AN IMPLICIT GRADIENT-TYPE METHOD FOR LINEARLY CONSTRAINED BILEVEL PROBLEMS

Ioannis Tsaknakis, Prashant Khanduri, Mingyi Hong

University of Minnesota Twin Cities
Department of Electrical and Computer Engineering
Minneapolis, MN, 55455, USA

ABSTRACT

In this work, we develop an implicit gradient-type (**IG-AL**) algorithm for bilevel optimization with *strongly convex linear inequality* constrained lower-level problems. Many learning problems of interest, including problems in distributed optimization, machine learning, economics, and transport research are captured by the above formulation. The key characteristics of the proposed algorithm are: (i) the use of a primal-dual augmented Lagrangian method for solving the lower-level problem, and (ii) construction of an implicit gradient (derived using the KKT conditions of the lower-level problem) for solving the upper-level problem. Importantly, the proposed algorithm avoids the (expensive) projection step to a half-space inherent to gradient descent-based alternatives. The performance of the proposed algorithm is evaluated on a set of numerical experiments.

Index Terms– constrained bilevel optimization, implicit gradient

1. INTRODUCTION

Bilevel optimization problems [1,2] form an important class of learning problems with two levels of hierarchy, wherein the solution of the *upper-level* problem depends on the solution of the *lower-level*. The importance of bilevel problems is demonstrated by their applicability in a wide range of applications including machine learning, specifically, problems such as hyperparameter optimization [3], meta-learning [4], adversarial learning [5], as well as in areas like economics [6] and network optimization [7].

In this paper, we focus on a special class of bilevel optimization problems, where the lower-level problem involves solving a *strongly convex problem* over a set of *linear inequality constraints*. To make this more concrete, we consider the following formulation:

$$\min_{\mathbf{x}\in\mathbb{R}^n} \Big\{ F(\mathbf{x}) := f(\mathbf{x}, \mathbf{y}^*(\mathbf{x})) \Big\}, \tag{1a}$$

$$\mathbf{y}^*(\mathbf{x}) \in \arg\min_{\mathbf{y}\in\mathbb{R}^m} \Big\{ g(\mathbf{x},\mathbf{y}) \;\big|\; A\mathbf{y} \leq \mathbf{b} \Big\}, \tag{1b}$$

where $f : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}$ is a (potentially) non-convex function, $g : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}$ is strongly-convex in $\mathbf{y}$, $A \in \mathbb{R}^{k\times n}$, and $b \in \mathbb{R}^k$; both f and g are continuously differentiable functions. Problem (1a) is referred to as the *Upper-Level (UL)*, while (1b) is called the *Lower-Level (LL)* problem. Also, note that in the above formulation, the constraints of the LL problem are independent of the UL variable $\mathbf{x}$, that is, we consider a *simple bilevel problem*.

Our goal in the above setting is to find the stationary points of function $F(\mathbf{x})$ defined in (1a). Note that since strong convexity of the LL problem's objective, $g(\mathbf{x},\mathbf{y})$, in $\mathbf{y}$ implies the uniqueness of the solution of the LL problem (for every fixed $\mathbf{x}$), the function $F(\mathbf{x})$ is well defined (i.e., for every $\mathbf{x}$ there exists a unique $\mathbf{y}^*(\mathbf{x})$, rather than a set of points). Moreover, provided the mapping $\mathbf{y}^* : \mathbb{R}^n \to \mathbb{R}^m$ is differentiable, using the chain rule we can derive the gradient of $F(\mathbf{x})$ from (1a) as

$$\nabla F(\mathbf{x}) = \nabla_x f(\mathbf{x}, \mathbf{y}^*(\mathbf{x})) + \big[\nabla \mathbf{y}^*(\mathbf{x})\big]^T \nabla_y f(\mathbf{x}, \mathbf{y}^*(\mathbf{x})). \tag{2}$$

The above gradient can be easily computed if an expression for $\nabla \mathbf{y}^*(\mathbf{x})$ can be derived. It is well known that this gradient can be easily derived using the implicit-function theorem for bilevel problems with unconstrained LL problems [8]. However, for bilevel problems with constrained LL problems computing $\nabla \mathbf{y}^*(\mathbf{x})$ is not feasible in general. In this work, we show that under some mild assumptions, not only $\mathbf{y}^*(\mathbf{x})$ returned by (1b) is differentiable, but $\nabla \mathbf{y}^*(\mathbf{x})$ can also be derived in closed-form for such problems. This makes it possible to develop gradient based algorithms to solve (1a), as the (approximate) gradient in (2) can be efficiently computed.

Prior Works. Research in bilevel optimization has a long-standing history in different fields, e.g., in Stackelberg games, economics, optimization theory [1, 2]. The classical approaches to solve bilevel optimization problems include penalty methods [9, 10], single-level reformulations of (convex) bilevel problems using the KKT conditions of the LL problem [11, 12], and trust-region methods [13, 14].

Recently, motivated by applications in machine learning, several *gradient-based methods* have been developed to tackle bilevel problems (see [15] for a comprehensive survey of these methods). For problems where the LL problem has a unique solution (as in our setting), there are two major classes of gradient-based methods depending on how we solve the LL problem and compute the gradient of (1a), *explicit* and *implicit gradient* methods.

