ACCELERATED ALGORITHMS FOR A BI-LEVEL FIXED-POINT LEARNING PROBLEM

Gabriele LEBLANC–SPANU

Abstract

In cybersecurity, automating defense strategies often comes down to solving graph-based optimization problems, and being able to compute their solution efficiently. In this work, we will use a model of bi-level optimization with fixed-point equilibrium constraints, adapted to certain cybersecurity defense problems. Classical iterative methods of first order, as Newton's method or Gradient Descent, solve this type of problems, but we propose here to apply a second-order inertial optimization algorithm called INNA [1], designed for deep learning problems, in order to accelerate its convergence. Hence we study the assumptions needed to apply INNA algorithm to our model, and thus obtain its convergence with this algorithm. We also present the results and properties we can expect from using INNA algorithm on our model.

Keywords: *bi-level optimization, deep learning, second-order methods*

BACKGROUND

Cybersecurity attacks have become more efficient in the last years, with the growing sophistication of cybercriminals, helped in particular by AI-enhanced tactics. Another factor has been the increasing complexity of cyberspace, that often enables threat actors to outpace traditional defenses [**world_economic_forum_global_2025**]. In this context, automating defense strategies is very important, because it gives the possibility to adapt to different type of attacks, and react to the attacker's actions. This automation can then be viewed as the defender solving a well-posed optimization problem, where the defender's objective is explicitly formulated (e.g., maximizing delay), and is subject to certain constraints. In general, these problems are graph-based optimization problems, as they represents the network the defender tries to protect, by strategically modifying it. We can model the defender strategically modifying the network, in reaction to an adaptive attacker, with an optimization problem containing a bi-level fixed-point condition. We will adopt this model in our study, as it enables to tackle a majority of defense problems that arise in cybersecurity nowadays.

In order to be efficient, the defender must be quick enough to respond optimally to the attacker, and this translates to accelerating the solving of our optimization problem, as it is often too computationally intensive to be directly applicable in real scenarios. In fact, classical solving algorithms are not very practical to solve this model, in reason of its complexity, in particular its non-convexity and non-smoothness. The use of different techniques is then necessary to accelerate its solving. We propose here to apply a second-order scheme called INNA [1] to accelerate convergence. The INNA algorithm combines Newton-like curvature information with inertia, and assures convergence of classical Deep Learning problems, even in non-smooth, non-convex settings. INNA is an algorithm designed for Deep Learning optimization problems, thus it can be easily applied to our bi-level optimization problem, under certain assumptions.

PRESENTATION OF THE BI-LEVEL FIXED-POINT PROBLEM

The aim of the problem is to solve:

$$\min_{\theta\in\Theta} \sum_{i=1}^{N} L(x_i(\theta), z_i), \quad L : \mathbb{R}^{d_x} \times \mathcal{Z} \to \mathbb{R} \tag{1}$$

where $\theta = (\theta_1, \dots, \theta_K)^\top \in \Theta \subseteq \mathbb{R}^K$ (closed, convex) is the decision vector. This problem is subject to, for each $i \in \{1, \dots, N\}$:

$$\text{State fixed point: } x_i(\theta) = f(x_i(\theta), u_i(\theta), P_i), \quad x_i(\theta) \in \mathbb{R}^{d_x}, \tag{2a}$$

$$\text{Control fixed point: } u_i(\theta) = \Gamma_{\mathcal{U}}\Big[u_i(\theta) - \alpha\nabla_u L\Big(\sum_{k=1}^{K} \theta_k \hat{x}_i^{(k)}(\theta, u_i(\theta)), z_i\Big)\Big], \quad u_i(\theta) \in \mathcal{U} \subseteq \mathbb{R}^{d_u}, \tag{2b}$$

where $f : \mathbb{R}^{d_x} \times \mathbb{R}^{d_u} \times \mathcal{P} \to \mathbb{R}^{d_x}$, $\Gamma_{\mathcal{U}} : \mathbb{R}^{d_u} \to \mathcal{U}$ is the Euclidean projection onto $\mathcal{U}$, $P_i \in \mathcal{P}$ is a problem instance parameter, $z_i \in \mathcal{Z}$ is a label/target, $\alpha > 0$ is a step-size, and the *roll-out features* $\left\{\hat{x}_i^{(k)}(\theta, u)\right\}_{k=0}^{K}$ are defined recursively by

$$\hat{x}_i^{(0)} := x_i^{(0)} \in \mathbb{R}^{d_x}, \qquad \hat{x}_i^{(k)}(\theta, u) := f(\hat{x}_i^{(k-1)}(\theta, u), u, P_i) \in \mathbb{R}^{d_x}, \quad k = 1, \dots, K. \tag{3}$$

