Non-convex Min-Max Optimization: Applications, Challenges, and Recent Theoretical Advances

Meisam Razaviyayn, Tianjian Huang, Songtao Lu, Maher Nouiehed, Maziar Sanjabi, Mingyi Hong

***Abstract*—The min-max optimization problem, also known as the saddle point problem, is a classical optimization problem which is also studied in the context of zero-sum games. Given a class of objective functions, the goal is to find a value for the argument which leads to a small objective value even for the worst-case function in the given class. Min-max optimization problems have recently become very popular in a wide range of signal and data processing applications such as fair beamforming, training generative adversarial networks (GANs), and robust machine learning (ML), to just name a few. The overarching goal of this article is to provide a survey of recent advances for an important subclass of min-max problem in which the minimization and maximization problems can be non-convex and/or non-concave. In particular, we first present a number of applications to showcase the importance of such min-max problems; then, we discuss key theoretical challenges, and provide a selective review of some exciting recent theoretical and algorithmic advances in tackling non-convex min-max problems. Finally, we point out open questions and future research directions.**[1]

I. Introduction

Recently, the class of non-convex min-max optimization problems has attracted significant attention across signal processing, optimization, and ML communities. The overarching goal of this paper is to provide a selective survey of the applications of such a new class of problem, discuss theoretical and algorithmic challenges, and present some recent advances in various directions.

To begin our discussion, let us consider the following generic problem formulation:

$$\min_{\mathbf{x}} \max_{\mathbf{y}} \quad f(\mathbf{x},\mathbf{y}) \qquad \text{(Min-Max)}$$
$$\text{s.t.} \quad \mathbf{x}\in\mathcal{X}\subseteq\mathbb{R}^d, \;\; \mathbf{y}\in\mathcal{Y}\subseteq\mathbb{R}^b,$$

where $f(\cdot,\cdot) : \mathbb{R}^d \times \mathbb{R}^b \to \mathbb{R}$ is differentiable with Lipschitz continuous gradient in $(\mathbf{x},\mathbf{y})$, possibly non-convex in $\mathbf{x}$ and possibly non-concave in $\mathbf{y}$; $\mathbf{x}\in\mathbb{R}^d$ and $\mathbf{y}\in\mathbb{R}^b$ are the optimization variables; $\mathcal{X}$ and $\mathcal{Y}$ are the feasible sets, which are assumed to be closed and convex. Notice that, while we present this article around the above min-max formulation, extending the ideas and discussions to max-min problems is straight forward.

When problem (Min-Max) is convex in $\mathbf{x}$ and concave in $\mathbf{y}$, the corresponding variational inequality (VI) becomes monotone, and a wide range of algorithms have been proposed for solving this problem; see, e.g., [1]–[4], and the references therein. However, as we will discuss in this article, solving min-max problems is challenging in non-convex setting. Such non-convex min-max optimization problems appear in different applications in signal processing (e.g., robust transceiver design,fair resource allocation [5], communication in the presence of jammers [6]), distributed signal processing [7], [8]), and ML (e.g., robust training of neural networks [9], training generative adversarial networks (GANs) [10], [11], and fair inference [12], [13]. More generally, any design problem in the presence of model uncertainty or adversary can be modeled as an optimization of the form (Min-Max). In this setup, $\mathbf{x}$ is the design parameter that should be optimized, while $\mathbf{y}$ is the uncertainty/adversary parameter which is not accurately measured, or may be adjusted by an adversary. In such scenarios, the goal in formulation (Min-Max) is to find a solution $\mathbf{x}=\bar{\mathbf{x}}$ that has a robust performance against all uncertainty/adversary values of $\mathbf{y}\in\mathcal{Y}$. Such a robustness requirement has long been deemed important in signal processing community, and it has recently played a crucial role in designing modern ML tools.

Despite the rising interests in nonconvex min-max problems, seldom have they been rigorously analyzed in either classical optimization or signal processing literature. In this article, we first present a number of applications to showcase the importance of such min-max problems, then we discuss key theoretical challenges, and provide a selective review of some recent theoretical and algorithmic advances in tackling the class of non-convex min-max problems. Finally, we will point out open questions and future research directions.

[1]The manuscript is accepted in IEEE Signal Processing Magazine.

II. Applications of non-convex min-max problems

To appreciate the importance of problem (Min-Max), let us first present a number of key applications of non-convex min-max optimization problems.

1) Generative adversarial networks (GANs): GANs [10] have recently gained tremendous popularity due to their unique ability to learn complex distributions and generate realistic samples, e.g., high resolution fake images. In the absence of labels, GANs aim at finding a mapping from a known distribution, e.g. Gaussian, to an unknown data distribution, which is only represented by empirical samples [10].

GANs consist of two neural networks: the generator and the discriminator. The goal of the generator is to generate *fake* samples which look like *real* samples in the distribution of interest. This process is done by taking i.i.d. samples from a known distribution such as Gaussian and transform it to samples similar to real ones via trained neural network. On the other hand, the discriminator's objective is to correctly classify the fake samples generated by the generator and the real samples drawn from the distribution of interest. The two-player game between the generator and the discriminator can be modeled as a min-max optimization problem [10]:

$$\min_{\mathbf{w}_g} \max_{\mathbf{w}_d} \; V(\mathbf{w}_g, \mathbf{w}_d), \tag{1}$$

where $\mathbf{w}_g$ is the generator's parameter; $\mathbf{w}_d$ is the discriminator's parameter; and $V(\cdot,\cdot)$ shows the cost function of the generator (which is equal to the negative of the discriminator's cost function). This min-max objective can be also justified as minimizing some distance between the distribution of real samples and the distribution of generated samples. In this interpretation, the distance between the two distributions is computed by solving a maximization (dual) problem; and the goal is to minimize the distance between the distribution of generated samples and the distribution of real samples. Various distance measures have been used for training GANs such as Jensen-Shannon divergence [10], f-divergence, and Wasserstein distance [11]. All these distances lead to non-convex non-concave min-max formulations for training GANs.

2) Fair ML: The past few years have witnessed several reported instances of ML algorithms suffering from systematic discrimination against individuals of certain protected groups; see, e.g., [13]–[15], and the references therein. Such instances stimulate strong interest in the field of fairness in ML which in addition to the typical goal of having an accurate learning model, brings fairness to the learning task. Imposing fairness on ML models can be done through three main approaches: preprocessing approaches, in-processing approaches, and postprocessing approaches.

To understand these three approaches, consider a ML task over a given random variables $\mathbf{X} \in \mathbb{R}^d$ representing the non-sensitive data attributes and $\mathbf{S} \in \mathbb{R}^k$ representing the sensitive attributes (such as age, gender, ethnicity, etc.). Pre-processing approaches tend to hinder discrimination by masking the training data before passing it to the decision making process. Among these methods, recent works [14], [15] have used an adversarial approach which seeks to learn a data representation $\mathbf{Z} = \zeta(\mathbf{X}, \mathbf{S})$ capable of minimizing the loss over the classifier $g(\mathbf{Z})$, and protecting the sensitive attributes $\mathbf{S}$ from an adversary $h(\mathbf{Z})$ that tries to reconstruct $\mathbf{S}$ from $\mathbf{Z}$. This requires solving the following min-max optimization problem:

$$\min_{\zeta, g} \max_{h} \; \mathbb{E}_{\mathbf{X},\mathbf{S}}\{\mathcal{L}(\zeta, g, h)\}.$$

Realizing the functions as neural networks, this formulation leads to non-convex min-max optimization problem.

Contrary to pre-processing methods, in-processing approaches impose fairness during training procedure. For example, they impose fairness by adding a regularization term that penalizes statistical dependence between the learning model output and the sensitive attributes $\mathbf{S}$. Let $g_{\boldsymbol{\theta}}(\mathbf{X}, \mathbf{S})$ be a certain output of the learning model. One can balance the learning accuracy and fairness by solving the following optimization problem:

$$\min_{\boldsymbol{\theta}} \; \mathbb{E}\left\{L\left(\boldsymbol{\theta}, \mathbf{X}\right)\right\} + \lambda \, \rho\left(g_{\boldsymbol{\theta}}(\mathbf{X}, \mathbf{S}), \mathbf{S}\right), \tag{2}$$

where $\rho(\cdot,\cdot)$ is a statistical independence measure and $L(\cdot,\cdot)$ denotes the training loss function. For example, in the classification task in which $\mathbf{X}$ contains both the input feature and the target variable, the function $L(\cdot,\cdot)$ measures the classification error of the trained classifier. Here, the parameter λ is a positive scalar balancing fairness and accuracy of the output model. When $\lambda \to \infty$, this optimization problem focuses more on making $g_{\boldsymbol{\theta}}(\mathbf{X}, \mathbf{S})$ and $\mathbf{S}$ independent, resulting in a fair inference. However, when $\lambda = 0$, no fairness is imposed and the focus is to maximize the accuracy of the model output.