In the explicit gradient methods, $\mathbf{y}^*(\mathbf{x})$ (in (1b)) is approximated using an iterative algorithm (e.g., gradient descent) to (approximately) solve (1b). Then, an estimate of $\nabla \mathbf{y}^*(\mathbf{x})$ is computed by backpropagating (unrolling) through the updates of the iterative algorithm [16, 17]. Note that for difficult problems (i.e., problems where multiple iterations are required to attain a satisfactory solution of the LL problem) this approach can be potentially time and space consuming, since it requires unrolling over a large number of updates. On the other hand, in implicit gradient methods $\nabla \mathbf{y}^*(\mathbf{x})$ is (approximately) computed using the implicit function theorem [8, 18]. However, as discussed earlier one of the limitations of these methods is that they can only be readily applied to unconstrained and (with some modifications) simple constrained LL problems, e.g., with linear equality constraints [19]; the major difficulty for such problems is computing an expression for $\nabla \mathbf{y}^*(\mathbf{x})$. In this work, we focus on one such simple setting and develop an implicit gradient method for the case where the optimization of the LL problem takes place over a set of linear inequality constraints.

Contributions. We develop an implicit gradient-based method, **IGAL**, for solving bilevel optimization problems with linear inequality constraints in the LL problem. We identify a number of important applications that fall under this class of problems. We propose an algorithm for solving the constrained bilevel problem, and provide its convergence analysis. The key feature of our approach lies in utilizing an augmented Lagrangian method for solving the LL problem. This allows us, (i) to use closed-form expressions for the variable updates, and (ii) to avoid computing (expensive) projections onto a half-space that are necessary for other gradient descent-type methods for solving the LL problem.

1.1. Motivating Applications

The class of bilevel programs introduced is broad enough to capture several interesting applications in the domains of information and signal processing. Below we give three such applications.

Distributed Optimization. In distributed optimization [20, 21], a set of N agents attempt to jointly minimize an objective function, g, over an undirected graph $G = (V, E)$. In the context of this work, we assume that the task of interest possesses a bilevel structure and the agents want to jointly optimize only the LL objective. The latter objective can be written as $g(\mathbf{x}, \mathbf{y}) = \sum_{i=1}^{N} g_i(\mathbf{x}, \mathbf{y}_i)$, where agent $i \in [N]$ has access only to the function g_i and its local variable $\mathbf{y}_i \in \mathbb{R}^{m_i}$, and $m = \sum_{i=1}^{N} m_i$. Then, the *consensus-based distributed optimization* problem can be formulated as follows:

$$\min_{\mathbf{x}\in\mathbb{R}^n} \Big\{ F(\mathbf{x}) := f(\mathbf{x}, \mathbf{y}^*(\mathbf{x})) \Big\}$$
$$\text{s.t. } \mathbf{y}^*(\mathbf{x}) \in \begin{cases} \arg\min_{\mathbf{y}\in\mathbb{R}^m} \sum_{i=1}^{N} g_i(\mathbf{x}, \mathbf{y}_i) \\ \text{s.t. } A\mathbf{y} = \mathbf{0}, \end{cases}$$

where $\mathbf{y} = (\mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_N) \in \mathbb{R}^m$, and the matrix $A \in \mathbb{R}^{m\times m}$ enforces consensus between neighboring nodes, that is, if $(i, j) \in E$ we have that $\mathbf{y}_i = \mathbf{y}_j$.

Adversarial Learning. Consider a learning problem with the aim to robustly train a model $h(\mathbf{x}, \mathbf{y})$, where $\mathbf{x}$ is the model parameter and $\mathbf{y}$ is the input, over a training set $\{(\mathbf{c}_i, \mathbf{d}_i)\}_{i=1}^{N}$ with $\mathbf{c}_i \in \mathbb{R}^m, \mathbf{d}_i \in \mathbb{R}$. Robustness in the context of our application means the ability to deal with adversarial examples [22], something that can be attained through an *adversarial training* procedure. This procedure can be formulated as a bilevel problem as follows:

$$\min_{\mathbf{x}\in\mathbb{R}^n} \Big\{ F(\mathbf{x}) := \frac{1}{2} \sum_{i=1}^{N} \|h(\mathbf{x}, \mathbf{c}_i + \mathbf{y}^*(\mathbf{x})) - \mathbf{d}_i\|^2 \Big\} \tag{3}$$
$$\text{s.t. } \mathbf{y}^*(\mathbf{x}) \in \begin{cases} \arg\min_{\mathbf{y}\in\mathbb{R}^m} -\frac{1}{2} \sum_{i=1}^{N} \|h(\mathbf{x}, \mathbf{c}_i + \mathbf{y}) - \mathbf{d}_i\|^2 + \frac{\gamma}{2}\|\mathbf{y}\|^2 \\ \text{s.t. } -\mathbf{b} \le \mathbf{y} \le \mathbf{b}, \end{cases}$$

where $\mathbf{y}$ is a universal (i.e., same across all training examples) perturbation which is bounded within a box $-\mathbf{b} \le \mathbf{y} \le \mathbf{b}$. The addition of the regularizer $\|\mathbf{y}\|^2$ aims to make the LL problem strongly-convex with γ as the regularization parameter.

