓ_{val} typically measure the discrepancy between the model prediction and the true label. Common choices for the validation loss ℓ_{val} include the Squared Error for regression tasks or the Cross-Entropy loss for classification. The key distinction is that ℓ_{tr} is evaluated on the training set and is often modulated by x. A prominent example is *data reweighting* [217], where x assigns a weight to each training sample, leading to the expression $\ell_{\mathrm{tr}}(z, x; \zeta_i) = \sigma(x_i)\ell(z; \zeta_i)$ (where σ ensures positivity), allowing the leader to downweight noisy or corrupted data.

The term $\mathcal{R}(z, x)$ represents a regularization term explicitly controlled by the hyperparameters. A standard example is weight decay, where $\mathcal{R}(z, x) = \sum_{i=1}^m \exp(x_i) z_i^2$, with x acting as the log-regularization coefficient for each weight. In practice, X often includes continuous and discrete parameters such as *regularization coefficients* and *architecture depth*. Seminal works have applied this framework to kernel methods and Support Vector Machines (SVMs), where the lower-level problem often enjoys convexity [47, 138, 23]. Other optimistic BL problems can be found here [182, 181, 233, 262, 255]. More general gradient-based approaches for continuous hyperparameters have been explored in [210, 97].

Historically, hyperparameter optimization was predominantly performed using Grid Search or Random Search. While effective for a small number of hyperparameters (e.g., $n < 10$), these "black-box" methods scale exponentially poorly with dimension. The BO perspective represented a paradigm shift by utilizing gradient information from the lower-level (via implicit differentiation or iterative differentiation). This allows for the simultaneous optimization of thousands of hyperparameters, such as assigning a unique regularization weight to every individual parameter or learning rate schedules, which is computationally intractable for search-based methods. This advancement has materialized into practical tools, with open-source libraries such as *Betty* [54], *Higher* [114], *TorchOpt* [216], and *JAXopt* [31] which leverage modern deep learning libraries that support automatic differentiation to enable efficient gradient-based hyperparameter optimization and meta-learning.

4.2. Data poisining attacks. For data poisoning attacks, it is important to note that in safety-critical scenarios, the leader (attacker) aims to maximize the learner's error by corrupting a subset of the training data. This creates a zero-sum game where the leader must guard against the follower (learner) selecting the robust solution $y \in S(x)$ that minimizes the damage. This corresponds to the pessimistic (or max-min) formulation:

$$\max_{x \in X} \inf_{y \in S(x)} F(x, y) := \sum_{\xi \in \mathcal{D}_{\mathrm{val}}} \ell(y; \xi). \tag{4.3}$$

In this case, the optimal solution set-valued mapping, $S : X \rightrightarrows \mathbb{R}^m$, of the training (lower-level) problem is defined by

$$S(x) := \underset{z \in \mathbb{R}^m}{\arg\min} f(x, z) := \sum_{\zeta \in \mathcal{D}_{\mathrm{clean}}} \ell(z; \zeta) + \sum_{x_k \in x} \ell(z; x_k). \tag{4.4}$$

Here, x represents the set of poisoned data points injected by the attacker, and x_k denotes the k-th individual poisoned sample within that set, whereas the lower-level variable y represents the *model parameters.* The set X defines the feasible attack space, often bounded to ensure the poisoned data remains imperceptible or within valid input ranges.

The pessimistic assumption ensures the attack is effective even against the best-case response of the learner. Seminal works in optimization-based poisoning include [192, 200], while further pessimistic BL problems can be found in [241, 170, 176, 129, 180, 179, 244]. Gradient-based approaches for pessimistic BL are explored in [116, 129].

It is worth noting that the literature often presents two sides of this coin. In *Data Poisoning*, as formulated above, the Attacker is the Leader (maximizing error) and the Learner is the Follower. Conversely, in *Adversarial Training*, the Learner is the Leader (minimizing error) who anticipates the Attacker (Follower) finding the worst-case perturbation. Both formulations are valid depending on whose perspective is being optimized. While the leader and follower clearly do not cooperate in these settings, many practical approaches rely on implicit function-based gradient methods. This is often the case because it is cheaper than using the pessimistic bilevel approach and it is mathematically justified when the lower-level problem has a unique solution (e.g., due to strong convexity or regularization). In such cases, the pessimistic and optimistic formulations coincide, $S(x)$ becomes a singleton, and the standard hypergradient derivations apply. For more details on this subject, interested readers are referred to [19, 39, 137].

4.3. Further applications and structural challenges. Beyond these examples, BO unifies diverse machine learning tasks [175]. *Meta-learning* fits this structure naturally, where the outer loop optimizes a meta-learner for fast adaptation on inner-loop tasks [97, 87]. Recently, BO has also become increasingly relevant for Large Language Models (LLMs). Architectures like *Titans* [16] and the *Nested Learning* framework [15] model long-term memory updates as an inner optimization loop. Similarly, *Test-Time Training* [237, 239] enables adaptation to long contexts via autoregressive training on the input sequence during inference, and in this context pretraining is similar to solving a BO problem where the lower-level is solved in an autoregressive fashion. *Deep equilibrium models* (DEQs) rely on fixed-point iterations, where training involves differentiating through the equilibrium state, a process mathematically equivalent to solving a bilevel program [10]. Similarly, *generative adversarial networks* formulate a min-max game between a generator and discriminator, often treated as a bilevel program [109, 211].

The practical application of BO in machine learning faces significant structural difficulties. For the remainder of this section, we discuss some structural challenges associated to BL problems. We start with the *high dimensionality and large-scale datasets.* A defining characteristic of modern machine learning is the scale of the problem. The lower-level variable y is almost invariably high-dimensional. In deep learning, y represents millions of neural network weights; in kernel methods, the number of dual variables scales with the dataset size. This makes it necessary to use approximate methods to compute the solution of the lower-level problem. The dimensionality of the upper-level variable x varies: it is typically small in classical HPO (< 10) but can be extremely large in Meta-learning or DEQs where x represents network initializations or parameters. Furthermore, the objective functions are defined as averages over massive datasets, precluding exact gradient evaluation and requiring the use of stochastic optimization methods.

Non-uniqueness of the lower-level solution. A central challenge is that the lower-level optimal solution set $S(x)$ is rarely a singleton for $x \in X$. The nature of this non-uniqueness varies by problem class. In the simplest case, such as linear regression with ℓ_2 regularization (Ridge Regression), the lower-level objective f is strongly convex with respect to y, guaranteeing a unique solution $S(x) = \{y(x)\}$. However, in the regime of *over-parameterized* linear models—where the number of parameters d exceeds the number of training samples m_1—if explicit regularization is absent, the objective is convex but not strongly convex. Consequently, $S(x)$ becomes an affine subspace containing infinitely many solutions that all achieve zero training error. The situation becomes even more complex in *deep learning*, where $f(x, y)$ is highly non-convex with respect to the neural network weights y (see, e.g., [164] for some graphical illustrations of the loss functions); In deep learning, non-uniqueness arises not just from flat valleys in the landscape but from fundamental symmetries (e.g., neuron permutation invariance) and the existence of multiple disconnected local minima. In this regime, the optimal solution set $S(x)$ is a disjoint union of manifolds, rendering the bilevel problem significantly harder to analyze.

The selection problem and algorithmic bilevel. This non-uniqueness leads directly to the selection problem: when $S(x)$ contains multiple solutions, which one does the training algorithm actually return? Classical formulations assume the leader can select the best (optimistic) or worst (pessimistic) $y \in S(x)$. In practice, however, the solution y is determined by the specific iterative algorithm used (e.g., Stochastic Gradient Descent) and its initialization. For instance, in over-parameterized linear regression initialized at zero, SGD converges specifically to the minimum Euclidean norm solution among the infinitely many minimizers. In deep learning, the "implicit bias" of the optimizer selects a specific attractor in the landscape. This algorithmic reality has led to "unrolled" formulations where the lower-level minimization is replaced by a dynamical system $y_{t+1} = \Phi_t(y_t, x)$ after T steps, denoted $y_T(x)$. This perspective allows optimizing optimization hyperparameters (e.g., learning rates) and aligns the theoretical formulation with the algorithmic output [110, 9, 188, 95].

Generalization and the test set. It is crucial to distinguish between the objective functions used in the bilevel optimization problem and the ultimate goal of the learning process. The upper-level objective F is typically evaluated on a validation set $\mathcal{D}_{\text{val}}$ to guide the selection of x. However, this is merely a proxy; the true performance metric is the generalization error on a strictly held-out *test set* $\mathcal{D}_{\text{test}}$, which is disjoint from both $\mathcal{D}_{\text{tr}}$ and $\mathcal{D}_{\text{val}}$ and is never used during the optimization of x or y. This distinction highlights the risk of "meta-overfitting", where hyperparameters are tuned to minimize validation error but fail to generalize to the test distribution.

Constrained/nonsmooth/nonconvex lower-level problems. Note that the HPO and data poisoning problems introduced above do not explicitly have lower-level constraints. However, explicit lower-level constraints are fundamental to the formulation of hyperparameter tuning for Support Vector Machines [21, 154], fairness-aware learning [263], sparse structure learning [26, 98], and safety-critical control [248, 141]. Further BL problems with lower-level constraints can be found in [269, 101, 177, 262, 259, 254, 256, 257, 139, 225, 179, 176, 171, 264]. These constraints necessitate the handling of non-differentiable projection operators and the breakdown of strict complementarity in KKT conditions [34], which complicate the estimation of hypergradients. Recent algorithmic literature has addressed these issues in various ways. For example using one-stage primal-dual formulations that update variables simultaneously [135, 233], smoothed implicit gradient approximations [139], and difference-of-convex relaxations for value-function constraints [100]. In BL, the lower-level problem can be nonsmooth (see, e.g., [173, 226, 202, 6, 26, 27]) or even nonconvex (see, e.g., [9, 177, 176, 179, 180, 171, 186]). In such scenarios, the classical algorithmic framework is not applicable, as it will be clear in Sections 5 and 6.

5. Implicit function-based methods for bilevel learning. The implicit function model ($\mathrm{P_i}$) is the framework for the classical algorithms for BL in the current literature. In this section, we examine the main techniques that have been developed so far from this perspective. More precisely, the main algorithmic idea is the gradient descent method tailored to the specific version

$$\min_{x\in\mathbb{R}^n} \mathcal{F}(x) := F(x, y(x)) \quad \text{with} \quad y(x) := \arg\min_{y\in\mathbb{R}^m} f(x,y) \tag{5.1}$$

of problem ($\mathrm{P_i}$), where in addition to the requirement that the lower-level problem has a unique optimal solution (for all upper-level variables), it is assumed that there is no upper– nor lower–level constraints (i.e., with $X := \mathbb{R}^n$ and $Y(x) := \mathbb{R}^m$). While upper-level constraints are generally manageable, incorporating lower-level constraints presents additional challenges which will be discussed in later sections. This formulation is well-defined under the basic assumption that $y(\cdot) : \mathbb{R}^n \to \mathbb{R}^m$ describes a vector-valued function such that $\{y(x)\} = S(x)$ (the optimal solution set is a singleton) for all $x \in \mathbb{R}^n$.

To have a sense of how this is still a very challenging problem, start by noting that we can easily find examples where the upper-level objective function F is convex and the lower-level objective function is strongly convex in the lower-level variable y, but the resulting function $x \mapsto \mathcal{F}(x)$ is still not necessarily convex. For example, let

$$F(x,y) := y \quad \text{and} \quad f(x,y) := \big(y - (x^3 - x)\big)^2 .$$

Clearly, $f(x,\cdot)$ is strongly convex for all $x \in \mathbb{R}$, and we can observe that the resulting function $x \mapsto \mathcal{F}(x) := F(x,y(x)) = x^3 - x$ is nonconvex. Hence, ($\mathrm{P_i}$) is a nonconvex optimization problem, even when all necessary requirements are satisfied for it to be well-defined. Additionally, as it will be clear in Section 7, the assumptions required for ($\mathrm{P_i}$) to be well-defined are generally only local. This means in many practical use cases the model is valid only locally and not necessarily on the whole domain of the reduced objective function $\mathcal{F}$ or the upper-level feasible set X.

We can also consider the stochastic setting where the upper- and lower-lever objective functions are expectations, which model situations where we have access to data sampled from unknown distributions. For such a setup, we have

$$F(x,y) = \mathbb{E}_\xi \hat{F}(x,y,\xi) \quad \text{and} \quad f(x,y) = \mathbb{E}_\zeta \hat{f}(x,y,\zeta), \tag{5.2}$$

where ξ and ζ are random variables. Some works study also the special case where expectations are replaced with their empirical estimates, i.e., they are finite sums

$$F(x,y) = \frac{1}{d_u}\sum_{i=1}^{d_u} \hat{F}(x,y,\xi_i) \quad \text{and} \quad f(x,y) = \frac{1}{d_l}\sum_{i=1}^{d_l} \hat{f}(x,y,\zeta_i). \tag{5.3}$$

As we mentioned in Section 4, in BL, n, m, d_u and d_l can be very large and f and F might be nonconvex: the main example is whenever the lower-level and/or upper-level variables are the parameters of a neural network. In the case where the function f is nonconvex w.r.t. y, the optimal solution set of the lower-level problem is not necessarily a singleton. However, algorithms are still applied in this setup with various degree of success.

5.1. General algorithmic framework. The core idea of the implicit function approaches is to compute the *hypergradient*, i.e. the gradient of $\mathcal{F}$ by exploiting the Implicit Function Theorem, which characterizes the dependence of the lower-level optimal solution on the upper-level variables. More precisely, if for a given x the lower-level problem admits a *unique* local minimizer $y(x)$, and if f is twice continuously differentiable with $\nabla^2_{yy} f(x,y(x))$ invertible, then the first-order optimality condition

$$\nabla_y f(x,y(x)) = 0, \tag{5.4}$$

defines y locally as an implicit function of x. Differentiating this stationarity condition with respect to x gives

$$\nabla^2_{xy} f(x,y(x)) + \nabla^2_{yy} f(x,y(x))\, \nabla y(x) = 0,$$

and, by the Implicit Function Theorem,

$$\nabla y(x) = -\left[\nabla^2_{yy} f(x,y(x))\right]^{-1} \nabla^2_{xy} f(x,y(x)),$$
$$\nabla \mathcal{F}(x) = \nabla_x F(x,y(x)) + \nabla y(x)^\top \nabla_y F(x,y(x)).$$

The gradient-based method that computes the exact hypergradient is described in Algorithm 5.1; see, e.g., [65].

The method relies on the well-defined nature of problem ($\mathrm{P_i}$), where the Jacobian $\nabla y(x^k)$ is computed by inverting the Hessian of the lower-level objective function. The commonly used sufficient condition to make the Jacobian well-defined is for the lower-level objective function to be strongly convex and twice continuously differentiable, which is satisfied, for example, by having an ℓ_2 regularization penalty in combination with a convex and twice differentiable objective such as in regularized logistic regression.

Algorithm 5.1 Classical hypergradient descent algorithm

Require: x^0 and sequence $\{\alpha_k\}$.
1: **for** $k = 0, 1, \dots, K$ **do**
2: $\quad y(x^k) = \arg\min_y f(x^k, y)$
3: $\quad \nabla y(x^k) = -\left[\nabla^2_{yy} f(x^k, y(x^k))\right]^{-1} \nabla^2_{xy} f(x^k, y(x^k))$
4: $\quad \nabla \mathcal{F}(x^k) = \nabla_x F(x^k, y(x^k)) + \nabla y(x^k)^\top \nabla_y F(x^k, y(x^k))$
5: $\quad x^{k+1} = x^k - \alpha_k \nabla \mathcal{F}(x^k)$
6: **end for**

In high-dimensional settings, however, Lines 2 and 3 of Algorithm 5.1 are computationally prohibitive. First, computing the lower-level solution $y(x^k)$ typically requires running an iterative lower–level solver to (near) convergence; in large-scale problems this can mean many gradient/Hessian–vector evaluations (often over large datasets), so the cost scales with both the lower-level dimension and the number of lower–level iterations. Second, the implicit-gradient term involves inverting the hessian. Forming the hessian already costs $\mathcal{O}(m^2)$ memory/time, and a direct factorization (e.g., Cholesky/LU) costs $\mathcal{O}(m^3)$ time and $\mathcal{O}(m^2)$ memory, which quickly becomes infeasible as m grows.

It is important to acknowledge that alternative frameworks exist which bypass the direct computation of the hypergradient. Notably, more recent efficient methods like BOME [171] adopt a value-function approach (See Section 8), reformulating the bilevel program into a single-level constrained optimization problem. This allows for the application of fully first-order methods that do not require computing second derivatives of the lower–level objective.

5.2. Efficient approximation schemes. To address the computational bottlenecks of Algorithm 5.1, various approximation schemes have been introduced. These methods generally aim to estimate the hypergradient efficiently without incurring the full cost of Hessian inversion or exact lower-level minimization.

A fundamental distinction in the literature is between *Approximate Implicit Differentiation (AID)* and *Iterative Differentiation (ITD)*. While AID approximately solves the linear system derived from the expression of the hypergradient, ITD approximates the hypergradient by backpropagating through the unrolled computational graph of the lower-level optimization algorithm (e.g., T steps of gradient descent). The two procedures are detailed in Algorithms 5.2 and 5.3, respectively. As explained in [188, 97], ITD computes the exact gradient of the proxy objective $x \mapsto F(x, y_T(x))$. However, its memory cost scales linearly with T, which can be prohibitive for problems requiring many lower–level steps. In contrast, AID allows for constant memory cost (using efficient linear solvers) but introduces a systematic bias if the linear system is not solved exactly or if the lower-level problem has not converged to the true solution $y^*(x)$. For a detailed comparison between these two strategies, interested readers are referred to [110].

In the AID framework, the computation of the hypergradient approximation (see Lines 4 and 5 in Algorithm 5.2) is typically reformulated using an adjoint vector v, which is the solution to the linear system $\nabla^2_{yy} f(x^k, \hat{y}_k)v = \nabla_y F(x^k, \hat{y}_k)$, so that

$$\nabla \mathcal{F}(x^k) \approx h_k = \nabla_x F(x^k, \hat{y}_k) + \nabla^2_{xy} f(x^k, \hat{y}_k)^\top v.$$

This linear system can be solved for example using Conjugate Gradient (CG) [210], which converges fast and relies solely on Hessian-vector products and avoids explicit matrix factorization. Alternatively, the linear system can be viewed as the optimality condition of a quadratic minimization problem and solved using (stochastic) gradient descent [110, 111]. This generalizes the Neumann series approximation (of the inverse hessian) [185], which corresponds to a specific gradient descent scheme.

To further reduce computational overhead, recent "Hessian–free" methods attempt to mitigate the cost of accessing second-order information directly. Unlike methods that simply

Algorithm 5.2 Bilevel Gradient Method with AID

Require: x^0, $\{\alpha_k\}$.
for $k = 0, 1, \dots, K$ **do**
  Lower–level optimization:
  Init solver at $\hat{y}_0$ (if warm start $\hat{y}_0 = \hat{y}_{k-1}$)
  Find $\hat{y}_k \approx \arg\min_y f(x^k, y)$
  Linear system:
  Init solver at $\hat{v}_0$ (if warm start $\hat{v}_0 = \hat{v}_{k-1}$)
  Find $\hat{v}_k$ by solving $\nabla^2_{yy} f(x^k, \hat{y}_k) v = \nabla_y F(x^k, \hat{y}_k)$
  Approximate Hypergradient:
  $h_k = \nabla_x F(x^k, \hat{y}_k) - \nabla^2_{xy} f(x^k, \hat{y}_k)^\top \hat{v}_k$
  Upper–level update:
  $x^{k+1} = x^k - \alpha_k h_k$
end for

ignore second-order terms, Hessian-free bilevel algorithms approximate the required Hessian-vector products to maintain convergence to the true stationary point. As analyzed by Sow et al. [234], one can estimate the product $\nabla^2_{yy} f(x, y)v$ using a finite difference approximation of gradients for a sufficiently small scalar $\delta > 0$. This technique allows the linear system to be solved using only first-order gradient oracles, avoiding explicit Hessian construction while strictly adhering to the implicit differentiation framework.

Algorithm 5.3 Bilevel Gradient Method with ITD

Require: x^0, $\{\alpha_k\}$, steps T.
for $k = 0, 1, \dots, K$ **do**
  Lower–level optimization:
  Init y_0 (if warm start set $y_0(x_k) = y_T(x_{k-1})$)
  for $t = 0, \dots, T-1$ **do**
    $y_{t+1}(x^k) = \Phi_t(x^k, y_t(x_k))$ {E.g., $\Phi_t(x, y) = y - \eta_t \nabla_y f(x, y)$}
  end for
  Set $\mathcal{F}_T(x^k) := F(x^k, y_T(x^k))$
  Approximate Hypergradient:
  Compute $h_k = \nabla \mathcal{F}_T(x^k)$ via backpropagation through $\Phi_0, \dots, \Phi_{T-1}$
  Upper-level update:
  $x^{k+1} = x^k - \alpha_k h_k$
end for

In the very common situation where we are dealing with large scale datasets, i.e. where d_u, d_l in (5.3) are large, the lower-level, the linear system and the final hypergradient computation (for AID methods), are solved with iterative algorithms relying on the *stochastic estimators* $\hat{f}$ and $\hat{F}$ of the lower and upper level objectives and their derivatives.