Various statistical dependence measures have been proposed for use in this formulation. For example, [13] proposed using Rényi correlation to impose fairness. The Rényi correlation between two random variables $\mathbf{A}$ and $\mathbf{B}$ is defined as $\rho(\mathbf{A}, \mathbf{B}) \triangleq \sup_{k,\ell} \rho_p(k(\mathbf{A}), \ell(\mathbf{B}))$ where ρ_p is the Pearson correlation coefficient and the supremum is over the set of measurable functions $k(\cdot)$ and $\ell(\cdot)$. Plugging the definition of Rényi correlation in (2) leads to a natural min-max formulation, which is the focus of this article.

3) Adversarial ML: The formulation (Min-Max) is also instrumental to model the dynamic process of *adversarial learning*, where the model training process involves some kind of "adversary". Depending on whether the goal is to break the ML model or to make it more robust, one can formulate different min-max optimization problems, as we briefly discuss in the following sections.

Adversarial attacks. First, let us take the viewpoint of the adversary, who would like to break a ML model so that it is more likely to produce wrong predictions. In this scenario, the adversary tries to increase the error of a well-trained ML model; therefore its behavior is modeled as the *outer* optimization problem, aiming to reduce the performance of the trained model. On the other hand, the training process is modeled as the *inner* optimization problem aiming to minimize the training error.

To be more specific, take the poisoning attack [16] as an example. Let $\mathcal{D} := \{\mathbf{u}_i, t_i\}_{i=1}^N$ denote the training dataset, where $\mathbf{u}_i$ and t_i represent the features and target labels of sample i respectively. Each data sample $\mathbf{u}_i$ can be corrupted by a perturbation vector $\boldsymbol{\delta}_i$ to generate a "poisoned" sample $\mathbf{u}_i + \boldsymbol{\delta}_i$. Let $\boldsymbol{\delta} := (\boldsymbol{\delta}_1, \ldots, \boldsymbol{\delta}_N)$ be the collection of all poisoning attacks. Then, the poisoning attack problem is formulated as

$$\max_{\boldsymbol{\delta}:\, \|\boldsymbol{\delta}_i\| \le \varepsilon} \min_{\mathbf{w}} \sum_{i=1}^{N} \ell(p(\mathbf{u}_i + \boldsymbol{\delta}_i; \mathbf{w}), t_i) \tag{3}$$

where $\mathbf{w}$ is the weight of the neural network; $p(\cdot)$ is the predicted output of the neural network; and $\ell(\cdot)$ is the loss function. The constraint $\|\boldsymbol{\delta}_i\| \le \varepsilon$ indicates that the poisoned samples should not be too different from the original ones, so that the attack is not easily detectable. Note that the "max-min" problem (3) can be written equivalently in the form of (Min-Max) by adding a minus sign to the objective.

Defense against adversarial attacks. It has been widely observed that ML models, especially neural networks, are highly vulnerable to adversarial attacks, including the poisoning attack discussed in the previous subseciton, or other popular attackes such as Fast Gradient Sign Method (FGSM) attack [17] and Projected Gradient Descent (PGD) attack [18]. These adversarial attacks show that a small perturbation in the data input can significantly change the output of a neural network and deceive different neural network architectures in a wide range of applications. To make ML models robust against adversarial attacks, one popular approach is to solve the following robust training problem [9] (using similar notations as in (3)):

$$\min_{\mathbf{w}} \sum_{i=1}^{N} \max_{\boldsymbol{\delta}:\, \|\boldsymbol{\delta}_i\| \le \varepsilon} \ell(p(\mathbf{u}_i + \boldsymbol{\delta}_i; \mathbf{w}), t_i).$$

Note that compared with (3), the roles of minimization and maximization have been switched. Clearly, this optimization problem is of the form (Min-Max).

4) Distributed processing: Some constrained nonconvex optimization problems could also be formulated as a min-max saddle point problem by leveraging the primal dual approach or the method of Lagrange multipliers. An example of that appears in distributed data processing over networks. Consider a network of N nodes in a graph $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$ with $|\mathcal{V}| = N$ vertices. The nodes can communicate with their neighbors, and their goal is to jointly solve the optimization problem:

$$\min_{z} \quad \sum_{i=1}^{N} g_i(z),$$

where each $g_i(\cdot)$ is a smooth function only known by node i. Further, for simplicity of presentation, assume that $z \in \mathbb{R}$.

Such a distributed optimization setting has been widely studied in the optimization and signal processing communities over the past few decades. Let x_i be node i's local copy of z. A standard first step in distributed optimization is to rewrite the above problem as:

$$\min_{x \in \mathbb{R}^N} g(\mathbf{x}) := \sum_{i=1}^{N} g_i(x_i) \qquad \text{s.t.} \quad \mathbf{A}\mathbf{x} = 0, \tag{4}$$

where $\mathbf{A} \in \mathbb{R}^{|\mathcal{E}| \times N}$ is the incidence matrix for graph $\mathcal{G}$ and $\mathbf{x} = (x_1, \ldots, x_N)$ is the concatenation of all copies of the decision variable. The linear constraint in (4) enforces $x_i = x_j$, if i, j are neighbors. Problem (4) can be rewritten as[2]

$$\max_{y \in \mathbb{R}^{|\mathcal{E}|}} \min_{x \in \mathbb{R}^N} \sum_{i=1}^{N} g_i(x_i) + \mathbf{y}^T \mathbf{A}\mathbf{x} \tag{5}$$

where $\mathbf{y}$ is the Lagrangian multiplier. Clearly, (5) is in the form of (Min-Max), where the coupling between $\mathbf{x}$ and $\mathbf{y}$ is *linear*. A number of algorithms have been developed for it; see a recent survey [20].

5) Max-Min fair transceiver design: Consider the problem of resource allocation in a wireless communication system, where N transmitter-receiver pairs are communicating. The goal is to maximize the minimum rate among all users. To be specific, consider a setting with K parallel channels. User i transmits with power $\mathbf{p}_i := [p_i^1; \cdots, p_i^K]$, and its rate is given by: $r_i(\mathbf{p}_1, \ldots, \mathbf{p}_N) = \sum_{k=1}^{K} \log\left(1 + \frac{a_{ii}^k p_i^k}{\sigma_i^2 + \sum_{j=1, j\neq i}^{N} a_{ji}^k p_j^k}\right)$ (assuming Gaussian signaling), which is a non-convex function in $\mathbf{p}$. Here a_{ji}^k denotes the channel gain between transmitter j

[2]It can be shown that finding a stationary solution of (5) is equivalent to finding a stationary solution for (4); see [19].

and receiver i on the k-th channel, and σ_i^2 is the noise power of user i. Let $\mathbf{p} := [\mathbf{p}_1; \cdots; \mathbf{p}_N]$, then the max-min fair power control problem is given by [5]

$$\max_{\mathbf{p}\in\mathcal{P}} \min_i \{r_i(\mathbf{p})\}_{i=1}^N, \tag{6}$$

where $\mathcal{P}$ denotes the set of feasible power allocations. While the inside minimization is over a discrete variable i, we can reformulate it as a minimization over continuous variables using transformation:

$$\max_{\mathbf{p}\in\mathcal{P}} \min_{\mathbf{y}\in\Delta} \quad \sum_{i=1}^N r_i(\mathbf{p}_1, \cdots, \mathbf{p}_N) \times y_i, \tag{7}$$

where $\Delta \triangleq \{\mathbf{y} \,|\, \mathbf{y} \geq \mathbf{0};\ \sum_{i=1}^N y_i = 1\}$ is the probability simplex. Notice that the inside minimization problem is linear in $\mathbf{y}$. Hence, there always *exists* a solution at one of the extreme points of the simplex Δ. Thus, the formulation (7) is equivalent to the formulation (6). By multiplying the objective by the negative sign, we can transform the above "max-min" formulation to "min-max" form consistent with (Min-Max), i.e.,

$$\min_{\mathbf{p}\in\mathcal{P}} \max_{\mathbf{y}\in\Delta} \quad \sum_{i=1}^N -r_i(\mathbf{p}_1, \cdots, \mathbf{p}_N) \times y_i$$