Toll Setting. The *toll setting problem* [23, 24] is a classical application in bilevel optimization where the goal is to find the optimal configuration of tolls (positioning, price, etc.) in a transportation network such that the profit (for the toll company) is maximized. To make this more concrete, consider a (transportation) graph $G = (V, E)$ and a set of nonnegative weights $\{w_e\}_{e\in E}$ which describe the cost of routing traffic through each edge (e.g., cost of gas). Since, the users of the transportation network attempt to minimize their cost, the toll setting problem can be naturally formulated as the following bilevel program:

$$\min_{\substack{\mathbf{x}\in\mathbb{R}^{|E|} \\ \mathbf{x}\ge\mathbf{0}, \mathbf{1}^T\mathbf{x}=\mathbf{c}}} \Big\{ F(\mathbf{x}) := -\sum_{(i,j)\in E} (w_{ij} + x_{ij}) y^*_{ij}(\mathbf{x}) \Big\}$$
$$\text{s.t. } \mathbf{y}^*(\mathbf{x}) \in \begin{cases} \arg\min_{\mathbf{y}\in\mathbb{R}^{|E|}} \sum_{(i,j)\in E} (w_{ij} + x_{ij}) y_{ij} + \frac{\gamma}{2}\|\mathbf{y}\|^2 \\ \text{s.t. } \begin{cases} \mathbf{0} \le \mathbf{y} \le \mathbf{b}_1 \\ \sum_{(i,j)\in E} y_{ij} - \sum_{(j,k)\in E} y_{jk} = 0, \forall j \in V \setminus \{s, t\} \\ \sum_{(s,i)\in E} y_{si} = \sum_{(j,t)\in E} y_{jt} = \mathbf{b}_2, \end{cases} \end{cases}$$

where $\mathbf{y} = (y_{ij})_{(i,j)\in E}$ is the amount of traffic routed through each edge. Note that $\mathbf{y}$ is upper bounded by $\mathbf{b}_1$ since the amount of traffic an edge can handle is limited, while the last constraint implies that the amount of traffic that is transferred from the origin (source-s) to the destination (sink-t) is fixed. Also, the variable $\mathbf{x} = (x_{ij})_{(i,j)\in E}$ is used to control the cost the tolls add to each edge; $\mathbf{c}$ is the total budget that can be used for increasing the cost of the edges. Finally, we add a regularizer in order to make the LL problem strongly convex with $\gamma > 0$ as the regularization parameter.

2. THE PROPOSED ALGORITHM AND ANALYSIS

2.1. Implicit Gradient

Our goal is to find the stationary solutions of problem (1a)-(1b) (i.e., stationary points of F) using an implicit gradient approach. At high level, this can be achieved via a gradient descent approach by iteratively computing the gradient of $F(\mathbf{x})$ given in (2). As discussed earlier, the key quantity that is required to be computed is the gradient of the LL problem's optimal solution, i.e., $\nabla\mathbf{y}^*(\mathbf{x})$. This is not possible in general, however, with the set of assumptions introduced next, we show that we can efficiently compute $\nabla\mathbf{y}^*(\mathbf{x})$.

Assumption 1. *For problem* (1a)-(1b)*, we assume*

1. *$f(\mathbf{x}, \mathbf{y})$ and $g(\mathbf{x}, \mathbf{y})$ are one and two times continuously differentiable functions, respectively.*
2. *For every $\mathbf{x} \in \mathbb{R}^n$, $g(\mathbf{x}, \mathbf{y})$ is strongly convex in $\mathbf{y}$.*
3. *There exists $\mathbf{y} \in \mathbb{R}^m$ such that $A\mathbf{y} \le \mathbf{b}$.*
4. *Let $\overline{A}(\mathbf{y})$ be the matrix that contains the rows of A that correspond to the active constraints of $A\mathbf{y} \le \mathbf{b}$, that is we have $\overline{A}(\mathbf{y})\mathbf{y} = \overline{\mathbf{b}}$, for the proper subvector $\overline{\mathbf{b}}$ of $\mathbf{b}$. Then, we assume [1] that $\overline{A}(\mathbf{y})$ is full row rank, for $\mathbf{y}^*(\mathbf{x}), \forall \mathbf{x} \in \mathbb{R}^n$. Also, we assume that strict complementarity [25] holds, i.e., for the Lagrange multipliers $\overline{\boldsymbol{\lambda}}^*(\mathbf{x})$ that correspond to the active constraints $\overline{A}(\mathbf{y}^*(\mathbf{x}))\mathbf{y}^*(\mathbf{x}) = \overline{\mathbf{b}}$ we have $\overline{\boldsymbol{\lambda}}^*(\mathbf{x}) > 0$.*

Remark 1. *In the special case where we have box constraints, i.e., $\mathbf{a} \le \mathbf{y} \le \mathbf{b}$, the only possible non-zero values in the matrix $\overline{A}$ are $+1, -1$. In addition, there is only one non-zero value at each column. Therefore, matrix $\overline{A}$ is full row rank, that is Assumption 1.4 is automatically satisfied.*

Under the above assumptions we can derive a closed-form expression for $\nabla\mathbf{y}^*(\mathbf{x})$.

[1] Note that this assumption is the LICQ condition [25] of the LL problem. It is commonly used in literature to ensure that the optimal solutions satisfy the KKT conditions.