A ubiquitous strategy to improve efficiency is "warm starting", which involves initializing the lower-level and linear system solvers at iteration k with the approximate solutions from iteration $k-1$. Intuitively, this exploits the smoothness of the solution map $y^*(x)$, implying that small changes in x result in small changes in the optimal y, making the previous solution a high-quality initial guess. Recent theoretical work by [133] has rigorously analyzed the impact of the computational "loops" (lower–level optimization steps T and linear solver steps Q) on convergence. They demonstrate that while substantial loops are often necessary for ITD, AID schemes utilizing warm-start strategies can effectively reduce the required loop count to $O(1)$ per iteration. The special case where only one step is done for the lower-level problem is referred to as single-loop (see e.g. [166, 108]), in contrast with double-loop bilevel algorithms which use more than one step. While this significantly improves practical efficiency, it complicates the theoretical analysis by introducing a strong coupling between the dynamics of the upper and lower-levels.

5.3. Convergence guarantees and complexity analysis. Formalizing the convergence of bilevel algorithms requires distinguishing between the quality of the *hypergradient approximation* and the convergence of the *optimization procedure* as a whole.

5.3.1. Convergence of the hypergradient approximation. Before analyzing the optimization trajectory, one must ensure that the estimated hypergradient h_k is a reliable proxy for the true hypergradient $\nabla\mathcal{F}(x_k)$. The true hypergradient depends on the exact lower-level solution $y^*(x_k)$ and the exact solution to the linear system $v^*(x_k)$, while in practice, we operate with approximations.

Lower-level fixed points. [110, 111] study the accuracy of (approximate) hypergradient computation for both AID and ITD bilevel problems in which the lower-level solution is defined implicitly as a fixed point, $y(x) = \Phi(x, y(x))$, under regularity assumptions that ensure well-posedness and stability of the implicit map. This framework subsumes the classical smooth lower level optimality condition $\nabla_y f(x, y(x)) = 0$ by choosing $\Phi(x, y)$ as a single step of a descent method, e.g., $\Phi(x, y) = y - \eta\nabla_y f(x, y)$, so that fixed points coincide with stationary points of the lower-level objective.

Importantly, the fixed-point viewpoint is strictly more general than convex optimization in the sense of minimizing a scalar potential: many equilibrium problems are naturally modeled by *monotone operator* formulations and solved by *operator-splitting* iterations (proximal point, forward–backward, Douglas–Rachford, etc.), which can be written as fixed-point iterations even when the underlying operator does *not* arise as the gradient (or subgradient) of any function [218, 13]. Intuitively, an operator fails to come from an optimization problem when it lacks a potential structure (e.g., it contains an intrinsically "rotational"/skew-symmetric component), as happens in general variational inequalities, game-theoretic equilibria, and saddle-point problems where the relevant stationarity conditions correspond to a monotone inclusion rather than minimization of a single scalar objective. This also connects to *implicit* or *equilibrium* architectures in deep learning, notably Deep Equilibrium Models [10], which define hidden states as solutions of a fixed-point equation.

Under the assumption that Φ (or ∇f) and F are Lipschits smooth, and that Φ is a *contraction* (Lipschitz with constant less than one) with respect to y (or alternatively if f is smooth and strongly convex with respect to y), the error in the AID hypergradient is bounded linearly by the errors in the lower-level and linear system subproblems:

$$\|h_k - \nabla\mathcal{F}(x_k)\| \leq L_1\|\hat{y}_k - y^*(x_k)\| + L_2\|\hat{v}_k - v^*(x_k)\|, \tag{5.5}$$

where L_1, L_2 are constants derived from the condition number κ of the lower-level problem and the Lipschitz constants of the gradient and/or fixed-point maps. Specifically, if we employ gradient descent for the lower-level and Conjugate Gradient (CG) or gradient descent for the linear system, [110] show that achieving an error $\|\nabla\mathcal{F}(x_k) - h_k\| \leq \epsilon$ requires $O(\log(1/\epsilon))$ iterations for both the lower-level solver and the linear system solver.

Under the same contractivity and smoothness assumptions, [110] also establish a *linear* (geometric) convergence of the ITD hypergradient to the true hypergradient. A similar results can be also found in the seminal automatic differentiation textbook by Griewank and Walther [115, Chapter 15]. In particular, for T lower–level gradient descent steps (unrolling length), [110, Theorem 2.1] shows a bound of the form

$$\left\|\nabla\mathcal{F}_T(x^k) - \nabla\mathcal{F}(x^k)\right\| \;\leq\; (c_1 T + c_2)\, q^T,$$

where $c_1, c_2 \geq 0$ and $0 \leq q < 1$ depend on x. This rate similar to the AID rate of cq^T, but multiplied by an additional prefactor growing like T. As a consequence, the iteration complexity to reach accuracy ϵ remains $O(\log(1/\epsilon))$, but the extra T prefactor yields a practical "delay" compared to AID bounds.

The *stochastic case* has been mainly studied for AID. The hypergradient estimator h_k admits an MSE decomposition into (i) terms controlled by the mean-square accuracy of the stochastic lower–level solvers used to compute $(\hat{y}_k, \hat{v}_k)$, and (ii) a variance term coming from stochastic Jacobian/Hessian-vector product estimators [111, 112]. As a result, the overall MSE essentially matches the convergence rate of the stochastic solvers employed at the lower-level and for the linear system, up to an additional variance contribution. In particular, the refined analysis in [112] yields bounds of the form

$$\mathbb{E}\|h_k - \nabla\mathcal{F}(x_T)\|^2 \;=\; O\left(\rho_T + \sigma_T + \frac{1}{b_T}\right),$$

where ρ_T and σ_T denote the MSEs of the lower-level and linear-system stochastic solvers (respectively), and b_T is number of samples used for some stochastic estimators. Hence, when both lower–level solvers are implemented with SGD-type methods so that $\rho_T = \Theta(1/T)$ and $\sigma_T = \Theta(1/T)$ and we also set $b_T = \Theta(T)$, the hypergradient MSE decreases as $O(1/T)$ as well, matching the canonical $O(1/T)$ rate of SGD for the lower-level problem. The ITD case has been studied only recently by [131], where they focus on the convergence of the derivative of SGD. They show that with a careful analysis it is possible to achieve a rate of $O(\log(T)^2/T)$, which is off only by a logarithmic factor compared to the AID rate.

While the classical theory relies on differentiability, recent work has extended the implicit differentiation framework to nonsmooth settings, such as when the lower-level problem involves L_1 regularization or nonsmooth activations. [34] and [27] established that under mild conditions (e.g., separability or specific non-degeneracy), implicit differentiation remains valid on the support of the solution, and convergence rates can still be derived. [113] further generalized the analysis to provide non-asymptotic convergence rates for nonsmooth AID, including the stochastic case.

Stochastic Setting			
Complexity	**Algorithm**	**WS**	**Loop**
$O(\epsilon^{-3})$	BSA [104]	No	Nested
$\tilde{O}(\epsilon^{-2.5})$	TTSA [124]	Yes	Single
$\tilde{O}(\epsilon^{-2})$	stocBiO [134]	Yes	Nested
	SMB [118], saBiAdam [126]	Yes	Single
	ALSET [53]	Yes	Single
$O(\epsilon^{-2})$	BSGM [112]	No	Nested
	Amigo [8]	Yes	Nested
Variance Reduction			
$O(\epsilon^{-2})$	STABLE [52], FSLA [165]	Yes	Single
$\tilde{O}(\epsilon^{-1.5})$	VRBO [260]	Yes	Nested
	STABLE-VR [118], SUSTAIN [140], VR-saBiAdam [126], MRBO [260]	Yes	Single
$O(d^{2/3}/\epsilon)$	SABA [61]	Yes	Nested

Deterministic Setting			
Complexity	**Algorithm**	**WS**	**Loop**
$O(\epsilon^{-5/4})$	BA [105]	No	Nested
$\tilde{O}(\epsilon^{-1})$	BiO-ITD [134]	Yes	Nested
	BGM [112]	No	Nested
$O(\epsilon^{-1})$	BiO-AID [134]	Yes	Nested
	Amigo [8]	Yes	Nested

Table 1: Sample complexity for finding an ϵ-stationary point (i.e. a point x such that $\mathbb{E}\|\nabla\mathcal{F}(x)\|^2 \leq \epsilon$) in implicit function based methods. *WS* indicates the use of warm-start for the lower-level problem. *Loop* specifies the structure: “Single” implies strictly 1 lower–level step is taken per upper–level step, while “Nested” implies a multi-step lower–level solver is used. The complexity of SABA is for the finite sum setting where $d = d_u + d_l$ is the total number of samples.

5.3.2. Convergence of the bilevel framework. To analyze the complexity of the full framework, we must first define the computational oracles involved. The analysis typically revolves around following operations: the *lower and upper level gradients* $\nabla_y f(x, y)$,

$\nabla_x \mathcal{F}(x,y)$,$\nabla_y \mathcal{F}(x,y)$ the *lower-level Hessian-vector product* $\nabla^2_{yy} f(x,y)v$, and the *mixed Hessian-vector product* $\nabla^2_{xy} f(x,y)^\top v$. In the stochastic case, these quantities are replaced instead by unbiased estimates with controlled variance. Thanks to Automatic Differentiation, computing these derivative operations has a cost which scales with the dimension of x and y in the same way as computing the function values. While the Hessian-vector product is typically implemented as a Jacobian-Vector Product (JVP) of the lower-level gradient map with respect to y, practical implementations often find that simple gradient evaluations are significantly cheaper (by a constant factor) than the second-order vector products. For convergence rates, these costs are often aggregated, measuring the total number of calls to these first-order and second-order oracles required to reach stationarity.

Since the the function $\mathcal{F}$ is nonconvex, the goal of the bilevel procedure is to find a stationary point. Therefore, the standard convergence metric is the number of iterations (or total oracle calls) required to find an ϵ-stationary point, defined as a point x satisfying $\mathbb{E}\|\nabla \mathcal{F}(x)\|^2 \leq \epsilon$. We summarize the main results in Table 1 and discuss them in detail below. In the deterministic case, the seminal work by [105] established a baseline complexity rate of $O(\epsilon^{-5/4})$ without warm start for AID. Subsequent research improved upon this by leveraging the warm-start strategy. By initializing the solvers with previous iterates, [134] and [8] demonstrated that it is possible to achieve the optimal complexity of $O(\epsilon^{-1})$, which matches the standard rate for single-level nonconvex optimization. The work [134] also establishes the first rate for ITD of $O(\epsilon^{-1}\log(\epsilon^{-1}))$ for ITD, which relies on warm-start.

The stochastic setting is significantly more challenging due to the bias-variance trade-off and the results are focused almost entirely on the AID method. The landscape here is defined by the sample complexity required to reach an ϵ-stationary point. The baseline complexity was established by the Bilevel Stochastic Approximation (BSA) algorithm [105], which does not use warm start and requires $O(\epsilon^{-3})$ samples and an increasing (as $\sqrt{k}$ where k is the upper-level iteration counter) number of lower–level steps to control bias. Subsequent improvements utilized warm start to reduce the lower–level loop complexity. The Two-Timescale Stochastic Approximation (TTSA) [124] achieves $\tilde{O}(\epsilon^{-2.5})$.

A significant leap in efficiency was marked by a class of algorithms achieving $\tilde{O}(\epsilon^{-2})$ complexity, matching the standard single-level stochastic baseline. This group includes stocBiO [134], SMB [118], ALSET [51], Amigo [8], STABLE [50], and FSLA [166]. Most of these methods achieve this efficiency by using warm start on the lower-level problem to reduce the number of lower–level iterations to $O(1)$ (often referred to as single-loop), although some, like Amigo, warm-start both the lower-level and linear system solvers which allows to remove the log factor and achieve $O(\epsilon^{-2})$ sample complexity, which interestingly matches that of a single level optimization problem. Variance-reduced methods such as SUSTAIN [140], VR-saBiAdam [126], and MRBO [260] push the theoretical boundary further, achieving a near-optimal complexity of $\tilde{O}(\epsilon^{-1.5})$, though they typically require more oracles per iteration and stronger assumptions. Specifically focusing on the finite-sum setting (Empirical Risk Minimization), [61] established tight lower bounds and proposed the SABA algorithm, which achieves optimal variance-reduced rates. This approach builds on their earlier general stochastic framework [60], which provided a unified analysis for single-loop aggregation schemes applicable to both finite-sum and infinite-horizon problems.

While warm start is prevalent, [112] challenged the assumption that it is necessary for optimal rates. They showed that a “cold start” procedure (solving the lower–level problem from scratch or fixed initialization) can still achieve the $\tilde{O}(\epsilon^{-2})$ sample complexity in the stochastic setting. The key is using larger mini-batches ($\Theta(\epsilon^{-1})$) for the hypergradient estimation and specific step-size schedules to control the variance. For the deterministic case, this cold-start approach improves the baseline to $O(\epsilon^{-1}\log(\epsilon^{-1}))$. While without warm-start the analysis is simpler since it decouples the lower–level and outer loop dynamics, it additionally requires the distance between the lower-level solution and the starting point of the lower-level solver to be uniformly bounded over the upper-level feasible set, which might explain why warm-start method often perform better in practice.

5.4. Meta-parameters and adaptive methods. It is worth emphasizing that many bilevel algorithms with strong convergence guarantees depend on several *algorithmic meta-parameters*, that is, quantities controlling how the numerical method is run rather than parameters of the learning model itself. For example the upper- and lower-level stepsizes, the number of lower-level updates performed per outer iteration, mini-batch sizes in the stochastic setting, regularization or damping parameters in implicit solvers, and tolerances or number of iterations for auxiliary linear-system solves. In many methods, these quantities cannot be chosen once and kept fixed, but instead must follow carefully designed schedules along the iterations in order to balance bias, variance, and stability. This tuning burden has motivated a recent line of work on adaptive and parameter-free bilevel methods. Early work in this direction includes *BiAdam*, which introduced adaptive learning rates for stochastic bilevel optimization and its variance-reduced variant VR-BiAdam [126].

More recently, *BiSLS/SPS* proposed stochastic line-search and Polyak-type rules to automatically choose the coupled upper- and lower-level stepsizes, with the explicit goal of improving stability and reducing manual tuning [85]. Going one step further, tuning-free methods such as D-TFBO and S-TFBO remove the need for prior knowledge of problem constants and choose stepsizes adaptively from cumulative gradient information, while attaining deterministic and stochastic rates that nearly match those of their well-tuned counterparts [261]. Related adaptive mirror-descent variants have also been proposed beyond the strongly-convex inner-level setting [7]. Adaptive ideas have also started to extend beyond the standard centralized Euclidean setting, including federated, Riemannian, and decentralized bilevel optimization, further indicating that robustness to solver meta-parameters is emerging as a broader theme in the bilevel literature [128, 227, 274].

5.5. Beyond classical assumptions. Some works have begun to relax classical assumptions in BL, to extend the gradient descent framework presented here to more general problem classes. In particular regarding nonconvex geometry, weak lower-level curvature, and fundamental complexity limits. Classical results focus on finding first-order stationary points, characterized solely by a small gradient norm. However, in nonconvex BO landscapes, such points may correspond to strict saddles of the upper-level objective function. To address this, [127] develops perturbed implicit-gradient methods that guarantee convergence to second-order stationary points. Common BO approaches to tackle problems with nonconvex lower-level players are discussed in Section 8.

The *lower-level singleton* assumption fails in many over-parameterized models, where the lower-level optimal solution may not be unique and the Hessian of the lower-level objective function is singular. To address this, recent papers have introduced Polyak-Łojasiewicz (PL) conditions for the lower-level problem. [179] propose a gradient-based framework that handles nonconvex followers by utilizing initialization auxiliaries, bypassing the need for strong convexity. Possible approaches to extend the gradient descent framework to problems where the lower-level singleton assumption (described in (2.1)) fails are discussed in the next section. Furthermore, [255] and [125] extend convergence guarantees to settings satisfying the PL condition, showing that implicit differentiation strategies remain tractable even when the argmin set is non-singleton. These works often rely on generalized inverses (such as the Moore–Penrose pseudoinverse) or iterative approximations to handle the singular Hessian matrices inherent in these relaxed settings.

Recent lower-bound results demonstrate that BO is fundamentally harder than minimax or single-level nonconvex optimization. [132] establish oracle lower bounds that match (up to logarithmic factors) the best known upper bounds, showing the intrinsic dependence on conditioning and cross-level coupling. In parallel, [61] prove tight lower bounds for bilevel empirical risk minimization and introduce an algorithm achieving near-optimal complexity.

6. Limitations of the implicit function-based framework. The gradient descent algorithmic framework discussed in the previous section has been very successful in solving various BL problem classes as highlighted previously. However, the assumptions for its implementation are very restrictive. At a high-level, the main restrictions are the requirement that *the lower-level player is unconstrained and can only have a unique optimal solution for all upper-level variables.* In this section, we analyze possible ways to extend the gradient descent method to the implicit function-type framework, while relaxing these assumptions. We start by first keeping assumption (2.1), while relaxing the assumption that the lower-level problem is unconstrained (i.e., the requirement that $Y(x) = \mathbb{R}^m$ for all $x \in X$).

6.1. Extension of the implicit function approach to lower-level constrained problems. We assume here that the lower-level feasible set has the form

$$Y(x) := \{y \in \mathbb{R}^m \mid \; g(x,y) \leq 0\}\,, \tag{6.1}$$

where $g : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}^q$ corresponds to the lower-level constraint function. Recall that the key requirement in the previous section to ensure the fulfillment of assumption (2.1) is to assume that the function $f(x,\cdot)$ is strongly convex for all $x \in X$. Next, we introduce assumptions that can ensure *strong local stability* for the lower-level problem; i.e., to enable the reduction of $y(\cdot)$ to a locally Lipschitz continuous vector-valued and even differentiable function, when the lower-level feasible is given by (6.1). Before, note that in Section 7 (resp. Section 8), we are going to specifically explore classical BO approaches to tackle bilevel programs where the lower-level problem is only convex (resp. nonconvex).

To ensure that assumption (2.1) holds under the lower-level feasible set (6.1), we can make the following assumptions associated to a point $(\bar{x}, \bar{y})$ such that $\bar{y} \in Y(\bar{x})$:

(A1) The function f and g are at least twice continuously differentiable near $(\bar{x}, \bar{y})$.

(A2) The lower-level linear independence constraint qualification (LLICQ) holds at $(\bar{x}, \bar{y})$; i.e., the vectors $\nabla_y g_i(\bar{x}, \bar{y})$ for $i \in I(\bar{x}, \bar{y})$ are linearly independent. Here, $I(\bar{x}, \bar{y})$ denotes the set of active indices for the lower-level constraint:

$$I(\bar{x}, \bar{y}) := \{i \in [q] \mid \; g_i(\bar{x}, \bar{y}) = 0\} \quad \text{with} \;\; [q] := \{i = 1, \ldots, q\}.$$

(A3) The lower-level second order sufficient condition (LSOSC) is satisfied at $(\bar{x}, \bar{y}, \bar{u})$, where $\bar{u}$ is a lower-level Lagrange multiplier associated to $(\bar{x}, \bar{y})$; i.e.,

$$d^\top \nabla^2_{yy} \ell(\bar{x}, \bar{y}, \bar{u}) d > 0 \quad \text{ for all } d \neq 0 \text{ s.t. } \left\{ \begin{array}{ll} \nabla_y g_i(\bar{x}, \bar{y}) d = 0 & i \in I(\bar{x}, \bar{y}), \\ \nabla_y g_i(\bar{x}, \bar{y}) d = 0 & i \in J(\bar{u}), \end{array} \right.$$

where ℓ is the lower-level Lagrangian function defined by

$$\ell(x,y,u) := f(x,y) + u^\top g(x,y) \tag{6.2}$$

and the index set $J(\bar{u})$ is given by

$$J(\bar{u}) := \{i \in [q] \mid \bar{u}_i > 0\}\,.$$

(A4) The lower-level strict complementary slackness (LSCS) is satisfied at the point $(\bar{x}, \bar{y}, \bar{u})$; i.e., it holds that $I(\bar{x}, \bar{y}) = J(\bar{u})$.