6) Communication in the presence of jammers: Consider a variation of the above problem, where M jammers participate in an N-user K-channel interference channel transmission. The jammers' objective is to reduce the sum rate of the system by transmitting noises, while the goal for the regular users is to transmit as much information as possible. We use p_i^k (resp. q_j^k) to denote the i-th regular user's (resp. j-th jammer's) power on the k-th channel. The corresponding sum-rate max-min problem can be formulated as:

$$\max_{\mathbf{p}} \min_{\mathbf{q}} \sum_{k,i,j} \log\left(1 + \frac{a_{ii}^k p_i^k}{\sigma_i^2 + \sum_{\ell=1,j\neq i}^N a_{\ell i}^k p_\ell^k + b_{ji}^k q_j^k}\right),$$
$$\text{s.t. } \mathbf{p} \in \mathcal{P},\ \ \mathbf{q} \in \mathcal{Q}, \tag{8}$$

where $a_{\ell i}^k$ and b_{ji}^k represent the k-th channels between the regular user pairs (ℓ, i) and regular and jammer pair (i, j), respectively. Here $\mathcal{P}$ and $\mathcal{Q}$ denote the set of feasible power allocation constraints for the users and the jammers. Many other related formulations have been considered, mostly from the game theory perspective [6]. Similar to the previous example, by multiplying the objective by a negative sign, we obtain an optimization problem of the form (Min-Max).

III. Challenges

Solving min-max problems even up to simple notions of stationarity could be extremely challenging in the non-convex setting. This is not only because of the non-convexity of the objective (which prevents us from finding global optima), but also is due to aiming for finding a min-max solution. To see the challenges of solving non-convex min-max problems, let us compare and contrast the optimization problem (Min-Max) with the regular smooth non-convex optimization problem:

$$\min_{\mathbf{z}\in\mathcal{Z}} \ h(\mathbf{z}). \tag{9}$$

where the gradient of function h is Lipschitz continuous.

While solving general non-convex optimization problem (9) to global optimality is hard, one can apply simple iterative algorithms such as projected gradient descent (PGD) to (9) by running the iterates

$$\mathbf{z}^{r+1} = \mathcal{P}_{\mathcal{Z}}(\mathbf{z}^r - \alpha \nabla h(\mathbf{z}^r)),$$

where r is the iteration count; $\mathcal{P}_{\mathcal{Z}}$ is projection to the set $\mathcal{Z}$; and α is the step-size. Algorithms like PGD enjoy two properties:

i) The quality of the iterates improve over time, i.e., $h(\mathbf{z}^{r+1}) \leq h(\mathbf{z}^r)$, where r is the iteration number.
ii) These algorithms are guaranteed to converge to (first-order) stationary points with global iteration complexity guarantees [21] under a mild set of assumptions.

The above two properties give enough confidence to researchers to apply projected gradient descent to many non-convex problems of the form (9) and expect to find "reasonably good" solutions in practice. In contrast, *there is no widely accepted optimization tool* for solving general non-convex min-max problem (Min-Max). A simple extension of the PGD to the min-max setting is the gradient-descent ascent algorithm (GDA). This popular algorithm simply alternates between a gradient descent step on $\mathbf{x}$ and a gradient ascent step on $\mathbf{y}$ through the update rules

$$\mathbf{x}^{r+1} = \mathcal{P}_{\mathcal{X}}(\mathbf{x}^r - \alpha \nabla_x f(\mathbf{x}^r, \mathbf{y}^r)),$$
$$\mathbf{y}^{r+1} = \mathcal{P}_{\mathcal{Y}}(\mathbf{y}^r + \alpha \nabla_y f(\mathbf{x}^r, \mathbf{y}^r)),$$

where $\mathcal{P}_{\mathcal{X}}$ and $\mathcal{P}_{\mathcal{Y}}$ are the projections to the sets $\mathcal{X}$ and $\mathcal{Y}$, respectively. The update rule of $\mathbf{x}$ and $\mathbf{y}$ can be done alternatively as well [i.e., $\mathbf{y}^{r+1} = \mathcal{P}_{\mathcal{Y}}(\mathbf{y}^r + \alpha \nabla_y f(\mathbf{x}^{r+1}, \mathbf{y}^r))$]. Despite popularity of this algorithm, it fails in many practical instances. Moreover, it is not hard to construct very simple examples for which this algorithm fails to converge to any meaningful point; see Fig. 1 for an illustration.

IV. Recent developments for solving non-convex min-max problems

To understand some of the recently developed algorithms for solving non-convex min-max problems, we first need to review and discuss stationarity and optimality conditions for such problems. Then, we highlight some of the ideas leading to algorithmic developments.

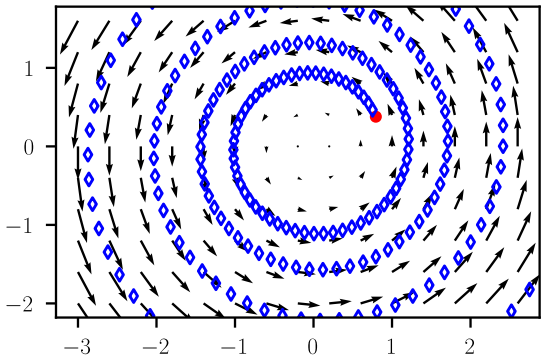

Fig. 1: GDA trajectory for the function $f(x, y) = xy$. The iterates of GDA diverge even in this simple scenario. The GDA algorithm starts from the red point and moves away from the origin (which is the optimal solution).

A. Optimality Conditions

Due to the non-convex nature of problem (Min-Max), finding the global solution is NP-hard in general [22]. Hence, the developed algorithms in the literature aimed at finding *"stationary solutions"* to this optimization problem. One approach for defining such stationarity concepts is to look at problem (Min-Max) as a game. In particular, one may ignore the order of minimization and maximization in problem (Min-Max) and view it as a zero-sum game between two players. In this game, one player is interested in solving the problem: $\min_{\mathbf{x}\in\mathcal{X}} \ f(\mathbf{x}, \mathbf{y})$, while the other player is interested in solving: $\max_{\mathbf{y}\in\mathcal{Y}} \ f(\mathbf{x}, \mathbf{y})$. Since the objective functions of both players are non-convex in general, finding a global Nash Equilibrium is not computationally tractable [22]. Hence, we may settle for finding a point satisfying first-order optimality conditions for each player's objective function, i.e., finding a point $(\bar{\mathbf{x}}, \bar{\mathbf{y}})$ satisfying

$$\begin{aligned} &\langle \nabla_{\mathbf{x}} f(\bar{\mathbf{x}}, \bar{\mathbf{y}}), \mathbf{x} - \bar{\mathbf{x}} \rangle \geq 0, \ \forall \mathbf{x} \in \mathcal{X} \\ &\qquad\qquad \text{and} \qquad\qquad \text{(Game-Stationary)} \\ &\langle \nabla_{\mathbf{y}} f(\bar{\mathbf{x}}, \bar{\mathbf{y}}), \mathbf{y} - \bar{\mathbf{y}} \rangle \leq 0, \ \forall \mathbf{y} \in \mathcal{Y}. \end{aligned}$$

This condition, which is also referred to as "quasi-Nash Equilibrium" condition in [23] or "First-order Nash Equilibrium" condition in [24], is in fact the solution of the VI corresponding to the min-max game. Moreover, one can use fixed point theorems and show existence of a point satisfying (Game-Stationary) condition under a mild set of assumptions; see, e.g., [25, Proposition 2]. In addition to existence, it is always easy to *check* whether a given point satisfies the condition (Game-Stationary). The ease of checkability and the game theoretic interpretation of the above (Game-Stationary) condition have attracted many researchers to focus on developing algorithms for finding a point satisfying this notion; see, e.g., [23]–[25], and the references therein.