Lemma 1 (Implicit Gradient)**.** *Under Assumption 1, we have*

$$\nabla_x \mathbf{y}^* = \left[\nabla^2_{yy} g(\mathbf{x}, \mathbf{y}^*)\right]^{-1} \left[-\nabla^2_{xy} g(\mathbf{x}, \mathbf{y}^*) - \overline{A}^T \nabla_x \overline{\boldsymbol{\lambda}}^*\right] \quad (4)$$

$$\nabla_x \overline{\boldsymbol{\lambda}}^* = -\left[\overline{A}\left[\nabla^2_{yy} g(\mathbf{x}, \mathbf{y}^*)\right]^{-1} \overline{A}^T\right]^{-1} \left[\overline{A}\left[\nabla^2_{yy} g(\mathbf{x}, \mathbf{y}^*)\right]^{-1} \nabla^2_{xy} g(\mathbf{x}, \mathbf{y}^*)\right]. \quad (5)$$

where we set $\mathbf{y}^* := \mathbf{y}^*(\mathbf{x})$ *and* $\overline{A} := \overline{A}(\mathbf{y}^*)$.

Proof. In this proof we follow a reasoning similar to [26, Thm. 1], where differently from this work they are dealing with Nash games. To begin with, the Lagrangian of problem (1b) is

$$L(\mathbf{x}, \mathbf{y}, \boldsymbol{\lambda}) = g(\mathbf{x}, \mathbf{y}) + \boldsymbol{\lambda}^T (A\mathbf{y} - \mathbf{b}).$$

Now, for some fixed $\mathbf{x} \in \mathbb{R}^n$, consider a KKT point $(\mathbf{y}^*, \boldsymbol{\lambda}^*) := (\mathbf{y}^*(\mathbf{x}), \boldsymbol{\lambda}^*(\mathbf{x}))$ of (1b); note that to simplify notation we omit the dependence on $\mathbf{x}$. For the KKT point it holds that,

- $\nabla_y L(\mathbf{x}, \mathbf{y}^*, \boldsymbol{\lambda}^*) = \nabla_y g(\mathbf{x}, \mathbf{y}^*) + A^T \boldsymbol{\lambda}^* = 0$
- $\boldsymbol{\lambda}^* (A\mathbf{y}^* - \mathbf{b}) = 0, \boldsymbol{\lambda}^* \geq 0, A\mathbf{y}^* - \mathbf{b} \leq 0.$

By considering only the active constraints at $(\mathbf{y}^*, \boldsymbol{\lambda}^*)$, the above set of KKT conditions can be equivalently written as

$$\nabla_y g(\mathbf{x}, \mathbf{y}^*) + \overline{A}^T \overline{\boldsymbol{\lambda}}^* = 0, \quad \overline{A}\mathbf{y}^* - \mathbf{b} = 0, \quad \overline{\boldsymbol{\lambda}}^* > 0, \quad (6)$$

where $\overline{\boldsymbol{\lambda}}^*$ denotes the elements of $\boldsymbol{\lambda}^*$ that correspond to the active constraints. Moreover, note that the point $(\mathbf{y}^*, \boldsymbol{\lambda}^*)$ is unique. The uniqueness of $\mathbf{y}^*$ results from the strong convexity of $g(\mathbf{x}, \mathbf{y})$ in $\mathbf{y}$, while the uniqueness of $\boldsymbol{\lambda}^*$ is the consequence of the full-rank of $\overline{A}$ (which ensures regularity, e.g., see [25]). In addition, we know from implicit function theorem [27] that $\mathbf{y}^*$ and $\boldsymbol{\lambda}^*$ are differentiable w.r.t. $\mathbf{x}$. Therefore, by computing the gradient of (6) w.r.t. $\mathbf{x}$ we get

$$\nabla^2_{xy} g(\mathbf{x}, \mathbf{y}^*) + \nabla^2_{yy} g(\mathbf{x}, \mathbf{y}^*) \nabla_x \mathbf{y}^* + \overline{A}^T \nabla_x \overline{\boldsymbol{\lambda}}^* = 0 \quad (7)$$

$$\overline{A} \nabla_x \mathbf{y}^* = 0. \quad (8)$$

Solving w.r.t. $\mathbf{y}^*$ in equation (7) we obtain,

$$\nabla_x \mathbf{y}^* = \left[\nabla^2_{yy} g(\mathbf{x}, \mathbf{y}^*)\right]^{-1} \left[-\nabla^2_{xy} g(\mathbf{x}, \mathbf{y}^*) - \overline{A}^T \nabla_x \overline{\boldsymbol{\lambda}}^*\right]. \quad (9)$$

Substituting (9) into (8) yields

$$\overline{A}\left[\nabla^2_{yy} g(\mathbf{x}, \mathbf{y}^*)\right]^{-1} \left[-\nabla^2_{xy} g(\mathbf{x}, \mathbf{y}^*) - \overline{A}^T \nabla_x \overline{\boldsymbol{\lambda}}^*\right] = 0 \Rightarrow$$

$$\nabla_x \overline{\boldsymbol{\lambda}}^* = -\left[\overline{A}\left[\nabla^2_{yy} g(\mathbf{x}, \mathbf{y}^*)^{-1}\right] \overline{A}^T\right]^{-1} \left[\overline{A}\left[\nabla^2_{yy} g(\mathbf{x}, \mathbf{y}^*)\right]^{-1} \nabla^2_{xy} g(\mathbf{x}, \mathbf{y}^*)\right].$$