Under assumptions (A1)—(A4), the lower-level optimal solution function $y(\cdot)$ is well-defined and continuously differentiable from a neighborhood of $\bar{x}$ to a neighborhood of $\bar{y}$, and its Jacobian can be written as follows:

$$\begin{aligned} \nabla y(\bar{x}) \;\; = \;\; & -\left(\nabla^2_{yy}\ell\right)^{-1} \Big\{ \mathrm{I} \\ & - \nabla_y g_{\bar{I}}^\top \left[\nabla_y g_{\bar{I}} \left(\nabla^2_{yy}\ell\right)^{-1} \nabla_y g_{\bar{I}}^\top \right]^{-1} \nabla_y g_{\bar{I}} \left(\nabla^2_{yy}\ell\right)^{-1} \Big\} \nabla^2_{xy}\ell \\ & - \left(\nabla^2_{yy}\ell\right)^{-1} \nabla_y g_{\bar{I}}^\top \left[\nabla_y g_{\bar{I}} \left(\nabla^2_{yy}\ell\right)^{-1} \nabla_y g_{\bar{I}}^\top \right]^{-1} \nabla_x g_{\bar{I}}, \end{aligned} \tag{6.3}$$

where I is the identity matrix of suitable dimensions and $\bar{I} \equiv I(\bar{x}, y(\bar{x}))$; see, e.g., Subsection 7.3.1 in the book [230] (or [86, Chapter 2]). Furthermore, for full clarity, note that in the formula (6.3), we have $\ell \equiv \ell(\bar{x}, \bar{y}, \bar{u})$ and $g_{\bar{I}} \equiv (g_i(\bar{x}, \bar{y}))_{i \in \bar{I}}$.

Algorithm 6.1 Gradient descent algorithm with lower-level constraints

Require: x^0 and $\{\alpha_k\}$;

1: **for** $k = 0, 1, \ldots, K$ **do**
2: $\quad y(x^k) = \arg\min_y \left\{ f(x^k, y) \;\middle|\; y \in Y(x^k) \right\}$;
3: $\quad I_k = I\left(x^k, y(x^k)\right)$;
4: $\quad$Calculate $\nabla y(x^k)$ using (6.3);
5: $\quad \nabla \mathcal{F}(x^k) = \nabla_x F(x^k, y(x^k)) + \nabla y(x^k)^\top \nabla_y F(x^k, y(x^k))$;
6: $\quad x^{k+1} = x^k - \alpha_k \nabla \mathcal{F}(x^k)$
7: **end for**

It goes without saying that under this framework, problem ($\mathrm{P_i}$) is well-defined and Algorithm 5.1 can be extended accordingly to this version of the problem with lower-level feasible set (6.1), as it can be seen in Algorithm 6.1.

Clearly, with the lower-level constraint, many more layers of difficulty appear in this extension of Algorithm 5.1. First, at each iteration, the active indices of the current iteration point $(x^k, y(x^k))$ are needed; cf. line 3. Secondly, the complexity of calculating $\nabla y(x^k)$ increases significantly, as at each iteration. The formula requires computing the inverses of both $\nabla^2_{yy}\ell$ and $\nabla_y g_{\bar{I}} \left(\nabla^2_{yy}\ell\right)^{-1} \nabla_y g_{\bar{I}}^\top$, as well as many other matrix operations, which need much more computing effort in line 4 of Algorithm 6.1. This is probably the key reason why almost all bilevel learning algorithms avoid lower-level constraints. It might be useful to observe that if $g \equiv 0$ in Algorithm 6.1, the method just reduces to Algorithm 5.1.

Besides the challenge in accommodating constraints in the lower-level problem, we would like to draw attention to the fact that even when $y(\cdot)$ is well-defined as a vector-valued function in some neighborhood of a point of interest, it is nonsmooth in general. The premise of having $y(\cdot)$ as a continuously differentiable function in some neighborhood of interest requires all the functions involved in the lower-level problem to be at least twice continuously differentiable, as required in almost all bilevel learning papers (also recalled this in the context of constraints above). However, the loss functions in a wide range of machine learning tasks (and by extension the lower-level objective functions of multiple bilevel learning applications) are nonsmooth; see, e.g., [240, 249] or [173, 226, 202, 6]. To restore smoothness of lower-level functions, some bilevel learning papers have applied smoothing techniques (see, e.g., [6, 202]) or use some usual tricks that result in possibly introducing constraints in problems that are initially unconstrained (see, e.g., [21, 153, 250]).

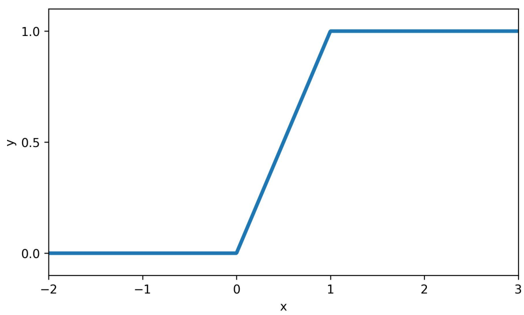

Fig. 2: Graph of the optimal solution function $y(\cdot)$ for the example in (6.4).

Moreover, it is important to observe that having twice continuously differentiable functions in the lower-level problem does not necessarily guarantee that $y(\cdot)$ is smooth. For instance, for the toy lower-level problem example

$$\min_y \left\{ \frac{1}{2}(y-x)^2 \;\middle|\; 0 \le y \le 1 \right\}, \tag{6.4}$$

the LLICQ (i.e., (A2)) and LSOSC (i.e., (A3)) are satisfied at $(0,0)$ and $(0,0,0)$, respectively, as $\bar{u} = 0$ here. The graph of $y(\cdot)$ for this example can be seen in Figure 2. There, you can see that $y(\cdot)$ is nondifferentiable at 0 because the LSCS fails, given that

$$I(0,0) = \{1\} \;\text{ and }\; J(0) = \emptyset \;\text{ (with } g_1(x,y) := -y \;\text{ and }\; g_2(x,y) := y-1).$$

As it can be seen in Figure 2, $y(\cdot)$ is a piecewise smooth function in the context of this example. This is a rather general observation for a constrained parametric optimization problem. To be more precise, consider the lower-level problem present in (BOP), where the lower-level feasible set is described by (6.1). If assumptions (A1)—(A4) are satisfied at some point $(\bar{x}, \bar{y})$, then, there are open neighborhoods U of $\bar{x}$ and V of $\bar{y}$, and a continuous function $y(\cdot)$ mapping U to V such that for each $x \in U$, $y(x)$ is the unique local solution of the lower-level problem (for x fixed) in V; and the function $y(\cdot)$ is piecewise continuously differentiable around $\bar{x}$. This means that $y(\cdot)$ is continuous and there is a finite family of continuously differentiable functions $y^1(x)$, ..., $y^N(x)$ defined in a neighborhood of $\bar{x}$, such that $y(x) \in \{y^l(x), \dots, y^N(x)\}$ for any $x \in U$; see [214, 65]. Under this framework, $y(\cdot)$ is local Lipschitz continuous near $\bar{x}$, as it is the case for any piecewise smooth function [222].

So, under relatively affordable assumptions, $y(\cdot)$ is a locally Lipschitz continuous function. There are multiple works on approaches to calculate generalized derivatives of $y(\cdot)$ (see [86, 214, 65] and references therein) and their use in the general field of bilevel optimization; see, e.g., [150, 219, 245]. However, these results do not seem to have been implemented in the context bilevel learning; and one of the main reasons for this is that these generalized derivatives of $y(\cdot)$ are generally abstract in nature or very difficult to calculate in practice. Recently, the paper [33] introduced the concept of *conservative derivative* to efficiently extend derivative approximation tools (such as *automatic differentiation*, a backbone object to enhance deep learning algorithms [14]) to nonsmooth functions. The concept of conservative derivative has been used in [113] to implement the classical iterative differentiation (ITD) and approximate implicit differentiation (AID) schemes, widespread in the practical implementation of lines 3 and 4 of Algorithm 5.1 in the smooth unconstrained-lower-level bilevel learning problem, to the case where $y(\cdot)$ is a nonsmooth function. More work is needed to generalized derivative approximation schemes to broader classes of BL problems.

6.2. Restoring the implicit function approach for problems with multiple lower-level optimal solutions. The biggest challenge for the state of the art method for bilevel learning, which is sketched in Algorithm 5.1, is the requirement to always have a unique solution for the lower–level problem; cf. (2.1). If this assumption fails, the implicit function approach ($\mathrm{P_i}$) is out of question. Ensuring that this condition holds requires very strong assumptions as we have discussed above. Unfortunately, this assumption cannot hold for a wide range of BL applications; see, e.g., [182, 181, 233, 262, 255, 116, 129], for a selection of such problems that do not satisfy condition (2.1). To address this challenge, a series of articles from the BO literature have proposed a regularization approach, which consists to introduce a perturbation to the lower-level problem of the form

$$\min_{y \in Y(x)} f(x,y) + \alpha \psi(x,y), \tag{6.5}$$

in order to force the fulfillment of condition (2.1). In fact, if the lower-level problem is *just* convex and the function $\psi : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}$ is strongly convex in y, then it holds that

$$\{x \in X | \;\; |S_\alpha(x)| \neq 1\} = \emptyset \;\text{ for all }\; \alpha > 0,$$

where S_α is the optimal solution mapping for the regularized problem (6.5); i.e.,

$$S_\alpha(x) := \underset{y\in Y(x)}{\operatorname{argmin}}\ f(x,y) + \alpha\psi(x,y).$$

In this context, model ($\mathrm{P_i}$) can be rescued as follows

$$\min_{x\in X} \mathcal{F}_\alpha(x) := F(x, y_\alpha(x)) \quad \text{with} \quad \{y_\alpha(x)\} = S_\alpha(x) \ \text{ for all } \ \alpha > 0. \tag{6.6}$$

With the additional caution ensuring that $y_\alpha(\cdot)$ describes a smooth function for all $\alpha > 0$, Algorithm 5.1 can be extended to problems with multiple optimal solutions if $Y(x) = \mathbb{R}^m$, and similarly, for problems with lower-level constraints described by (6.1), a suitable version of Algorithm 6.1 can also be extended to this case.

To make such an iterative procedure rigorous, a sequence $(\alpha_n)_n$ such that $\alpha_n > 0$ for all $n \in \mathbb{N}$ with $\alpha_n \downarrow 0$ can be introduced with

$$x(\alpha_n) \in \underset{x\in X}{\operatorname{argmin}}\ \mathcal{F}_{\alpha_n}(x) \ \text{ for } \ n \in \mathbb{N}. \tag{6.7}$$

The regularization in (6.5), due to Tikhonov [242], has been used in [75] to develop a gradient descent-type algorithm, with $\psi(x,y) := F(x,y)$ if $\nabla^2_{yy}F(x,y)$ is positive definite for all (x,y), to solve a constrained lower-level bilevel program. On the other hand, the regularization term $\psi(x,y) := \|y\|^2$ is used in [64, 67] to develop bundle algorithms for different classes of the bilevel optimization with lower-level constraints; further details on regularization methods for bilevel programs can be found in [65].

Despite suitable technical assumptions in the aforementioned papers to ensure the convergence of such algorithms, very simple examples can be constructed to show that the resulting limit point $\bar{x} = \lim_{n\to\infty} x(\alpha_n)$ can be quite far away from the true optimal solution or stationary point of the corresponding version of problem (BOP); see, e.g., [75, 199]. However, it can be shown that under mild assumptions, the limit point $(\bar{x}, y(\bar{x}))$ resulting from an algorithm based on the Tikhonov regularization above converges to a *lower Stackelberg equilibrium*; i.e., a point that satisfies

$$\bar{x} \in X, \ \ \bar{y} \in S(\bar{x}), \quad F(\bar{x},\bar{y}) \le \inf_{x\in X} \sup_{y\in S(\bar{x})} F(x,y).$$

This obviously implies that a lower Stackelberg equilibrium lies between optimistic and pessimistic optimal solution of problem (BOP) given that

$$\inf_{x\in X} \inf_{y\in S(\bar{x})} F(x,y) \le \ F(\bar{x},\bar{y}) \ \le \inf_{x\in X} \sup_{y\in S(\bar{x})} F(x,y).$$

What this could mean in bilevel learning is that the Tikhonov regularization approach above could be a tractable framework to extend Algorithms 5.1 and 6.1 to problems with multiple lower-level optimal solutions if one is inclined to accept lower Stackelberg equilibrium. Also note that the concept of lower Stackelberg equilibrium is closely connected to the notion of *subgame perfect Nash equilibrium*, introduced by the Nobel laureate Reinhard Selten in [224] and which is widely used in economics.

6.3. Implicit set-valued model. A direct approach to deal with the failure of condition (2.1) could simply be to insert the lower-level optimal solution mapping S in the upper-level objective function, therefore leading to the set-valued optimization problem

$$\min_{x\in X} \mathcal{F}_S(x) := F(x, S(x)). \tag{$\mathrm{P_S}$}$$

This model, which is a direct extension of the implicit function problem ($\mathrm{P_i}$) to bilevel optimization problem where the lower-level optimal solution mapping is set-valued, was first studied in [271] using the corresponding Pareto optimal solution concept. Namely, in the latter paper, optimality conditions for problem ($\mathrm{P_S}$) were derived and shown to capture all the stationary notions known for the optimistic problem ($\mathrm{P_o}$). More recently, it was shown in [232] that problem ($\mathrm{P_S}$) can be equivalent to the optimistic problem ($\mathrm{P_o}$) (resp. pessimistic problem ($\mathrm{P_p}$)), while considering the l-minimal optimal solution (resp. u-minimal optimal solution) for problem ($\mathrm{P_S}$), in the sense of set-valued optimization, under mild assumptions.

It must be said that so far, the model ($\mathrm{P_S}$) is only a theoretical object, which can enable some enhanced insights on the bilevel optimization problem when the lower-level player has multiple options to make their decision for some choices of the upper-level player. However, there is not much progress in solving set-valued optimization problems in the current literature. For some recent attempts to compute optimal solutions for the problem, interested readers are referred to [35, 106, 183], for example.

7. Karush-Kuhn-Tucker reformulation-based methods. We have seen in the previous section that when the lower-level problem is constrained, a direct extension of the classical bilevel learning algorithm presented in Section 5 is intractable; cf. Algorithm 6.1. Throughout this section, we continue with the assumption that the lower-level problem is constrained by the set (6.1). However, unlike in the previous section, where we need the very strong assumptions (A1)–(A4) for the corresponding version of ($\mathrm{P_i}$) to be well-defined as a smooth optimization problem, here, we only need to impose that the lower-level problem is convex (i.e., for all $x \in X$, the functions $f(x,.)$ and $g_i(x,.)$, for $i = 1,\dots,q$, are convex) and satisfies a constraint qualification (CQ) that can enable us to write the Karush-Kuhn-Tucker (KKT) conditions that characterize the inclusion $y \in S(x)$.

As we will refer to the Mangasarian-Fromovitz constraint qualification (MFCQ) multiple times in the sequel, note that for a general constrained optimization problem

$$\text{(7.1)} \qquad \begin{aligned} &\min \quad \mathfrak{f}(x) \\ &\text{s.t.} \quad x \in \mathcal{C} := \{x \in \mathbb{R}^n \mid \ \mathfrak{g}(x) \le 0, \ \ \mathfrak{h}(x) = 0\} \end{aligned}$$

with continuously differentiable functions $\mathfrak{f} : \mathbb{R}^n \to \mathbb{R}$, $\mathfrak{g} : \mathbb{R}^n \to \mathbb{R}^p$, and $\mathfrak{h} : \mathbb{R}^n \to \mathbb{R}^q$, it will be said to hold at a point $\bar{x} \in \mathcal{C}$ if the following condition is satisfied:

$$\text{(MFCQ)} \qquad \left.\begin{aligned} \nabla\mathfrak{g}(\bar{x})^\top\alpha + \nabla\mathfrak{h}(\bar{x})^\top\beta = 0 \\ \alpha \ge 0, \ \ \alpha^\top\mathfrak{g}(\bar{x}) = 0 \end{aligned}\right\} \Longrightarrow \left\{\begin{aligned} \alpha = 0, \\ \beta = 0. \end{aligned}\right.$$

Throughout this section, we assume here that the lower-level problem is convex and (MFCQ) holds at all $(x,y) \in \operatorname{gph} Y$ with $x \in X$ and $y \in S(x)$, then, as mentioned in Section 3, historically, the basic approach to solve the corresponding standard optimistic bilevel program (P) has consisted to first write it as a single-level optimization problem

$$\text{(KKT)} \qquad \begin{aligned} \min_{x,y,u} \quad & F(x,y) \\ \text{s.t.} \quad & x \in X, \ \ \nabla_y \ell(x,y,u) = 0, \\ & u \ge 0, \ g(x,y) \le 0, \ u^\top g(x,y) = 0, \end{aligned}$$

known as KKT reformulation. Here, the lower-level Lagrangian function ℓ is defined as in (6.2). Problem (KKT) has been one of the main *go to* frameworks to develop numerical methods for the optimistic bilevel optimization problem (P) since its introduction in the field of mathematical optimization; see, e.g., [65, 57, 76, 78] and references therein.

After some discussion on the background the KKT reformulation problem (KKT), we provide an overview of classical ideas from the BO literature that remain mostly unexplored in BL. At the end of the section we present some works on BL that have applied the KKT reformulation. Overall, the material here could serve as base for investigations on the scalability of these techniques, especially in terms of their approximations with ideas similar to the ones that have been at the core of the success of the gradient descent method for bilevel learning described in Section 5.

The basic idea behind reformulation (KKT) is the fact that under the lower-level convexity assumption and the fulfillment of the LLICQ at a point $(x, y) \in \operatorname{gph} S$,

$$y \in S(x) \iff \exists u \in \mathbb{R}^q : \begin{cases} \nabla_y \ell(x, y, u) = 0, \\ u \geq 0,\ g(x, y) \leq 0,\ u^\top g(x, y) = 0. \end{cases} \tag{7.2}$$

Despite this equivalence, the relationship between problems (P) and (KKT) is a bit tricky due to the appearance of the Lagrange multiplier u in the new problem. Both problems are globally equivalent in some sense [68]. However, locally, to get a local optimal solution $(\bar{x}, \bar{y})$ of problem (P) from (KKT), one needs to ensure that $(\bar{x}, \bar{y}, u)$ is locally optimal for the latter problem for all $u \in \Lambda(\bar{x}, \bar{y})$, where

$$\Lambda(\bar{x}, \bar{y}) := \left\{ u \in \mathbb{R}^q \mid \nabla_y \ell(\bar{x}, \bar{y}, u) = 0,\ u \geq 0,\ g(\bar{x}, \bar{y}) \leq 0,\ u^\top g(\bar{x}, \bar{y}) = 0 \right\}$$

denotes the set of lower-level Lagrange multipliers corresponding to the lower-level optimal solution $\bar{y}$ when x is fixed at $\bar{x}$. Given that the set $\Lambda(\bar{x}, \bar{y})$ can be made of infinitely many points, this is an intractable prospect, even for very small toy examples, not to imagine this in the context of BL. For more details on this connection between problems (P) and (KKT), see [68]. For a systematic analysis of single-level reformulations of problems (KKT) based on transformations of the lower-level problem (especially from the duality theory perspective), interested readers are referred to [72].

As a transition point to discuss methods to solve problem (KKT), it would be useful to note that some care is needed in handling the problem, and not just treat it as any random mathematical program with equality and inequality constraints. For example, if we do so, the first issue that we are faced with is that many classical constraint qualifications, including the MFCQ, will fail; see, e.g., [220]. The challenges in solving problem (KKT) are due to the complementarity conditions, present in the last line of the feasible set, and which do not allow for a feasible point to strictly satisfy the inequality constraints to exist. This problematic structure of the feasible set of problem (KKT) has motivated the development of specially tailored algorithms to solve it. Throughout the literature, there are roughly four algorithmic techniques to tackle the problem, and we briefly outline them below.

7.1. Partial penalization–based methods. Partial penalization consists of penalizing the term $u^\top g(x, y)$ by moving it from the feasible set to the (upper-level) objective function. This results in the new problem

$$(\mathrm{KKT}_\lambda) \qquad \begin{aligned} \min_{x,y,u} \quad & F(x, y) - \lambda u^\top g(x, y) \\ \text{s.t.} \quad & x \in X,\ \nabla_y \ell(x, y, u) = 0,\ u \geq 0,\ g(x, y) \leq 0 \end{aligned}$$

with penalization parameter $\lambda > 0$. The feasible set becomes much easier to handle with transformation (KKT_λ), as the (MFCQ), for example, can usually be satisfied for this penalized problem. Problem (KKT_λ) has been widely used since the early days, as a based to solve problem (KKT), and therefore the corresponding bilevel optimization (P). It was possibly first used to tackle the fully linear bilevel program in [40] (with improvements later provided in [251]), where the penalty parameter is sequentially increased until a stopping criterion is achieved. The penalization model (KKT_λ) with a sequentially varying penalization parameter was also used in [163] to develop an interior-point-type method for the general MPEC problem. For such a class of problem, the partial penalization model of the form (KKT_λ) is thoroughly studied in [215] but in the case of a fixed parameter.