A potential drawback of the above stationarity notation is its ignorance to the order of the minimization and maximization players. Notice that the Sion's min-max theorem shows that when $f(\mathbf{x}, \mathbf{y})$ is convex in $\mathbf{x}$ and concave in $\mathbf{y}$ the minimization and maximization can interchange in (Min-Max), under the mild additional assumption that either $\mathcal{X}$ or $\mathcal{Y}$ is compact. However, for the general non-convex problems, the minimization and maximization cannot interchange, i.e.,

$$\min_{\mathbf{x}\in\mathcal{X}} \max_{\mathbf{y}\in\mathcal{Y}} \ f(\mathbf{x}, \mathbf{y}) \ \neq \ \max_{\mathbf{y}\in\mathcal{Y}} \min_{\mathbf{x}\in\mathcal{X}} \ f(\mathbf{x}, \mathbf{y}).$$

Moreover, the two problems may have different solutions. Therefore, the (Game-Stationary) notion might not be practical in applications in which the minimization and maximization order is important, such as defense against adversarial attacks to neural networks (as discussed in the previous section). To modify the definition and considering the minimization and maximization order, one can define the stationarity notion by rewriting the optimization problem (Min-Max) as

$$\min_{\mathbf{x}\in\mathcal{X}} \ g(\mathbf{x}) \tag{10}$$

where $g(\mathbf{x}) \triangleq \max_{\mathbf{y}\in\mathcal{Y}} f(\mathbf{x}, \mathbf{y})$ when $\mathbf{x} \in \mathcal{X}$ and $g(\mathbf{x}) = +\infty$ when $\mathbf{x} \notin \mathcal{X}$. Using this viewpoint, we can define a point $\bar{\mathbf{x}}$ as a stationary point of (Min-Max) if $\bar{\mathbf{x}}$ is a first-order stationary point of the non-convex non-smooth optimization (10). In other words,

$$\mathbf{0} \in \partial g(\bar{\mathbf{x}}), \qquad \text{(Optimization-Stationary)}$$

where $\partial g(\bar{\mathbf{x}})$ is Fréchet sub-differential of a function $g(\cdot)$ at the point $\bar{\mathbf{x}}$, i.e., $\partial g(\bar{\mathbf{x}}) \triangleq \{\mathbf{v} \mid \liminf_{\mathbf{x}'\mapsto\mathbf{x}} \big(g(\mathbf{x}') - g(\mathbf{x}) - \langle \mathbf{v}, \mathbf{x}' - \mathbf{x}\rangle\big)/\big(\|\mathbf{x}' - \mathbf{x}\|\big) \geq 0\}$. It is again not hard to show existence of a point satisfying (Optimization-Stationary) under a mild set of assumptions such as compactness of the feasible set and continuity of the function $f(\cdot, \cdot)$. This is because of the fact that any continuous function on a compact set attains its minimum. Thus, at least the global minimum of the optimization problem (10) satisfies the optimality condition (Optimization-Stationary). This is in contrast to the (Game-Stationary) notion in which even the global minimum of (10) may not satisfy (Game-Stationary) condition. The following example, which is borrowed from [26], illustrates this fact.

Example 1: Consider the optimization problem (Min-Max) where the function $f(x, y) = 0.2xy - \cos(y)$ in the region $[-1, 1] \times [-2\pi, 2\pi]$. It is not hard to check that this min-max optimization problem has two global solutions $(x^*, y^*) = (0, -\pi)$ and $(0, \pi)$. However,

neither of these two points satisfy the condition (Game-Stationary). ■

One criticism of the (Optimization-Stationary) notion is the high computational cost of its evaluation for general non-convex problems. More precisely, unlike the (Game-Stationary) notion, checking (Optimization-Stationary) for a given point $\bar{\mathbf{x}}$ could be computationally intractable for general non-convex function $f(\mathbf{x}, \mathbf{y})$. Finally, it is worth mentioning that, although the two stationary notions (Optimization-Stationary) and (Game-Stationary) lead to different definitions of stationarity (as illustrated in Example 1), the two notions could coincide in special cases such as when the function $f(\mathbf{x}, \mathbf{y})$ is concave in $\mathbf{y}$ and its gradient is Lipschitz continuous; see [27] for more detailed discussion.

B. Algorithms Based on Potential Reduction

Constructing a potential and developing an algorithm to optimize the potential function is a popular way to solve different types of games. A natural potential to minimize is the function $g(\mathbf{x})$, defined in the previous section. In order to solve (10) using standard first-order algorithms, one needs to have access to the (sub-)gradients of the function $g(\cdot)$. While presenting the function $g(\cdot)$ in closed-form may not be possible, calculating its gradient at a given point may still be feasible via Danskin's theorem stated below.

Danskin's Theorem [28]: Assume the function $f(\mathbf{x}, \mathbf{y})$ is differentiable in $\mathbf{x}$ for all $\mathbf{x} \in \mathcal{X}$. Furthermore, assume that $f(\mathbf{x}, \mathbf{y})$ is strongly concave in $\mathbf{y}$ and that $\mathcal{Y}$ is compact. Then, the function $g(\mathbf{x})$ is differentiable in $\mathbf{x}$. Moreover, for any $\mathbf{x} \in \mathcal{X}$, we have $\nabla g(\mathbf{x}) = \nabla_{\mathbf{x}} f(\mathbf{x}, \mathbf{y}^*(\mathbf{x}))$, where $\mathbf{y}^*(\mathbf{x}) = \arg\max_{\mathbf{y} \in \mathcal{Y}} f(\mathbf{x}, \mathbf{y})$.

This theorem states that one can compute the gradient of the function $g(\mathbf{x})$ through the gradient of the function $f(\cdot, \cdot)$ when the inner problem is strongly concave, i.e. $f(\mathbf{x}, \cdot)$ is strongly concave for all $\mathbf{x}$. Therefore, under the assumptions of Danskin's theorem, to apply gradient descent algorithm to (10), one needs to run the following iterative procedure:

$$\mathbf{y}^{r+1} = \arg\max_{\mathbf{y} \in \mathcal{Y}} \; f(\mathbf{x}^r, \mathbf{y}) \tag{11a}$$

$$\mathbf{x}^{r+1} = \mathcal{P}_{\mathcal{X}}(\mathbf{x}^r - \alpha \nabla f(\mathbf{x}^r, \mathbf{y}^{r+1})). \tag{11b}$$

More precisely, the dynamics obtained in (11) is equivalent to the gradient descent dynamics $\mathbf{x}^{r+1} = \mathcal{P}_{\mathcal{X}}(\mathbf{x}^r - \alpha \nabla g(\mathbf{x}^r))$, according to Danskin's theorem. Notice that computing the value of $\mathbf{y}^{r+1}$ in (11) requires finding the exact solution of the optimization problem in (11a). In practice, finding such an exact solution may not be computationally possible. Luckily, even an inexact version of this algorithm is guaranteed to converge as long as the point $\mathbf{y}^{r+1}$ is computed accurately enough in (11a). In particular, [26] showed that the iterative algorithm

$$\text{Find } \mathbf{y}^{r+1} \text{ s.t. } f(\mathbf{x}^r, \mathbf{y}^{r+1}) \geq \max_{\mathbf{y} \in \mathcal{Y}} \; f(\mathbf{x}^r, \mathbf{y}) - \epsilon \tag{12a}$$

$$\mathbf{x}^{r+1} = \mathcal{P}_{\mathcal{X}}(\mathbf{x}^r - \alpha \nabla f(\mathbf{x}^r, \mathbf{y}^{r+1})). \tag{12b}$$

is guaranteed to find an "approximate stationary point" where the approximation accuracy depends on the value of ϵ. Interestingly, even the strong concavity assumption could be relaxed for convergence of this algorithm as long as step (12a) is computationally affordable (see [26, Theorem 35]). The rate of convergence of this algorithm is accelerated for the case in which the function $f(\mathbf{x}, \mathbf{y})$ is concave in $\mathbf{y}$ (and general nonconvex in $\mathbf{x}$) in [24] and further improved in [29] through a proximal-based acceleration procedure. These works (locally) create strongly convex approximation of the function by adding proper regularizers, and then apply accelerated iterative first-order procedures for solving the approximation. The case in which the function $f(\mathbf{x}, \mathbf{y})$ is concave in $\mathbf{y}$ has also applications in solving finite max problems of the form:

$$\min_{\mathbf{x} \in \mathcal{X}} \; \max \{f_1(\mathbf{x}), \ldots, f_n(\mathbf{x})\}. \tag{13}$$

This is due to the fact that this optimization problem can be rewritten as

$$\min_{\mathbf{x} \in \mathcal{X}} \; \max_{\mathbf{y} \in \Delta} \; \sum_{i=1}^{n} y_i f_i(\mathbf{x}), \tag{14}$$

where $\Delta \triangleq \{\mathbf{y} \,|\, \mathbf{y} \geq \mathbf{0};\ \sum_{i=1}^n y_i = 1\}$. Clearly, this optimization problem is concave in $\mathbf{y}$.