AIM AND METHOD

The objective of this work is to study the possible application of INNA algorithm [1], to the problem presented earlier, as it can be a way to tackle its non-convexity and non-smoothness. As said before, what we expect is to obtain the convergence of the INNA algorithm, when applied to our problem, and to also obtain the convergence rates of INNA. For this, we need first to verify that our problem is well suited for the INNA algorithm, in other words that it verifies the assumptions needed to have convergence with INNA. These assumptions are in particular some characteristics of the loss function, which is L in our problem, that need to be tame and locally Lipschitz continuous. We will adopt a formal mathematical reasoning to prove that our bi-level fixed-point problem is well suited for INNA algorithm, and we will discuss under what assumptions this is true. To obtain the convergence rates of INNA, we will also prove mathematically that our problem satisfies the assumptions needed for having these convergence rates, and we will discuss under what assumptions this is true. To prove these two points, we will mostly rely on the theoretical work detailed in [2, 3], and we will unroll the fixed-point equation into a finite number of iterations, in order to simplify our study.

RESULTS

INNA algorithm uses only few hypothesis on the loss function to ensure that the algorithm converges. In particular, it needs that the loss is tame and locally Lipstchitz continuous. In our case, the loss function of our problem can be expressed as the composition of different functions (due to the unrolling of the fixed points). Because the composition of tame functions is not tame, we study definable functions, that are always tame. Using the founding work of [5, 2], we give (and prove) the different possible operations on definable functions, that keep the definable property of the resulting function. In particular, the composition of two definable functions is definable, and this ensures that if every function defined in our problem is definable then our loss function is definable. We also prove that if the set $\mathcal{U}$ is definable, then the projection $\Gamma_{\mathcal{U}}$ is also definable. We prove in a similar way that our loss function is locally Lipschitz continuous only if every function composing it is locally Lipschitz continuous. With this last hypothesis, this gives us the convergence of INNA for our problem. In a similar way, we obtain the convergence rates of INNA for our problem, only if the functions composing our loss function are semi-algebraic.

These hypothesis on the functions of our problem are easily obtainable, as most of the functions used today in optimization problems are definable. The main problem comes from the projection on the set $\mathcal{U}$, as this set needs to be definable, which can be harder to obtain. For example, if $\mathcal{U}$ is a polytope, it will be definable, but more complex sets may fail to be definable. The convergence rates of INNA need stronger assumptions, and in most cases we cannot ensure these convergence rates for our problem, without transforming or restricting the functions used.

To illustrate our results, we adapted the implementation used in [4] to compare the convergence time of INNA with other Deep Learning algorithms. This implementation integrates fixed points constraints, and projection on polytopes, and without being a full implementation of INNA for our problem, it shows how INNA behaves for similar problems.

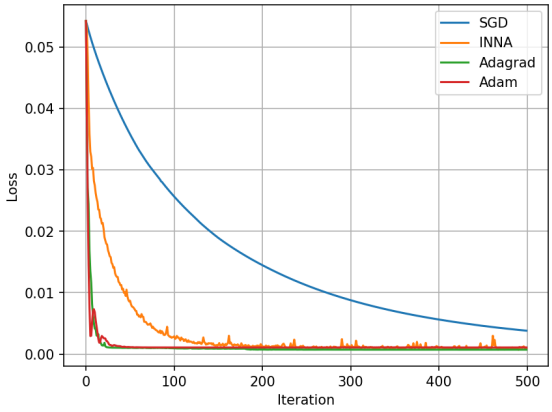

Figure 1. Loss functions for different DL algorithms

References

[1] Camille Castera et al. *An Inertial Newton Algorithm for Deep Learning*. arXiv:1905.12278 [cs]. July 2021. DOI: 10.48550/arXiv.1905.12278. URL: http://arxiv.org/abs/1905.12278 (visited on 10/03/2025).

[2] Michel Coste. *An introduction to O-minimal geometry*. eng. Dottorato di ricerca in matematica / Università di Pisa, Dipartimento di Matematica. Pisa: Istituti Editoriali e Poligrafici Internazionali, 2000. ISBN: 978-88-8147-226-0.

[3] A. D. Ioffe. "An Invitation to Tame Optimization". en. In: *SIAM Journal on Optimization* 19.4 (Jan. 2009), pp. 1894–1917. ISSN: 1052-6234, 1095-7189. DOI: 10.1137/080722059. URL: http://epubs.siam.org/doi/10.1137/080722059 (visited on 10/23/2025).

[4] Zico Kolter, David Duvenaud, and Matt Johnson. *Differentiable optimization (NeurIPS 2020 Tutorial)*. 2020. URL: https://implicit-layers-tutorial.org/differentiable_optimization/.

[5] Lou Van Den Dries and Chris Miller. "Geometric categories and o-minimal structures". In: *Duke Mathematical Journal* 84.2 (Aug. 1996). ISSN: 0012-7094. DOI: 10.1215/S0012-7094-96-08416-1. URL: https://projecteuclid.org/journals/duke-mathematical-journal/volume-84/issue-2/Geometric-categories-and-o-minimal-structures/10.1215/S0012-7094-96-08416-1.full (visited on 12/05/2025).