Partial penalization has also been used in [267] (also see [215]) as a form of constraint qualification to derive necessary optimality conditions for problem (KKT). More recently, detailed algorithmic studies on the partial penalization approach in (KKT_λ) have been conducted in [273, 250] for the standard optimistic bilevel optimization problem (P). As it is common in exact penalization methods, finding a good value for the parameter λ for problem (KKT$_\lambda$) is difficult. The paper [273] also includes an empirical study on the subject.

7.2. Relaxation–based methods. They consist of a process that starts with the enlargement of the feasible set of problem (KKT) by introducing a relaxation function to generate a more tractable subproblem. The standard relaxation schemes for problem (KKT) can be summarized in the compact model

$$(\mathrm{KKT}_t) \qquad \begin{array}{ll} \min\limits_{x,y,u} & F(x,y) \\ \text{s.t.} & x \in X,\ \nabla_y \ell(x,y,u) = 0,\ \ \phi^t_{i,\mathcal{R}}(x,y,u) \leq 0,\ i \in [q]. \end{array}$$

Here, $t > 0$ denotes the relaxation parameter, while $\mathcal{R}$ is used to represent a specific relaxation from the literature; more precisely, $\mathcal{R} \in \{\mathrm{S}, \mathrm{LF}, \mathrm{KDB}, \mathrm{SU}, \mathrm{KS}\}$, where S, LF, KDB, SU, and KS respectively represents the Scholtes, Lin and Fukushima, Kadrani, Dussault and Benchakroun, Steffensen and Ulbrich, and Kanzow and Schwartz relaxation of problem (KKT). For $t > 0$ and $i = 1, \dots, q$, the function $\phi^t_{i,\mathcal{R}}$ can be defined by

$$(7.3) \qquad \phi^t_{i,\mathcal{R}}(x,y,u) := \begin{cases} \begin{pmatrix} g_i(x,y) \\ -u_i \\ -u_i g_i(x,y) - t \end{pmatrix} & \text{if} \quad \mathcal{R} := S, \\ \begin{pmatrix} -(u_i g_i(x,y) + t^2) \\ -(u_i + t)(-g_i(x,y) + t) + t^2 \end{pmatrix} & \text{if} \quad \mathcal{R} := LF, \\ \begin{pmatrix} g_i(x,y) - t \\ -u_i - t \\ -(u_i - t)(g_i(x,y) + t) \end{pmatrix} & \text{if} \quad \mathcal{R} := KDB, \\ \begin{pmatrix} g_i(x,y) \\ -u_i \\ \varphi^t_{i,SU}(x,y,u) \end{pmatrix} & \text{if} \quad \mathcal{R} := SU, \\ \begin{pmatrix} g_i(x,y) \\ -u_i \\ \varphi^t_{i,KS}(x,y,u) \end{pmatrix} & \text{if} \quad \mathcal{R} := KS, \end{cases}$$

where, for $t > 0$ and $i = 1, \dots, q$, the function $\varphi^t_{i,SU}$ is defined as

$$\varphi^t_{i,SU}(x,y,u) := \begin{cases} 2u_i & \text{if} \quad g_i(x,y) + u_i \leq -t, \\ -2g_i(x,y) & \text{if} \quad g_i(x,y) + u_i \geq t, \\ u_i - g_i(x,y) - t\theta\left(\dfrac{u_i + g_i(x,y)}{t}\right) & \text{if} \quad |u_i + g_i(x,y)| < t. \end{cases}$$

As for the function $\theta(\cdot)$, it represents a suitable regularization function (see details in [236]) and $\varphi^t_{i,KS}$ is defined by

$$\varphi^t_{i,KS}(x,y,u) := \begin{cases} (u_i - t)(-g_i(x,y) - t) & \text{if} \quad u_i - g_i(x,y) \geq 2t, \\ -\dfrac{1}{2}\left((u_i - t)^2 + (-g_i(x,y) - t)^2\right) & \text{if} \quad u_i - g_i(x,y) < 2t. \end{cases}$$

For an overview and detailed study of these relaxations in the broader context of mathematical programs with complementarity constraints, interested readers are referred to [123] and references therein, where specific advantages and drawbacks of each relaxation are given and compared. It might be useful to highlight that the S-relaxation, which is probably the first and simplest one, introduced in [221], seems to usually show the best numerical performance (even tough convergence is typically to a C-stationary point, which is relatively weak).

Note that the complementarity conditions present in (KKT) make the feasible set of the problem very thin (as it is the union of segments of the axes), and hence, the process for a numerical procedure to search for an optimal point from it to be quite tricky. For a given relaxation above, the parameter $t > 0$ helps to control the enlargement of the feasible set such that as $t \downarrow 0$, one generally recaptures the feasible set of problem (KKT). These relaxation methods typically lead to C or M-stationarity points, depending on the assumptions made to establish their convergence results. For more details on the related theory and the definitions of these stationarity concepts, interested readers are referred to the article [123]; if one is interested in the implementation of these relaxations in the context of the pessimistic bilevel program ($\mathrm{P_p}$), see [17, 18], for example.

7.3. NCP function–based methods. These are algorithms to solve problem (KKT) that proceed by first transforming the complementarity conditions into a system of equations in such a way to get an equivalent problem of the form

$$\text{(NCP)} \qquad \begin{array}{ll} \min\limits_{x,y,u} & F(x,y) \\ \text{s.t.} & x \in X, \;\; \nabla_y \ell(x,y,u) = 0, \;\; \phi_{i,\mathcal{N}}(x,y,u) = 0, \, i \in [q], \end{array}$$

where $\phi_{i,\mathcal{N}}$, $i = 1, \ldots, q$, represents a so-called nonlinear complementarity problem (NCP) function. Precisely, for a given index $i = 1, \ldots, q$, an NCP function $\phi_{i,\mathcal{N}}$ is a real-valued function constructed such that we have

$$\phi_{i,\mathcal{N}}(x,y,u) = 0 \iff [u_i \geq 0, \; g_i(x,y) \leq 0, \; u_i g_i(x,y) = 0]\,. \tag{7.4}$$

The concept of NCP function, introduced by Mangasarian in [189], has been widely studied in the literature, considering the occurrence of complementarity conditions in many practical applications in areas such as economics, engineering, and science, just to name a few. The most famous NCP functions are probably the *Fischer–Burmeister* and *min* operator functions, which are respectively defined from $\mathbb{R}^2$ to $\mathbb{R}$ by

$$\phi_{\mathrm{FB}} := \sqrt{a^2 + b^2} - (a+b) \;\; \text{and} \;\; \phi_{\min}(a,b) := \min\{a, \, b\},$$

which each vanishes at a point $(a,b) \in \mathbb{R}^2$ if and only if $a \geq 0$, $b \geq 0$, and $ab = 0$. Note that using this NCP function in (7.4), we have $\phi_{i,\mathcal{N}}(x,y,u) := \phi_{\mathrm{FB}}\left(-g_i(x,y), \; u_i\right)$ for $i \in [q]$. The Fischer-Burmeister function, introduced in [88], has been particularly prominent due to the fact that the associated merit function $\psi(a,b) := \frac{1}{2}\|\phi_{\mathrm{FB}}(a,b)\|^2$ for the system $\phi_{\mathrm{FB}}(a,b) = 0$ is continuously differentiable. This has enable the development of powerful algorithms for semismooth systems of equations involving complementarity conditions; see, e.g., [62] for an important optimization algorithm in this context.

Note that there are so many ways to construct NCP functions; see, e.g., [5, 99] for some recent studies on the subject. With regards to solving (KKT), the NLPEC solver (https://www.gams.com/50/docs/S_NLPEC.html) works by automatically enabling the selection of an NCP function for a given general MPEC problem, to design a process to compute solutions for problem (NCP). Recently, special NCP functions that improve the development of efficient methods for neural network computations have been discovered; see, e.g., [4, 258]. It must however be said that a detailed study of the impact of NCP functions in the numerical development of methods for the KKT reformulation (KKT) for the optimistic bilevel optimization problem (P) has not yet been done.

7.4. Nonsmooth equation system–based methods. Given that (KKT) is a nonconvex optimization problem all the methods discussed so far to solve it can typically be shown to theoretically only converge to stationary points. However, considering the nature of the feasible set of the problem, defined by complementarity conditions, there are multiple different types of stationarity concepts, depending on the specific approach used or the properties imposed for the problem data to ensure theoretical convergence. The main types of stationarity concepts for problem (KKT) are the C–, M–, and S–stationarity concepts, where the latter is the *strong* stationarity concept, which is equivalent to the KKT conditions of problem (KKT) when it is viewed as a usual optimization problem with equality and inequality constraints. As for M–and C–, they stand for the *Mordukhovich* and *Clarke* stationarity conditions, respectively, due to the variational analysis tools applied to compute the generalized derivative of the involved nonsmooth functions; for more details on these concepts and how to derive them for local optimal solutions of (KKT), see, e.g., [220, 92, 78].

Another important common point of the methods described so far for problem (KKT) is that they all rely on first building a *nice* auxiliary problem that is subsequently solved with the hope compute a *solution* of the original problem. As we have just said above, theretically, these algorithms can only be shown to compute stationary points. However, another classical philosophy in solving constrained optimization problems, with continuous variables, is to directly compute these stationary points. This has led to the stream of work on semismooth Newton methods around the early 1990s for general smooth constrained problems; see, e.g., [88, 212, 207]. This class of methods has recently been explored to compute M-stationarity points for mathematical programs with complementarity constraints, and can therefore be used to tackle the KKT reformulation (KKT); see, e.g., [117, 119, 253].

To get a flavor of how a nonsmooth equation-based method can work in practice, consider the KKT reformulation based Lagrangian function $L_{\rm K}$ defined for any $(x, y, u, \alpha, \beta, \gamma)$ with $(x, y, u) \in \mathbb{R}^n \times \mathbb{R}^m \times \mathbb{R}^q$ and $(\alpha, \beta, \gamma) \in \mathbb{R}^p \times \mathbb{R}^q \times \mathbb{R}^m$ by

$$L_{\rm K}(x, y, u, \alpha, \beta, \gamma) := F(x, y) + \alpha^\top G(x) + \beta^\top g(x, y) + \gamma^\top \ell(x, y, u),$$

where, $G : \mathbb{R}^n \to \mathbb{R}^p$ is a continuously differentiable function that describes the upper-level feasible set as follows:

$$X := \{x \in \mathbb{R}^n \mid G(x) \le 0\}. \tag{7.5}$$

Additionally, for any feasible point $(x, y, u) \in \mathbb{R}^n \times \mathbb{R}^m \times \mathbb{R}^q$ of problem (KKT), the classical partition of the index set associated to the complementarity conditions that partly describe the feasible set of the problem is given by

$$\begin{aligned} \eta &:= \eta(x, y, u) := \{i = 1, \ldots, q \mid \; u_i = 0, \;\; g_i(x, y) > 0\}, \\ \mu &:= \mu(x, y, u) := \{i = 1, \ldots, q \mid \; u_i = 0, \;\; g_i(x, y) = 0\}, \\ \nu &:= \nu(x, y, u) := \{i = 1, \ldots, q \mid \; u_i > 0, \;\; g_i(x, y) = 0\}. \end{aligned}$$

Based on this notation, a feasible point $(x, y, u) \in \mathbb{R}^n \times \mathbb{R}^m \times \mathbb{R}^q$ of problem (KKT) is said to be a M-stationary point if there exists Lagrange multipliers $\alpha \in \mathbb{R}^p$, $\beta \in \mathbb{R}^q$ and $\gamma \in \mathbb{R}^m$ such that the following necessary optimality conditions hold:

$$\nabla_{x,y} L_{\rm K}(x, y, \alpha, \beta, \gamma) = 0, \tag{7.6}$$

$$\alpha \ge 0, \;\; G(x) \le 0, \;\; \alpha^\top G(x) = 0, \tag{7.7}$$

$$\nabla_y g_\nu(x, y)\gamma = 0, \;\; \beta_\eta = 0, \tag{7.8}$$

$$\forall i \in \mu : \;\; (\beta_i > 0 \wedge \nabla_y g_i(x, y)\gamma > 0) \vee (\beta_i \nabla_y g_i(x, y)\gamma) = 0. \tag{7.9}$$

For details on how to obtain these conditions for a given local optimal solution of problem [78, 79]. The optimality conditions (7.6)–(7.9) can be written as a nonlinear system of equations if we consider the function the Fischer-Burmeister function (introduced in [88])

$$\phi_{\rm FB}(a, b) := \sqrt{a^2 + b^2} - (a + b) \;\; \text{for} \;\; (a, b) \in \mathbb{R}^2 \tag{7.10}$$

and the M-stationarity function (introduced in [119])

$$\phi_{\rm M}(a, b, c, d) := \min \left\{ \begin{array}{c} \max\{-a, \, |b|, \, |c|\}, \\ \max\{-b, \, |a|, \, |d|\}, \\ \max\{|a|, \, |b|, \, c, \, d\} \end{array} \right\} \;\; \text{for} \;\; (a, b, c, d) \in \mathbb{R}^4.$$

The M-stationarity system (7.6)–(7.9) can be equivalently written as

$$\Phi(x,y,u,\alpha,\beta,\gamma) := \begin{bmatrix} \nabla_{x,y} L_{\mathrm{K}}(x,y,u,\alpha,\beta,\gamma) \\ (\phi_{\mathrm{FB}}(\alpha_j,\ -G_j(x)))_{j=1,\ldots,p} \\ (\phi_{\mathrm{M}}\,(u_i,\ -g_i(x,y),\ \nabla_y g_i(x,y)\gamma,\ \beta_i))_{i=1,\ldots,q} \end{bmatrix} = 0. \tag{7.11}$$

A careful analysis of a nonsmooth Newton method to solve this system is conducted in [119]. Different approaches to construct numerical methods to directly compute different types of statioarity concepts for problem (KKT) can be found in [117, 253].

A typical challenges for a method to directly compute stationary points for problem (KKT) via a nonsmooth system of equations reside in the selection of transformation functions (such as ϕ_{FB} and ϕ_{M} above) and corresponding adequate generalized differentiation objects for Newton or Levenberg–Marquardt step; see Section 10 for a discussion on this topic and some potential ideas on how the current bilevel learning machinery can be used in this context to scale the corresponding techniques up.

7.5. The Big–M strategy. It is one of the most common approaches to solve problem (KKT) in practice. It consists of replacing the product term $u^\top g(x,y)$ in the complementarity constraints of problem (KKT) by the two conditions

$$u_j \le v_j M_D, \ \ -g_j(x,y) \le (1-v_j)M_P, \ \ , v_j \in \{0,1\}, \ j = 1,\ldots,q, \tag{7.12}$$

where $M_D > 0$ and $M_P > 0$ are constants assumed to be *large enough*; hence, they are called *Big-M*s. This approach is typically used for fully linear bilevel optimization problems; i.e., the problem (BOP) where all the functions involved are linear in (x,y). Observe that for such a problem, replacing $u^\top g(x,y)$ in (KKT) with the system (7.12) will lead to a linear problem, which is parameterized by the big-Ms. In this case, the resulting problem would be a linear program, which can then be embedded in an algorithmic process requiring standard off-the-shelf tools for linear programs. The first main challenge with this approach is that the big-Ms need to be chosen such that an optimal solution to (KKT) is not cut off.

In [145], it is shown that identifying a suitable value for the big-M is an NP-hard problem. Moreover, even when a suitable value for the big–M could be found, solving the resulting optimization problem with constraint of the form is mixed-integer problem, which would not be scalable in the bilevel learning context. Note that an alternative to address the challenge with identifying a suitable big-M is to use an SOS1 scheme to construct a different constraint system, leading to a mixed-integer optimization that is much easier to solve; see, [146] for a detailed exposition of the SOS1 scheme and its advantages of the big–M strategy.

7.6. KKT reformulation in bilevel learning. The reformulation (KKT) was the primary approach in the series of papers [21, 25, 152, 22, 23, 153, 195, 24] by Kristin Bennett and her co-authors, focused on special version of the hyperparameter optimization problem (4.1)–(4.2), especially support vector problems with linear kernel. These papers played a key role in the promotion of applications of bilevel optimization in machine learning, as already discussed earlier in Section 3. The approach used in all these papers is based on first transforming the corresponding problems into the form (KKT) and then applying off–the-shelf solvers on them. More recently, we have had many other papers studying hyperparameters optimization problems for linear or nonlinear support vector machines [168, 213, 167, 58, 250], where the transformation (KKT) is also the base for the numerical methods. In many of these papers (see, e.g., [58, 152, 153, 195, 213, 250, 168]), it is shown that the BO approach can lead to algorithms that are more efficient than classical methods to conduct this process, such as grid search and Bayesian optimization, if these techniques are programmed under the same framework. Of course, no complexity analysis has been conducted on such methods. However, the message that we can draw from the mentioned references is that if the lower-level problem is constrained the KKT reformulation discussed here has the potential to lead to efficient methods for BL.

To close this section, observe that if the lower-level problem in (BOP) is unconstrained, the problem (KKT) reduces to a problem with the constraint $\nabla_y f(x, y) = 0$, and hence no complementarity constraints appear in this case. In the papers [6, 202], this approach is used to deal with hyperparameter optimization problems where the lower-level problem is unconstrained (or with the lower-level constraints embedded to the lower-level objective function). However, as the lower-level objective function is nonsmooth there, the approach is extended by means a smoothing technique that replaces this objective function by an approximation function that enable the recovery of the some useful information (e.g., element from the the generalized Jacobian) from the original nonsmooth function when the involved parameter is driven to zero. In the papers [6, 202], it is also shown that the smoothing algorithm developed there is far much faster than grid search and Bayesian optimization methods for hyperparameter optimization.

8. Lower-level value function reformulation-based methods. One of the common points between the implicit function model (P_i), used in classical bilevel learning algorithmic framework presented in Section 5, and the KKT reformulation (KKT) covered in the previous section is that they both require second (resp. third) order derivative information on the lower-level problem for the development of *first* (resp. *second*) order methods. This higher order derivative requirement can be avoided if the model

$$\text{(LLVF)} \qquad \min_{x,y} F(x, y) \text{ s.t. } x \in X, \; y \in Y(x), \; f(x, y) - \varphi(x) \le 0,$$

known as the lower-level value function (LLVF) reformulation, is considered as single-level transformation in the context of the standard optimistic problem (P). Note that here, φ represents the lower-level (optimal) value function

$$\varphi(x) := \min_{y \in Y(x)} f(x, y). \tag{8.1}$$

It is important to observe that unlike for the implicit function model and the KKT reformulation, where lower-level convexity and constraint qualifications are needed in the transformation process of problem (P), no assumption is required to write problem (LLVF). This is due to the fact that by definition, it holds that

$$S(x) = \{ y \in Y(x) | \; f(x, y) \le \varphi(x) \}.$$

Therefore, problems (P) and (LLVF) are globally and locally equivalent for free.

Before embarking on further in-depth analysis of problem (LLVF), let us place the approach in the historical context. Over 30 years ago, the problem (LLVF) was considered in the literature from at least three different perspectives. First, Outrata [204, 205] explored the approach as a framework to numerically solve problem (P). Around the same time, Loridan and Morgan (see, e.g., [198, 184]) used (LLVF) to establish regularity and stability results, where most often, the aim is to relax the value function constraint $f(x, y) - \varphi(x) \le \epsilon$ (with regularization parameter $\epsilon > 0$) and establish some sequential convergence properties in relation to optimal solutions and value functions associated to the original optimistic and pessimistic models (P_o) and (P_p), respectively. Work on sequential stability analysis of this type has continued to be very active (see, e.g., [46, 45]). Subsequently, around the early 1990s, Ye and Zhu [270] introduced the study of necessary optimality conditions for problem (P) based on the LLVF reformulation. This is the stream of work that has mostly come to prominence in the last 30 years in relation to problem (LLVF).