Finally, it is worth emphasizing that, the algorithms developed based on Danskin's theorem (or its variations) can only be applied to problems where step (12a) can be computed efficiently. While non-convex non-concave min-max problems do not satisfy this assumption in general, one may be able to approximate the objective function with another objective function for which this assumption is satisfied. The following example illustrates this possibility in a particular application.

Example 2: Consider the defense problem against adversarial attacks explained in the previous section where the training of a neural network requires solving the optimization problem (using similar notations as in (3)):

$$\min_{\mathbf{w}} \; \sum_{i=1}^{N} \max_{\|\boldsymbol{\delta}_i\| \leq \varepsilon} \ell(p(\mathbf{u}_i + \boldsymbol{\delta}_i; \mathbf{w}), t_i). \tag{15}$$

Clearly, the objective function is non-convex in $\mathbf{w}$ and non-concave in $\boldsymbol{\delta}$. Although finding the strongest adversarial attacker $\boldsymbol{\delta}_i$, that maximizes the inner problem in (15), might be intractable, it is usually possible

		A [9]	B [31]	Proposed [24]
Natural		98.58%	97.21%	98.20%
FGSM [17]	$\varepsilon = 0.2$	96.09%	96.19%	97.04%
	$\varepsilon = 0.3$	94.82%	96.17%	96.66%
	$\varepsilon = 0.4$	89.84%	96.14%	96.23%
PGD [18]	$\varepsilon = 0.2$	94.64%	95.01%	96.00%
	$\varepsilon = 0.3$	91.41%	94.36%	95.17%
	$\varepsilon = 0.4$	78.67%	94.11%	94.22%

TABLE I: The performance of different defense algorithms for training neural network on MNIST dataset. The first row is the accuracy when no attack is present. The second and the third row show the performance of different defense algorithms under "FGSM" and "PGD" attack, respectively. Different ε values show the magnitude of the attack. Different columns show different defense strategies. The defense method A (proposed in [9]) and the defense mechanism B (proposed in [31]) are compared against the proposed method in [24]. More details on the experiment can be found in [24].

to obtain a finite set of weak attackers. In practice, these attackers could be obtained using heuristics, e.g. projected gradient ascent or its variants [24]. Thus, [24] proposes to approximate the above problem with the following more tractable version:

$$\min_{\mathbf{w}} \frac{1}{N}\sum_{i=1}^{N} \max\left\{\ell\left(p\left(\mathbf{u}_i + \boldsymbol{\delta}_k(\mathbf{u}_i, \mathbf{w}); \mathbf{w}\right), t_i\right)\right\}_{k=1}^{K}, \tag{16}$$

where $\{\boldsymbol{\delta}_k(\mathbf{u}_i, \mathbf{w})\}_{k=1}^{K}$ is a set of K-weak attackers to data point $\mathbf{u}_i$ using the neural network's weight $\mathbf{w}$. Now the maximization is over a finite number of adversaries and hence can be transformed to a concave inner maximization problem using the transformation described in (13) and (14). The performance of this simple reformulation of the problem is depicted in Table I. As can be seen in this table, the proposed reformulation yields comparable results against state-of-the-art algorithms. In addition, unlike the other two algorithms, the proposed method enjoys theoretical convergence guarantees. ■

In the optimization problem (15), the inner optimization problem is non-concave in $\boldsymbol{\delta}$. We approximate this non-concave function with a concave function by generating a finite set of adversarial instances in (16) and used the transformation described in (13),(14) to obtain a concave inner maximization probelm. This technique can be useful in solving other general optimization problems of the form (Min-Max) by approximating the inner problem with a finite set of points in the set $\mathcal{Y}$. Another commonly used technique to approximate the inner maximization in (Min-Max) with a concave problem is to add a proper regularizer in y. This technique has been used in [30] to obtain a stable training procedure for generative adversarial networks.

C. Algorithms Based on Solving VI

Another perspective that leads to the development of algorithms is the game theoretic perspective. To present the ideas, first notice that the (Game-Stationary) notion defined in the previous section can be summarized as

$$\langle F(\bar{\mathbf{z}}), \mathbf{z} - \bar{\mathbf{z}}\rangle \geq 0, \ \forall \mathbf{z} \in \mathcal{Z} \tag{17}$$

where $\mathcal{Z} \triangleq \mathcal{X} \times \mathcal{Y}$, $\mathbf{z} = \begin{bmatrix} \mathbf{x} \\ \mathbf{y} \end{bmatrix}$, and $F(\cdot)$ is a mapping induced by the objective function $f(\mathbf{x}, \mathbf{y})$ in (Min-Max), defined by

$$F(\bar{\mathbf{z}}) \triangleq F\left(\begin{bmatrix} \bar{\mathbf{x}} \\ \bar{\mathbf{y}} \end{bmatrix}\right) = \begin{bmatrix} \nabla_{\mathbf{x}} f(\bar{\mathbf{x}}, \bar{\mathbf{y}}) \\ -\nabla_{\mathbf{y}} f(\bar{\mathbf{x}}, \bar{\mathbf{y}}) \end{bmatrix}.$$

This way of looking at the min-max optimization problem naturally leads to the design of algorithms that compute the solution of the (Stampacchia) VI (17).

When problem (Min-Max) is (strongly) convex in $\mathbf{x}$ and (strongly) concave in $\mathbf{y}$, the mapping $F(\mathbf{z})$ is (strongly) monotone[3], therefore classical methods for solving variational inequalities (VI) such as extra-gradient can be applied [1]. However, in the non-convex and/or non-concave setting of interest to this article, the strong monotonicity property no longer holds; and hence the classical algorithms cannot be used in this setting. To overcome this barrier, a natural approach is to approximate the mapping $F(\cdot)$ with a series of strongly monotone mappings and solve a series of strongly monotone VIs. The work [32] builds upon this idea and creates a series of strongly monotone VIs using proximal point algorithm, and proposes an iterative procedure named inexact proximal point (IPP) method, as given below:

$$\text{Let } F^{\gamma}_{\mathbf{z}^r}(\mathbf{z}) = F(\mathbf{z}) + \gamma^{-1}(\mathbf{z} - \mathbf{z}^r) \tag{18a}$$

$$\text{Let } \mathbf{z}^{r+1} \text{ be the (approx) solution of VI } F^{\gamma}_{\mathbf{z}^r}(\cdot). \tag{18b}$$

In (18), $\gamma > 0$ is chosen to be small enough so that the mapping $F^{\gamma}_{\mathbf{z}^r}(\mathbf{z})$ becomes strongly monotone (in $\mathbf{z}$). The strongly monotone mapping $F^{\gamma}_{\mathbf{z}^r}(\mathbf{z})$ can be solved using another iterative procedure such as extra gradient method, or the iterative procedure

$$\mathbf{z}^{t+1} = \mathcal{P}_{\mathcal{Z}}(\mathbf{z}^t - \beta F(\mathbf{z}^t)) \tag{19}$$

where β denotes the stepsize and $\mathcal{P}_{\mathcal{Z}}$ is the projection to the set $\mathcal{Z}$.

Combining the dynamics in (18) with the iterative procedure in (19) leads to a natural double-loop algorithm. This double-loop algorithm is not always guaranteed to solve the VI in (17). Instead, it has been shown that this double-loop procedure computes a solution $\mathbf{z}^*$ to the following *Minty VI*:

$$\langle F(\mathbf{z}), \mathbf{z} - \mathbf{z}^*\rangle \geq 0, \ \forall \mathbf{z} \in \mathcal{Z}. \tag{20}$$

[3]A strongly monotone mapping $F(\cdot)$ satisfies the following $\langle F(\mathbf{z}) - F(\mathbf{v}), \mathbf{z} - \mathbf{v}\rangle \geq \sigma\|\mathbf{v} - \mathbf{z}\|^2$, $\forall\ \mathbf{v}, \mathbf{z} \in \mathcal{Z}$, for some constant $\sigma > 0$. If it satisfies this inequality for $\sigma = 0$, we say the VI is monotone.