Finally, notice that due to the strong convexity of g the KKT point $\mathbf{y}^*$ corresponds to the unique optimal solution of problem (1b). Therefore, we have the proof. □

2.2. Convergence Analysis

Lemma 1 above provides us with a closed-form expression for the computation, at least in theory, of the quantity $\nabla \mathbf{y}^*(\mathbf{x})$ and thus of the gradient $\nabla F(\mathbf{x})$ (cf. (2)). However, the expressions derived in (4) and (5) require the knowledge of the exact solution $\mathbf{y}^*(\mathbf{x})$ of the LL problem (for every $\mathbf{x}$), which is not possible in general. In practice, we are only able to compute an estimate $\hat{\mathbf{y}}(\mathbf{x})$ of $\mathbf{y}^*(\mathbf{x})$. Assuming that we have access to $\hat{\mathbf{y}}(\mathbf{x})$ we can compute a gradient estimate $\hat{\nabla} F(\mathbf{x})$ by substituting $\mathbf{y}^*(\mathbf{x})$ with $\hat{\mathbf{y}}(\mathbf{x})$ in expressions (2), (4) and (5). Then, we can use this approximate gradient $\hat{\nabla} F(\mathbf{x})$ to perform one gradient descent step for solving (1a)-(1b). To summarize, we consider the following algorithmic scheme for the solving problem (1a)-(1b), which we call **IG** (i.e., implicit gradient):

1. Run the following procedure for T iterations. At iteration r we perform the following steps:
   (a) Find an approximate solution $\hat{\mathbf{y}}(\mathbf{x}^r)$ of the LL problem (1b) such that,
      i. $\|\hat{\mathbf{y}}(\mathbf{x}^r) - \mathbf{y}^*(\mathbf{x}^r)\| \leq \delta$,
      ii. $\hat{\mathbf{y}}(\mathbf{x}^r)$ is a feasible point, i.e., $A\hat{\mathbf{y}}(\mathbf{x}^r) \leq \mathbf{b}$,
      iii. It holds that $\overline{A}(\mathbf{y}^*(\mathbf{x}^r)) = \overline{A}(\hat{\mathbf{y}}(\mathbf{x}^r))$.
   (b) Compute $\hat{\nabla} F(\mathbf{x}^r)$ using (2), (4) and (5) by substituting $\hat{\mathbf{y}}(\mathbf{x}^r)$ in place of $\mathbf{y}^*(\mathbf{x}^r)$.
   (c) Gradient descent step: $\mathbf{x}^{r+1} = \mathbf{x}^r - \beta \hat{\nabla} F(\mathbf{x}^r)$.

Before we proceed with the convergence analysis, we introduce a second set of assumptions.

Assumption 2. *For problem* (1a)-(1b)*, we assume that the following conditions hold for every* $\mathbf{x}, \overline{\mathbf{x}} \in \mathbb{R}^n, \mathbf{y}, \overline{\mathbf{y}} \in \mathbb{R}^m$

1. $\|\nabla_y f(\mathbf{x}, \mathbf{y})\| \leq \overline{L}_f$
2. $\|\nabla f(\mathbf{x}, \mathbf{y}) - \nabla f(\mathbf{x}, \overline{\mathbf{y}})\| \leq L_f \|\mathbf{y} - \overline{\mathbf{y}}\|$
3. $\|\nabla_y g(\mathbf{x}, \mathbf{y}) - \nabla_y g(\mathbf{x}, \overline{\mathbf{y}})\| \leq L_g \|\mathbf{y} - \overline{\mathbf{y}}\|$
4. $\|\nabla^2_{yy} g(\mathbf{x}, \mathbf{y}) - \nabla^2_{yy} g(\mathbf{x}, \overline{\mathbf{y}})\| \leq L_{g_{yy}} \|\mathbf{y} - \overline{\mathbf{y}}\|$
5. $\|\nabla^2_{xy} g(\mathbf{x}, \mathbf{y})\| \leq \overline{L}_{g_{xy}}$
6. $\|\nabla^2_{xy} g(\mathbf{x}, \mathbf{y}) - \nabla^2_{xy} g(\mathbf{x}, \overline{\mathbf{y}})\| \leq L_{g_{xy}} \|\mathbf{y} - \overline{\mathbf{y}}\|$
7. $\|\nabla F(\mathbf{x}) - \nabla F(\overline{\mathbf{x}})\| \leq L \|\mathbf{x} - \overline{\mathbf{x}}\|$

Theorem 1 (Convergence of **IG**)**.** *Suppose Assumptions 1 and 2 hold. Also, assume that at each iteration* r *of the* ***IG*** *procedure we have access to* $\hat{\mathbf{y}}(\mathbf{x}^r)$ *for which the three conditions described in Step 1(a) hold, and* $1/L \leq \beta \leq 3/(2L)$*. Then, if we run* ***IG*** *for* T *iterations we get*

$$\frac{1}{T} \sum_{r=0}^{T-1} \|\nabla F(\mathbf{x}^r)\|^2 \leq \frac{c_1}{\left(\frac{3\beta}{2} - L\beta^2\right) T} + \frac{\left(L\beta^2 - \frac{\beta}{2}\right) c_2^2 \delta^2}{\left(\frac{3\beta}{2} - L\beta^2\right)},$$

where c_1, c_2 *are constants;* c_2 *depends on* $\overline{L}_f$*,* L_f*,* L_g*,* $L_{g_{yy}}$*,* $\overline{L}_{g_{xy}}$*, and* $L_{g_{xy}}$.