8.1. Differentiability of the lower-level optimal value function. We start here by considering the lower-level problem (LL) while letting U be an open neighborhood of a point $\bar{x} \in X$. Then the following implication holds:

$$y(\cdot) \in \mathcal{C}^1(U) \quad \Longrightarrow \quad \varphi \in \mathcal{C}^1(U). \tag{8.2}$$

In fact, if $y(\cdot) \in \mathcal{C}^1(U)$, then for all $x \in U$, $\varphi(x) = f(x, y(x))$ and therefore, similarly to the gradient formula for $\mathcal{F}$ in Section 5, we have

$$\nabla\varphi(x) = \nabla_x f(x, y(x)) + \nabla y(x)^\top \nabla_y f(x, y(x)). \tag{8.3}$$

Obviously, if the lower-level problem is unconstrained, it follows from the first order optimality conditions $\nabla_y f(x, y(x)) = 0$, that for all $x \in U$,

$$\nabla\varphi(x) = \nabla_x f(x, y(x)). \tag{8.4}$$

Otherwise, if the lower-lower feasible set-valued mapping Y is defined as in (6.1), i.e., $Y(x) := \{y \in \mathbb{R}^m \mid g(x, y) \leq 0\}$, then it results from the Lagrange multiplier rule (see the left-to-right implication in (7.2)) that for any $x \in U$, under a lower-level constraint qualification such as the MFCQ, for example, at the point $(x, y(x))$, then we have

$$\nabla_y f(x, y(x)) + \nabla_y g(x, y(x))^\top u(x) = 0. \tag{8.5}$$

On the other hand, with $I \equiv I(\bar{x}, y(\bar{x}))$, it follows from (A1)–(A4) that for some open neighborhood $U_0 \subset U$ of $\bar{x}$, we have

$$g_i(x, y(x)) = 0 \quad \text{for} \quad i \in I, \ \ x \in U_0.$$

Hence, by applying the chain rule once again,

$$0 = \nabla_x g_i(x, y(x)) + \nabla y(x)^\top \nabla_y g_i(x, y(x)) \quad \text{for} \quad i \in I, \ \ x \in U_0. \tag{8.6}$$

It follows from a combination of (8.3), (8.5), and (8.6) that

$$\begin{aligned} \nabla\varphi(x) &= \nabla_x f(x, y(x)) + \nabla y(x)^\top \left[-\textstyle\sum_{i\in I} u_i(x) \nabla_y g_i(x, y(x))\right], \\ &= \nabla_x f(x, y(x)) - \sum_{i \in I} u_i(x) \left[\nabla y(x)^\top \nabla_y g_i(x, y(x))\right], \\ &= \nabla_x f(x, y(x)) + \sum_{i \in I} u_i(x) \nabla_x g_i(x, y(x)). \end{aligned} \tag{8.7}$$

Moreover, it is clear from (8.4) and the last line of equation (8.7) that when the value function φ is smooth, unlike in the context of $y(\cdot)$, the expression of its gradient needs only first order information for the functions involved in the lower-level problem. This remains true even if φ is not smooth. Before we discuss this aspect, note that the converse of implication (8.2) is not true. In particular, for the example of parametric problem in (6.4), the optimal solution function $y(\cdot)$ is nonsmooth at 0 and 1 (see Figure 2), while the value function φ is continuously differentiable at these same points (see Figure 3).

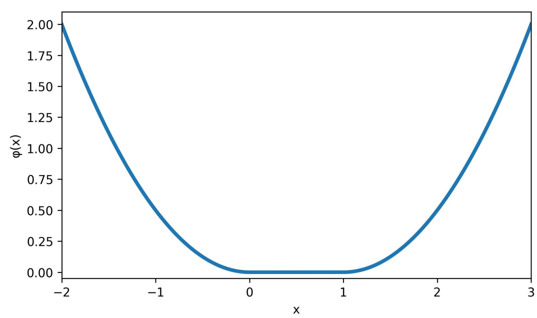

Fig. 3: Graph of the lower-level value function φ associated to problem (6.4).

Now, observe that even when φ is nonsmooth, a subgradient of the function can still typically have the form in the last line of equation (8.7). To see this, assume that the lower-level feasible set-valued mapping Y is given in (6.1) and let the functions f and g_i for $i = 1, \ldots, q$ are smooth and fully convex; i.e., convex in the joint variable (x, y). Furthermore, let $S(x) \neq \emptyset$ for all $x \in U \supset X$. Then φ is locally Lipschitz continuous near any point $x \in X$; furthermore, for any $y \in S(x)$, there exists $u \in \mathbb{R}^q$ such that

$$u \in \Lambda(x, y) \quad \text{and} \quad \nabla_x f(x, y) + \nabla_x g(x, y)^\top u \ \in \partial\varphi(x), \tag{8.8}$$

where $\partial\varphi(x)$ represents the subdifferential of φ at x, in the sense of convex analysis. More precisely, if a constraint qualification (e.g., the MFCQ) holds at $(x, y) \in \mathrm{gph}S$ for the lower-level constraint, then it holds that

$$\partial\varphi(x) = \{\, \nabla_x \ell(x, y, u) |\ u \in \Lambda(x, y)\}\,; \tag{8.9}$$

see, e.g., [269] for details on how to generate this formula. Recall that in the formula (8.9), ℓ denotes the lower-level Lagrangian function (6.2). In the case where the full convexity assumption on the lower-level problem is not satisfied, according to Gauvin and Dubeau [102], if we assume that the set-valued mapping Y (6.1) is nonempty and uniformly compact near x and the MFCQ holds at y in $Y(x)$ for fixed x (for all $y \in S(x)$), then φ is Lipschitz continuous near x and its Clarke subdifferential can be estimated as

$$\partial\varphi(x) \subseteq \mathrm{co} \left\{ \bigcup_{y \in S(x)} \bigcup_{u \in \Lambda(x,y)} \{\nabla_x \ell(x, y, u)\} \right\}, \tag{8.10}$$

where "co" stands for the convex hull of the corresponding set. This formula can be simplified in various ways, depending on the adjustments made on the assumptions; see, [272] and references therein for an overview of possible simplification scenarios. For instance, it naturally results from (8.10) that φ is strictly differentiable with

$$\nabla\varphi(x) = \nabla_x \ell(x, y, u) \ \text{ provided that } \ \{y\} = S(x) \text{ and } \{u\} = \Lambda(x, y). \tag{8.11}$$

Thanks to the formulas of the subdifferential of φ that have emerged from this discussion, *pure* first and second order methods can be constructed to solve problem (LLVF), as it will be clear below when we discuss solution algorithms. However, this does not necessarily make the problem easy to solve. For instance, similarly to problem (KKT), most classical CQs fail for problem (LLVF); see, e.g., [77, 270]. Additionally, analogously to ($\mathrm{P_i}$), problem (LLVF) is only implicitly defined and nonsmooth in general.

8.2. Optimality conditions. Many papers have been written on optimality conditions for problem (LLVF), mainly focusing on the use of variational analysis tool, considering the possible nonsmoothness of φ; see, e.g., [270, 69, 77] and references therein. If (x, y) is a local optimal solution of problem (LLVF), then, the possibly simplest class of necessary optimality conditions for problem (LLVF) at this point is

$$\left.\begin{aligned} \nabla_x F(x, y) + \nabla_x g(x, y)^\top (u - \lambda w) + \nabla G(x)^\top v = 0 \\ \nabla_y F(x, y) + \nabla_y g(x, y)^\top (u - \lambda w) = 0 \\ \nabla_y f(x, y) + \nabla_y g(x, y)^\top w = 0 \\ u \geq 0, \ \ g(x, y) \leq 0, \ \ u^\top g(x, y) = 0 \\ v \geq 0, \ \ G(x) \leq 0, \ \ v^\top G(x) = 0 \\ w \geq 0, \ \ g(x, y) \leq 0, \ \ w^\top g(x, y) = 0 \\ \lambda \geq 0 \end{aligned}\right\} \tag{StatsCond}$$

with u, v, and λ representing upper-level Lagrange multipliers, respectively associated to the constraints $y \in Y(x)$, $x \in X$, and $f(x, y) - \varphi(x) \leq 0$, while using the description of the upper-level feasible set in (7.5). Note that here, the vector w corresponds to the lower-level Lagrange multiplier associated to the calculation of a subgradient of φ at x in the spirit of the formulas (8.9), (8.10), and (8.11).

An interesting feature of the stationarity system (StatsCond) is that it does not explicitly depend on the optimal value function φ. However, feasibility is satisfied if the lower-level problem is convex, given that this ensures that $y \in S(x)$ if and only if there exist $w \in \mathbb{R}^q$ such that the conditions in lines three and six are satisfied; cf. (7.2).

The optimality conditions (StatsCond) can hold at a point (x, y) under the following assumptions (see [270, 69, 77], for example, for relevant theory):

(i) Calmness of the set-valued mapping $\Phi : \mathbb{R} \rightrightarrows \mathbb{R}^m$ defined below at $(0, x, y)$:

$$\Phi(\theta) := \{(x, y) \in \mathbb{R}^n \times \mathbb{R}^m \mid y \in Y(x),\;\; f(x, y) - \varphi(x) \leq \theta\};$$

(ii) Fulfillment of the MFCQ at x for the upper-level constraint described in (7.5);

(iii) Fulfillment of the MFCQ at y in $Y(x)$;

(iv) Inner semicontinuous of S at (x, y) or full convexity of the lower-level problem.

Note that the lower-level problem is fully convex if the functions f and g_i, for $i = 1, \ldots, p$, are convex in (x, y). The concept of inner semicontinuity of a set-valued mapping (see, e.g., [196]) is weaker than, but closely related to, the notion of lower semicontinuity, while, similarly, the calmness of a set-valued mapping is weaker than the Aubin property (extension of Lipschitz continuity to set-valued mappings). For the precise mathematical definition of calmness and its characterizations and related applications, interested readers are referred to [120, 121]. For further discussion on all these assumptions in the context of bilevel optimization and the derivation of necessary optimality conditions for problem (LLVF) interested readers can consult the references [69, 81, 197, 161, 270, 266, 268, 74, 77, 190].

For the avoidance of any doubt, it is worth recalling that necessary optimality conditions such as those in (StatsCond) can be used to achieve at least three goals in relation to (P): (a) They can be used as stopping criteria for numerical algorithms to solve problem (LLVF). (b) They could be combined with suitable second order sufficient conditions to establish that a point is locally optimal for (LLVF); suitable second order sufficient conditions ensuring that necessary conditions of the type in (StatsCond) can lead to locally optimal points for (LLVF) are developed in [89, 191]. (c) Conditions such as (StatsCond) can be used to build second order methods. More details on works related to points (a) and (c) in the development of numerical methods for (LLVF) will be discussed later in this section.

8.3. Pure first order–type methods. In [204], a usual augmented Lagrangian function is considered and minimized with a bundle method; in this paper, we have unperturbed lower-level constraints, but no upper-level constraint. In [205], the generalized equation, consisting of replacing $y \in S(x)$ by

$$0 \in \nabla_y f(x, y) + N_{Y(x)}(y),$$

where $N_{Y(x)}(y)$ denotes a normal cone to $Y(x)$ at the point y, is considered together with the implicit function and LLVF reformulations to address constrained lower-level optimistic bilevel programs, with special focus on the latter two models. The constraints of problem (LLVF) are fully penalized, under the assumption that the calmness condition, in the sense of Clarke [56], is satisfied. For each of the transformations (i.e., the corresponding ($\mathrm{P_i}$) and (LLVF) problems), focus in [205] is on how to compute elements from the subdifferentials of the corresponding nonsmooth objective functions. This enables the use of the NDO (nondifferentiable optimization) solver based on the work in [142] (see latest edition in [143]) to compute solutions for the implicit function and LLVF models.

Overall, in terms of developing first order methods, any algorithmic technique that requires only first order information to proceed can be explored in the context of problem (LLVF). Recently, a few papers on pure first order methods have appeared in the bilevel learning literature; see, e.g., [171, 156, 155, 100, 269]. The main approach in this context has consisted of solving the penalized problem

$$\min_{x,y} F(x,y) + \lambda\left(f(x,y) - \varphi(x)\right), \tag{8.12}$$

provided there is no upper– nor lower-level constraints. In this case, it obviously follows from (8.11) that $\nabla\varphi(x) = \nabla_x f(x,y)$ with $\{y\} = S(x)$, under suitable assumptions. In this context, a gradient descent scheme, similarly to Algorithm 5.1 for problem ($\mathrm{P_i}$), can be developed for (8.12). One of the main challenges here identifying suitable values for the penalization parameter λ; see some relevant analysis in the paper [243].

A notable representative of this line is the fully first-order stochastic approximation (F2SA) method of [156], which provides one of the cleanest complexity theories for the penalized formulation (8.12) in the unconstrained setting. The core difficulty is that the penalty term introduces a bias (since $\lambda < \infty$ does not enforce $y \in S(x)$ exactly), but taking λ too large makes the penalized objective increasingly ill-conditioned, forcing smaller step sizes. F2SA resolves this trade-off by using a *scheduled* penalty parameter $\{\lambda_k\}_{k\geq 0}$: starting from $\lambda_0 > 0$, it *increases* λ_k at a controlled polynomial rate (equivalently, it *decreases* the effective penalty tolerance $1/\lambda_k$), while simultaneously shrinking the primal step sizes to maintain stability as the penalized landscape sharpens. This careful co-evolution of $(\lambda_k, \alpha_k, \beta_k)$ yields explicit non-asymptotic rates in both deterministic and stochastic regimes under the regularity assumptions used by implicit function methods, including iteration complexities to reach an ϵ-stationary point scaling as $\tilde{\mathcal{O}}(\varepsilon^{-3/2})$ in deterministic settings and $\tilde{\mathcal{O}}(\varepsilon^{-5/2})$–$\tilde{\mathcal{O}}(\varepsilon^{-7/2})$ in stochastic settings depending on whether noise affects one or both levels. Closely related penalty-based value function methods, such as [155], develop finite-time guarantees for first-order schemes by linking approximate stationarity of the penalized problem to approximate bilevel optimality through suitable choices of the penalty magnitude.

It is also instructive to contrast these lower-level value function penalization-based rates with those obtained by the barrier-style value function method named BOME, published in [171]. While BOME likewise avoids implicit differentiation by exploiting the value-function constraint, its convergence guarantees are established for a *different* notion of progress, namely a KKT-like residual tailored to the value-function constrained reformulation (rather than directly bounding the norm of the hypergradient.

When a general form of problem (LLVF), involving upper and/or lower-level constraints, is considered, penalizing only the value function constraint as done in (8.12), corresponds to a partial penalization model, given that it remains constrained by both the upper- and lower-level constraints. This partial penalization approach was introduced in [270, 266], via a specialized version of the Clarke concept, label as *partial calmness*, to derive necessary optimality conditions for problem (LLVF). The concept of partial calmness has since then become very prominent in the study of the LLVF reformulation and other optimization problem classes (see, e.g., [267, 172, 273]). For instance, it can be used to replace the calmness concept in (i) above, as a qualification condition, to derive the necessary optimality conditions in (StatsCond). However, in this case, λ will be positive and instead represent the penalty parameter, and not a Lagrange multiplier [270, 69, 77, 265].

Another approach that also builds on the partial penalization model in (8.12) is the difference-of-convex functions (DCA) method. To get a flavor of how this works, recall that if the lower-level objective and constraint functions in problem (P) are fully convex, then the value function φ (8.1) is convex. If we additionally assume that the upper-level objective function F is convex in (x,y), then the functions $F + \lambda f$ and φ are both convex, and therefore, the objective function in (8.12) is the difference of two convex functions for a fixed $\lambda > 0$. Building on a well-established literature on DCA programming, the paper [100] studies this approach in the context of a bilevel hyperparameter selection problem; a particular feature of the DCA method, which differentiates it from the gradient descent-type approaches in the previous papers (e.g., [204] or more generally in the bilevel learning literature), is that at each iteration, a linear approximation of the value function is built using an element from its subdifferential. Furthermore, in [100], the subproblem is made strongly convex by the addition of a proximal term. This approach is generalized in [269] to problems with DC upper-level objective function. The penalty methods in these two papers are same as in [205], with only two differences: (i) the value function is approximated by a linear function based subgradient of φ along the lines of the formulas (8.9), (8.10), and (8.11); and (ii) a proximal term is added to the subproblem for it to be strongly convex.

8.4. Pure second order–type methods. Fixing $\lambda > 0$ in the optimality conditions provided in (StatsCond), as it would be the case if the partial calmness concept is used as a qualification condition, the system can be written as

$$\Phi_\lambda(x,y,u,v,w) := \begin{bmatrix} \nabla_{x,y} L_\lambda(x,y,u,v,w) \\ \nabla_y \ell(x,y,w) \\ \left(\phi_{\mathrm{FB}}(u_i,\ -g_i(x,y))\right)_{j=1,\ldots,q} \\ \left(\phi_{\mathrm{FB}}(v_j,\ -G_j(x))\right)_{j=1,\ldots,p} \\ \left(\phi_{\mathrm{FB}}(w_i,\ -g_i(x,y))\right)_{j=1,\ldots,q} \end{bmatrix} = 0, \tag{8.13}$$

where the lower-level Lagrangian function ℓ and the Fischer-Burmeister function ϕ_{FB} are defined in (6.2) and (7.10), respectively. Note that the Lagrangian function L_λ is defined for any (x,y,u,v,w) with $(x,y) \in \mathbb{R}^n \times \mathbb{R}^m$ and $(u,w,v) \in \mathbb{R}^{2q} \times \mathbb{R}^p$ by

$$L_\lambda(x,y,u,v,w) := F(x,y) + v^\top G(x) + (u - \lambda w)^\top g(x,y).$$

Unlike the system (7.11) that involved second order information for the lower-level objective and constraint functions, (8.13) is a pure first order system. In fact, (8.13) is a $(n+m+p+2q) \times (n+2m+p+2q)$ semismooth system of equations. Based on these observations, Gauss-Newton and Levenberg–Marquardt methods are developed to solve the system in [93, 243, 136]. Since the system has m more equations than variables, substituting y in the second and fourth block of the right-hand-side of equation (8.13) with a dummy variable z, we regularize (8.13) to a square semismooth system of equations, which can then enable the development of a semismooth Newton method; see [273, 89] for details in this direction, with a comparison of the numerical performance of (KKT) and (LLVF) conducted in [273]. This comparison suggests that (LLVF) generally leads to a better performance for the examples from the BOLIB library [277], which are essenitally small toy problems.

An important point that needs to be highlighted for the aforementioned works on solving the system (8.13) is that they are all *pure* second order methods, as they do not require any third order derivative information. This would not be the case in the context of problems ($\mathrm{P_i}$) and (KKT) as such techniques will require third order derivative information for the functions involved in the corresponding lower-level problems.

8.5. Other types of LLVF–based methods. Let us observe that the value function constraint, which represents the main component of the feasible set of problem (LLVF), can be equivalently written as

$$f(x,y) - f(x,z) \leq 0 \ \text{ for } \ z \in Y(x),$$

which is a generalized semi-infinite constraint. Considering this representation, the series of papers [194, 82, 147, 252] exploits semi-infinite type reformulations or related relaxation techniques to build global optimization algorithms for problem (P), using branch and bound procedures as key ingredients to ensure numerical efficiency. The LLVF reformulation has also been used (see, e.g., [238, 90, 91]) to develop cutting plane–type numerical algorithms for mixed-integer bilevel programs in the case where all involved functions are linear.

The paper [169] proposes an approximation algorithm that relies on a concept of entropy integral function as a smoothing function for the optimal value function φ when $Y(x) = Y$ (i.e., unperturbed). We also have an algorithm in [70], where the value function (8.1) for $f(x, y) := x^\top y$ and $Y(x) := \{y \in \mathbb{R}^m | \ Ay \le b\}$ is iteratively approximated by a linear approximation. The advantage with the structure of the lower-level problem here is that as Y is an unperturbed polyhedral set, an optimal solution for the lower-level problem can be found at one of its extreme points for all $x \in X$, under a mild assumption (e.g., if Y is a bounded polyhedral). For a fully linear upper-level problem, it is shown that the algorithm converges to a global or local optimal solution, depending on the specific assumption scenario considered. The idea in [70] is later extended to more general problem classes in [71]. However, similarly to most of the schemes just described above, the scalability of this class of method to relatively large problem classes is uncertain. Hence, their applicability to bilevel learning problems might be very limited.

It might also be useful to mention that in [158], the reformulation (LLVF) is used as base for the construction of a Generalized Nash equilibrium problem (GNEP) that is closely related to problem (P). This GNEP model is then exploited in [159] to build a numerical method to compute approximate stationarity points for problem (LLVF).

9. Comparing the reformulations of the optimistic bilevel program. The main focus of this paper is the standard optimistic bilevel optimization problem (P), expressed in the implicit function model ($\mathrm{P_i}$) when condition (2.1) is satisfied, and otherwise in the KKT and LLVF reformulations (KKT) and (LLVF), respectively, when (2.2) holds. It therefore seems natural to take a little moment in this section to briefly compare the single-level reformulations ($\mathrm{P_i}$), (KKT), and (LLVF) of problem (P).

Of course, these three problems are so significantly different from each other, with the challenging component of ($\mathrm{P_i}$) appearing in its objective function, while problems (KKT) and (LLVF) have very complex feasible sets. In terms of the smoothness of ($\mathrm{P_i}$) and (LLVF), we have condition (8.2), which implies that under suitable conditions, (LLVF) will be automatically smooth if ($\mathrm{P_i}$) is. However, the converse of this implication is not true as demonstrated by the example in (6.4); cf. Figures 2 and 3. For specific comparisons between (KKT) and (LLVF) from multiple perspectives, interested readers are referred to [273].