Notice that this solution concept is different than the solution $\bar{\mathbf{z}}$ in (17) as it has $F(\mathbf{z})$ instead of $F(\bar{\mathbf{z}})$ in the left hand side. While these two solution concepts are different in general, it is known that if $\mathcal{Z}$ is a convex set, then any $\mathbf{z}^*$ satisfying (20) also satisfies (17). Furthermore, if $F(\cdot)$ is monotone (or when $f(\cdot,\cdot)$ is convex in $\mathbf{x}$ and concave in $\mathbf{y}$), then any solution to (17) is also a solution to (20). While such monotonicity requirement can be slightly relaxed to cover a wider range of non-convex problems (see e.g. [33]), it is important to note that for generic non-convex, and/or non-convave function $f(\cdot,\cdot)$, there may not exist $\mathbf{z}^*$ that satisfies (20); see Example 3.

Example 3: Consider the following function which is non-convex in x, but concave in y:

$$f(x,y) = x^3 + 2xy - y^2,\ \mathcal{X}\times\mathcal{Y} = [-1,1]\times[-1,1].$$

One can verify that there are only two points, $(0,0)$ and $(-1,-1)$ that satisfy (17). However, none of the above solutions satisfies (20). To see this, one can verify that $\langle F(z), z - z^*\rangle = -4 < 0$ for $z = (0,-1)$ and $z^* = (-1,-1)$ and that $\langle F(z), z - z^*\rangle = -3 < 0$ for $z = (-1,0)$ and $z^* = (0,0)$. Since any $\mathbf{z}^*$ satisfying (20) will satisfy (17), we conclude that there is no point satisfying (20) for this min-max problem. ■

In conclusion, the VIs (17) and (20) offer new perspectives to analyze problem (Min-Max); but the existing algorithms such as (18) cannot deal with many problems covered by the potential based methods discussed in Sec. IV-B (for example when $f(\mathbf{x},\mathbf{y})$ is non-convex in x and concave in $\mathbf{y}$ or when a point satisfying (20) does not exists as we explained in Example 3). Moreover, the VI-based algorithms completely ignore the order of maximization and minimization in (Min-Max) and, hence, cannot be applied to problems in which the order of min and max is crucial.

D. Algorithms Using Single-Loop Update

The algorithms discussed in the previous two subsections are all *double loop* algorithms, in which one variable (e.g. $\mathbf{y}$) is updated in a few consecutive iterations before another variable gets updated. In many practical applications, however, *single loop* algorithms which update $\mathbf{x}$ and $\mathbf{y}$ either alternatingly or simultaneously are preferred. For example, in problem (8), the jammer often pretends to be the regular user, so it updates simultaneously with the regular users [6]. However, it is challenging to design and analyze single loop algorithms for problem (Min-Max) — even for the simplest linear problem where $f(\mathbf{x},\mathbf{y}) = \langle \mathbf{x},\mathbf{y}\rangle$, the single-loop algorithm GDA diverges; see the discussion in Sec. III.

To overcome the above challenges, [19] proposes a single loop algorithm called Hybrid Block Successive Approximation (HiBSA), whose iterations are given by

$$\mathbf{x}^{r+1} = \mathcal{P}_{\mathcal{X}}(\mathbf{x}^r - \beta^r \nabla_{\mathbf{x}} f(\mathbf{x}^r,\mathbf{y}^r)), \tag{21a}$$

$$\mathbf{y}^{r+1} = \mathcal{P}_{\mathcal{Y}}\big((1+\gamma^r\rho)\mathbf{y}^r + \rho\nabla_{\mathbf{y}} f\left(\mathbf{x}^{r+1},\mathbf{y}^r\right)\big), \tag{21b}$$

where $\beta^r, \rho > 0$ are the step sizes; $\gamma^r > 0$ is some perturbation parameter. This algorithm can be further generalized to optimize certain approximation functions of x and y, similarly to the successive convex approximation strategies used in min-only problems [34], [35]. The "hybrid" in the name refers to the fact that this algorithm contains both the descent and ascent steps.

The HiBSA iteration is very similar to the GDA algorithm mentioned previously, in which gradient descent and ascent steps are performed alternatingly. The key difference here is that the $\mathbf{y}$ update includes an additional term $\gamma^r\rho\mathbf{y}^r$, so that at each iteration the $\mathbf{y}$ update represents a "perturbed" version of the original gradient ascent step. The idea is that after the perturbation, the new iteration $\mathbf{y}^{r+1}$ is "closer" to the old iteration $\mathbf{y}^r$, so it can avoid the divergent patterns depicted in Fig. 1. Intuitively, as long as the perturbation eventually goes to zero, the algorithm will still converge to the desired solutions. Specifically, it is shown in [19] that, if $f(\mathbf{x},\mathbf{y})$ is strongly concave in $\mathbf{y}$, then one can simply remove the perturbation term (by setting $\gamma^r = 0$ for all r), and the HiBSA will converge to a point satisfying condition (Game-Stationary). Further, if $f(\mathbf{x},\mathbf{y})$ is only concave in $\mathbf{y}$, then one needs to choose $\beta^r = \mathcal{O}(1/r^{1/2})$, and $\gamma^r = \mathcal{O}(1/r^{1/4})$ to converge to a point satisfying condition (Game-Stationary).

Example 4: We apply the HiBSA to the power control problem (8). It is easy to verify that the jammer's objective is strongly concave over the feasible set.

We compare HiBSA with two classic algorithms: interference pricing [36], and the WMMSE [37], both of which are designed for power control problem *without* assuming the presence of the jammer. Our problem is tested using the following setting. We construct a network with 10 regular user and a single jammer. The interference channel among the users and the jammer is generated using uncorrelated fading channel model with channel coefficients generated from the complex zero-mean Gaussian distribution with unit covariance.

From Fig. 2 (top), we see that the pricing algorithm monotonically increases the sum rate (as is predicted by theory), while HiBSA behaves differently: after some initial oscillation, the algorithm converges to a value that has lower sum-rate. Further in Fig. 2 (bottom), we do see that by using the proposed algorithm, the jammer is able to effectively reduce the total sum rate of the system. ■

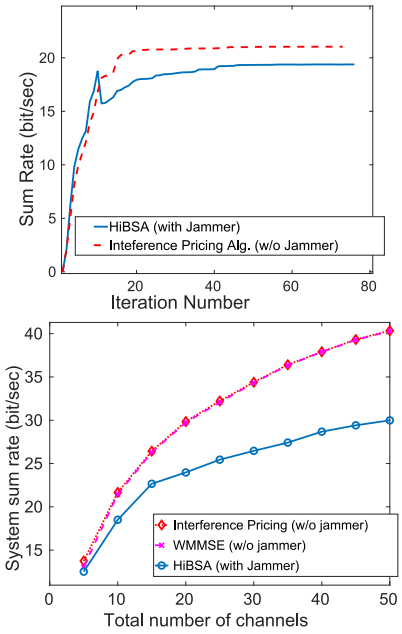

Fig. 2: The convergence curves and total averaged system performance comparing three algorithms: WMMSE, Interference Pricing and HiBSA. All users' power budget is fixed at $P = 10^{\mathrm{SNR}/10}$. For test cases without a jammer, we set $\sigma_k^2 = 1$ for all k. For test cases with a jammer, we set $\sigma_k^2 = 1/2$ for all k and let the jammer have the rest of the noise power, i.e., $p_{0,\max} = N/2$. Figure taken from [19].

E. Extension to Zeroth-order Based Algorithms

Up to now, all the algorithms reviewed require first-order (gradient) information. In this subsection, we discucss a useful extension when only *zeroth-order* (ZO) information is available. That is, we only have access to the objective values $f(\mathbf{x}, \mathbf{y})$ at a given point $(\mathbf{x}, \mathbf{y})$ at every iteration. This type of algorithm is useful, for example, in practical adversarial attack scenario where the attacker only has access to the output of the ML model [16].