Proof Sketch. Below, we provide a proof outline for the above result.

1. Derive upper bounds for the gradients $\|\nabla \mathbf{y}^*(\mathbf{x})\|$ and $\|\nabla \lambda^*(\mathbf{x})\|$.
2. Derive upper bounds for the quantities $\|\nabla \mathbf{y}^*(\mathbf{x}) - \widehat{\nabla} \mathbf{y}(\mathbf{x})\|$ and $\|\nabla \overline{\boldsymbol{\lambda}}^*(\mathbf{x}) - \widehat{\nabla} \boldsymbol{\lambda}(\mathbf{x})\|$; $\widehat{\nabla} \mathbf{y}(\mathbf{x})$ and $\widehat{\nabla} \boldsymbol{\lambda}(\mathbf{x})$ denote the formulas in eq. (4), (5), respectively, where in place of the exact solution $\mathbf{y}^*(\mathbf{x})$ we have the approximate one $\hat{\mathbf{y}}(\mathbf{x})$.
3. Then, we bound the error of the gradient estimate of the implicit function F, $\|\nabla F(\mathbf{x}) - \widehat{\nabla} F(\mathbf{x})\|$. Note that $\widehat{\nabla} F(\mathbf{x})$ is given in (2), with $\mathbf{y}^*(\mathbf{x})$ replaced by the approximate solution $\hat{\mathbf{y}}(\mathbf{x})$, and $\nabla \mathbf{y}^*(\mathbf{x})$ with $\widehat{\nabla} \mathbf{y}(\mathbf{x})$. Note that this error originates as a result of approximately solving the LL problem.
4. Using the descent lemma for F and the bound $\|\nabla F(\mathbf{x}) - \widehat{\nabla} F(\mathbf{x})\|$ we show convergence to a stationary solution of $F(\mathbf{x})$ at a rate of $\frac{1}{T}$, up to some additive error term. Moreover, note that the additive error δ can be made arbitrarily small by solving the LL problem to sufficient accuracy.

□

2.3. Augmented Lagrangian method for the LL problem

The above convergence result assumes access to an approximate solution of the LL problem (1b) at every iteration, however no concrete method is described above that finds that solution. Next, we utilize an *augmented Lagrangian method* to solve LL problem (cf. (1b)) approximately. Specifically, we consider the following augmented Lagrangian [28]:

$$L_\rho(\mathbf{x}, \mathbf{y}, \boldsymbol{\mu}) = g(\mathbf{x}, \mathbf{y}) + \sum_{i=1}^{k} \frac{\left[\rho \mathbf{a}_i^T \mathbf{y} + \mu_i\right]_+^2 - \mu_i^2}{2\rho},$$

where $\mathbf{a}_i$ is the ith row of the of matrix A, $[x]_+$ denotes the projection on $\mathbb{R}_+$, $\boldsymbol{\mu}$ is the Lagrange multipliers vector, and ρ is a penalty parameter. Then, we can utilize the Aug-PDG algorithm described in [28] to solve the LL problem. The pseudocode of the proposed approach (procedure **IG** with the use of Aug-PDG for the LL problem; we call it **IG-AL**) is described in Algorithm 1.

Algorithm 1 **IG** with Aug-PDG for the LL problem (**IG-AL**)

Input: $\mathbf{x}^0, \mathbf{y}^0, \boldsymbol{\mu}^0, \alpha, \beta, \rho, \delta, T$.
for $r = 0, 1, \ldots, T-1$ **do**
  $j \leftarrow 0,\ \mathbf{y}_0^r \leftarrow \mathbf{y}^0,\ \boldsymbol{\mu}_0^r \leftarrow \boldsymbol{\mu}^0$
  while $\|\mathbf{y}^*(\mathbf{x}^r) - \mathbf{y}_j^r\|^2 + \|\boldsymbol{\mu}^*(\mathbf{x}^r) - \boldsymbol{\mu}_j^r\|^2 > \delta^2$ **do**
    $\mathbf{y}_{j+1}^r \leftarrow \mathbf{y}_j^r - \alpha \nabla_y L_\rho(\mathbf{x}^r, \mathbf{y}_j^r, \boldsymbol{\mu}_j^r)$
    $\boldsymbol{\mu}_{j+1}^r \leftarrow \text{proj}_{\mathbb{R}_+^k}\left(\boldsymbol{\mu}_j^r + \alpha \nabla_\lambda L_\rho(\mathbf{x}^r, \mathbf{y}_j^r, \boldsymbol{\mu}_j^r)\right)$
    $j \leftarrow j + 1$
  end while
  $\mathbf{y}^{r+1} \leftarrow \mathbf{y}_{j-1}^r$
  Compute $\hat{\nabla}\mathbf{y}(\mathbf{x}^r)$ using $\mathbf{y}^{r+1}$ in place of $\mathbf{y}^*$ in (4),(5)
  $\hat{\nabla} F(\mathbf{x}^r) \leftarrow \nabla_x f(\mathbf{x}^r, \mathbf{y}^{r+1}) + \hat{\nabla}\mathbf{y}(\mathbf{x}^r) \nabla_y f(\mathbf{x}^r, \mathbf{y}^{r+1})$
  $\mathbf{x}^{r+1} \leftarrow \mathbf{x}^r - \beta \hat{\nabla} F(\mathbf{x}^r)$
end for

3. EXPERIMENTS

To validate the performance of the proposed **IG-AL** algorithm, we conduct a number of experiments on the toll setting problem. Specifically, we consider the following setting: we generate a complete graph with n nodes, where the capacity $\mathbf{b}_1$ and the weights (i.e., costs) of each edge are generated uniformly at random in the interval $[1, 2]$; the amount of traffic $\mathbf{b}_2$ is set as $p\%$ of the total capacity of the edges exiting the source.