Table 2: Some basic comparisons of the reformulations of the standard optimistic bilevel optimization problem. $\sim$ denotes partial guarantees (often for smoothed/MPEC surrogates, requiring bounded iterates). * denotes rate results proved under lower-level strong convexity and lower-level Hessian smoothness (as in implicit-function analyses).

	Property	($\mathrm{P_i}$)	(KKT)	(LLVF)
SLR requirements	Smoothness	✓	✓	✗
	Convexity	✓	✓	✗
	Strong convexity	✓	✗	✗
	LLCQ	✓	✓	✗
ULCQ fulfilment	UMFCQ	✓	✗	✗
	Can UMFCQ be restored?	✓	✓	✗
Derivative requirement for methods	1D1OM	✗	✗	✓
	2D2OM	✗	✗	✓
Convergence	Deterministic rates	✓	$\sim$	✓*
	Stochastic rates	✓	✗	✓*

Overall, Table 2 summarizes key comparisons between ($\mathrm{P_i}$), (KKT), and (LLVF) from four perspectives (while assuming that the lower-level problem is constrained): (a) The requirements needed to formally write the corresponding single-level reformulation (SLR) of

(P) with LLCQ standing for the *lower-level constraint qualification* in reference to whether one is needed to write the corresponding reformulation. (b) The behavior of each reformulation with regards to a suitable version of the MFCQ that we label here as upper-level constraint qualification (ULCQ); a key thing to note is that the MFCQ fails for both the KKT and LLVF reformulations, when their constraints are treated as usual equality and inequality constraints. However, the MFCQ can be restored for (KKT) by suitably addressing the combinatorial structure in the complementarity conditions; see, e.g., [78, 220]. So far, no restoration approach for the MFCQ has been discovered for problem (LLVF); see, e.g., [273] for a related discussion. (c) The third block of Table 2 corresponds to the derivative requirements for the corresponding reformulation in the sense that the abbreviation 1D1OM is used to refer to *whether only first order derivatives are enough to develop a first order method* for the corresponding reformulation, while 2D2OM refers to *whether only second order derivatives are enough to develop second order methods.*

Finally, the fourth aspect (d) concerns the convergence guarantees currently available in the BL literature, distinguishing between deterministic and stochastic rates. From the standpoint of *available complexity guarantees*, the picture is currently quite uneven across reformulations. For the LLVF route, recent work has established explicit non-asymptotic rates for *pure first-order* schemes, most prominently via barrier/penalty mechanisms; see BOME [171] and the penalty-based methods in [156, 155]. These guarantees are typically developed for the *unconstrained* bilevel setting (or after handling constraints separately) and require lower-level regularity such as strong convexity/PL-type conditions and smoothness to control the bias induced by finite penalty/barrier parameters and finite inner-loop accuracy. In contrast, for KKT-type reformulations, while classical theory clarifies how constraint qualifications may be recovered by treating complementarity carefully [220, 78], finite-time rate results in modern machine learning settings are comparatively limited and often appear in forms that are *partial* in the sense of Table 2: existing analyses typically provide decay bounds for a *KKT residual* of a smoothed/regularized surrogate and rely on additional technical conditions such as bounded iterates/compactness and smoothing schedules (e.g., method-of-multipliers/augmented-Lagrangian developments); see, e.g., [187, 178].

10. Conclusions and final remarks. Based on the survey of the BL and BO literature conducted in this paper, the following concluding observations can be made:

(i) The implicit function approach, labelled as $(\mathrm{P_i})$, has been working very well in solving specific classes of BL problems, thanks to efficient approximations of the lower-level optimal solution function $y(\cdot)$ and its Jacobian $\nabla y(\cdot)$, when both functions are well-defined. However, this approach has many limitations, as not only, ensuring that the required basic assumption (2.1) is satisfied is very difficult, but making sure that the lower-level optimal solution function is smooth function is even harder; cf. discussion in Section 6. As also highlighted in the latter section, things get worse when the lower-level problem is contrained.

(ii) For some special classes of BL problems, alternative methods to state of the art techniques, have been shown not only to better capture the corresponding task, but numerically, their resulting BO formulation can be more efficiently solved, even when lower-level problem is constrained. This is the case, for example, for hyperparameter optimization is machine learning, as discussed in Subsection 7.6. The BO formulation of the problem can be numerically solved more efficiently and accurately in comparison to the state of the art grid search and Bayesian approaches, which are standard in the main stream machine learning literature, and also the most widely used in practice. However, such BO–based tools have not yet made their way to main stream machine learning infrastructures such as widely used open source libraries. A likely reason is not only limited awareness, but also the practical complexity of current BO solvers: despite their principled formulation and strong numerical results, many methods rely on several algorithmic *meta-parameters*—for example, the number of inner iterations, truncation depth, damping/regularization, linear-solver tolerances, and step sizes for both upper- and lower-level updates. These choices can have a major effect on stability, memory footprint, and runtime, which makes robust off-the-shelf deployment in mainstream machine-learning libraries more difficult. The adaptive methods discussed in Subsection 5.4 might help mitigate these issues but are in early stage of development.

(iii) There is an exponential number of applications of bilevel optimization in machine learning. But for most of these problem–types, state of the art BL methods cannot be applied. For example, we can mention problems with lower-level constraints, where the lower-level optimal solution set-valued is not single-valued for some upper-level variables, as well problems with nonsmooth lower-level objective or constraint functions, as outlined in Section 4. For many of these problems, like the pessimistic case that results from (2.2), for example, not much progress has been made in solving them in the general BO literature.

(iv) There is also a wide range of numerical techniques in the BO literature that remain unexplored in the context of BL; this paper has provided a brief overview of these approaches, with the hope that in the near future, they will draw the attention that they deserve. As highlighted in Sections 7–8, a key limitation of the classical BO algorithms is that many are hard to scale, especially when they involve lower-level constraints. The powerful derivative approximation schemes from state of the art numerical schemes for BL (cf. Section 5) are a potential way forward in the context of methods for problems (KKT) and (LLVF). Work to deploy such ideas to solve the LLVF reformulation has started in the context of unconstrained lower-level problems as highlighted in Section 8.

The second challenge that comes with inequality constraints is the combinatorial nature of the formula involved in BO numerical schemes (see, e.g., the combinatorial nature of the systems (6.3) and (8.13), as well as in the complementarity conditions in the feasible set of problem (KKT)). With regards to this, there is a good chance that GPU–based methods to scale up optimization algorithms, as outlined in [223, 30], for example, could be potential ways forward to enable classical BO methods to solve reasonable size BL problems. Additionally, the emergence of quantum computing and its potential to enable the scaling of enumeration–based algorithms (see, e.g., [162]) is also a path that, in combination with derivative approximation–based schemes, could provide paths to accelerate classical BO methods to solve realistic BL problems.

Acknowledgments. Pradeep Sharma collected and organized some of the references that helped in the writing of Section 4. The fourth author would like to thank Stefan Güttel (The University of Manchester), Coralia Cartis (University of Oxford), Harry Zheng and Panos Parpas (Imperial College), and Oliver Stein (Karlsruhe Institute of Technology) for their seminar invitations, where some of the material in this paper was presented, and the interesting discussions that inspired the development of some of the elements in the text.

REFERENCES

[1] A. Aboussoror, S. Adly, and F. E. Saissi, *Strong-weak nonlinear bilevel problems: existence of solutions in a sequential setting*, Set-valued and Variational Analysis, 25 (2017), pp. 113–132.
[2] A. Aboussoror and P. Loridan, *Strong-weak Stackelberg problems in finite dimensional spaces*, Serdica Mathematical Journal, 21 (1995), pp. 151p–170p.
[3] H. Akaike, *A new look at the statistical model identification*, IEEE Transactions on Automatic Control, 19 (1974), pp. 716–723.
[4] J. H. Alcantara and J.-S. Chen, *Neural networks based on three classes of NCP-functions for solving nonlinear complementarity problems*, Neurocomputing, 359 (2019), pp. 102–113.
[5] J. H. Alcantara, C.-H. Lee, C. T. Nguyen, Y.-L. Chang, and J.-S. Chen, *On construction of new NCP functions*, Operations Research Letters, 48 (2020), pp. 115–121.
[6] J. H. Alcantara, C. T. Nguyen, T. Okuno, A. Takeda, and J.-S. Chen, *Unified smoothing approach for best hyperparameter selection problem using a bilevel optimization strategy*, Mathematical Programming, 212 (2025), pp. 479–518.
[7] K. Antonakopoulos, S. Sabach, L. Viano, M. Hong, and V. Cevher, *Adaptive bilevel optimization*, ACM/IMS Journal of Data Science, 2 (2025), pp. 1–29.
[8] M. Arbel and J. Mairal, *Amortized implicit differentiation for stochastic bilevel optimization*, in International Conference on Learning Representations, 2022.
[9] M. Arbel and J. Mairal, *Non-convex bilevel games with critical point selection maps*, Advances in Neural Information Processing Systems, 35 (2022), pp. 8013–8026.
[10] S. Bai, J. Z. Kolter, and V. Koltun, *Deep equilibrium models*, in Advances in Neural Information

Processing Systems, vol. 32, 2019.
[11] J. F. Bard, *Practical bilevel optimization: algorithms and applications*, vol. 30, Kluwer Academic Publishers, 1998.
[12] J. F. Bard and J. E. Falk, *An explicit solution to the multi-level programming problem*, Computers & Operations Research, 9 (1982), pp. 77–100.
[13] H. H. Bauschke and P. L. Combettes, *Correction to: convex analysis and monotone operator theory in Hilbert spaces*, in Convex analysis and monotone operator theory in Hilbert spaces, Springer, 2020, pp. C1–C4.
[14] A. G. Baydin, B. A. Pearlmutter, A. A. Radul, and J. M. Siskind, *Automatic differentiation in machine learning: a survey*, Journal of Machine Learning Research, 18 (2018), pp. 1–43.
[15] A. Behrouz, M. Razaviyayn, P. Zhong, and V. Mirrokni, *Nested learning: The illusion of deep learning architectures*, Arxiv Preprint Arxiv:2512.24695, (2025).
[16] A. Behrouz, P. Zhong, and V. Mirrokni, *Titans: Learning to memorize at test time*, Arxiv Preprint Arxiv:2501.00663, (2024).
[17] I. Benchouk, L. Jolaoso, K. Nachi, and A. Zemkoho, *Scholtes relaxation method for pessimistic bilevel optimization*, Set-valued and Variational Analysis, 33 (2025), p. 10.
[18] I. Benchouk, L. Jolaoso, K. Nachi, and A. Zemkoho, *Relaxation methods for pessimistic bilevel optimization*, Set-valued and Variational Analysis, 34 (2026), p. 1.
[19] D. Benfield, S. Coniglio, M. Kunc, P. T. Vuong, and A. Zemkoho, *Classification under strategic adversary manipulation using pessimistic bilevel optimisation*, Arxiv:2410.20284, (2024).
[20] Y. Bengio, *Gradient-based optimization of hyperparameters*, Neural Computation, 12 (2000), pp. 1889–1900, https://doi.org/10.1162/089976600300015187.
[21] K. P. Bennett, J. Hu, X. Ji, G. Kunapuli, and J.-S. Pang, *Model selection via bilevel optimization*, in The 2006 IEEE International Joint Conference on Neural Network Proceedings, IEEE, 2006, pp. 1922–1929, https://doi.org/10.1109/IJCNN.2006.246935.
[22] K. P. Bennett and G. Kunapuli, *A bilevel optimization approach to machine learning*, 2008, https://api.semanticscholar.org/CorpusID:3444089.
[23] K. P. Bennett, G. Kunapuli, J. Hu, and J.-S. Pang, *Bilevel optimization and machine learning*, in IEEE world congress on computational intelligence, Springer, 2008, pp. 25–47.
[24] K. P. Bennett and G. M. Moore, *Bilevel programming algorithms for machine learning model selection*, 2010, https://api.semanticscholar.org/CorpusID:124751727.
[25] K. P. Bennett and E. Parrado-Hernández, *The interplay of optimization and machine learning research*, The Journal of Machine Learning Research, 7 (2006), pp. 1265–1281.
[26] Q. Bertrand, Q. Klopfenstein, M. Blondel, S. Vaiter, A. Gramfort, and J. Salmon, *Implicit differentiation of lasso-type models for hyperparameter optimization*, in International Conference on Machine Learning, PMLR, 2020, pp. 810–821.
[27] Q. Bertrand, Q. Klopfenstein, M. Massias, M. Blondel, S. Vaiter, A. Gramfort, and J. Salmon, *Implicit differentiation for fast hyperparameter selection in non-smooth convex learning*, The Journal of Machine Learning Research, 23 (2022), pp. 6680–6722.
[28] W. Bialas and M. Karwan, *On two-level optimization*, IEEE Transactions on Automatic Control, 27 (1982), pp. 211–214.
[29] W. F. Bialas and M. H. Karwan, *Two-level linear programming*, Management Science, 30 (1984), pp. 1004–1020, http://www.jstor.org/stable/2631591 (accessed 2025-04-17).
[30] A. L. Bishop, J. Z. Zhang, S. Gurumurthy, K. Tracy, and Z. Manchester, *Relu-qp: A gpu-accelerated quadratic programming solver for model-predictive control*, in 2024 IEEE International Conference on Robotics and Automation (ICRA), IEEE, 2024, pp. 13285–13292.
[31] M. Blondel, Q. Berthet, M. Cuturi, R. Frostig, S. Hoyer, F. Llinares-López, F. Pedregosa, and J.-P. Vert, *Efficient and modular implicit differentiation*, in Advances in Neural Information Processing Systems, vol. 35, 2022, pp. 5230–5242.
[32] G. M. Bollas, P. I. Barton, and A. Mitsos, *Bilevel optimization formulation for parameter estimation in vapor–liquid (–liquid) phase equilibrium problems*, Chemical Engineering Science, 64 (2009), pp. 1768–1783.
[33] J. Bolte and E. Pauwels, *Conservative set valued fields, automatic differentiation, stochastic gradient methods and deep learning*, Mathematical Programming, 188 (2021), pp. 19–51.
[34] J. Bolte, L. Tam, and E. Pauwels, *Nonsmooth implicit differentiation for machine learning*, in Advances in Neural Information Processing Systems, vol. 35, 2022, pp. 12714–12726.
[35] G. Bouza, E. Quintana, and C. Tammer, *A steepest descent method for set optimization problems with set-valued mappings of finite cardinality*, Journal of Optimization Theory and Applications, 190 (2021), pp. 711–743.
[36] J. Bracken and J. T. McGill, *Mathematical programs with optimization problems in the constraints*, Operations Research, 21 (1973), pp. 37–44.
[37] J. Bracken and J. T. McGill, *Defense applications of mathematical programs with optimization problems in the constraints*, Operations Research, 22 (1974), pp. 1086–1096.
[38] J. Bracken and J. T. McGill, *A method for solving mathematical programs with nonlinear programs in the constraints*, Operations Research, 22 (1974), pp. 1097–1101.
[39] M. Brückner and T. Scheffer, *Stackelberg games for adversarial prediction problems*, in Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data

mining, 2011, pp. 547–555.

[40] M. Campelo, S. Dantas, and S. Scheimberg, *A note on a penalty function approach for solving bilevel linear programs*, Journal of Global Optimization, 16 (2000), p. 245.

[41] W. Candler and R. Norton, *Multi-level programming and development policy*, Technical Report 258, World Bank Staff, Washington D.c., 1977.

[42] W. Candler and R. Norton, *Multilevel programming*, Technical Report 20, World Bank Staff, Washington D.c., 1977.

[43] W. Candler and R. Townsley, *A linear two-level programming problem*, Computers & Operations Research, 9 (1982), pp. 59–76, https://doi.org/https://doi.org/10.1016/0305-0548(82)90006-5, https://www.sciencedirect.com/science/article/pii/0305054882900065.

[44] D. Cao and L. C. Leung, *A partial cooperation model for non-unique linear two-level decision problems*, European Journal of Operational Research, 140 (2002), pp. 134–141.

[45] F. Caruso, M. C. Ceparano, and J. Morgan, *Lower Stackelberg equilibria: from bilevel optimization to Stackelberg games*, Optimization, 74 (2025), pp. 2857–2883.

[46] F. Caruso, M. B. Lignola, and J. Morgan, *Regularization and approximation methods in Stackelberg games and bilevel optimization*, in Bilevel optimization: Advances and next challenges, Springer, 2020, pp. 77–138.

[47] O. Chapelle, V. Vapnik, O. Bousquet, and S. Mukherjee, *Choosing multiple parameters for support vector machines*, Machine Learning, 46 (2002), pp. 131–159.

[48] C. Chen, X. Chen, C. Ma, Z. Liu, and X. Liu, *Gradient-based bi-level optimization for deep learning: A survey*, 2022, https://arxiv.org/abs/2207.11719.

[49] C. Chen and J. Cruz, *Stackelburg solution for two-person games with biased information patterns*, IEEE Transactions on Automatic Control, 17 (1972), pp. 791–798, https://doi.org/10.1109/TAC.1972.1100179.

[50] T. Chen, Y. Sun, Q. Xiao, and W. Yin, *A single-timescale method for stochastic bilevel optimization*, in International Conference on Artificial Intelligence and Statistics, PMLR, 2022, pp. 2466–2488.

[51] T. Chen, Y. Sun, and W. Yin, *Closing the gap: Tighter analysis of alternating stochastic gradient methods for bilevel problems*, in Advances in Neural Information Processing Systems, vol. 34, 2021, pp. 25294–25307.

[52] T. Chen, Y. Sun, and W. Yin, *A single-timescale stochastic bilevel optimization method*, Arxiv Preprint Arxiv:2102.04671, (2021).

[53] T. Chen, Y. Sun, and W. Yin, *Tighter analysis of alternating stochastic gradient method for stochastic nested problems*, Arxiv Preprint Arxiv:2106.13781, (2021).

[54] S. K. Choe and W. Neiswanger, *Betty: An automatic differentiation library for multilevel optimization*, ICLR 2023, (2023).

[55] A. E. Cinà, K. Grosse, A. Demontis, S. Vascon, W. Zellinger, B. A. Moser, A. Oprea, B. Biggio, M. Pelillo, and F. Roli, *Wild patterns reloaded: A survey of machine learning security against training data poisoning*, ACM Computing Surveys, 55 (2023), pp. 1–39.

[56] F. H. Clarke, *Optimization and nonsmooth analysis*, SIAM, 1990.

[57] B. Colson, P. Marcotte, and G. Savard, *An overview of bilevel optimization*, Annals of Operations Research, 153 (2007), pp. 235–256.

[58] S. Coniglio, A. Dunn, Q. Li, and A. Zemkoho, *Bilevel hyperparameter optimiza-tion for nonlinear support vector machines*, Https://Optimization-online. Org, (2023), pp. 1–78.

[59] C. Crockett, J. A. Fessler, et al., *Bilevel methods for image reconstruction*, Foundations and Trends® in Signal Processing, 15 (2022), pp. 121–289.

[60] M. Dagréou, P. Ablin, S. Vaiter, and T. Moreau, *A framework for bilevel optimization that enables stochastic and global variance reduction algorithms*, Advances in Neural Information Processing Systems, 35 (2022), pp. 26698–26710.

[61] M. Dagréou, T. Moreau, S. Vaiter, and P. Ablin, *A lower bound and a near-optimal algorithm for bilevel empirical risk minimization*, in Proceedings of the 27th International Conference on Artificial Intelligence and Statistics, PMLR, 2024, pp. 82–90.

[62] T. De Luca, F. Facchinei, and C. Kanzow, *A semismooth equation approach to the solution of nonlinear complementarity problems*, Mathematical Programming, 75 (1996), pp. 407–439.

[63] A. H. De Silva, *Sensitivity formulas for nonlinear factorable programming and their application to the solution of an implicitly defined optimization model of United States crude oil production*, Doctoral Dissertation, George Washington University, Washington, D.C., 1979.

[64] S. Dempe, *A bundle algorithm applied to bilevel programming problems with non-unique lower level solutions*, Computational Optimization and Applications, 15 (2000), pp. 145–166.

[65] S. Dempe, *Foundations of Bilevel Programming*, vol. 61 of Nonconvex Optimization and Its Applications, Kluwer Academic Publishers, Dordrecht, 2002.

[66] S. Dempe, *Annotated bibliography on bilevel programming and mathematical programs with equilibrium constraints*, Optimization, 52 (2003), pp. 333–359.

[67] S. Dempe and J. F. Bard, *Bundle trust-region algorithm for bilinear bilevel programming*, Journal of Optimization Theory and Applications, 110 (2001), pp. 265–288.

[68] S. Dempe and J. Dutta, *Is bilevel programming a special case of a mathematical program with complementarity constraints?*, Mathematical Programming, 131 (2012), pp. 37–48.