To design algorithms in the ZO setting, one typically replaces the gradient $\nabla h(\mathbf{x})$ with some kind of *gradient estimate*. One popular estimate is given by

$$\widehat{\nabla}_{\mathbf{x}} h(\mathbf{x}) = \frac{1}{q} \sum_{i=1}^{q} \frac{d[h(\mathbf{x} + \mu \mathbf{u}_i) - h(\mathbf{x})]}{\mu} \mathbf{u}_i,$$

where $\{\mathbf{u}_i\}_{i=1}^q$ are q *i.i.d.* random direction vectors drawn uniformly from the unit sphere, and $\mu > 0$ is a smoothing parameter. We note that the ZO gradient estimator involves the random direction sampling w.r.t. $\mathbf{u}_i$. It is known that $\widehat{\nabla}_{\mathbf{x}} h(\mathbf{x})$ provides an unbiased estimate of the gradient of the smoothing function of f rather than the true gradient of f. Here the smoothing function of f is defined by $h_\mu(\mathbf{x}) = \mathbb{E}_{\mathbf{v}}[h(\mathbf{x} + \mu \mathbf{v})]$, where $\mathbf{v}$ follows the uniform distribution over the unit Euclidean ball. Such a gradient estimate is used in [16] to develop a ZO algorithm for solving min-max problems.

Example 5: To showcase the performance comparison between ZO based and first-order (FO) based algorithm, we consider applying HiBSA and its ZO version to the data poisoning problem (3). In particular, let us consider attacking the data set used to train a logistic regression model. We first set the poisoning ratio, i.e., the percentage of the training samples attacked, to 15%. Fig. 3 (top) demonstrates the testing accuracy (against iterations) of the model learnt from poisoned training data, where the poisoning attack is generated by ZO min-max (where the adversarial only has access to victim model outputs) and FO min-max (where the adversarial has access to details of the victim model). As we can see from Fig. 3, the poisoning attack can significantly reduce the testing accuracy compared to the clean model. Further, the ZO min-max yields promising attacking performance comparable to the FO min-max. Additionally, in Fig. 3 (bottom), we present the testing accuracy of the learned model under different data poisoning ratios. As we can see, only 5% poisoned training data can significantly break the testing accuracy of a well-trained model. The details of this experiment can be found in [16]. ■

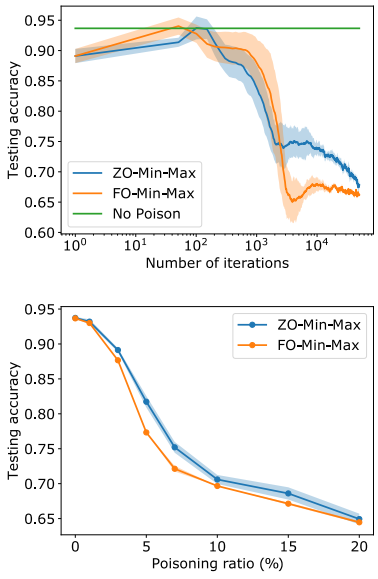

Fig. 3: Empirical performance of ZO/FO-Min-Max in poisoning attacks, where the ZO/FO-Min-Max algorithm refers to HiBSA with either ZO or FO oracle: (top) testing accuracy versus iterations (the shaded region represents variance of 10 random trials), and (bottom) testing accuracy versus data poisoning ratio. Figure taken from [16].

V. Connections among algorithms and optimality conditions

In this section, we summarize our discussion on various optimality conditions as well as algorithm performance. First, in Fig. 4, we describe the relationship between the Minty condition (20), the (Optimization-Stationary) and (Game-Stationary). Second, we compare the properties of different algorithms, such as their convergence conditions and optimality criteria in Table II. Despite the possible equivalence between the optimality conditions, we still keep the column "optimality criteria" because these are the criteria based on which the algorithms are originally designed.

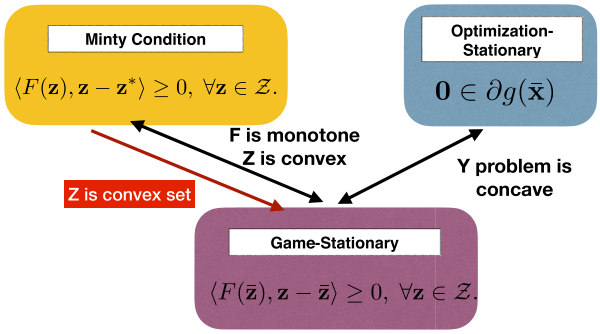

Fig. 4: Relations of different optimality conditions

.

VI. Conclusion and Future research directions

Non-convex min-max optimization problems appear in a wide range of applications. Despite the recent developments in solving these problems, the available tool sets and theories are still very limited. In particular, as discussed in section IV, these algorithms require at least one of the two following assumptions:

i) The objective of one of the two player is easy to optimize. For example, the object function in (Min-Max) is concave in $\mathbf{y}$.
ii) The Minty solutions satisfying (20) for the min-max game are the solutions to the Stampchia VI (17).

While the first assumption is mostly easy to check, the second assumption might not be easy to verify. Nevertheless, these two conditions do not imply each other. Moreover, there is a wide range of non-convex min-max problem instances that does not satisfy either of these assumptions. For solving those problems, it might be helpful to approximate them with a min-max problem satisfying one of these two assumptions.

As future research directions, a natural first open direction is toward the development of algorithms that can work under a more relaxed set of assumptions. We emphasize that the class of problems that are provably solvable (to satisfy either (Optimization-Stationary) or (Game-Stationary)) is still very limited, so it is important to extend solvable problems to a more general set of non-convex non-concave functions. One possible first step to address this is to start from algorithms that converge to desired solutions when initialized close enough to them, i.e. *local* convergence; for recent developments on this topic see [38]. Another natural research direction is about the development of the algorithms in the absence of smoothness assumptions. When the objective function of the players are non-smooth but "proximal-gradient friendly", many of the results presented in this review can still be used by simply using proximal gradient instead of gradient [39]. These scenarios happen, for example, when the objective function of the players is a summation of a smooth function and a convex non-smooth function. Additionally, it is of interest to customize the existing generic non-convex min-max algorithms to practical applications, for example to reinforcement learning [40].

One of the main applications of non-convex min-max problems, as mentioned in section II, is to design systems that are robust against uncertain parameters or the existence of adversaries. A major question in these applications is whether one can provide "robustness certificate" after solving the optimization problem. In particular, can we guarantee or measure the robustness level in these applications? The answer to this question is closely tied to the development of algorithms for solving non-convex min-max problems.

Another natural research direction is about the rate of convergence of the developed algorithms. For example, while we know solving min-max problems to (Optimization-Stationary) is easy when (Min-Max) is concave in $\mathbf{y}$ (and possibly non-convex in $\mathbf{x}$), the optimal rate of convergence (using gradient information) is still not known. Moreover, it is natural to ask whether knowing the Hessian or higher order derivatives of the objective function could improve the performance of these algorithms. So far, most of the algorithms developed for non-convex min-max problems rely only on gradient information.

References

[1] A. Nemirovski, "Prox-method with rate of convergence o(1/t) for variational inequalities with lipschitz continuous monotone operators and smooth convex-concave saddle point problems," *SIAM Journal on Optimization*, vol. 15, no. 1, pp. 229–251, 2004.
[2] A. Juditsky and A. Nemirovski, "Solving variational inequalities with monotone operators on domains given by linear minimization oracles," *Mathematical Programming*, vol. 156, no. 1-2, pp. 221–256, 2016.
[3] P. Tseng, "On accelerated proximal gradient methods for convex-concave optimization," *submitted to SIAM Journal on Optimization*, vol. 2, p. 3, 2008.

Algorithm	Optimality Criterion	Oracles	Assumptions		Other Comments
Multi-Step GDA [24]	Game-Stationary	FO	f NC in x, concave in y		Det. & DL, & Acc.
CDIAG [29]	Optimization-Stationary	FO	f NC in x, concave in y		Det. & TL, & Acc.
PG-SMD/ PGSVRG [33]	Optimization-Stationary	FO		f concave in y	St. & DL
			$f = \frac{1}{n}\sum_{i=1}^{n} f_i$ $f_i(x, y)$ NC in x	f_i concave in y	
IPP for VI [32]	Minty-Condition	FO	NC in x, y		Det. & DL
GDA [27]	Optimization-Stationary	FO	f NC in x, concave in y		Det. & St. & DL y-step has small stepsize
HiBSA [19] [16]	Game-Stationary	FO & ZO	f NC in x, concave in y		Det. & SL

TABLE II: Summary of algorithms for the min-max optimization problem (Min-Max) along with their convergence guarantees. Note that in the third column, we characterize the type of the oracle used, i.e., FO or ZO. In the last column are other comments about the algorithms, i.e deterministic (Det.) or stochastic (St.), single loop (SL) or double loop (DL) or triple loop (TL), Acceleration (Acc.) or not. Moreover, we use the abbreviations NC for non-convex.