We run the **IG-AL** algorithm for $T = 100$ iterations, and for simplicity (deviating from the exact algorithm description, since it performs sufficiently well) we fix the number of iterations of the Aug-PDG loop to only 10. We choose the rest of the parameters as $\alpha = \beta = 5 \times 10^{-2}$, $\gamma = 1$ and $\rho = 1$. In order to assess the performance of the proposed algorithm we perform a comparison of the **IG-AL** algorithm with the following heuristics: a) Random assignment (RND) where the total toll budget is distributed randomly across the edges, b) Max capacity (MAX) where the total toll budget is used entirely on the edge of maximum capacity, c) Uniform (UNI) where the total toll budget is split uniformly over all the edges.

The evaluation of the different approaches is conducted as follows. First, we execute the algorithm and obtain the allocation of the total toll budget across the edges (i.e., vector $\mathbf{x}$). Then, on the new network (where the cost in each edge is increased according to the tolls budget allocation), we compute the traffic allocation (i.e., vector $\mathbf{y}$) of minimum total cost, i.e., we solve a minimum cost flow problem. The performance of each method is measured using the following quantity:

$$r = \left(c_{\text{after}} - c_{\text{before}}\right) / c_{\text{before}},$$

where $c_{\text{after}}, c_{\text{before}}$ are the costs of the minimum cost traffic allocation after and before the tolls are enforced, respectively.

We test two different settings: 1) $n = 10$, $p = 75\%$, 2) $n = 14$, $p = 90\%$. For each setting, we plot the evolution of our performance measure as a function of the total budget $\mathbf{c}$ available for the tolls. The experiment results are averaged over 10 independent runs and their variance (i.e., the range of values obtained in the 10 runs) is illustrated with the shaded regions around the respective plots. Notice that **IG-AL** performs better compared to the other approaches over the whole range of budgets. In fact, notice that the max capacity and uniform allocations are not useful in practice.

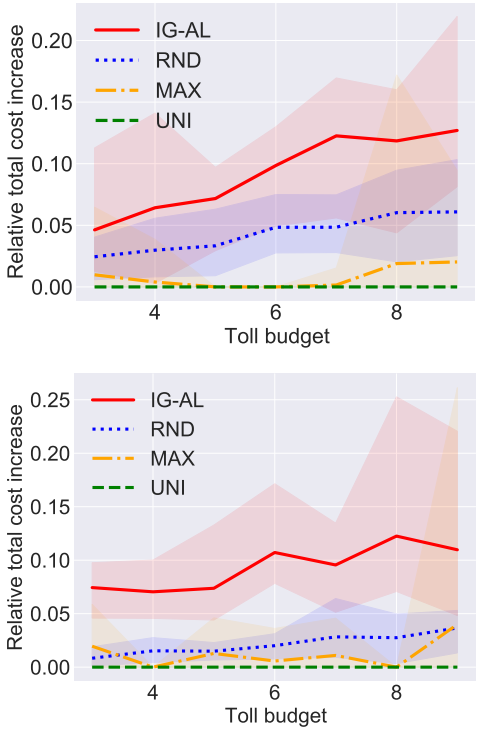

Fig. 1. Evolution of the relative total cost increase r as a function of the total toll budget. (a) $n = 10$, $p = 75\%$, (b) $n = 14$, $p = 90\%$

4. CONCLUSION

This work proposed, **IG-AL**, an implicit gradient-type algorithm for solving simple bilevel problems with linear inequality constraints in the LL problem. The core of the proposed algorithm consists of an augmented Lagrangian method that is used for solving the LL problem. The combination of implicit gradient computation for solving UL and the augmented Lagrangian to solve the LL problem leads to closed-from updates and circumvents the use of difficult projection operations. Moreover, note that implicit gradient methods in general cannot be applied directly to bilevel problems with constrained LL problems. Therefore, in the future it would be interesting to explore other simple constraint sets (e.g., ball constraints) for the LL problem, for which one can develop algorithms that admit closed form updates and at the same time the respective bilevel problem has practical applications.

5. REFERENCES

[1] Benoît Colson, Patrice Marcotte, and Gilles Savard, "Bilevel programming: A survey," *4or*, vol. 3, no. 2, pp. 87–107, 2005.

[2] Stephan Dempe and Alain Zemkoho, *Bilevel optimization*, Springer, 2020.

[3] Ankur Sinha, Tanmay Khandait, and Raja Mohanty, "A gradient-based bilevel optimization approach for tuning hyperparameters in machine learning," *arXiv preprint arXiv:2007.11022*, 2020.

[4] Aravind Rajeswaran, Chelsea Finn, Sham M Kakade, and Sergey Levine, "Meta-learning with implicit gradients," in *Advances in Neural Information Processing Systems*, H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, Eds. 2019, vol. 32, Curran Associates, Inc.