[69] S. Dempe, J. Dutta, and B. Mordukhovich, *New necessary optimality conditions in optimistic bilevel programming*, Optimization, 56 (2007), pp. 577–604.
[70] S. Dempe and S. Franke, *Solution algorithm for an optimistic linear Stackelberg problem*, Computers & Operations Research, 41 (2014), pp. 277–281.
[71] S. Dempe and S. Franke, *On the solution of convex bilevel optimization problems*, Computational Optimization and Applications, 63 (2016), pp. 685–703.
[72] S. Dempe and P. Mehlitz, *Duality-based single-level reformulations of bilevel optimization problems*, Journal of Optimization Theory and Applications, 205 (2025), p. 26.
[73] S. Dempe, B. S. Mordukhovich, and A. B. Zemkoho, *Sensitivity analysis for two-level value functions with applications to bilevel programming*, SIAM Journal on Optimization, 22 (2012), pp. 1309–1343.
[74] S. Dempe, B. S. Mordukhovich, and A. B. Zemkoho, *Necessary optimality conditions in pessimistic bilevel programming*, Optimization, 63 (2014), pp. 505–533.
[75] S. Dempe and H. Schmidt, *On an algorithm solving two-level programming problems with nonunique lower level solutions*, Computational Optimization and Applications, 6 (1996), pp. 227–249.
[76] S. Dempe and A. Zemkoho, *Bilevel optimization*, in Springer optimization and its applications, vol. 161, Springer, 2020.
[77] S. Dempe and A. B. Zemkoho, *The generalized Mangasarian-Fromowitz constraint qualification and optimality conditions for bilevel programs*, Journal of Optimization Theory and Applications, 148 (2011), pp. 46–68.
[78] S. Dempe and A. B. Zemkoho, *On the Karush–Kuhn–Tucker reformulation of the bilevel optimization problem*, Nonlinear Analysis: Theory, Methods & Applications, 75 (2012), pp. 1202–1218.
[79] S. Dempe and A. B. Zemkoho, *The bilevel programming problem: reformulations, constraint qualifications and optimality conditions*, Mathematical Programming, 138 (2013), pp. 447–473.
[80] A. H. deSilva and G. P. McCormick, *Implicitly defined optimization problems*, Annals of Operations Research, 34 (1992), pp. 107–124.
[81] N. Dinh, B. Mordukhovich, and T. T. Nghia, *Subdifferentials of value functions and optimality conditions for dc and bilevel infinite and semi-infinite programs*, Mathematical Programming, 123 (2010), pp. 101–138.
[82] H. Djelassi, A. Mitsos, and O. Stein, *Recent advances in nonconvex semi-infinite programming: Applications and algorithms*, EURO Journal on Computational Optimization, 9 (2021), p. 100006.
[83] N. R. Draper and H. Smith, *Applied regression analysis*, vol. 326, John Wiley & Sons, 1998.
[84] L. Eriksson, E. Johansson, N. Kettaneh-Wold, C. Wikström, and S. Wold, *Design of experiments*, Principles and Applications, Learn Ways Ab, Stockholm, (2000).
[85] C. Fan, G. Choné-Ducasse, M. Schmidt, and C. Thrampoulidis, *Bisls/sps: Auto-tune step sizes for stable bi-level optimization*, Arxiv Preprint Arxiv:2305.18666, (2023).
[86] A. V. Fiacco, *Introduction to sensitivity and stability analysis in non linear programming*, New York: Academic Press,, 1983.
[87] C. Finn, P. Abbeel, and S. Levine, *Model-agnostic meta-learning for fast adaptation of deep networks*, in Proceedings of the 34th International Conference on Machine Learning (ICML), 2017, pp. 1126–1135, https://proceedings.mlr.press/v70/finn17a.html.
[88] A. Fischer, *A special Newton-type optimization method*, Optimization, 24 (1992), pp. 269–284.
[89] A. Fischer, A. B. Zemkoho, and S. Zhou, *Semismooth Newton-type method for bilevel optimization: global convergence and extensive numerical experiments*, Optimization Methods and Software, 37 (2022), pp. 1770–1804.
[90] M. Fischetti, I. Ljubić, M. Monaci, and M. Sinnl, *A new general-purpose algorithm for mixed-integer bilevel linear programs*, Operations Research, 65 (2017), pp. 1615–1637.
[91] M. Fischetti, I. Ljubić, M. Monaci, and M. Sinnl, *On the use of intersection cuts for bilevel optimization*, Mathematical Programming, 172 (2018), pp. 77–103.
[92] M. L. Flegel and C. Kanzow, *On m-stationary points for mathematical programs with equilibrium constraints*, Journal of Mathematical Analysis and Applications, 310 (2005), pp. 286–302.
[93] J. Fliege, A. Tin, and A. Zemkoho, *Gauss–Newton-type methods for bilevel optimization*, Computational Optimization and Applications, 78 (2021), pp. 793–824.
[94] J. Fortuny-Amat and B. McCarl, *A representation and economic interpretation of a two-level programming problem*, The Journal of the Operational Research Society, 32 (1981), pp. 783–792, http://www.jstor.org/stable/2581394 (accessed 2025-04-17).
[95] L. Franceschi, M. Donini, P. Frasconi, and M. Pontil, *Forward and reverse gradient-based hyperparameter optimization*, 2017, https://arxiv.org/abs/1703.01785.
[96] L. Franceschi, M. Donini, V. Perrone, A. Klein, C. Archambeau, M. Seeger, M. Pontil, and P. Frasconi, *Hyperparameter optimization in machine learning*, Foundations and Trends® in Machine Learning, 18 (2025), pp. 975–1109.
[97] L. Franceschi, P. Frasconi, S. Salzo, R. Grazzi, and M. Pontil, *Bilevel programming for hyperparameter optimization and meta-learning*, 2018, https://arxiv.org/abs/1806.04910.
[98] J. Frecon, S. Salzo, and M. Pontil, *Bilevel learning of deep representations.*
[99] A. Galántai, *Properties and construction of NCP functions*, Computational Optimization and Applications, 52 (2012), pp. 805–824.

[100] L. L. Gao, J. Ye, H. Yin, S. Zeng, and J. Zhang, *Value function based difference-of-convex algorithm for bilevel hyperparameter selection problems*, in International Conference on Machine Learning, PMLR, 2022, pp. 7164–7182.

[101] L. L. Gao, J. J. Ye, H. Yin, S. Zeng, and J. Zhang, *Moreau envelope based difference-of-weakly-convex reformulation and algorithm for bilevel programs*, Arxiv:2306.16761, (2023).

[102] J. Gauvin and F. Dubeau, *Differential properties of the marginal function in mathematical programming*, in Optimality and Stability in Mathematical Programming, Springer, 2009, pp. 101–119.

[103] A. Geoffrion, *Coordination of two-level organizations with multiple objectives*, Techniques of Optimization, (1972).

[104] S. Ghadimi and M. Wang, *Approximation methods for bilevel programming*, Arxiv Preprint Arxiv:1802.02246, (2018).

[105] S. Ghadimi and M. Wang, *Approximation methods for bilevel programming*, Mathematical Programming, 179 (2020), pp. 79–115.

[106] D. Ghosh, Anshika, J.-C. Yao, and X. Zhao, *Quasi-Newton method for set optimization problems with set-valued mapping given by finitely many vector-valued functions*, Numerical Functional Analysis and Optimization, (2025), pp. 1–41.

[107] M. H. Glass, *Bilevel optimization for parameter estimation in thermodynamics*, PhD thesis, Dissertation, Rheinisch-Westfälische Technische Hochschule Aachen, 2018.

[108] X. Gong, J. Hao, and M. Liu, *A nearly optimal single loop algorithm for stochastic bilevel optimization under unbounded smoothness*, in International Conference on Machine Learning, PMLR, 2024, pp. 15854–15892.

[109] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, *Generative adversarial nets*, in Advances in Neural Information Processing Systems, 2014, https://papers.nips.cc/paper/5423-generative-adversarial-nets.pdf.

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

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

[112] R. Grazzi, M. Pontil, and S. Salzo, *Bilevel optimization with a lower-level contraction: Optimal sample complexity without warm-start*, Journal of Machine Learning Research, 24 (2023), pp. 1–37.

[113] R. Grazzi, M. Pontil, and S. Salzo, *Nonsmooth implicit differentiation: deterministic and stochastic convergence rates*, in Proceedings of the 41st International Conference on Machine Learning, 2024, pp. 16250–16274.

[114] E. Grefenstette, B. Amos, D. Yarats, P. M. Htut, A. Molchanov, F. Meier, D. Kiela, K. Cho, and S. Chintala, *Generalized inner loop meta-learning*, 2019, https://arxiv.org/abs/1910.01727.

[115] A. Griewank and A. Walther, *Evaluating derivatives: principles and techniques of algorithmic differentiation*, SIAM, 2008.

[116] Z. Guan, D. Sow, S. Lin, and Y. Liang, *Gradient-based algorithms for pessimistic bilevel optimization*, Journal Placeholder, (2022).

[117] L. Guo, G.-H. Lin, and J. J. Ye, *Solving mathematical programs with equilibrium constraints*, Journal of Optimization Theory and Applications, 166 (2015), pp. 234–256.

[118] Z. Guo, Q. Hu, L. Zhang, and T. Yang, *Randomized stochastic variance-reduced methods for multi-task stochastic bilevel optimization*, Arxiv Preprint Arxiv:2105.02266, (2021).

[119] F. Harder, P. Mehlitz, and G. Wachsmuth, *Reformulation of the m-stationarity conditions as a system of discontinuous equations and its solution by a semismooth Newton method*, SIAM Journal on Optimization, 31 (2021), pp. 1459–1488.

[120] R. Henrion, A. Jourani, and J. Outrata, *On the calmness of a class of multifunctions*, SIAM Journal on Optimization, 13 (2002), pp. 603–618.

[121] R. Henrion and J. V. Outrata, *Calmness of constraint systems with applications*, Mathematical Programming, 104 (2005), pp. 437–464.

[122] R. Hettich and K. O. Kortanek, *Semi-infinite programming: theory, methods, and applications*, SIAM Review, 35 (1993), pp. 380–429.

[123] T. Hoheisel, C. Kanzow, and A. Schwartz, *Theoretical and numerical comparison of relaxation methods for mathematical programs with complementarity constraints*, Mathematical Programming, 137 (2013), pp. 257–288.

[124] M. Hong, H.-T. Wai, Z. Wang, and Z. Yang, *A two-timescale stochastic algorithm framework for bilevel optimization: Complexity analysis and application to actor-critic*, SIAM Journal on Optimization, 33 (2023), pp. 147–180.

[125] F. Huang, *On momentum-based gradient methods for bilevel optimization with nonconvex lower-level*, Arxiv Preprint Arxiv:2303.03944, (2023).

[126] F. Huang and H. Huang, *Biadam: Fast adaptive bilevel optimization methods*, Arxiv Preprint Arxiv:2106.11396, (2021).

[127] M. Huang, X. Chen, K. Ji, S. Ma, and L. Lai, *Efficiently escaping saddle points in bilevel optimization*, Journal of Machine Learning Research, 26 (2025), pp. 1–61.

[128] Y. Huang, Q. Lin, N. Street, and S. Baek, *Federated learning on adaptively weighted nodes by bilevel optimization*, Arxiv Preprint Arxiv:2207.10751, (2022).

[129] H. Huo, R. Liu, and Z. Su, *A perturbed value-function-based interior-point method for perturbed pessimistic bilevel problems*, Arxiv Preprint Arxiv:2401.03636, (2024).
[130] L. Hurwicz, *The theory of economic behavior*, The American Economic Review, 35 (1945), pp. 909–925, http://www.jstor.org/stable/1812602 (accessed 2025-04-01).
[131] F. Iutzeler, E. Pauwels, and S. Vaiter, *Derivatives of stochastic gradient descent in parametric optimization*, Advances in Neural Information Processing Systems, 37 (2024), pp. 118859–118882.
[132] K. Ji and Y. Liang, *Lower bounds and accelerated algorithms for bilevel optimization*, Journal of Machine Learning Research, 24 (2023), pp. 1–56.
[133] K. Ji, M. Liu, Y. Liang, and L. Ying, *Will bilevel optimizers benefit from loops?*, in Advances in Neural Information Processing Systems, vol. 35, 2022, pp. 13828–13839.
[134] K. Ji, J. Yang, and Y. Liang, *Bilevel optimization: Convergence analysis and enhanced design*, in International Conference on Machine Learning, PMLR, 2021, pp. 4859–4869.
[135] L. Jiang, Q. Xiao, V. M. Tenorio, F. Real-Rojas, A. G. Marques, and T. Chen, *A primal-dual-assisted penalty approach to bilevel optimization with coupled constraints*, in Advances in Neural Information Processing Systems, vol. 37, 2024.
[136] L. O. Jolaoso, P. Mehlitz, and A. B. Zemkoho, *A fresh look at nonsmooth Levenberg–Marquardt methods with applications to bilevel optimization*, Optimization, 74 (2025), pp. 2745–2792.
[137] M. Kantarcioğlu, B. Xi, and C. Clifton, *Classifier evaluation and attribute selection against active adversaries*, Data Mining and Knowledge Discovery, 22 (2011), pp. 291–335.
[138] S. S. Keerthi, V. Sindhwani, and O. Chapelle, *An efficient method for gradient-based adaptation of hyperparameters in svm models*, in Advances in Neural Information Processing Systems, 2006, https://proceedings.neurips.cc/paper_files/paper/2006.
[139] P. Khanduri, I. Tsaknakis, Y. Zhang, J. Liu, S. Liu, J. Zhang, and M. Hong, *Linearly constrained bilevel optimization: A smoothed implicit gradient approach*, in International Conference on Machine Learning, PMLR, 2023, pp. 16291–16325.
[140] P. Khanduri, S. Zeng, M. Hong, H.-T. Wai, Z. Wang, and Z. Yang, *A near-optimal algorithm for stochastic bilevel optimization via double-momentum*, in Advances in Neural Information Processing Systems, vol. 34, 2021, pp. 30271–30283.
[141] D. Kim, T. Cho, S. Han, H. Chung, K. Lee, and S. Oh, *Spectral-risk safe reinforcement learning with convergence guarantees*, in Advances in Neural Information Processing Systems, vol. 37, 2024.
[142] K. KIWIEL, *Methods of descent for nondifferentiable optimization*, Lecture Notes in Mathematics, (1985).
[143] K. C. Kiwiel, *Methods of descent for nondifferentiable optimization*, vol. 1133, Springer, 2006.
[144] T. Kleinert, M. Labbé, I. Ljubić, and M. Schmidt, *A survey on mixed-integer programming techniques in bilevel optimization*, Euro Journal on Computational Optimization, 9 (2021), p. 100007.
[145] T. Kleinert, M. Labbé, F. a. Plein, and M. Schmidt, *There's no free lunch: on the hardness of choosing a correct big-m in bilevel optimization*, Operations Research, 68 (2020), pp. 1716–1721.
[146] T. Kleinert and M. Schmidt, *Why there is no need to use a big-m in linear bilevel optimization: A computational study of two ready-to-use approaches*, Computational Management Science, 20 (2023), p. 3.
[147] P.-M. Kleniati and C. S. Adjiman, *Branch-and-sandwich: a deterministic global optimization algorithm for optimistic bilevel programming problems. part i: Theoretical development*, Journal of Global Optimization, 60 (2014), pp. 425–458.
[148] C. D. Kolstad, *A review of the literature on bi-level mathematical programming*, Technical Report La-10284-ms, Us-32, Los Alamos National Laboratory, (1985).
[149] C. D. Kolstad, *Derivative evaluation and computational experience with large bi-level mathematical programs*, Bebr Faculty Working Paper; No. 1266, (1986).
[150] C. D. Kolstad and L. S. Lasdon, *Derivative evaluation and computational experience with large bilevel mathematical programs*, Journal of Optimization Theory and Applications, 65 (1990), pp. 485–499.
[151] J. Kornai and T. Lipták, *Two-level planning*, Econometrica, 33 (1965), pp. 141–169, http://www.jstor.org/stable/1911892 (accessed 2025-04-01).
[152] G. Kunapuli, K. P. Bennett, J. Hu, and J. S. Pang, *Bilevel model selection for support vector machines*, 2007, https://api.semanticscholar.org/CorpusID:2606853.
[153] G. Kunapuli, K. P. Bennett, J. Hu, and J. S. Pang, *Classification model selection via bilevel programming*, Optimization Methods and Software, 23 (2008), pp. 475 – 489, https://api.semanticscholar.org/CorpusID:15800567.
[154] G. Kunapuli, K. P. Bennett, J. Hu, and J.-S. Pang, *Classification model selection via bilevel programming*, Optimization Methods and Software, 23 (2008), pp. 475–489, https://www.tandfonline.com/doi/abs/10.1080/10556780802102586.
[155] J. Kwon, D. Kwon, S. Wright, and R. Nowak, *On penalty methods for nonconvex bilevel optimization and first-order stochastic approximation*, Arxiv Preprint Arxiv:2309.01753, (2023).
[156] J. Kwon, D. Kwon, S. Wright, and R. D. Nowak, *A fully first-order method for stochastic bilevel optimization*, in International Conference on Machine Learning, PMLR, 2023, pp. 18083–18113.
[157] T. Lagos and O. A. Prokopyev, *On complexity of finding strong-weak solutions in bilevel linear programming*, Operations Research Letters, 51 (2023), pp. 612–617.

[158] L. Lampariello and S. Sagratella, *A bridge between bilevel programs and nash games*, Journal of Optimization Theory and Applications, 174 (2017), pp. 613–635.
[159] L. Lampariello and S. Sagratella, *Numerically tractable optimistic bilevel problems*, Computational Optimization and Applications, 76 (2020), pp. 277–303.
[160] L. Lampariello, S. Sagratella, and O. Stein, *The standard pessimistic bilevel problem*, SIAM Journal on Optimization, 29 (2019), pp. 1634–1656.
[161] P. Le Hai, F. Lara, and B. S. Mordukhovich, *Regular subgradients of marginal functions with applications to calculus and bilevel programming*, Journal of Optimization Theory and Applications, 205 (2025), pp. 1–30.
[162] L. Leenders, M. Sollich, C. Reinert, and A. Bardow, *Integrating quantum and classical computing for multi-energy system optimization using benders decomposition*, Computers & Chemical Engineering, 188 (2024), p. 108763.
[163] S. Leyffer, G. López-Calva, and J. Nocedal, *Interior methods for mathematical programs with complementarity constraints*, SIAM Journal on Optimization, 17 (2006), pp. 52–77.
[164] H. Li, Z. Xu, G. Taylor, C. Studer, and T. Goldstein, *Visualizing the loss landscape of neural nets*, Advances in Neural Information Processing Systems, 31 (2018).
[165] J. Li, B. Gu, and H. Huang, *A fully single loop algorithm for bilevel optimization without hessian inverse*, 2021, https://arxiv.org/abs/2112.04660.
[166] J. Li, B. Gu, and H. Huang, *A fully single loop algorithm for bilevel optimization without hessian inverse*, in Proceedings of the AAAI Conference on Artificial Intelligence, vol. 36, 2022, pp. 7426–7434.
[167] J. Li, Q. Li, and A. Zemkoho, *On constraint qualifications for MPECs with applications to bilevel hyperparameter optimization for machine learning*, Arxiv Preprint Arxiv:2508.12850, (2025).
[168] Q. Li, Z. Li, and A. Zemkoho, *Bilevel hyperparameter optimization for support vector classification: theoretical analysis and a solution method*, Mathematical Methods of Operations Research, 96 (2022), pp. 315–350.
[169] G.-H. Lin, M. Xu, and J. J. Ye, *On solving simple bilevel programs with a nonconvex lower level program*, Mathematical Programming, 144 (2014), pp. 277–305.
[170] Q. Lin, Z. Fang, Y. Chen, K. C. Tan, and Y. Li, *Evolutionary architectural search for generative adversarial networks*, IEEE Transactions on Emerging Topics in Computational Intelligence, 6 (2022), pp. 783–794.
[171] B. Liu, M. Ye, S. Wright, P. Stone, and Q. Liu, *Bome! bilevel optimization made easy: A simple first-order approach*, Advances in Neural Information Processing Systems, 35 (2022), pp. 17248–17262.
[172] G. Liu, J. Ye, and J. Zhu, *Partial exact penalty for mathematical programs with equilibrium constraints*, Set-valued Analysis, 16 (2008), pp. 785–804.
[173] H. Liu and S. Satoh, *Rethinking adversarial training with a simple baseline*, Arxiv Preprint Arxiv:2306.07613, (2023).
[174] H. Liu, K. Simonyan, and Y. Yang, *Darts: Differentiable architecture search*, 2019, https://arxiv.org/abs/1806.09055.
[175] R. Liu, J. Gao, J. Zhang, D. Meng, and Z. Lin, *Investigating bi-level optimization for learning and vision from a unified perspective: A survey and beyond*, IEEE Transactions on Pattern Analysis and Machine Intelligence, 44 (2021), pp. 10045–10067.
[176] R. Liu, X. Liu, X. Yuan, S. Zeng, and J. Zhang, *A value-function-based interior-point method for non-convex bi-level optimization*, in International Conference on Machine Learning, PMLR, 2021, pp. 6882–6892.
[177] R. Liu, X. Liu, S. Zeng, J. Zhang, and Y. Zhang, *Value-function-based sequential minimization for bi-level optimization*, IEEE Transactions on Pattern Analysis and Machine Intelligence, (2023).
[178] R. Liu, Y. Liu, W. Yao, S. Zeng, and J. Zhang, *Averaged method of multipliers for bi-level optimization without lower-level strong convexity*, 2023, https://arxiv.org/abs/2302.03407.
[179] R. Liu, Y. Liu, S. Zeng, and J. Zhang, *Towards gradient-based bilevel optimization with nonconvex followers and beyond*, Advances in Neural Information Processing Systems, 34 (2021), pp. 8662–8675.
[180] R. Liu, Y. Liu, S. Zeng, and J. Zhang, *Augmenting iterative trajectory for bilevel optimization: Methodology, analysis and extensions*, 2023, https://arxiv.org/abs/2303.16397.
[181] R. Liu, P. Mu, X. Yuan, S. Zeng, and J. Zhang, *A generic first-order algorithmic framework for bi-level programming beyond lower-level singleton*, 2020, https://arxiv.org/abs/2006.04045.
[182] R. Liu, P. Mu, X. Yuan, S. Zeng, and J. Zhang, *A general descent aggregation framework for gradient-based bi-level optimization*, 2022, https://arxiv.org/abs/2102.07976.
[183] A. Löhne, *A solution method for arbitrary polyhedral convex set optimization problems*, SIAM Journal on Optimization, 35 (2025), pp. 330–346.
[184] P. Loridan and J. Morgan, *New results on approximate solution in two-level optimization*, Optimization, 20 (1989), pp. 819–836.
[185] J. Lorraine, P. Vicol, and D. Duvenaud, *Optimizing millions of hyperparameters by implicit differentiation*, in International Conference on Artificial Intelligence and Statistics, vol. 108, PMLR, 2020, pp. 1540–1552.
[186] S. Lu, *Slm: A smoothed first-order lagrangian method for structured constrained nonconvex opti-*