[4] Y. Nesterov, "Dual extrapolation and its applications to solving variational inequalities and related problems," *Mathematical Programming*, vol. 109, no. 2-3, pp. 319–344, 2007.

[5] Y.-F. Liu, Y.-H. Dai, and Z.-Q. Luo, "Max-min fairness linear transceiver design for a multi-user MIMO interference channel," in *Proc. of International Conference on Communications*, 2011.

[6] R. H. Gohary, Y. Huang, Z.-Q. Luo, and J.-S. Pang, "Generallized iterative water-filling algorithm for distributed power control in the presence of a jammer," *IEEE Transactions On Signal Processing*, vol. 57, no. 7, pp. 2660–2674, 2009.

[7] G. B. Giannakis, Q. Ling, G. Mateos, I. D. Schizas, and H. Zhu, "Decentralized learning for wireless communications and networking," in *Splitting Methods in Communication and Imaging*. Springer New York, 2015.

[8] A. Nedić, A. Olshevsky, and M. G. Rabbat, "Network topology and communication-computation tradeoffs in decentralized optimization," *Proceedings of the IEEE*, vol. 106, no. 5, pp. 953–976, May 2018.

[9] A. Madry, A. Makelov, L. Schmidt, D. Tsipras, and A. Vladu, "Towards deep learning models resistant to adversarial attacks," in *International Conference on Learning Representations*, 2018.

[10] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in *Proc. of Advances in Neural Information Processing systems*, 2014, pp. 2672–2680.

[11] M. Arjovsky, S. Chintala, and L. Bottou, "Wasserstein generative adversarial networks," in *Proc. of Interantional Conference on Machine Learning*, 2017.

[12] B. H. Zhang, B. Lemoine, and M. Mitchell, "Mitigating unwanted biases with adversarial learning," in *Proceedings of the 2018 AAAI/ACM Conference on AI, Ethics, and Society*. ACM, 2018, pp. 335–340.

[13] S. Baharlouei, M. Nouiehed, A. Beirami, and M. Razaviyayn, "Rényi fair inference," in *Proc. of International Conference on Learning Representation*, 2020.

[14] D. Xu, S. Yuan, L. Zhang, and X. Wu, "Fairgan: Fairness-aware generative adversarial networks," in *2018 IEEE International Conference on Big Data (Big Data)*. IEEE, 2018, pp. 570–575.

[15] D. Madras, E. Creager, T. Pitassi, and R. S. Zemel, "Learning adversarially fair and transferable representations," in *Proc. of International Conference on Machine Learning*, 2018, pp. 3381–3390.

[16] S. Liu, S. Lu, X. Chen, Y. Feng, K. Xu, A. Al-Dujaili, M. Hong, and U.-M. Obelilly, "Min-max optimization without gradients: convergence and applications to black-box evasion and poisoning attacks," in *Proc. of International Conference on Machine Learning*, 2020.

[17] I. J. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and harnessing adversarial examples," in *Proc. of International Conference on Learning Representations*, 2015.

[18] A. Kurakin, I. Goodfellow, and S. Bengio, "Adversarial machine learning at scale," in *Proc. of International Conference on Learning Representations*, 2017.

[19] S. Lu, I. Tsaknakis, M. Hong, and Y. Chen, "Hybrid block successive approximation for one-sided non-convex min-max problems: algorithms and applications," *IEEE Transactions on Signal Processing*, 2020, accepted for publication.

[20] T. Chang, M. Hong, H. Wai, X. Zhang, and S. 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.

[21] Y. Nesterov, *Introductory lectures on convex optimization: A basic course*. Springer Science & Business Media, 2013, vol. 87.

[22] K. G. Murty and S. N. Kabadi, "Some NP-complete problems in quadratic and nonlinear programming," *Mathematical Programming*, vol. 39, no. 2, pp. 117–129, Jun 1987. [Online]. Available: http://dx.doi.org/10.1007/BF02592948

[23] J.-S. Pang and G. Scutari, "Nonconvex games with side constraints," *SIAM Journal on Optimization*, vol. 21, no. 4, pp. 1491–1522, 2011.

[24] M. Nouiehed, M. Sanjabi, T. Huang, J. D. Lee, and M. Razaviyayn, "Solving a class of non-convex min-max games using iterative first order methods," in *Advances in Neural Information Processing Systems 32*, 2019, pp. 14 905–14 916.

[25] J.-S. Pang and M. Razaviyayn, "A unified distributed algorithm for non-cooperative games," *Big Data over Networks*, p. 101, 2016.

[26] C. Jin, P. Netrapalli, and M. I. Jordan, "What is local optimality in nonconvex-nonconcave minimax optimization?" in *Proc. of Interantional Conference on Machine Learning*, 2020.

[27] T. Lin, C. Jin, and M. I. Jordan, "On gradient descent ascent for nonconvex-concave minimax problems," in *Proc. of International Conference on Machine Learning*, 2020.

[28] J. M. Danskin, "The theory of max-min, with applications," *SIAM Journal on Applied Mathematics*, vol. 14, no. 4, pp. 641–664, 1966.

[29] K. K. Thekumparampil, P. Jain, P. Netrapalli, and S. Oh, "Efficient algorithms for smooth minimax optimization," in *Advances in Neural Information Processing Systems 32*, 2019, pp. 12 659–12 670.

[30] M. Sanjabi, J. Ba, M. Razaviyayn, and J. D. Lee, "On the convergence and robustness of training gans with regularized optimal transport," in *Advances in Neural Information Processing Systems*, 2018, pp. 7091–7101.

[31] H. Zhang, Y. Yu, J. Jiao, E. Xing, L. E. Ghaoui, and M. Jordan, "Theoretically principled trade-off between robustness and accuracy," in *International Conference on Machine Learning*, 2019, pp. 7472–7482.

[32] Q. Lin, M. Liu, H. Rafique, and T. Yang, "Solving weakly-convex-weakly-concave saddle-point problems as weakly-monotone variational inequality," *arXiv preprint arXiv:1810.10207*, 2018.

[33] H. Rafique, M. Liu, Q. Lin, and T. Yang, "Non-convex min-max optimization: Provable algorithms and applications in machine learning," *Smooth Games Optimization and Machine Learning Workshop (NIPS 2018), arXiv preprint arXiv:1810.02060*, 2018.

[34] M. Hong, M. Razaviyayn, Z.-Q. Luo, and J.-S. Pang, "A unified algorithmic framework for block-structured optimization involving big data," *IEEE Signal Processing Magazine*, vol. 33, no. 1, pp. 57 – 77, 2016.

[35] M. Razaviyayn, M. Hong, and Z.-Q. Luo, "A unified convergence analysis of block successive minimization methods for nonsmooth optimization," *SIAM Journal on Optimization*, vol. 23, no. 2, pp. 1126–1153, 2013.

[36] D. A. Schmidt, C. Shi, R. A. Berry, M. L. Honig, and W. Utschick, "Comparison of distributed beamforming algorithms for mimo interference networks," *IEEE Transactions on Signal Processing*, vol. 61, no. 13, pp. 3476–3489, July 2013.

[37] Q. Shi, M. Razaviyayn, Z.-Q. Luo, and C. He, "An iteratively weighted MMSE approach to distributed sum-utility maximization for a MIMO interfering broadcast channel," *IEEE Transactions on Signal Processing*, vol. 59, no. 9, pp. 4331–4340, 2011.

[38] L. Adolphs, H. Daneshmand, A. Lucchi, and T. Hofmann, "Local saddle point optimization: A curvature exploitation approach," in *The 22nd International Conference on Artificial Intelligence and Statistics*, 2019, pp. 486–495.

[39] B. Barazandeh and M. Razaviyayn, "Solving non-convex non-differentiable min-max games using proximal gradient method," in *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2020, pp. 3162–3166.

[40] H.-T. Wai, M. Hong, Z. Yang, Z. Wang, and K. Tang, "Variance reduced policy evaluation with smooth function approximation," in *Advances in Neural Information Processing Systems 32*, 2019, pp. 5784–5795.