[5] Yihua Zhang, Guanhua Zhang, Prashant Khanduri, Mingyi Hong, Shiyu Chang, and Sijia Liu, "Revisiting and advancing fast adversarial training through the lens of bi-level optimization," *arXiv: 2112.12376*, 2021.

[6] Mark Cecchini, Joseph Ecker, Michael Kupferschmid, and Robert Leitch, "Solving nonlinear principal-agent problems using bilevel programming," *European Journal of Operational Research*, vol. 230, no. 2, pp. 364–373, 2013.

[7] Athanasios Migdalas, "Bilevel programming in traffic planning: Models, methods and challenge," *Journal of global optimization*, vol. 7, no. 4, pp. 381–405, 1995.

[8] Saeed Ghadimi and Mengdi Wang, "Approximation methods for bilevel programming," *arXiv preprint arXiv:1802.02246*, 2018.

[9] JJ Ye and DL Zhu, "Optimality conditions for bilevel programming problems," *Optimization*, vol. 33, no. 1, pp. 9–27, 1995.

[10] Akshay Mehra and Jihun Hamm, "Penalty method for inversion-free deep bilevel optimization," *arXiv preprint arXiv:1911.03432*, 2019.

[11] Wayne F. Bialas and Mark H. Karwan, "Two-level linear programming," *Management Science*, vol. 30, no. 8, pp. 1004–1020, 1984.

[12] Chenggen Shi, Jie Lu, and Guangquan Zhang, "An extended kuhn–tucker approach for linear bilevel programming," *Applied Mathematics and Computation*, vol. 162, no. 1, pp. 51–63, 2005.

[13] Patrice Marcotte, Gilles Savard, and D. Zhu, "A trust region algorithm for nonlinear bilevel programming," *Oper. Res. Lett.*, vol. 29, pp. 171–179, 11 2001.

[14] Benoît Colson, Patrice Marcotte, and Gilles Savard, "A trust-region method for nonlinear bilevel programming: Algorithm and computational experience," *Computational Optimization and Applications*, vol. 30, pp. 211–227, 03 2005.

[15] Risheng Liu, Jiaxin Gao, Jin Zhang, Deyu Meng, and Zhouchen Lin, "Investigating bi-level optimization for learning and vision from a unified perspective: A survey and beyond," *arXiv*, 2021.

[16] Luca Franceschi, Michele Donini, Paolo Frasconi, and Massimiliano Pontil, "Forward and reverse gradient-based hyperparameter optimization," in *International Conference on Machine Learning*. PMLR, 2017, pp. 1165–1173.

[17] Amirreza Shaban, Ching-An Cheng, Nathan Hatch, and Byron Boots, "Truncated back-propagation for bilevel optimization," in *The 22nd International Conference on Artificial Intelligence and Statistics*. PMLR, 2019, pp. 1723–1732.

[18] Prashant Khanduri, Siliang Zeng, Mingyi Hong, Hoi-To Wai, Zhaoran Wang, and Zhuoran Yang, "A near-optimal algorithm for stochastic bilevel optimization via double-momentum," *Advances in Neural Information Processing Systems*, vol. 34, 2021.

[19] Stephen Gould, Basura Fernando, Anoop Cherian, Peter Anderson, Rodrigo Cruz, and Edison Guo, "On differentiating parameterized argmin and argmax problems with application to bi-level optimization," *arXiv*, 07 2016.

[20] Tao Yang, Xinlei Yi, Junfeng Wu, Ye Yuan, Di Wu, Ziyang Meng, Yiguang Hong, Hong Wang, Zongli Lin, and Karl H Johansson, "A survey of distributed optimization," *Annual Reviews in Control*, vol. 47, pp. 278–305, 2019.

[21] Tsung-Hui Chang, Mingyi Hong, Hoi-To Wai, Xinwei Zhang, and Songtao Lu, "Distributed learning in the nonconvex world: From batch data to streaming and beyond," *IEEE Signal Processing Magazine*, vol. 37, no. 3, pp. 26–38, 2020.

[22] Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy, "Explaining and harnessing adversarial examples," *arXiv preprint arXiv:1412.6572*, 2014.

[23] Mohamed Didi-Biha, Patrice Marcotte, and Gilles Savard, *Path-based formulations of a bilevel toll setting problem*, pp. 29–50, Springer US, Boston, MA, 2006.

[24] Dr. Vyacheslav Kalashnikov, José-Fernando Camacho-Vallejo, Ronald Askin, and Nataliya Kalashnykova, "Comparison of algorithms for solving a bi-level toll setting problem," *International journal of innovative computing, information & control: IJICIC*, vol. 6, pp. 3529–3549, 08 2010.

[25] Dimitri P Bertsekas, *Nonlinear programming*, Athena Scientific Belmont, MA, 1998.

[26] Francesca Parise and Asuman Ozdaglar, "Sensitivity analysis for network aggregative games," in *2017 IEEE 56th Annual Conference on Decision and Control (CDC)*. IEEE, 2017, pp. 3200–3205.

[27] R Tyrrell Rockafellar and Roger J-B Wets, *Variational analysis*, vol. 317, Springer Science & Business Media, 2009.

[28] Min Meng and Xiuxian Li, "Aug-pdg: Linear convergence of convex optimization with inequality constraints," *arXiv: Optimization and Control*, 2020.