mization, Advances in Neural Information Processing Systems, 36 (2024).
[187] S. Lu, S. Zeng, X. Cui, M. Squillante, L. Horesh, B. Kingsbury, J. Liu, and M. Hong, *A stochastic linearized augmented lagrangian method for decentralized bilevel optimization*, Advances in Neural Information Processing Systems, 35 (2022), pp. 30638–30650.
[188] D. Maclaurin, D. Duvenaud, and R. Adams, *Gradient-based hyperparameter optimization through reversible learning*, in International Conference on Machine Learning, PMLR, 2015, pp. 2113–2122.
[189] O. L. Mangasarian, *Equivalence of the complementarity problem to a system of nonlinear equations*, SIAM Journal on Applied Mathematics, 31 (1976), pp. 89–92.
[190] P. Mehlitz, L. I. Minchenko, and A. B. Zemkoho, *A note on partial calmness for bilevel optimization problems with linearly structured lower level*, Optimization Letters, 15 (2021), pp. 1277–1291.
[191] P. Mehlitz and A. B. Zemkoho, *Sufficient optimality conditions in bilevel programming*, Mathematics of Operations Research, 46 (2021), pp. 1573–1598.
[192] S. Mei and X. Zhu, *Using machine teaching to identify optimal training-set attacks on machine learners*, in Proceedings of the AAAI Conference on Artificial Intelligence, 2015, pp. 2871–2877, https://aaai.org/ojs/index.php/AAAI/article/view/7928.
[193] A. Mitsos, G. M. Bollas, and P. I. Barton, *Bilevel optimization formulation for parameter estimation in liquid–liquid phase equilibrium problems*, Chemical Engineering Science, 64 (2009), pp. 548–559.
[194] A. Mitsos, P. Lemonidis, and P. I. Barton, *Global solution of bilevel programs with a nonconvex inner program*, Journal of Global Optimization, 42 (2008), pp. 475–513.
[195] G. M. Moore, C. Bergeron, and K. P. Bennett, *Nonsmooth bilevel programming for hyperparameter selection*, in 2009 IEEE International Conference on Data Mining Workshops, IEEE, 2009, pp. 374–381.
[196] B. S. Mordukhovich, *Variational analysis and applications*, Springer, 2018.
[197] B. S. Mordukhovich, N. M. Nam, and H. M. Phan, *Variational analysis of marginal functions with applications to bilevel programming*, Journal of Optimization Theory and Applications, 152 (2012), pp. 557–586.
[198] J. Morgan, *Constrained well-posed two-level optimization problems*, in Nonsmooth Optimization and Related Topics, Springer, 1989, pp. 307–325.
[199] J. Morgan and F. Patrone, *Stackelberg problems: Subgame perfect equilibria via Tikhonov regularization*, in Advances in dynamic games: Applications to economics, management science, engineering, and environmental management, Springer, 2006, pp. 209–221.
[200] L. Muñoz-González, B. Biggio, A. Demontis, A. Paudice, V. Wongrassamee, E. C. Lupu, and F. Roli, *Towards poisoning of deep learning algorithms with back-gradient optimization*, in Proceedings of the 10th ACM workshop on artificial intelligence and security, 2017, pp. 27–38.
[201] S. C. Narula and A. D. Nwosu, *Two-level hierarchical programming problem*, in Essays and Surveys on Multiple Criteria Decision Making, P. Hansen, ed., Berlin, Heidelberg, 1983, Springer Berlin Heidelberg, pp. 290–299.
[202] T. Okuno, A. Takeda, A. Kawana, and M. Watanabe, *On l_p-hyperparameter learning via bilevel nonsmooth optimization*, The Journal of Machine Learning Research, 22 (2021), pp. 11093–11139.
[203] J. Outrata, M. Kocvara, and J. Zowe, *Nonsmooth approach to optimization problems with equilibrium constraints: theory, applications and numerical results*, vol. 28, Springer Science & Business Media, 1998.
[204] J. V. Outrata, *A note on the usage of nondifferentiable exact penalties in some special optimization problems*, Kybernetika, 24 (1988), pp. 251–258.
[205] J. V. Outrata, *On the numerical solution of a class of Stackelberg problems*, Zeitschrift Für Operations Research, 34 (1990), pp. 255–277.
[206] R. Pan, D. Zhang, H. Zhang, X. Pan, M. Xu, J. Zhang, R. Pi, X. Wang, and T. Zhang, *Scalebio: Scalable bilevel optimization for LLM data reweighting*, in Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), 2025, pp. 31959–31982.
[207] J.-S. Pang and L. Qi, *Nonsmooth equations: motivation and algorithms*, SIAM Journal on Optimization, 3 (1993), pp. 443–465.
[208] G. P. Papavassilopoulos, *Algorithms for static Stackelberg games with linear costs and polyhedra constraints*, in 1982 21st IEEE Conference on Decision and Control, 1982, pp. 647–652, https://doi.org/10.1109/CDC.1982.268221.
[209] R. Paulavičius, P.-M. Kleniati, and C. S. Adjiman, *Global optimization of nonconvex bilevel problems: implementation and computational study of the branch-and-sandwich algorithm*, in Computer Aided Chemical Engineering, vol. 38, Elsevier, 2016, pp. 1977–1982.
[210] F. Pedregosa, *Hyperparameter optimization with approximate gradient*, in International Conference on Machine Learning, PMLR, 2016, pp. 737–746.
[211] D. Pfau and O. Vinyals, *Connecting generative adversarial networks and actor-critic methods*, Arxiv Preprint Arxiv:1610.01945, (2016).
[212] L. Qi and J. Sun, *A nonsmooth version of Newton's method*, Mathematical Programming, 58 (1993), pp. 353–367.
[213] Y. Qian, Q. Li, and A. Zemkoho, *Global relaxation-based LP-Newton method for multiple hy-*

perparameter selection in support vector classification with feature selection, Arxiv Preprint Arxiv:2312.10848, (2023).

[214] D. Ralph and S. Dempe, *Directional derivatives of the solution of a parametric nonlinear program*, Mathematical Programming, 70 (1995), pp. 159–172.

[215] D. Ralph and S. J. Wright, *Some properties of regularization and penalization schemes for mpecs*, Optimization Methods and Software, 19 (2004), pp. 527–556.

[216] J. Ren, X. Feng, B. Liu, X. Pan, Y. Fu, L. Mai, and Y. Yang, *Torchopt: An efficient library for differentiable optimization*, Journal of Machine Learning Research, 24 (2023), pp. 1–14.

[217] M. Ren, W. Zeng, B. Yang, and R. Urtasun, *Learning to reweight examples for robust deep learning*, in International conference on machine learning, PMLR, 2018, pp. 4334–4343.

[218] E. K. Ryu and S. Boyd, *Primer on monotone operator methods*, Applied and Computational Mathematics, 15 (2016), pp. 3–43.

[219] G. Savard and J. Gauvin, *The steepest descent direction for the nonlinear bilevel programming problem*, Operations Research Letters, 15 (1994), pp. 265–272.

[220] H. Scheel and S. Scholtes, *Mathematical programs with complementarity constraints: Stationarity, optimality, and sensitivity*, Mathematics of Operations Research, 25 (2000), pp. 1–22.

[221] S. Scholtes, *Convergence properties of a regularization scheme for mathematical programs with complementarity constraints*, SIAM Journal on Optimization, 11 (2001), pp. 918–936.

[222] S. Scholtes, *Introduction to piecewise differentiable equations*, Springer, 2012.

[223] M. Schubiger, G. Banjac, and J. Lygeros, *Gpu acceleration of admm for large-scale quadratic programming*, Journal of Parallel and Distributed Computing, 144 (2020), pp. 55–67.

[224] R. Selten, *Spieltheoretische behandlung eines oligopolmodells mit nachfrageträgheit: Teil i: Bestimmung des dynamischen preisgleichgewichts*, Zeitschrift Für die gesamte Staatswissenschaft/Journal of Institutional and Theoretical Economics, (1965), pp. 301–324.

[225] W. Shi, Y. Chang, and B. Gu, *Double momentum method for lower-level constrained bilevel optimization*, (2023).

[226] W. Shi, H. Huang, and B. Gu, *Efficient bi-level optimization for non-smooth optimization*, 2022, https://openreview.net/forum?id=qy4uO5c_OB.

[227] X. Shi, R. Xiao, and R. Jiang, *An adaptive algorithm for bilevel optimization on riemannian manifolds*, in The Thirty-ninth Annual Conference on Neural Information Processing Systems.

[228] J. Shihui, *A new descent method for solving ill-posed bilevel programming problems via maxmin model*, in 2013 Fourth International Conference on Digital Manufacturing & Automation, IEEE, 2013, pp. 47–50.

[229] K. Shimizu and E. Aiyoshi, *A new computational method for Stackelberg and min-max problems by use of a penalty method*, IEEE Transactions on Automatic Control, 26 (1981), pp. 460–466.

[230] K. Shimizu, Y. Ishizuka, and J. F. Bard, *Nondifferentiable and two-level mathematical programming*, Kluwer Academic Publishers, 1997.

[231] A. Sinha, P. Malo, and K. Deb, *A review on bilevel optimization: From classical to evolutionary approaches and applications*, IEEE Transactions on Evolutionary Computation, 22 (2017), pp. 276–295.

[232] K. Som, D. Thirumulanathan, and J. Dutta, *Bilevel programming problems: a view through set-valued optimization*, Annals of Operations Research, (2025), pp. 1–26.

[233] D. Sow, K. Ji, Z. Guan, and Y. Liang, *A primal-dual approach to bilevel optimization with multiple inner minima*, Arxiv Preprint Arxiv:2203.01123, (2022).

[234] D. Sow, K. Ji, and Y. Liang, *On the convergence theory for hessian-free bilevel algorithms*, in Advances in Neural Information Processing Systems, vol. 35, 2022, pp. 21425–21439.

[235] F. v. Stackelberg, *Marktform und Gleichgewicht*, 1934.

[236] S. Steffensen and M. Ulbrich, *A new relaxation scheme for mathematical programs with equilibrium constraints*, SIAM Journal on Optimization, 20 (2010), pp. 2504–2539.

[237] Y. Sun, X. Li, K. Dalal, J. Xu, A. Vikram, G. Zhang, Y. Dubois, X. Chen, X. Wang, S. Koyejo, et al., *Learning to (learn at test time): Rnns with expressive hidden states*, in Forty-second International Conference on Machine Learning.

[238] S. Tahernejad, T. K. Ralphs, and S. T. DeNegre, *A branch-and-cut algorithm for mixed integer bilevel linear optimization problems and its implementation*, Mathematical Programming Computation, 12 (2020), pp. 529–568.

[239] A. Tandon, K. Dalal, X. Li, D. Koceja, M. Rød, S. Buchanan, X. Wang, J. Leskovec, S. Koyejo, T. Hashimoto, et al., *End-to-end test-time training for long context*, Arxiv Preprint Arxiv:2512.23675, (2025).

[240] J. Terven, D.-M. Cordova-Esparza, J.-A. Romero-González, A. Ramírez-Pedraza, and E. Chávez-Urbiola, *A comprehensive survey of loss functions and metrics in deep learning*, Artificial Intelligence Review, 58 (2025), p. 195.

[241] Y. Tian, L. Shen, G. Su, Z. Li, and W. Liu, *Alphagan: Fully differentiable architecture search for generative adversarial networks*, IEEE Transactions on Pattern Analysis and Machine Intelligence, 44 (2021), pp. 6752–6766.

[242] A. N. Tikhonov, *Regularisation methods for optimal control problems*, in Doklady Akademii Nauk, vol. 162, Russian Academy of Sciences, 1965, pp. 763–765.

[243] A. Tin and A. B. Zemkoho, *Levenberg–Marquardt method and partial exact penalty parameter se-*

lection in bilevel optimization, Optimization and Engineering, 24 (2023), pp. 1343–1385.
[244] M. A. Ustun, L. Xu, B. Zeng, and X. Qian, *Hyperparameter tuning through pessimistic bilevel optimization*, Arxiv Preprint Arxiv:2412.03666, (2024).
[245] L. Vicente, G. Savard, and J. Júdice, *Descent approaches for quadratic bilevel programming*, Journal of Optimization Theory and Applications, 81 (1994), pp. 379–399.
[246] L. N. Vicente and P. H. Calamai, *Bilevel and multilevel programming: A bibliography review*, Journal of Global Optimization, 5 (1994), pp. 291–306.
[247] H. Von Stackelberg, *Market structure and equilibrium*, Springer, 2011.
[248] J. Wang, Y. Zhu, Z. Wang, Y. Zheng, J. Hao, and C. Chen, *Bierl: A meta evolutionary reinforcement learning framework via bilevel optimization*, in ECAI 2023, Ios Press, 2023, pp. 2568–2575.
[249] Q. Wang, Y. Ma, K. Zhao, and Y. Tian, *A comprehensive survey of loss functions in machine learning*, Annals of Data Science, 9 (2022), pp. 187–212.
[250] S. Ward, A. Zemkoho, and S. Ahipasaoglu, *Mathematical programs with complementarity constraints and application to hyperparameter tuning for nonlinear support vector machines*, Arxiv Preprint Arxiv:2504.13006, (2025).
[251] D. J. White and G. Anandalingam, *A penalty function approach for solving bi-level linear programs*, Journal of Global Optimization, 3 (1993), pp. 397–419.
[252] W. Wiesemann, A. Tsoukalas, P.-M. Kleniati, and B. Rustem, *Pessimistic bilevel optimization*, SIAM Journal on Optimization, 23 (2013), pp. 353–380.
[253] J. Wu, L. Zhang, and Y. Zhang, *An inexact Newton method for stationary points of mathematical programs constrained by parameterized quasi-variational inequalities*, Numerical Algorithms, 69 (2015), pp. 713–735.
[254] Q. Xiao, S. Lu, and T. Chen, *An alternating optimization method for bilevel problems under the polyak-łojasiewicz condition*, in Thirty-seventh Conference on Neural Information Processing Systems, 2023.
[255] Q. Xiao, S. Lu, and T. Chen, *A generalized alternating method for bilevel learning under the polyak-łojasiewicz condition*, 2023, https://arxiv.org/abs/2306.02422.
[256] Q. Xiao, H. Shen, W. Yin, and T. Chen, *Alternating implicit projected sgd and its efficient variants for equality-constrained bilevel optimization*, 2023, https://arxiv.org/abs/2211.07096.
[257] Q.-W. Xiao, H. Shen, W. Yin, and T. Chen, *Alternating projected sgd for equality-constrained bilevel optimization*, in International Conference on Artificial Intelligence and Statistics, PMLR, 2023, pp. 987–1023, https://api.semanticscholar.org/CorpusID:259104711.
[258] Y.-J. Xie and Y.-F. Ke, *Neural network approaches based on new NCP-functions for solving tensor complementarity problem*, Journal of Applied Mathematics and Computing, 67 (2021), pp. 833–853.
[259] X. Xu, J. Zhang, F. Liu, M. Sugiyama, and M. Kankanhalli, *Efficient adversarial contrastive learning via robustness-aware coreset selection*, Arxiv Preprint Arxiv:2302.03857, (2023).
[260] J. Yang, K. Ji, and Y. Liang, *Provably faster algorithms for bilevel optimization*, 2021, https://arxiv.org/abs/2106.04692.
[261] Y. Yang, H. Ban, M. Huang, S. Ma, and K. Ji, *Tuning-free bilevel optimization: New algorithms and convergence analysis*, Arxiv Preprint Arxiv:2410.05140, (2024).
[262] W. Yao, C. Yu, S. Zeng, and J. Zhang, *Constrained bi-level optimization: Proximal lagrangian value function approach and hessian-free algorithm*, Arxiv Preprint Arxiv:2401.16164, (2024).
[263] M. Yazdani-Jahromi, A. Tayebi, M. S. Ozdayi, R. K. Iyer, and M. Kantarcioglu, *Fair bilevel neural network (fairbinn): On balancing fairness and accuracy via Stackelberg equilibrium*, in Advances in Neural Information Processing Systems, vol. 37, 2024.
[264] F. Ye, B. Lin, X. Cao, Y. Zhang, and I. Tsang, *A first-order multi-gradient algorithm for multi-objective bi-level optimization*, Arxiv Preprint Arxiv:2401.09257, (2024).
[265] J. Ye, *New uniform parametric error bounds*, Journal of Optimization Theory and Applications, 98 (1998), pp. 197–219.
[266] J. Ye and D. Zhu, *A note on optimality conditions for bilevel programming problems*, Optimization, 39 (1997), pp. 361–366.
[267] J. Ye, D. Zhu, and Q. J. Zhu, *Exact penalization and necessary optimality conditions for generalized bilevel programming problems*, SIAM Journal on Optimization, 7 (1997), pp. 481–507.
[268] J. J. Ye, *Nondifferentiable multiplier rules for optimization and bilevel optimization problems*, SIAM Journal on Optimization, 15 (2004), pp. 252–274.
[269] J. J. Ye, X. Yuan, S. Zeng, and J. Zhang, *Difference of convex algorithms for bilevel programs with applications in hyperparameter selection*, Mathematical Programming, 198 (2023), pp. 1583–1616.
[270] J. J. Ye and D. Zhu, *Optimality conditions for bilevel programming problems*, Optimization, 33 (1995), pp. 9–27.
[271] A. B. Zemkoho, *Solving ill-posed bilevel programs*, Set-valued and Variational Analysis, 24 (2016), pp. 423–448.
[272] A. B. Zemkoho, *Estimates of generalized Hessians for optimal value functions in mathematical programming*, Set-valued and Variational Analysis, 30 (2022), pp. 847–871.
[273] A. B. Zemkoho and S. Zhou, *Theoretical and numerical comparison of the Karush–Kuhn–Tucker and value function reformulations in bilevel optimization*, Computational Optimization and Ap-

plications, 78 (2021), pp. 625–674.

[274] Z. Zhai, W. Yan, and Y.-J. A. Zhang, *Problem-parameter-free decentralized bilevel optimization*, in The Thirty-ninth Annual Conference on Neural Information Processing Systems.

[275] Y. Zhang, P. Khanduri, I. Tsaknakis, Y. Yao, M. Hong, and S. Liu, *An introduction to bi-level optimization: Foundations and applications in signal processing and machine learning*, 2023, https://arxiv.org/abs/2308.00788.

[276] Y. Zhang, P. Khanduri, I. Tsaknakis, Y. Yao, M. Hong, and S. Liu, *An introduction to bilevel optimization: Foundations and applications in signal processing and machine learning*, IEEE Signal Processing Magazine, 41 (2024), pp. 38–59.

[277] S. Zhou, A. B. Zemkoho, and A. Tin, *BOLIB: Bilevel Optimization LIBrary of test problems*, in Bilevel Optimization: Advances and Next Challenges, Springer, 2020, pp. 563–580.